KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

ROZPRAWA DOKTORSKA

Przetwarzanie wizji przez roboty autonomiczne z wykorzystaniem systemów rozproszonych

Michał PODPORA

Promotor:
dr hab. inż. Jan SADECKI, prof. PO

Opole, 2011

Z całego serca pragnę podziękować mojemu promotorowi, **profesorowi Janowi Sadeckiemu**, którego niezwykła życzliwość i skromność będą zawsze w mojej pamięci.

W sposób szczególny dziękuję również wszystkim bliskim mi osobom, które okazały mi wsparcie i/lub wyrozumiałość dla mojej determinacji. Wśród tych osób na pierwszym miejscu dziękuję **Ani – mojej kochanej żonie**, która nieraz w chwili zwątpienia użyczała mi swojej siły i wytrwałości.

Dziękuję Władzom Instytutu Automatyki i Informatyki Wydziału Elektrotechniki Automatyki i Informatyki oraz Dyrekcji Zespołu Szkół Elektrycznych im.T.Kościuszki w Opolu za umożliwienie nieograniczonego dostępu do klastrów komputerowych, którymi dysponują.

W latach 2008-2010 praca była współfinansowana ze środków Europejskiego Funduszu Społecznego w ramach następujących projektów:

I. Stypendium projektu „Stypendia dla słuchaczy technicznych studiów doktoranckich w Politechnice Opolskiej” (przyznane na rok akademicki 2008/2009 oraz przyznane na rok 2009 w roku akademickim 2009/2010)

II. Stypendium projektu „Stypendia dla wyróżniających się doktorantów Politechniki Opolskiej” (przyznane na rok 2010 w roku akademickim 2009/2010)

Spis treści

Definicje wybranych pojęć i skrótów użytych w pracy

Agent upostaciowiony – robot zdolny do świadczenia usług materialnych, wyposażony w sensory oraz efektory (strukturę mechaniczną robota wraz z silnikami i elektroniką realizującą regulację niskopoziomową) określane łącznie jako warstwa sprzętowa. Warstwa ta wymaga odpowiedniego układu sterującego, który zrealizuje zlecone zadanie. Z punktu widzenia realizacji usług, agenta upostaciowionego stanowią: oprogramowanie układu sterującego oraz warstwa sprzętowa [34].

AI – *Artificial Intelligence* – sztuczna inteligencja (patrz: Dodatek A, rozdział A.1).

AV – *Active Vision* – aktywna wizja – ogół technik fragmentarycznej analizy obrazu źródłowego (patrz: rozdział 3.2.1).

DWT – *Discrete Wavelet Transform* – dyskretna transformata falkowa (patrz: rozdział 3.2.1).

HTM – *Hierarchical Temporal Memory* – pamięć hierarchiczna (patrz: rozdział 3.3).

IU – *Image Understanding* – rozumienie wizji (patrz: rozdział 1.2).

Klaster komputerowy – maszyna równoległa będąca grupą niezależnych, zazwyczaj homogenicznych komputerów, umiejscowionych w bezpośrednim sąsiedztwie, połączonych siecią komputerową, z uruchomionym środowiskiem (biblioteką lub demonem) umożliwiającym wykorzystanie mocy obliczeniowej wszystkich komputerów przy realizacji wspólnych aplikacji rozproszonych. W szczególnym przypadku: jeden komputer (por.: rozdział 3.4).

LRF – *Laser Range Finder* – laserowy skaner dalmierzowy.

Obiekt – w niniejszej pracy słowo „obiekt" jest używane w dwóch rozłącznych znaczeniach, zależnie od kontekstu (patrz: rozdział 1.2):

- w znaczeniu zgodnym z dyscypliną Automatyka i Robotyka – jako „obiekt sterowania" (głównie w rozdziale 3.6),
- lub częściej, w znaczeniu zgodnym z terminologią przetwarzania obrazu i systemów wizyjnych – jako „element otoczenia" (sceny), który można

wyodrębnić na podstawie jego cech (np. przedmiot lub osoba, znajdujące się w polu widzenia systemu wizyjnego robota).

Obliczenia równoległe – *Parallel Computing* – taki sposób przetwarzania, w którym wiele instrukcji wykonywanych jest jednocześnie [5].

POI – *Point-of-Interest* – wirtualny odpowiednik biologicznego „punktu fiksacji" (patrz: rozdział 3.2.2).

Przetwarzanie obrazu – zespół operacji prowadzących do zamiany rastrowego obrazu pochodzącego z kamer(y) systemu wizyjnego na dane wystarczające do wyodrębnienia logicznych elementów sceny (rozdział 1.2).

Punkt fiksacji – punkt sceny, na którym zatrzymuje się linia wzroku, punkt z którego pochodzi fragment obrazu stymulujący plamkę żółtą obu oczu [77].

ROI – *Region-of-Interest* – wg standardu JPEG2000: prostokątny fragment obrazu o lepszej jakości obrazu (innej wartości parametru *threshold*) [48].

Rozproszony system wizyjny – *Distributed Vision System* – Poprawne względem zasady działania tłumaczenie to „system rozproszonej wizji" – w systemach tego typu używa się wielu kamer obserwujących scenę z różnych lokalizacji [24], nie jest poruszane zagadnienie ilości węzłów/komputerów. (patrz: rozdział 4.4.4) Niniejsza praca **nie** dotyczy rozwiązania tego typu (patrz: „Rozproszony (klastrowy) system wizyjny").

Rozproszony (klastrowy) system wizyjny – system wizyjny, którego algorytmy przetwarzania i wnioskowania korzystają z klastra komputerowego (lub innej maszyny równoległej).

Rozumienie obrazu – całokształt operacji niezbędnych do podejmowania decyzji na podstawie danych z systemu wizyjnego (rozdział 1.2).

Ruchy sakkadowe – krótkie, szybkie ruchy gałek ocznych, powodujące zmianę punktu fiksacji [77] (np. przy czytaniu), również przy śledzeniu obiektów lub w reakcji na nowy obiekt w scenie.

Scena – obraz rejestrowany przez kamerę, pole widzenia.

1. Wstęp

1.1. Wprowadzenie

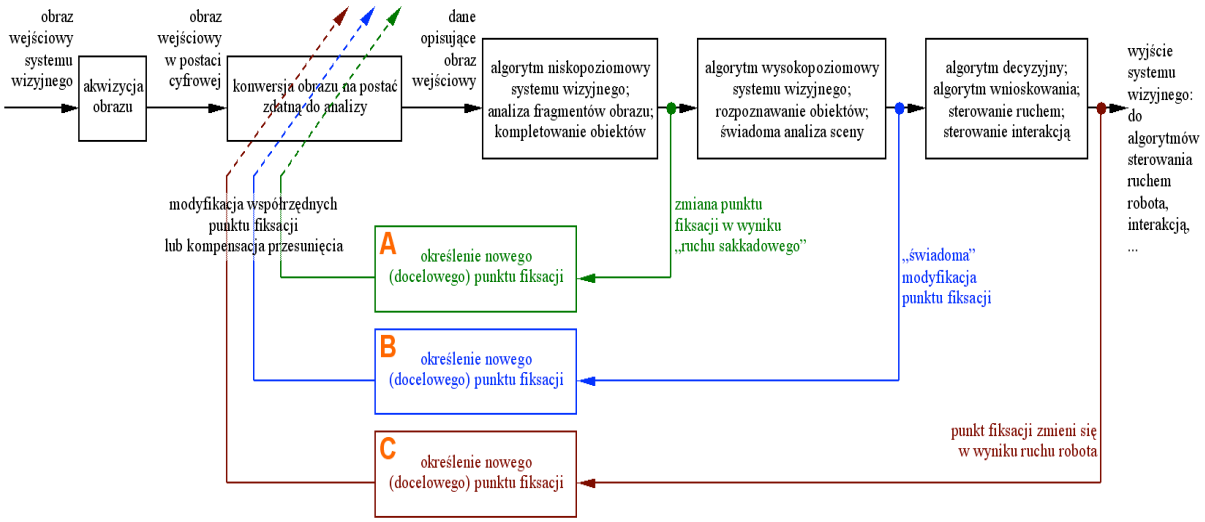
W niniejszej pracy przedstawiono wyniki rozważań oraz eksperymentów dotyczących możliwości zastosowania klastrów komputerowych w celu rozszerzenia mocy obliczeniowej robotów mobilnych. Dzisiejsze roboty mobilne nie wykorzystują klastrów do przetwarzania wizji z powodu istniejących ograniczeń technicznych. W pracy zaproponowano i zweryfikowano praktycznie sposób ominięcia tych ograniczeń. Przeniesienie części algorytmu rozumienia wizji do klastra komputerowego wiąże się nie tylko z dostępem do większej mocy obliczeniowej dla danego robota, ale również z możliwością współdzielenia przez grupę robotów wspólnej wysokopoziomowej pamięci skojarzeniowej i procedur dostępu. Ponieważ praca dotyczy możliwości wykorzystania informacji z systemu wizyjnego i algorytmów rozumienia wizji wzorowanych na ludzkim wzroku w robotach i systemach dedykowanych zadaniom interakcji człowiek-maszyna, dlatego dołożono wszelkich starań, aby cały proces przetwarzania danych wizualnych był zbliżony do biologicznego odpowiednika. Wszelkie podobieństwa i analogie są tu zabiegiem zamierzonym, wynikającym bezpośrednio z potrzeby podejmowania wysiłków badawczych w kierunku wytworzenia systemu zdolnego do natywnej (naturalnej) interakcji z człowiekiem. W tym zakresie niniejsza praca w wyraźny sposób wkomponowuje się w Propozycję Strategicznego Programu Badawczego przedstawioną przez Komitet Automatyki i Robotyki Polskiej Akademii Nauk [34] oraz współgra z działaniami Komisji Europejskiej prowadzonymi w ramach programu ICT Challenge 2 [20]. Tematyka rozprawy w szczególnym stopniu nawiązuje do następujących zadań badawczych Propozycji Strategicznego Programu Badawczego KAiR PAN [34]:

- 4.1 Algorytmy (…) rozpoznawania obrazów,
- 4.4 Algorytmy sterowania (…) interakcją z otoczeniem,
- 4.6 Percepcja i kognitywistyka,

- 4.7 Planowanie, (…) i wspomaganie decyzji,
- 4.9 Współdziałanie i inteligentne środowisko,
- 4.12 Protokoły do wykonania złożonych usług (…).

Zaproponowane w pracy rozwiązania można rozważać jako dedykowane pojedynczym zestawom robot-klaster (bez połączenia z globalną siecią i innymi robotami/klastrami), ale o wiele bardziej widoczna będzie ich użyteczność, gdy zaznaczona w pracy granica między robotem a klastrem wkomponuje się w (teoretyczną jak dotąd) granicę między wymienionymi w [34] elementami warstwy realizacji proponowanego tam rozszerzenia Internetu – pomiędzy agentem upostaciowionym (robotem) a agentem wirtualnym (aplikacją klastrową, świadczącą usługi robotowi).

Fundamentem dla opisanych w niniejszej pracy rozwiązań oraz wytworzonych technologii i algorytmów jest system wizyjny rozumiany jako układ regulacji, który można przedstawić za pomocą schematu blokowego przedstawionego na rysunku Rys. 1.1.



Rys. 1.1. Schemat blokowy przedstawiający system wizyjny jako złożony układ regulacji. Rysunek i koncepcję omówiono w rozdziale 3 pt. „Istota rozwiązania", a w szczególności w podrozdziale 3.6 – „Elekcja POI (sterowanie współrzędnymi punktu)".

W pracy przedstawiono koncepcję i przykładową implementację sprzężenia oznaczonego na powyższym rysunku literą „A". Ten fragment przedstawionego układu regulacji ma zasadnicze znaczenie dla niniejszej pracy, ponieważ tu, zdaniem autora, znajduje się dotychczasowe ograniczenie uniemożliwiające dalszy rozwój „inteligencji" robotów mobilnych przeznaczonych do interakcji z człowiekiem. Nietrywialne metody wnioskowania i interakcji z otoczeniem wymagają sporej mocy obliczeniowej, podobnie implementacje architektur emergentnych, tymczasem dzisiejsze roboty mobilne nie dysponują

wystarczającym w tym zakresie potencjałem. Idealnym rozwiązaniem byłoby użycie klastra komputerowego, niestety – systemy wizyjne działające w niedeterministycznym otoczeniu (a takim jest środowisko człowieka) nie doczekały się jak dotąd udanych uniwersalnych implementacji rozproszonych. Przyczyną takiego stanu rzeczy jest przede wszystkim problem wynikający z ograniczeń dzisiejszej technologii – zadanie rozsyłania strumienia wideo (lub nawet pojedynczych klatek obrazu) do węzłów klastra implikuje długi czas komunikacji międzywęzłowej, co utrudnia osiągnięcie korzyści ze zrównoleglenia algorytmów robota mobilnego. Rozwiązanie tego problemu przedstawiono w niniejszej pracy.

Osobnym, niezmiernie istotnym problemem są zagadnienia wykorzystania aktywnej wizji do sterowania realizowanego w ramach pętli B oraz C (Rys. 1.1), jak też, idąc jeszcze dalej, do sterowania ruchem robota mobilnego związanym z uwzględnieniem specyfiki celów sterowania sformułowanych dla konkretnego robota i konkretnego problemu sterowania. Problemy te wiążą się dodatkowo z szeregiem dobrze znanych zagadnień związanych z projektowaniem, własnościami oraz realizacją tego typu układów [11] [52]. Biorąc jednak pod uwagę zakres niniejszej pracy związany bezpośrednio z pewnymi aspektami sterowania realizowanego w obrębie pętli A – zagadnienia te zostały w niniejszej pracy pominięte, ze świadomością jednak konieczności ich uwzględniania przy konstruowaniu praktycznych rozwiązań robotów mobilnych.

1.2. Interakcja robot-człowiek w kontekście wizji

Podstawowym źródłem informacji o otoczeniu, dostarczającym danych dla robotów mających podejmować interakcję z ludźmi, jest (powinien być) obraz. Jednak dla większości robotów obraz odgrywa dużo mniejszą rolę niż dla człowieka. Konstruktorzy w rozmaity sposób usiłują obejść niedostatki wynikające z niepełnego wykorzystania wizji, stosując inne metody pozyskiwania wiedzy o otoczeniu robota, jak na przykład: czujniki ultradźwiękowe (*ang. sonar ring*, sonar), czujniki podczerwone (*ang. infrared sensors*, IR), dwuwymiarowe laserowe skanery dalmierzowe (*ang. laser range finder*, LRF) [51]. W większości konstrukcji wizja pozostaje mimo wszystko fakultatywnym źródłem informacji. Dzieje się tak dlatego, że dane wizualne są niezwykle skomplikowanym źródłem informacji, podczas gdy sonar i IR są w typowych zastosowaniach zupełnie wystarczające do bezkolizyjnego kierowania robotem, a LRF w zupełności wystarcza do nawigacji nawet po najbardziej skomplikowanych

trajektoriach. Urządzenia te okazują się jednak bezużyteczne, gdy w grę wchodzi nawiązanie interakcji z człowiekiem.

Rozumienie obrazu

Rozumienie obrazu (*ang. Image Understanding*) jest zagadnieniem niezwykle obszernym [16]. Opracowano wiele algorytmów i praktycznych przykładów zastosowań, jednak nadal jedyne całkowicie działające implementacje dotyczą zastosowań przemysłowych, z ograniczoną liczbą rozpoznawalnych obiektów[1]. W chwili obecnej nadal nie istnieje „uniwersalne" rozwiązanie, które można zastosować w każdym robocie. Popularność tematyki rozumienia wizji częściowo pokazuje internetowe kompendium wiedzy Computer Vision Online [21], prowadzone przez m.in. R. Fisher'a z Uniwersytetu w Edynburgu. Złożoność zagadnienia rozumienia wizji w sposób usystematyzowany przedstawiono w [16].

W znacznym uproszczeniu, termin „rozumienie obrazu" oznacza[2] całokształt operacji niezbędnych do podejmowania decyzji na podstawie danych z systemu wizyjnego [81]. Jeżeli dane wizualne mają być wykorzystywane w algorytmach systemu decyzyjnego robota, to rozumienie obrazu jest niezbędnym etapem toru danych.

W klasycznym podejściu podstawową i jednocześnie najtrudniejszą częścią rozumienia obrazu jest przetwarzanie obrazu.

Przetwarzanie obrazu

Przetwarzanie obrazu (*ang. Image Processing*) w znaczeniu ogólnym określa dowolne operacje związane z modyfikacją (cyfrową lub analogową) zawartości obrazu [30]. Przetwarzanie obrazu, w kontekście funkcjonowania systemu wizyjnego, to zespół operacji prowadzących do zamiany rastrowego obrazu (pochodzącego z kamer(y) systemu wizyjnego) na dane wystarczające do dalszej swobodnej analizy informacji w nim zawartej [49]. Zazwyczaj przetwarzanie obrazu wiąże się z koniecznością zastosowania kilku-kilkunastu operacji na obrazie wejściowym, spośród których najbardziej typowymi są: wyostrzanie

[1] W niniejszej pracy słowo „obiekt" jest używane w dwóch rozłącznych znaczeniach, zależnie od kontekstu – jako „obiekt sterowania" (głównie w rozdziale 3.6) lub częściej jako element otoczenia (sceny), który można wyodrębnić na podstawie jego cech (np. przedmiot lub osoba, znajdujące się w polu widzenia systemu wizyjnego robota).

[2] Bardziej rozbudowaną definicję IU można znaleźć w [125].

krawędzi, modyfikacje histogramów, filtry, rotacje, skalowania. [49] [135] [151] [50] Operacje te należy koniecznie uwzględnić przy szacowaniu potrzebnej mocy obliczeniowej, szczególnie w przypadku, gdy przetwarzany obraz ma wysoką rozdzielczość lub gdy liczba rozpoznawalnych obiektów w scenie jest większa.

Rozpoznawanie niewielu prostych obiektów w scenie jest dziś często spotykane i to nie tylko w przemyśle (na przykład wyszukiwanie twarzy zaimplementowane w oprogramowaniu aparatów fotograficznych), jednak rozpoznawanie wielu różnych obiektów (przedmiotów, ludzi, etc.) w otoczeniu człowieka jest już dużym wyzwaniem. Rzadkością są systemy zdolne do wizualnego uczenia się nowych obiektów, takie jak eksperymentalne oprogramowanie japońskiego robota ASIMO opracowane w Niemczech przez zespół prof. E. Körner'a [64], pokazane w działaniu w materiale wideo [71].

Tematyka przetwarzania obrazu i rozumienia obrazu jest niezwykle obszerna i interesująca, dlatego też jest to zagadnienie nadzwyczaj intensywnie eksplorowane. Każdy cel może być osiągnięty na wiele sposobów, wiele aspektów procesu przetwarzania może podlegać optymalizacji. Istnieje wiele czasopism bezpośrednio poświęconych tej tematyce (m.in. „*Computer Vision and Image Understanding*", „*IEEE Transactions on Image Processing*", „*IEEE Transactions on Pattern Analysis and Machine Intelligence*", „*Image and Vision Computing*", „*Machine Vision and Applications*").

Niestety znacząca ilość prac nie przekłada się bezpośrednio na postęp w tematyce inteligencji maszyn. Większość z nich nie ma znaczącego wpływu na zagadnienia istniejące poza torem danych wizualnych systemu wizyjnego (mowa tu przede wszystkim o wnioskowaniu i o podejmowaniu działań).

W kontekście algorytmów wnioskowania największą popularność mają obecnie prace umożliwiające systemom wizyjnym rozpoznawanie predefiniowanych obiektów lub obiektów o predefiniowanych cechach.

Problematyka naukowo-badawcza dotycząca systemów wizyjnych korzystających z danych przetwarzanych w algorytmach wnioskowania jest w chwili obecnej eksplorowana ciągle jeszcze w niewystarczającym stopniu.[3] W niniejszej pracy, w rozdziale trzecim, opisane będzie dokładniej zagadnienie aktywnej wizji (*ang. Active Vision*, AV), które pojawiwszy się w roku 1987 [15], do dziś bywa źródłem inspiracji, jednak w swojej rdzennej

[3] Takich systemów wizyjnych, które dzięki sprzężeniu zwrotnemu korzystają z wiedzy gromadzonej przez system wnioskujący.

postaci[4] nie jest już zagadnieniem tak masywnie eksplorowanym[5]. Powszechnym zagadnieniem jest natomiast podejście odwrotne – sterowanie działaniami robota przy (komplementarnym) uwzględnieniu danych systemu wizyjnego. Zdecydowana większość obecnie produkowanych wielozadaniowych lub edukacyjnych robotów mobilnych wyposażana jest w sonary i/lub skanery LRF, mniej – w tradycyjne kamery lub kamery stereowizyjne, ponieważ o wiele łatwiej jest wykorzystać informacje z tych pierwszych. Bardzo interesujące połączenie obu technik[6] przedstawiono w artykule [9].

W chwili obecnej algorytmy systemów wizyjnych reprezentują o wiele wyższy poziom niż dawniej, a roboty niezaprzeczalnie lepiej radzą sobie w otoczeniu człowieka – przykładem może być tu postęp poczyniony między rokiem 2009 (robot skonstruowany przez firmę Panasonic i Tokyo University zmywa naczynia[7] [47]), a rokiem 2010 (robot HERB współtworzony przez Intel Labs Pittsburgh i Carnegie Mellon University zmywa naczynia[8] [75]). Postęp ten po części możliwy był dzięki ciągłemu rozwojowi algorytmów rozumienia wizji, a w szczególności rozróżniania obiektów w scenie.

Wspomniane sukcesy robotów i ich konstruktorów są jednak zaledwie małymi krokami na drodze ewolucji robotów inteligentnych. Ilość rozróżnianych obiektów, warunki akwizycji, a przede wszystkim możliwość (w zasadzie brak możliwości)[9] uczenia nowych obiektów są głównymi powodami sporej dozy sceptycyzmu wśród zwolenników kognitywistyki (względem istniejących koncepcji systemów wizyjnych), jak również nowych możliwości implementacyjnych w zakresie tworzenia maszyn autonomicznych.

[4] W *active vision* zazwyczaj zakładano (dla przyspieszenia obliczeń), że tylko określony fragment obrazu jest istotny, podczas gdy reszta obrazu może nie być przetwarzana. Dziś tak daleko idąca oszczędność jest stosowana tylko w specyficznych, przewidywalnych warunkach. Więcej o *active vision* w rozdziale drugim.

[5] Większość dziś stosowanych algorytmów rozumienia wizji zakłada możliwość przetwarzania całości pozyskanego obrazu, cały obraz poddawany jest operacjom filtrowania i analizy.

[6] System autonomicznej nawigacji klasyfikujący rodzaj terenu (obszary przejezdne, zieleń i obiekty nieprzejezdne, itd.) na podstawie danych z LRF oraz kamery autonomicznego mobilnego terenowego robota typu UAV (*ang. unmanned autonomous vehicle*). Ponieważ skrót UAV jest również używany dla *unmanned aerial vehicle*, pojazdy typu UAV określa się również nazwą UGV (*ang. unmanned ground vehicle*).

[7] Robot firmy Panasonic to system złożony z ramy przewożącej kamerę, zamontowanej nad blatem kuchennym zawierającym zlew oraz z ramienia robota przemysłowego. Film [75] pokazuje pełne stanowisko oraz najważniejsze umiejętności robota.

[8] Mobilny robot (zasilany przewodowo) na ruchomej platformie z ramieniem, wyposażony w algorytmy rozpoznawania obiektów (w tym rozróżniania spośród innych obiektów naczyń a nawet naczyń zawierających płyn). System wizyjny to kamera umieszczona na krótkim wysięgniku oraz skaner LRF zamontowany w pomysłowy sposób – na uchwycie obracającym go dla uzyskania dwuwymiarowej reprezentacji danych trzeciego wymiaru.

[9] Uczenie rozpoznawania nowych obiektów, aby miało sens, musi być dopełnione możliwością rozpoznania wyuczonych obiektów. Niestety ilość rozpoznawalnych obiektów nie jest wartością łatwo skalowalną – często dodanie kolejnego pojedynczego obiektu do bazy rozpoznawalnych obiektów jest problemem.

1.3. Przyszłość robotów, czyli o motywacji badawczej

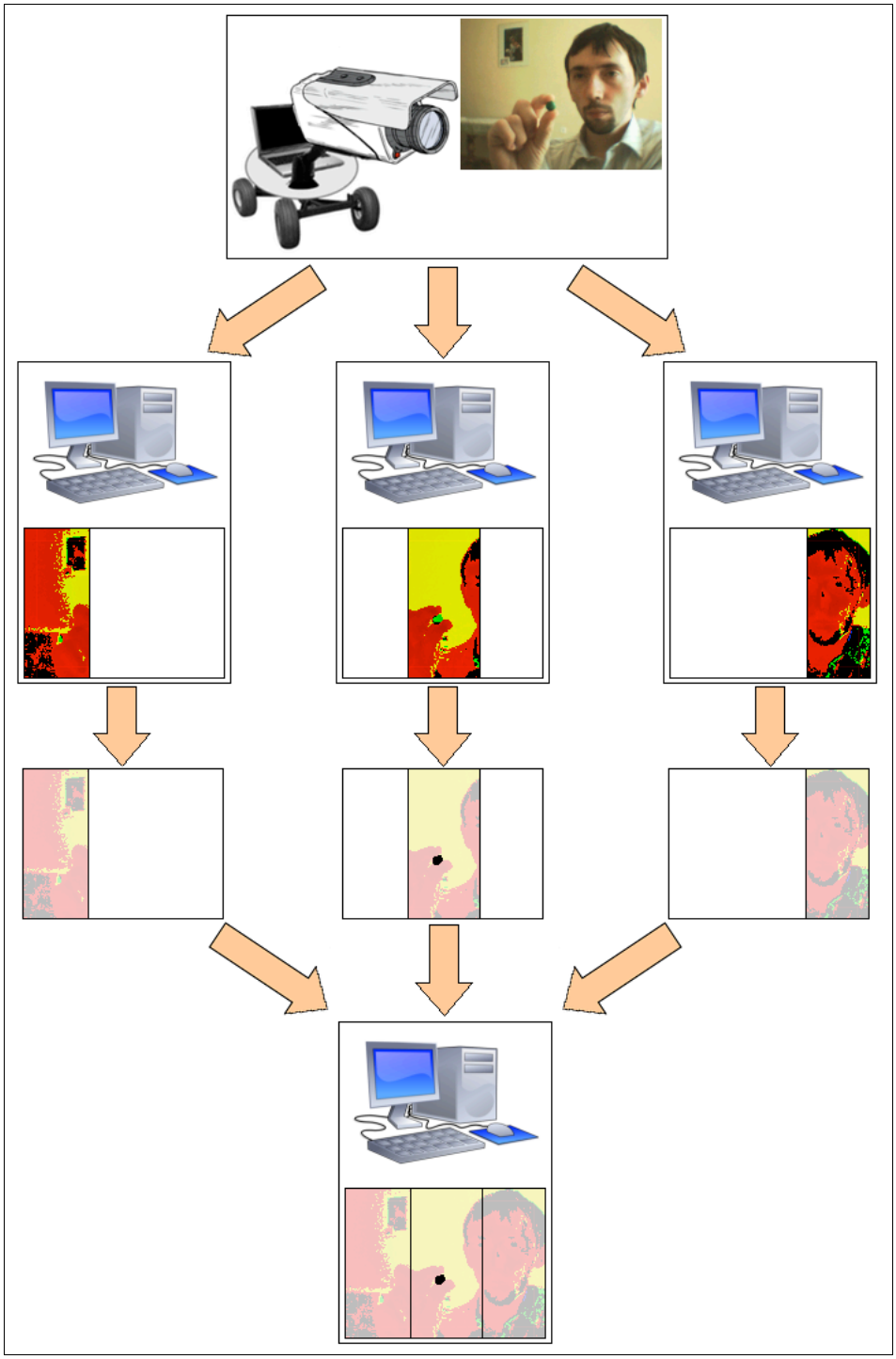
Postęp w tworzonych robotach humanoidalnych jest niezaprzeczalny. Znacznej poprawie uległa konstrukcja, napędy, dynamika, nawet energooszczędność. Niestety rozwój „inteligencji" zdaje się nadal tkwić w martwym punkcie. Może to być spowodowane powszechnie stosowanym, błędnym podejściem do zagadnienia sztucznej inteligencji – podejściem związanym z jej rdzenną definicją[10], podejściem mało elastycznym. Nie jest bowiem możliwe zaprogramowanie wszystkich możliwych scenariuszy, wyuczenie wszystkich istniejących obiektów. Dlatego tworzenie systemów wizyjnych robotów „inteligentnych" w ten sam sposób w jaki działają systemy wizyjne robotów przemysłowych mija się z celem – właśnie z powodu braku uniwersalności. Na bieżącym etapie rozwoju robotów „inteligentnych" należy położyć nacisk na wytworzenie algorytmu umożliwiającego robotom (nienadzorowane lub nadzorowane) samodzielne poznawanie świata ludzi – bez każdorazowej interwencji programistów. Jest to możliwe na drodze ciągłego doskonalenia algorytmów przetwarzania obrazu, rozumienia wizji i wnioskowania, niestety to podejście wiąże się z wzrostem zapotrzebowania systemu na moc obliczeniową. Istnieje jeszcze drugie rozwiązanie – wzorowane na biologicznym odpowiedniku systemu wizyjnego.

Zmysł wzroku jest zmysłem stosunkowo dobrze poznanym, mimo to stworzenie jego wiernie odwzorowanego sztucznego odpowiednika jest nadal problemem trudnym do rozwiązania. O ile akwizycja obrazu, procesy chemiczne komórek siatkówki oka, są dobrze znane, to szczegóły procesu rozumienia wizji nadal są nie do końca jasne. Wiadomo, że informacja trafia nerwem wzrokowym do mózgu, tam przetwarzana jest w rejonach kory, rozpoznawany jest ruch, podstawowe kształty, itd. Wiedza ta jest jednak nadal niewystarczająca do pełnego odtworzenia tego procesu w świecie maszyn [96][11]. Jednym z podstawowych problemów, uniemożliwiających dalsze badania tej metody, jest typowy problem algorytmów rozumienia wizji o rozbudowanej bazie obiektów – do rozróżniania wielu (często podobnych) obiektów potrzebna jest rozbudowana baza cech, duża moc obliczeniowa oraz możliwie najlepsza jakość obrazu źródłowego [29]. Problem rozmiaru bazy cech oraz niedostatku mocy obliczeniowej można łatwo rozwiązać stosując potężny dedykowany komputer wieloprocesorowy, lub nawet klaster złożony z wieloprocesorowych

[10] Zagadnienie „inteligencji robotów" omówiono szczegółowo w rozdziale Dodatek A.

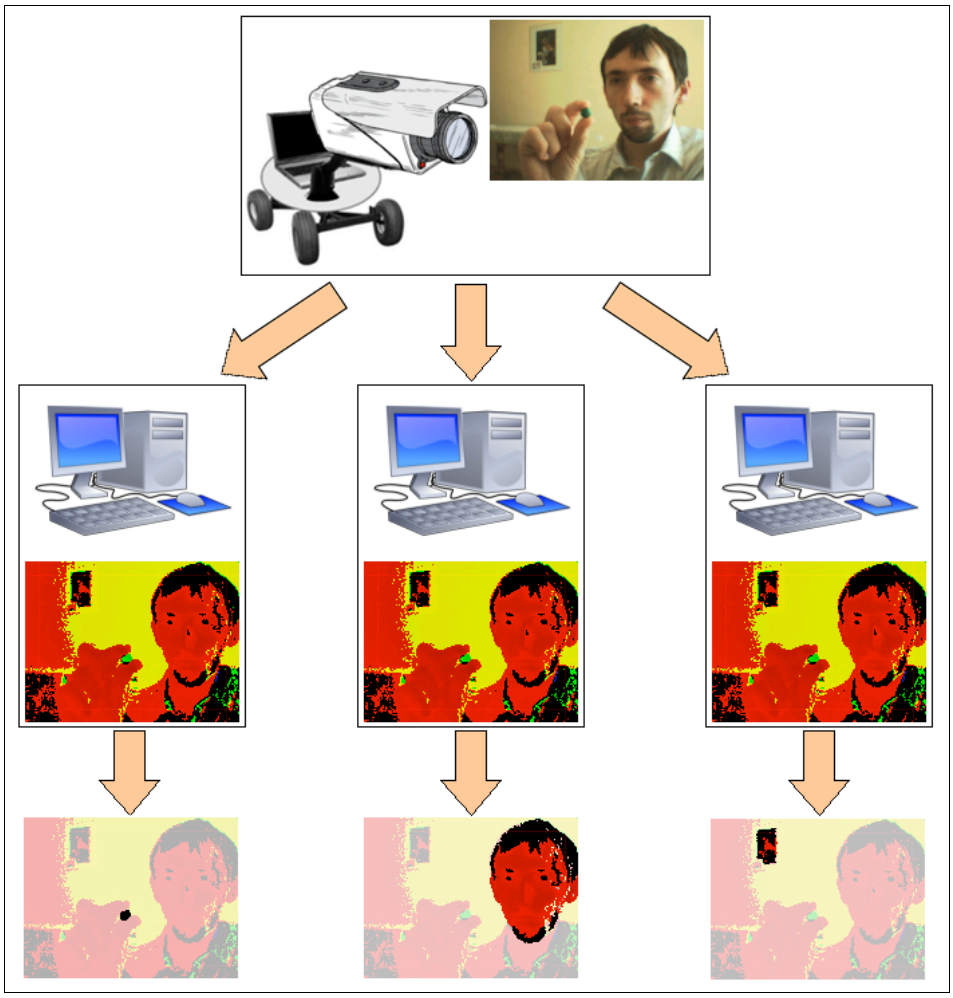
[11] W cytowanym artykule autorzy wspominają, że niektóre mechanizmy rozumienia wizji realizowane przez mózg nie zostały jeszcze do końca poznane.

węzłów. Takie rozwiązanie jest niestety niezwykle trudne do zrównoleglenia. Niektóre operacje etapu przetwarzania można podzielić na kilka procesów, niestety później i tak trzeba je połączyć przed algorytmem rozpoznawania (Rys. 1.2). Operacje związane z podziałem zadań, synchronizacją i przesyłaniem dodatkowo wydłużają czas przetwarzania. Czas wnioskowania (najbardziej zależny od liczby obiektów) pozostaje wówczas bez zmian. Istnieje ryzyko pominięcia obiektu istniejącego na pograniczu rejonów.



Rys. 1.2. Przykładowa metoda realizacji procesu rozumienia wizji w środowisku równoległym polegająca na dekompozycji obrazu na fragmenty (tzw. dekompozycja danych - [23])

Innym podejściem jest zrównoleglanie (a w zasadzie powielanie) całego procesu rozpoznawania wraz z wnioskowaniem we wszystkich węzłach, gdy każdy węzeł ma możliwość rozpoznawania innych obiektów (Rys. 1.3).

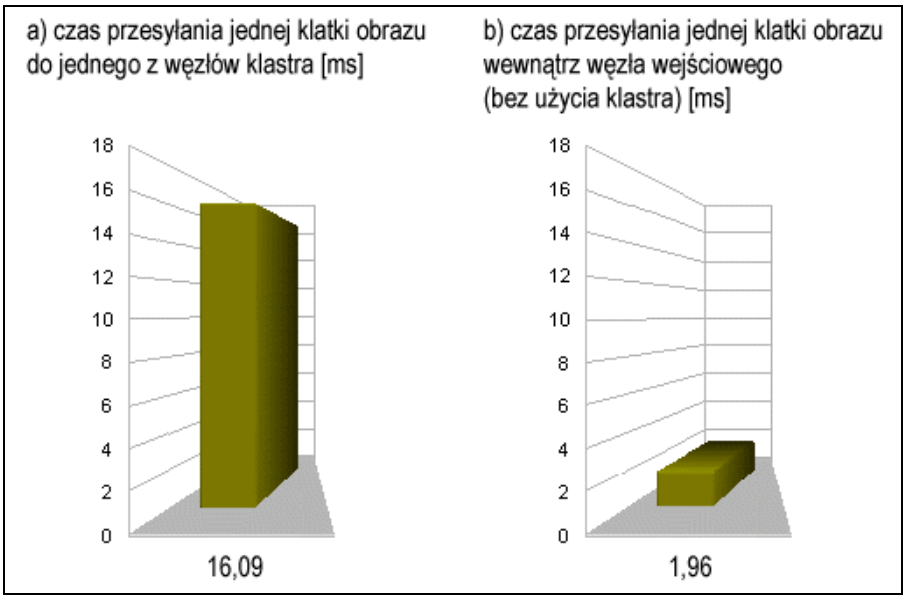


Rys. 1.3. Przykładowa metoda realizacji procesu rozumienia wizji w środowisku równoległym polegająca na dekompozycji obrazu na fragmenty (tzw. dekompozycja funkcjonalna - [23])

To rozwiązanie niestety również ma wady – sztywna, nadal ograniczona ilość rozpoznawanych obiektów (zależna od liczby węzłów), a przede wszystkim problem ograniczonej przepustowości utrudniający przesyłanie obrazu w klastrze. Mimo to od pewnego czasu zaczynają się pojawiać prace dotyczące algorytmów współbieżnych i rozproszonych dedykowanych zagadnieniom rozumienia wizji [117] [98], jednak prace te nie tylko zakładają nieograniczony dostęp do maszyny równoległej (w analizowanym w niniejszej pracy przypadku maszyna równoległa nie stanowi części agenta upostaciowionego), lecz również nie deklarują możliwości wykorzystania zaawansowanych metod analizy otoczenia. Według pracy [21], w historii systemów wizyjnych było kilka zespołów badawczych próbujących implementować algorytmy rozumienia wizji robotów

poruszających się w niezdeterminowanym otoczeniu na komputerach klastrowych, jednak w bibliografii zauważa się brak opracowań dotyczących wyników tych projektów. Natomiast autorom pracy [120][12] udało się dotrzeć do dwóch (niewymienionych w [21]) projektów, jednak również bez wyników. Przeprowadzone doświadczenia [122] wyjaśniły, że problem polegał na „zapychaniu" węzła mobilnego[13] oraz klastra zadaniami rozsyłania wizji (Rys. 1.4).

Aktualnie znane są nieliczne implementacje systemów wizyjnych używających klastra komputerowego w sposób przedstawiony na Rys. 1.2 – w sytuacjach, gdy na podstawie danych wizualnych możliwe jest wnioskowanie na poziomie indywidualnego węzła, na przykład w systemach wizyjnych kontroli jakości produktu na linii produkcyjnej [79][14].



Rys. 1.4. Wyniki pomiarów[15] czasu transmisji pojedynczej klatki obrazu: a) do drugiego węzła (fizycznie odrębnego komputera), b) wewnątrz węzła wejściowego (komputera z kamerą)

Niezwykle istotne i celowe jest wobec tego rozwiązanie problemu komunikacji i tym samym umożliwienie efektywnego wykorzystania klastra komputerowego jako mocy obliczeniowej autonomicznego robota mobilnego.

[12] Publikacja [120] (stanowiąca podsumowanie badań prowadzonych w ramach dwóch grantów) opisywała uzyskane przez autorów obiecujące wyniki, aczkolwiek należy do ich wartości podchodzić niezwykle ostrożnie – zostały otrzymane ponad dziesięć lat temu.

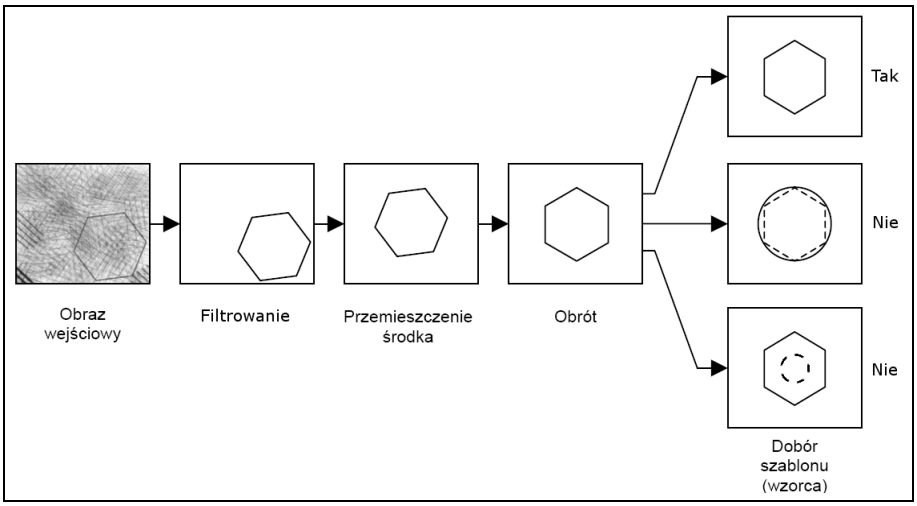
[13] Komputera będącego pokładowym komputerem robota mobilnego (embedded lub przewożony komputer przenośny) i równocześnie będącego węzłem klastra

[14] Za przykład może posłużyć opisana w literaturze [79] aplikacja rozproszona sprawdzająca poprawność wykonania paneli LCD i płyt PCB.

[15] Dla obrazu wejściowego o rozdzielczości 320x240 pikseli i głębi 24bpp z kamery USB, na komputerze stacjonarnym PC, z dwurdzeniowym procesorem Intel Core2Duo 2.33GHz, systemem operacyjnym Fedora3.

1.4. Cel, teza i zakres pracy

Zaproponowane w pracy rozwiązanie wymaga zmiany podejścia do systemu wizyjnego, który dziś najczęściej rozumiany jest jako system o jednokierunkowym przepływie danych (Rys. 1.5) [25] – od urządzenia dokonującego akwizycji informacji źródłowej, aż po algorytm wnioskowania.



Rys. 1.5. Sekwencja filtrów i przekształceń prostego klasycznego systemu wizyjnego

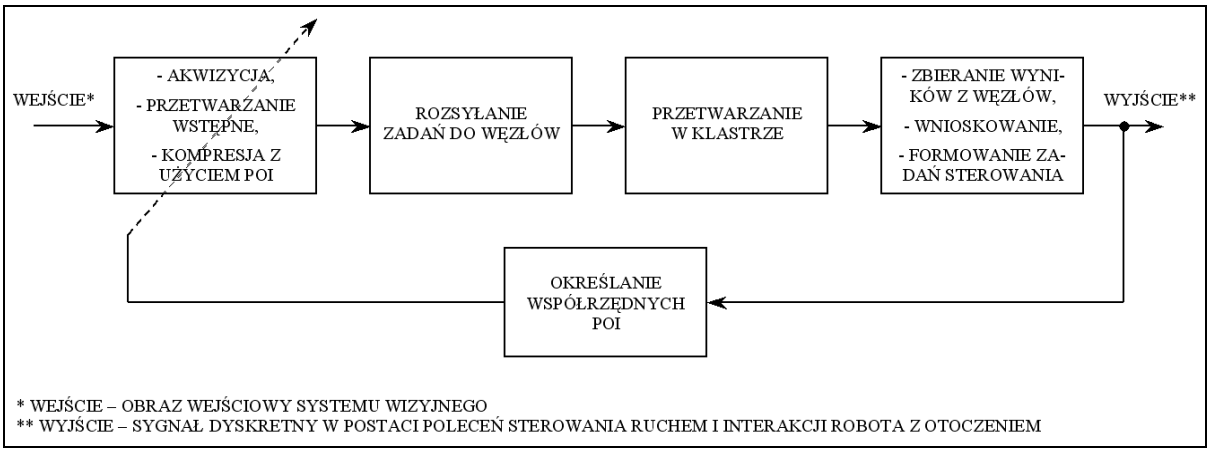
Proponowany w pracy nowy system wizyjny nie jest sekwencją operacji, lecz może być rozpatrywany jako układ regulacji, w którym właśnie działanie wprowadzonego sprzężenia zwrotnego odgrywa kluczową rolę na wszystkich etapach – od akwizycji do wnioskowania. Pomysł wzorowany jest na biologicznym odpowiedniku systemu wizyjnego, w którym nie rozróżnia się procesów filtrowania, translacji, obrotu obiektu, ani porównywania[16] z bazą znanych obiektów. Nierozłącznym elementem działania ludzkiego wzroku są ruchy gałki ocznej, w tym tzw. ruchy sakkadowe [13], jednak w drodze uproszczenia w większości sztucznych systemów wizyjnych ruch gałki ocznej jest ignorowany[17].

[16] Samo „porównywanie" nadal istnieje w niektórych teoriach dotyczących sposobu działania pamięci ludzkiego mózgu, ale coraz częściej naukowcy się wycofują z tych teorii, przechodząc do obozu zwolenników tzw. „pamięci rozproszonej" lub „pamięci holograficznej".

[17] W niektórych opracowaniach i implementacjach opartych o technikę przetwarzania „*Active Vision*" (AV) pojawia się co prawda podejście które dziś określa się „*Region-of-Interest*", jednak w AV pozostała część obrazu jest ignorowana, co znów stanowi kolejne uproszczenie (i utrudnienie jednocześnie).

W niniejszej pracy zaproponowano nowe, kompleksowe rozwiązanie (ideę, algorytm, metody oraz przykładową implementację) aktywnego systemu wizyjnego.

Schemat procesu regulacji, która czyni ten system wizyjny „aktywnym", w skrócie przedstawiono na Rys. 1.6. Bardziej szczegółowo został on opisany w rozdziale 3.6.



Rys. 1.6. Uproszczony schemat blokowy proponowanego aktywnego systemu wizyjnego

Problem badawczy, teza

Schemat przedstawiony na rysunku Rys. 1.6 będzie realizowalny, jeżeli w fazie projektowania nie zostaną pominięte następujące – uściślone w niniejszej pracy – wskazówki:

- fragmenty obrazu poddawane analizie muszą być przesyłane z maksymalną dostępną jakością,
- fragmenty obrazu niepoddawane analizie powinny być poddane kompresji, aby zminimalizować czas transmisji między węzłami klastra.

Dzisiejsze systemy wizyjne zazwyczaj nie dzielą obrazu na fragmenty, nie zezwalają również na używanie kompresji w torze danych. Proponowany system wizyjny jest jednak odmienny. Kompresja, a nawet stratna kompresja, należy do jednej z jego największych zalet – właśnie dzięki niej możliwe jest rozsyłanie klatek obrazu do węzłów klastra.

W wyniku wstępnie przeprowadzonych analiz, symulacji i testowych implementacji fragmentarycznych udało się naszkicować kierunek badań szczegółowych, umożliwiających wyznaczenie skutecznych i powtarzalnych schematów postępowania zmierzających do umożliwienia uruchomienia procesów wnioskujących systemu wizyjnego, zlokalizowanych w węzłach klastra komputerowego. Przeprowadzone czynności przygotowawcze umożliwiły sprecyzowanie tezy:

Dzięki odpowiedniemu przygotowaniu danych wizualnych oraz właściwemu doborowi topologii klastra komputerowego możliwe jest wykorzystanie go jako efektywnego narzędzia obliczeniowego systemu wizyjnego robota autonomicznego.

W celu udowodnienia prawdziwości powyższej tezy niezbędne było rozwiązanie szeregu problemów badawczych, spośród których najbardziej istotnymi są:

- zaproponowanie struktury i zasady działania systemu wizyjnego (rozdział 1.1 i 3.2.2),
- zaproponowanie nowej metody reprezentacji/konwersji danych wejściowych (obrazu) dla potrzeb projektowanego systemu wizyjnego (rozdział 3.2.2),
- zaproponowanie sposobu uwzględniania wyników działania algorytmu wnioskującego (w węzłach klastra) podczas przygotowywania kolejnej klatki obrazu (rozdział 3.6),
- przeprowadzenie analizy czasowej zaproponowanego rozwiązania (rozdział 4.3),
- przeprowadzenie weryfikacji implementacyjnej potwierdzającej możliwość zastosowania wyników pracy w systemach wizyjnych (rozdział 4.4 i 4.5).

Po wstępnym studium wykonalności wyznaczono cel główny pracy oraz cele pomocnicze:

Cel główny:

Studium wykonalności, opracowanie i weryfikacja implementacyjna systemu wizyjnego robota mobilnego korzystającego z klastra komputerowego, oraz porównanie efektywności jego działania z systemem nie używającym klastra.

Udowodnienie tezy oraz realizacja celu głównego pracy wymagały wykonania następujących szczegółowych zadań implementacyjnych (cele pomocnicze):

- opracowanie przykładowej implementacji systemu wizyjnego robota mobilnego,

- przeniesienie części przetwarzania do węzłów klastra komputerowego,
- implementacja opracowanej metody reprezentacji danych wizualnych,
- weryfikacja implementacyjna przeprowadzonej analizy czasowej opracowanego systemu wizyjnego w odniesieniu do systemu nie używającego klastra.

Praktyczna realizacja aktywnego systemu wizyjnego w środowisku rozproszonym przeznaczonego do interakcji człowiek-maszyna jest możliwa tylko przy skróceniu czasu komunikacji. Implikuje to pewien przyczynowo-skutkowy ciąg zależności, a więc i jednocześnie przebieg badań.

Badania i eksperymenty wykonywano w kolejności umożliwiającej szybkie i trafne określenie wykonalności postawionego zadania badawczego, jednak w rozprawie wyniki badań zostały przedstawione w sposób usystematyzowany, jako wkomponowane w działający system wizyjny dowodzący celowości podejmowanych działań.

Charakterystyka rozwiązania

W niniejszej dysertacji przedstawiono innowacyjny aktywny system wizyjny korzystający bezpośrednio z opracowanych autorskich rozwiązań, współdziałający z algorytmem rozproszonym uruchomionym w środowisku klastrowym. Dotychczasowe próby realizacji algorytmów systemów wizyjnych robotów mobilnych w środowiskach klastrowych [21] nie są dostatecznie zadowalające[18] ze względu na przyjmowane przez projektantów wymagania/ograniczenia:

- klatki strumienia wideo zawierają obraz o najlepszej (ale uzasadnionej technologicznie) dostępnej jakości (m.in. rozdzielczości, głębi koloru)
- klatki strumienia wideo jeżeli muszą być kompresowane, są kompresowane przy użyciu kompresji bezstratnej

Wymagania te są w zupełności zasadne, inaczej nie byłoby możliwe skuteczne stosowanie procedur bezkontekstowego przetwarzania obrazu. Niestety założenia te całkowicie uniemożliwiają osiągnięcie korzyści z prób zrównoleglania któregokolwiek z etapów algorytmów przetwarzania czy wnioskowania[19] rozbudowanych systemów wizyjnych robotów mobilnych.

[18] Nie dotyczy zastosowań przemysłowych i innych o małej bazie rozpoznawanych obiektów i o powtarzalnym i przewidywalnym otoczeniu.

[19] Mowa tu o algorytmach wnioskowania będących częścią podsystemu wizyjnego. Wnioskowanie podsystemu decyzyjnego lub podsystemu sztucznej inteligencji nie jest tutaj rozważane, może podlegać innym warunkom.

Rozwiązanie przedstawione w pracy bazuje na zupełnie odmiennym podejściu do danych wejściowych systemu wizyjnego. W proponowanym rozwiązaniu klatki obrazu mogą być poddawane kompresji, nawet kompresji stratnej, ponieważ sposób analizowania danych jest zdecydowanie odmienny od klasycznego.

Podstawową różnicą pomiędzy klasycznymi systemami wizyjnymi a proponowanym w pracy rozwiązaniem jest występujące w nim sprzężenie zwrotne (Rys. 1.1). W klasycznych systemach wizyjnych wynik działania algorytmów rozpoznawania, wnioskowania i decyzyjnych nie miał wpływu na proces pozyskiwania i przetwarzania obrazu, jedynie na ruch mechanicznych komponentów robota (w tym niekiedy uchwytu kamery[20]), ale nie na działanie samego systemu wizyjnego[21]. W proponowanym systemie wnioskowanie stanowi nie tylko integralną część procesu przetwarzania a nawet pozyskiwania obrazu, ale jest wręcz spoiwem procesów akwizycji, przetwarzania, wnioskowania a nawet algorytmu decyzyjnego (Rys. 1.6).

Sedno proponowanego rozwiązania polega na przesyłaniu i przetwarzaniu klatek o zróżnicowanej powierzchniowo jakości obrazu. Rozwiązanie zaczerpnięte z mechanizmu funkcjonowania ludzkiego wzroku zakłada cykliczne definiowanie współrzędnych punktu zainteresowania, którego najbliższe otoczenie będzie poddawane analizie i precyzyjnemu przetwarzaniu. Jakość w analizowanym fragmencie powinna być możliwie najlepsza, ale w odległych rejonach przechwytywanego obrazu nie jest parametrem krytycznym.

Wnioskowanie na podstawie tak uzyskanych informacji o otoczeniu może się wydawać niepotrzebną komplikacją w postaci potrzeby ponownego składania wirtualnej reprezentacji rejestrowanego obrazu, jednak proponowane rozwiązanie zakłada również radykalne zmiany na poziomie algorytmów wnioskowania. Rozpoznawanie obiektów nie jest przeprowadzane w drodze porównywania krawędzi[22] przetworzonego rastra z bazą kształtów (tak jak przedstawiono to na rysunku Rys. 1.5), lecz w oparciu o (opisane szczegółowo w rozdziale 3.3) sieci HTM.

[20] Ruch uchwytu kamery wpływa na wejście systemu wizyjnego pośrednio, co może być traktowane jako zabieg zamierzony lub jako zakłócenie.

[21] Wyjątkiem jest tu część rozwiązań implementujących *active vision*, w których algorytm po wstępnej analizie całego obrazu definiował region zainteresowania do analizy precyzyjnej, np. rozpoznawanie znaków alfanumerycznych z tablic rejestracyjnych przejeżdżających samochodów.

[22] Podana metoda stanowi najczęściej stosowaną w systemach wizyjnych, została tu przytoczona jako reprezentatywna przeciwstawność ogółu klasycznych (obecnie uznanych i stosowanych) metod przetwarzania i wnioskowania do proponowanej metody.

Reasumując, algorytm przedstawionego w pracy aktywnego systemu wizyjnego robota mobilnego, używającego klastra jako narzędzia obliczeniowego, zawiera m.in. sekwencję następujących zadań:

1. akwizycję danych wizualnych (pojedynczej klatki strumienia wideo)
2. konwersję klatki transformatą DWT w sposób zbliżony do opisanego w standardzie JPEG2000 [48] [138], ale używający zmiennej przestrzennie wartości parametru progu (*ang. threshold*), zależnej od odległości od podanego punktu POI (konwersja jest operacją stratną dla niezerowych wartości progu)
3. kompresję bezstratną lub kompresję stratną o modyfikowalnym przestrzennie współczynniku kompresji
4. w węzłach klastra (asynchronicznie):
   - pobranie klatki obrazu przez węzeł klastra komputerowego (podział gruboziarnisty funkcjonalny)
   - dekompresję oraz transformatę odwrotną – IDWT
   - podanie otrzymanego fragmentu obrazu na wejścia warstwy sensorycznej lokalnej sieci HTM
   - cykl działania sieci HTM (formowanie/aktywacja tzw. niezmiennych reprezentacji[23] w wyższych warstwach sieci, rozpoznawanie obiektu na podstawie niepełnych danych, dodatkowo określenie kolejnego punktu zainteresowania) lub dowolnej innej implementacji pamięci skojarzeniowej; lokalna sieć HTM może być siecią rozproszoną, również korzystającą z tego samego klastra
   - sygnalizację wykrycia w obrazie danego obiektu, zgłoszenie wykrycia algorytmowi decyzyjnemu lub/i przesłanie propozycji nowych współrzędnych punktu zainteresowania do węzła wejściowego (robota mobilnego)
5. zebranie propozycji nowych współrzędnych punktu zainteresowania, uwzględnienie ich przy akwizycji i konwersji kolejnej klatki obrazu wejściowego

Dzięki zaproponowanemu sposobowi zastosowania opisanej reprezentacji obrazu możliwe jest przesłanie każdej klatki obrazu do kilku-kilkunastu węzłów klastra. Dzięki temu, że algorytm działania sieci HTM łatwo poddaje się zrównolegleniu, cały system dodatkowo znacząco zyskuje na skalowalności.

[23] „niezmienne reprezentacje" pojawiają się też czasem w literaturze jako „prototypy pojęć".

1.5. Struktura pracy

Rozprawę podzielono na pięć rozdziałów. Pierwsze dwa rozdziały mają charakter wprowadzenia, obejmując nie tylko cel i zakres pracy, ale również motywację i umiejscowienie pracy wśród aktualnych trendów i badań w rozpatrywanym obszarze.

Rozdział trzeci zawiera opis zarówno aktualnych jak i zaproponowanych oryginalnych rozwiązań i koncepcji teoretycznych, umiejscowiony ściśle w kontekście systemu wizyjnego jako fragmentu złożonego układu regulacji. W rozdziale tym podsumowano jedynie wybrane istniejące sposoby postępowania z danymi systemów wizyjnych robotów oraz wyjaśniono sposób rozwiązania postawionego zadania badawczego.

Rozdział czwarty zawiera prezentację sposobu organizacji przeprowadzonych eksperymentów badawczych (implementacyjnych) w postaci: opisu środowiska eksperymentów, opisu i interpretacji przeprowadzonych eksperymentów oraz oceny wyników opartą m.in. o przeprowadzoną analizę czasową. Aby ułatwić lekturę, wszelkie szczegóły implementacyjne przeniesiono do Dodatku Dodatek B niniejszej pracy. Informacje tam zawarte są również ograniczone – do umożliwiających odtworzenie przebiegu i wyników przeprowadzonych eksperymentów.

Ostatni z rozdziałów podsumowuje pracę oraz wskazuje dalsze możliwości rozwijania zaproponowanego w pracy rozwiązania.

Do pracy dołączono nośnik CD zawierający kod źródłowy projektów prób implementacyjnych, wraz z dodatkową dokumentacją w postaci plików pdf, zrzutów ekranu oraz plików wideo. Zawartość nośnika opisano szczegółowo w Dodatku Dodatek C.

2. Analiza obrazu w dzisiejszych systemach wizyjnych

W aktualnie tworzonych/wykorzystywanych systemach wizyjnych przetwarzanie wstępne obrazu (całokształt operacji, które muszą być wykonane, aby możliwe było rozpoczęcie wnioskowania na podstawie danych wizualnych) nie jest rozważane jako obiekt sterowania, lecz jako prosta sekwencja przekształceń. Brak sprzężeń między poszczególnymi etapami przetwarzania wstępnego wynika między innymi z braku potrzeby modyfikacji/uściślania którekolwiek z etapów, ponieważ wszystkie operacje przeprowadzane są na kompletnych danych (całym obrazie i w najwyższej[24] jakości).

Zbiór operacji przetwarzających wstępnie obraz (na postać stosowną dla wejść algorytmu wnioskującego) według literatury [148] [135] może być podzielony na dwie grupy:

- operacje bezkontekstowe wstępnego przetwarzania obrazu,
- operacje kontekstowe filtracji obrazu.

Wprowadzenie takiego podziału, choć zupełnie zasadne w kontekście cytowanej publikacji[25] , w przypadku proponowanego w niniejszej pracy systemu wizyjnego (podobnie jak w większości nowopowstających systemów wizyjnych) traci znaczenie implementacyjne. Natomiast niezwykle sugestywne jest samo nazewnictwo użyte w tym rozróżnieniu. Istnieje bowiem potrzeba wytyczenia jasnej i czytelnej granicy pomiędzy operacjami wykonywanymi bez względu na zawartość[26] danych, a operacjami przeprowadzanymi w sposób zależny (i ściśle powiązany) z zawartością i rodzajem zawartości danych. Granica ta jest niezmiernie istotna przy określaniu, które spośród operacji przetwarzania obrazu powinny być wykonane przed rozesłaniem danych do węzłów klastra, a które powinny być wykonywane później, w węzłach klastra. Tak określona granica powinna się wkomponowywać w topologię proponowanego w pracy rozproszonego (klastrowego) systemu wizyjnego.

[24] Najwyższej nie tylko w kontekście rozdzielczości ale najwyższej uzasadnionej ekonomicznie.

[25] operacji strumieniowego przetwarzania obrazów realizowanych przez dedykowane procesory sprzętowe; publikacja z 2003 roku

[26] obecność w scenie przedmiotów/osób/detali wymagających zainteresowania/analizy

Aby możliwe było użycie do dalszej analizy obrazu pozyskanego z kamery lub dowolnego innego urządzenia przechwytywania (w warunkach innych niż laboratoryjne), należy przygotować pozyskane dane poprzez wykonanie na nich serii operacji zapewniających powtarzalne parametry jakościowe przetwarzanego obrazu.

Dodatkowo, przed rozesłaniem obrazu do węzłów klastra, warto przeprowadzić takie operacje przetwarzania, które mogą zaowocować znacznym zmniejszeniem ilości danych. Ze względu na wprowadzony podział funkcjonalny zadań realizowanych w węzłach klastra, można wskazać węzły, które nie potrzebują kompletnych informacji, a wystarczy odpowiednio spreparowany obraz monochromatyczny.

W proponowanym w pracy systemie wizyjnym niektóre z operacji wymienionych poniżej niejednokrotnie mogą być de facto realizowane już w węzłach klastra.[27]

Poniżej wymieniono podstawowe operacje wstępnego przetwarzania danych wizualnych (*ang. Image Processing*), stosowane w dzisiejszych systemach wizyjnych.

Konwersja formatu danych wejściowych

Konwersja formatu danych wejściowych zwykle jest pierwszą operacją wykonywaną na pozyskanym obrazie (zaraz po samej akwizycji). Tego etapu przetwarzania nie można zazwyczaj wyeliminować, ponieważ sprzętowa realizacja konwersji fal widzialnych na postać cyfrową wiąże się ze zmianą postaci danych. Projektant systemu wizyjnego może jednak mieć pewien wpływ na ten proces – niektóre kamery pozwalają wybrać z poziomu sterownika kamery format danych wyjściowych. W niektórych sytuacjach konwersję do pożądanego formatu graficznego trzeba wykonać programowo.

[27] Przykładowo: dodatkowa filtracja obrazu kolorowego – jeżeli do węzła przesyłany jest kompletny kolorowy obraz i jego przetwarzanie w węźle wejściowym nie spowoduje (znaczącego) zmniejszenia jego rozmiarów, to przetwarzanie to można zaimplementować w węzłach wnioskujących (zamiast wejściowego).

Normalizacja (korekcja jasności)

Normalizacja polega na zmianie przedziału wartości pikseli. Używana jest przy przetwarzaniu wstępnym do zgrubnej korekty poziomu jasności obrazu oraz po operacjach przekształceń arytmetycznych celem zabezpieczenia zakresu. Zwykle wartości pikseli w obrazie są skalowane do nowego zakresu przy użyciu zależności liniowej.

Korekcja gamma (korekcja kontrastu)

Korekcja gamma umożliwia korekcję kontrastu, poprzez przeliczenie wartości każdego piksela podnosząc jego wartość do potęgi będącej parametrem przekształcenia. W połączeniu z normalizacją stanowi niezastąpione narzędzie umożliwiające „wydobycie" obrazu zdatnego do dalszej obróbki z obrazów pobranych w niesprzyjających okolicznościach (m.in. mgła, zachmurzenie, niedoświetlenie, prześwietlenie).

Skalowanie według funkcji i wycinanie

Skalowanie według funkcji polega na przeprowadzeniu operacji korekcji przy zastosowaniu innych, specyficznych, odwzorowań. Wycinanie polega na usunięciu informacji o danym pikselu dla określonego zakresu wartości. Celem tych operacji jest uwypuklenie specyficznych cech obrazu, na przykład poszukiwanie w obrazie detalu o ściśle określonej barwie.

Modyfikacja histogramu

Operacje przeprowadzane na obrazie z uwzględnieniem jego histogramu należą do najbardziej użytecznych operacji przekształceń punktowych obrazu, ponieważ umożliwiają racjonalizację wykorzystania zakresu wartości jakie mogą być przypisane pikselom. Poprzez prostą operację wyrównania histogramu można znacznie ułatwić implementację i zmniejszyć stopień złożoności kolejnych algorytmów przetwarzania, choćby poprzez wykorzystanie całego zakresu wartości do poszukiwania cech lub detali w scenie. Korzyści wynikające z zastosowania operacji wyrównania histogramu ciekawie zilustrowano w [135].

Tworzenie postaci monochromatycznej

Postać monochromatyczna obrazu tworzona jest poprzez wygenerowanie takiej dwuwymiarowej macierzy pikseli, w której każdy piksel reprezentowany jest jedną wartością, odczytywaną najczęściej w skali szarości. W przypadku tworzenia obrazu monochromatycznego na podstawie obrazu kolorowego dodatkowo istotny jest sposób zamiany wartości kolorów składowych na wartość docelową. Często spotykanym problemem jest (oczywista) niejednoznaczność w postaci braku możliwości rozróżnienia niektórych kolorów po przekształceniu. Powstało wiele prac opisujących dedykowane algorytmy umożliwiające taką konwersję obrazu kolorowego do monochromatycznego, aby rozróżnienie najważniejszych dla człowieka kolorów było nadal możliwe, m.in. [128].

Postaci monochromatycznej czasami używa się w celu przedstawienia występowania danej cechy na powierzchni obrazu. Za przykład może tu posłużyć dwuwymiarowa macierz głębokości, uzyskiwana jako efekt działania algorytmów stereowizji – każdy piksel definiowany jest za pomocą jednej wartości, która określa odległość pewnego punktu w przestrzeni od kamery. Innym przykładem może być obraz przygotowany do przetwarzania w kontekście jednej określonej cechy, na przykład natężenie określonej barwy lub spreparowanie obrazu różnicowego zawierającego informację o ruchu lub przemieszczeniu pomiędzy dwoma klatkami obrazu.

Binaryzacja

Pod pojęciem binaryzacji kryje się wiele metod konwersji obrazu wejściowego do postaci, w której każdy piksel jest reprezentowany za pomocą jednego bitu. Oprócz binaryzacji z górnym progiem znane są jeszcze m.in.: binaryzacja z podwójnym ograniczeniem, binaryzacja z histerezą i binaryzacja wielokryterialna.

Filtracja

Operacje filtracji modyfikują wartość danego piksela biorąc pod uwagę jego otoczenie. Potrafią dzięki temu zmienić informacje opisujące obraz w sposób umożliwiający wyeliminowanie lub uwypuklenie pewnych jego cech. Operacji filtracji używa się zazwyczaj w celu [135]:

- usunięcia (lub zmniejszenia) szumu w obrazie,

- uwypuklenia cech zgodnych ze znanym wzorcem,
- usunięcia specyficznych wad obrazu (zagniecenia zdjęć, zarysowania na kliszy),
- wyostrzenie krawędzi,
- rekonstrukcja fragmentów obrazu.

Do najbardziej popularnych rodzajów operacji filtracji należą:

- konwolucje (stosowane w celu eliminacji szumów metodą zbliżoną do średnich lokalnych),
- filtry dolnoprzepustowe (implementujące konwolucję dyskretną w postaci macierzy konwolucji; działają znacznie szybciej niż implementacje konwolucji ciągłej; dają niepożądany efekt rozmycia),
- filtry górnoprzepustowe – gradienty [135]: gradient Robertsa, maska Prewitta, maska Sobela, m.in. (wykrywają szybkie zmiany poziomu jasności – krawędzie, detale, pojedyncze piksele),
- filtry górnoprzepustowe wykrywające krawędzie – laplasjany (umożliwiają wykrywanie krawędzi bez typowej dla metod konwolucyjnych kierunkowości),
- filtry górnoprzepustowe wykrywające narożniki,
- filtry nieliniowe (kombinowane) [135] (poprzez zastosowanie nieliniowej kombinacji dwóch filtrów liniowych dają lepsze rezultaty niż laplasjany),
- filtry logiczne (umożliwiają zastosowanie maski z warunkami wynikającymi z określonych przez programistę reguł/operacji logicznych),
- filtry medianowe (wartości mnożników w masce uzależniają od mediany wartości lokalnych pikseli pod maską; usuwają szumy nie usuwając cennych szczegółów obrazu; stosowane powszechnie w analizie obrazów medycznych),
- filtry adaptacyjne (umożliwiają przełączanie pomiędzy technikami filtracji lub ich łączenie w zależności od lokalnie określonego charakteru obrazu),
- filtracja w dziedzinie Fouriera (filtracja przeprowadzana na danych wynikowych Dyskretnej Transformaty Fouriera; szczegółowo opisane w [135]),
- filtracja transformatą falkową (redukcja wysokoczęstotliwościowego szumu lub redukcja trendu [63]).

Operacje morfologiczne

Przekształcenia morfologiczne uznawane są za jedne z najważniejszych w przetwarzaniu obrazów, ponieważ umiejętnie połączone i zaimplementowane stanowią podstawę dla dalszej analizy kształtów, rozmiarów i wzajemnego położenia elementów w scenie. Operacje morfologiczne najłatwiej stosować na spreparowanym do tego celu obrazie binarnym, wyróżniającym analizowane elementy obrazu od pozostałych i tła. Do najbardziej popularnych prostych przekształceń morfologicznych należą:

- erozja i dylatacja (umożliwiają generalizację obrazu i wyodrębnienie „masywnych" elementów obrazu),

- otwarcie i zamknięcie, automediana (umożliwiają usunięcie części zakłóceń utrudniających/uniemożliwiających wyodrębnienie granic danego elementu),
- morfologiczne metody geodezyjne [50] (umożliwiają przeprowadzanie operacji morfologicznych warunkowych, na podstawie dodatkowego obrazu wzorcowego),
- szkieletyzacja, szkieletyzacja z pocienianiem, homotopia (umożliwia zachowanie właściwości topologicznych danego elementu obrazu, przy znacznym zmniejszeniu liczby pikseli niezbędnych do jego opisania; w połączeniu z algorytmem obcinania gałęzi generuje bardzo intuicyjną postać reprezentacji topologii danego elementu).

Oprócz wymienionych powyżej operacji wstępnego przetwarzania obrazu, celem rozpoczęcia wnioskowania na podstawie przesłanych danych, aktualne systemy wizyjne stosują rozmaite strategie analizy obrazu, prowadzące bezpośrednio do wnioskowania lub współgrające nierozerwalnie z algorytmami wnioskowania.

Poniżej wymieniono kilka najbardziej popularnych metod z zakresu analizy obrazu („rozumienia wizji", *ang. Image Understanding*). [50] [118] [135] [151]

Segmentacja

Segmentacja jest to taka operacja, która powoduje wyodrębnienie ze sceny pojedynczych elementów celem ich późniejszego rozpoznawania. Istnieje kilka technik segmentacji:

- segmentacja obszarowa (przez podział lub rozrost obszaru),
- segmentacja konturowa (z udziałem alg. Wykrywania krawędzi),
- inne

Implementacji segmentacji zawsze towarzyszy konieczność rozwiązania problemu segmentacji obiektów stykających się. Dodatkowym problemem – w przypadku obrazów innych niż binarne – jest brak możliwości stosowania „wprost" wyżej wymienionych metod segmentacji.

Zwykle procesowi segmentacji towarzyszy indeksacja.

Indeksacja

Celem indeksacji jest przyporządkowanie określonym (w drodze segmentacji) elementom obrazu indywidualnych etykiet oraz oznaczenie nią wszystkich pikseli tych obiektów.

Cechy obrazu

Aby możliwe było rozpoznanie elementów w scenie, oprócz segmentacji i indeksacji niezbędne jest określenie ich podstawowych parametrów. Rodzaj parametrów brany pod uwagę jest ściśle zależny od specyfiki zadania rozpoznawania i może być zarówno zadaniem łatwym[28] jak i niezwykle złożonym. W wielu systemach wizyjnych, aby możliwe było przeprowadzenie wspomnianego rozróżnienia, niezbędne jest przeprowadzenie tzw. etapu pomiarów. Mierzone mogą być dowolne parametry (głównie morfologiczne/geometryczne) pozwalające na rozróżnienie i późniejsze przyporządkowanie danego elementu do jednej z grup. Bardziej złożone cechy obrazu, jak m.in. cechy szkieletowe [50], liczba Eulera [151], momenty bezwładności[29] i współczynniki kształtu [135], w niektórych zastosowaniach pozwalają na rozróżnianie elementów nawet bez udziału miar morfologicznych.

Miary morfologiczne

Oprócz cech obrazu, w celu rozróżnienia jego elementów, stosuje się również tzw. miary morfologiczne w znacznym stopniu ułatwiające opisywanie kształtów w scenie. Wśród nich najważniejszymi miarami są granulometria („odsiewanie" elementów o różnej powierzchni) i kowariancja morfologiczna (określanie odległości pomiędzy powtarzalnymi, położonymi w regularnych odstępach, elementami obrazu). Do bardziej zaawansowanych metod rozróżniania morfologicznego obiektów należą ekstraktory klas, opisane m.in. w [50].

Dodatkowe cechy elementów obrazu

W dzisiejszych systemach wizyjnych implementuje się kilka niezależnych sposobów opisywania cech elementów obrazu. Oprócz cech i miar opisanych wcześniej, dodatkowo

[28] np. rozróżnienie śruby od nakrętki na taśmie montażowej

[29] momenty bezwładności pierwszego rzędu określają środek ciężkości danej figury

bardzo popularne jest wykorzystanie ruchu [118], głębi lub termowizji, co każdorazowo stanowi spore ułatwienie implementacji procesu segmentacji oraz znacząco zwiększa skuteczność algorytmu segmentacji.

Analiza Top-Down

Po odnalezieniu i rozpoznaniu danego elementu w scenie nie jest konieczne przeprowadzanie kolejnego jego rozpoznawania w następnej klatce strumienia wideo. Można przyjąć, że w kolejnej klatce obrazu będzie się on również znajdował, w zbliżonej lokalizacji. Takie podejście (gdy analiza potwierdza/obala wcześniejszą hipotezę dotyczącą elementów obrazu) nazywa się w literaturze „*Top-Down*" (w przeciwieństwie do klasycznej, „*Bottom-Up*").

Architektury kognitywne

Działanie systemu wizyjnego, a w szczególności jego algorytmów rozumienia wizji, w naturalny sposób ewoluuje/dąży do rozpoznawania elementów sceny, czyli do przyporządkowania danemu zbiorowi pikseli właściwej nazwy, właściwych parametrów, a w przypadku robota właściwych sposobów interakcji (obsługi). Systemy wizyjne robotów przemysłowych niejednokrotnie nie potrzebują zaawansowanych metod rozumienia wizji, ponieważ rozróżniają ograniczoną liczbę klas elementów oraz pracują w niezmiennych warunkach. W przypadku robotów przeznaczonych do interakcji z człowiekiem, należy stosować/rozwijać algorytmy umożliwiające korzystanie z bazy wiedzy wykraczającej poza możliwości prostych systemów wizyjnych z bazą kształtów. Rozwiązaniem mogą się okazać architektury kognitywne, a dokładniej architektury kognitywne, symboliczne oraz emergentne. Krótki przegląd architektur kognitywnych opisuje m.in. W. Duch w [136]. Rozwiązanie zaproponowane w niniejszej pracy (oraz wykonane eksperymentalne implementacje) zostały zaprojektowane z myślą o użyciu architektury emergentnej – Sieci HTM – jako metody rozumienia wizji i wnioskowania.

3. Istota rozwiązania

Zmysł wzroku można nazwać biologicznym konwerterem pewnego wycinka fal elektromagnetycznych na impulsy elektryczne. W połączeniu z procesami zachodzącymi podczas przesyłania i obróbki tych impulsów elektrycznych (w nerwie wzrokowym i mózgu) do wspomnianej konwersji należy dodać złożone zagadnienie interpretacji sygnału. Aby człowiek potrafił utworzyć, rozumieć i kontrolować „sztuczny zmysł wzroku", należy dopilnować, aby cały proces konwersji i interpretacji był maksymalnie zbliżony do biologicznego odpowiednika. Oczywiście celowe jest wprowadzanie fakultatywnych modyfikacji w znaczący sposób zwiększających użyteczność danych wejściowych, takich jak m.in.:

- szersze rozmieszczenie oczu/kamer skutkuje zwiększeniem czytelności różnic w obrazie stereoskopowym[30],
- zastosowanie kamery działającej w tzw. bliskiej podczerwieni poprawia „widoczność" w przypadku mgły/dymu i innych sytuacji związanych z niepożądanym rozpraszaniem światła lub zbyt słabym oświetleniem sceny,
- zastosowanie kamery termowizyjnej daje dużo informacji w przypadku istot żywych lub innych źródeł ciepła,
- zwiększenie rozdzielczości kamery (podsystemu akwizycji) zwiększa szczegółowość przechwytywanego obrazu.

Jednak w przypadku implementowania takich lub innych dodatkowych funkcjonalności trzeba brać pod uwagę złożoność wynikowego procesu widzenia, który otrzymamy w ten sposób.

[30] szczególnie w przypadku obrazów stosunkowo niskiej rozdzielczości

3.1. Działanie wzroku u człowieka

Ludzkie oko jest narządem, który zadziwia prostotą funkcjonowania: światło pada przez soczewkę na siatkówkę, w której pręciki i słupki są pierwszym etapem zamiany światła widzialnego na impulsy przesyłane nerwem wzrokowym do mózgu, gdzie obraz jest analizowany i interpretowany. Mimo że funkcjonowanie oka można streścić w jednym zdaniu, do tej pory nie udało się stworzyć syntetycznego układu imitującego proces widzenia. Prostota działania wzroku okazuje się ułudą, gdy pod uwagę zostaną wzięte wszystkie szczegóły procesu pozyskiwania i wstępnego przetwarzania informacji wizualnej.[31] Jeżeli przy rozważaniach uwzględniony zostanie sposób przetwarzania i rozumienia wizji w mózgu, wspomniana prostota po dziś dzień ustępuje miejsca pytaniom pozostawionym bez odpowiedzi i domniemaniom.

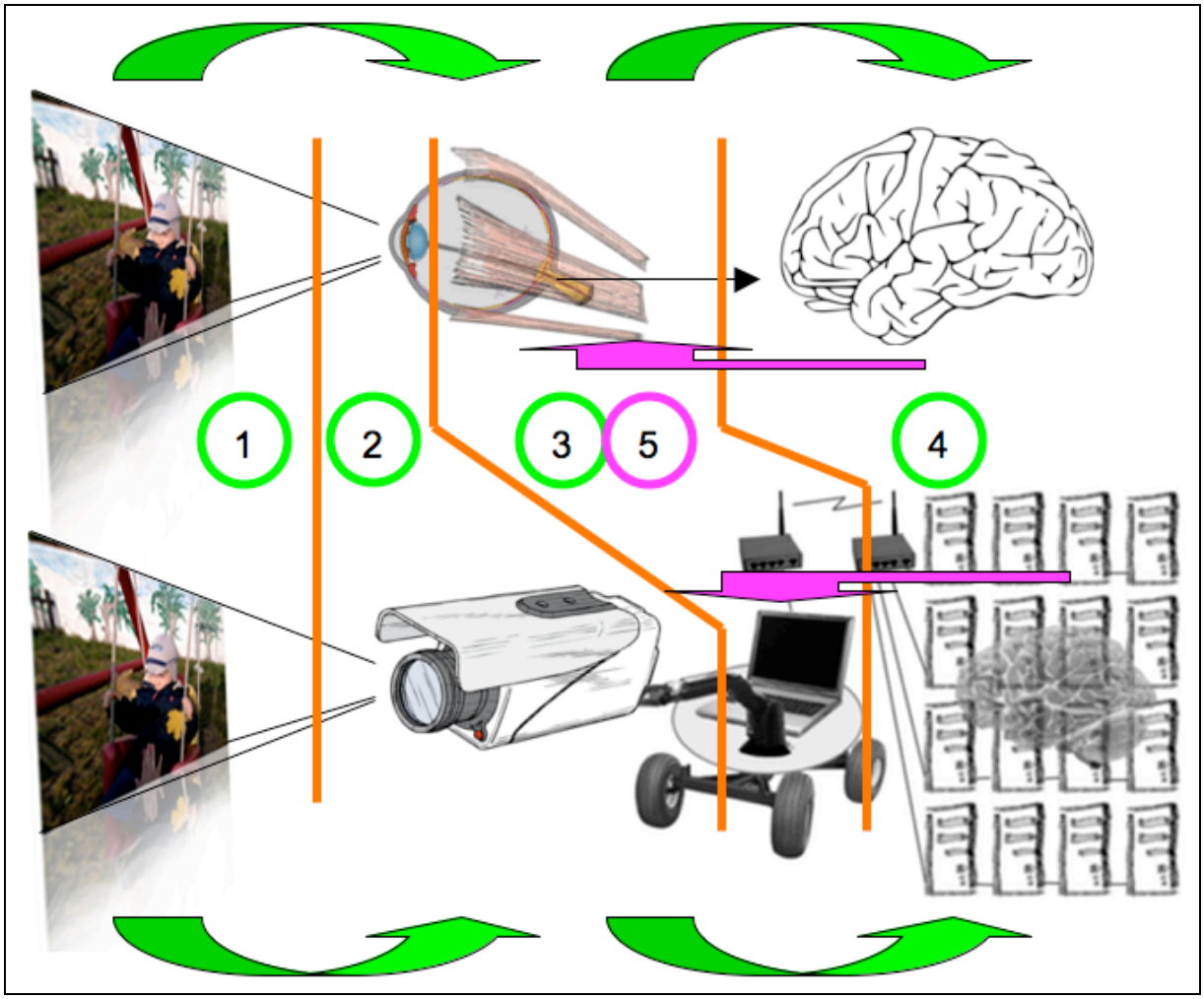
Próby stworzenia syntetycznego układu imitującego biologiczny proces widzenia praktycznie stoją w miejscu. Owszem, komputerowe systemy wizyjne są intensywnie eksplorowane, ale nie przekłada się to na postęp w próbach implementacyjnego modelowania ludzkiego wzroku. Przyczyny takiego stanu rzeczy można się dopatrywać w szczegółach technicznych istniejących dziś systemów wizyjnych. Takie cechy systemów jak duża rozdzielczość czy wysoka jakość pozyskiwanego obrazu stały się niezbędnym warunkiem poprawnego funkcjonowania coraz bardziej wyrafinowanych algorytmów przetwarzania obrazu. Twórcy dzisiejszych systemów wizyjnych komponentami elektronicznymi (kamerą i komputerem) reprezentują narządy biologiczne (oko i mózg), znacznie oddalając się od biologicznego pierwowzoru. Systemy te stają się bardziej podobne w działaniu do oka złożonego[32] [86] niż do ludzkiego – oka prostego. Tymczasem siatkówka oka ludzkiego nie jest jednolicie pokryta czopkami i pręcikami – istnieje fragment o strategicznym znaczeniu, cechujący się zwiększonym zagęszczeniem tych komórek, zwany żółtą plamką. Dlaczego nie wykorzystać tego faktu w celu redukcji ilości danych wraz z odległością od obserwowanego detalu?

Ponadto traktowanie oka jako takiego wejścia biologicznego systemu wizyjnego, z którego można jedynie odczytywać dane, nie do końca jest właściwe. Z biologicznym procesem widzenia nierozerwalnie związane jest zagadnienie sterowania – m.in. w postaci

[31] Za wymienione w tym zdaniu badania i odkrycia przyznano wiele prestiżowych nagród, w tym dwie Nagrody Nobla [87] [88]

[32] Oczy złożone (mozaikowe) występują m.in. u owadów

mechanizmu sakkad. Obraz pochodzący z centralnego punktu, na który skierowany jest wzrok, pada na obszar siatkówki zwany żółtą plamką (Rys. 3.1). Aby możliwe było rozumienie całej sceny lub choćby obiektu, potrzebne jest przesunięcie linii wzroku na kolejny punkt/detal obrazu. To przesunięcie zwane jest sakkadą. Powstała w ten sposób sekwencja czasowa kolejnych wzorców przestrzennych (obrazów) dekodowana jest w mózgu umożliwiając podjęcie akcji lub dalszą analizę.



Rys. 3.1. Widzenie oraz droga wzrokowa w systemie wizyjnym – na górze: biologiczny pierwowzór (uproszczenie), na dole: proponowane rozwiązanie. 1) otoczenie, 2) akwizycja, 3) zmiana reprezentacji na POI z użyciem współrzędnych punktu fiksacji, 4) analiza danych (przetwarzanie, wnioskowanie), 5) informacja zwrotna dot. nowego punktu fiksacji.

W biologii ruchy sakkadowe są badane już od pół wieku. Choć znane były jeszcze wcześniej, dopiero praca A.L.Yarbus'a [152][33] uważana jest za pierwszy przypadek badania sakkad poprzez zarejestrowanie ruchów pomiędzy punktami fiksacji. Dziś badanie sakkad należy do jednego z najciekawszych i jednocześnie najprostszych, „akademickich" przykładów ruchu i sterowania ruchem gałki ocznej [129]. Niebywale motywującym do

[33] opublikowana pierwszy raz w 1962 roku, w języku rosyjskim

dalszych badań jest również fakt, że pomimo pozornie małej złożoności zasady działania sakkad nadal zagadką są kluczowe zagadnienia[34] związane z sakkadami jako obiektem sterowania. Sam mechanizm działania łącza układu nerwowego z mięśniami (szybkie działanie mięśnia w trakcie istnienia potencjału czynnościowego neuronów sterujących) w sposób ciekawy i przystępny opisano w pracy [129].

Aspekt sterowania, istniejący w każdym z rodzajów ruchów biologicznej gałki ocznej, wspaniale opisano również pracy [116], kładąc mniej nacisku na aspekty biologiczne, za to szczegółowo analizując model matematyczny aż po równania opisujące ruch.

W materiałach multimedialnych dostępnych na oficjalnej stronie projektu GazeTracker [27] można odnaleźć plik wideo [26] pokazujący przemieszczanie się linii wzroku osoby czytającej zawartość strony internetowej. Zaskakujący jest stopień „niedokładności" z jaką ludzkie oko skanuje/ogarnia kolejne linie tekstu – nie rozpoznając poszczególnych liter, ale zatrzymując wzrok co pewien czas zaledwie w okolicy danej linii tekstu. Przeanalizowanie tego procesu pozwoliło ocenić „obszar roboczy" żółtej plamki, a co za tym idzie określić docelową powierzchnię obszaru o najlepszej jakości obrazu w tworzonym systemie wizyjnym.

3.2. Systemy wizyjne naśladujące działanie ludzkiego wzroku

Algorytmy przetwarzania obrazu w dzisiejszych systemach wizyjnych mają jedną niefortunną[35] cechę wspólną – nadmiarowość danych, wynikającą z jednolitej rastrowej reprezentacji obrazu wejściowego systemu wizyjnego. W przypadku transmisji często używa się kompresji (bezstratnej), jednak przy przetwarzaniu kompresja nie jest żadnym ułatwieniem (wręcz czyni proces przetwarzania bardziej czasochłonnym). Kompresja stratna jest uważana za niedopuszczalną, ponieważ pogarsza wyniki działania filtrów i algorytmów przetwarzania. Dlatego część twórców systemów wizyjnych rezygnuje całkowicie z kompresji. Tymczasem sposób działania biologicznego systemu wizyjnego (dla oka prostego) pokazuje, że koncepcja jednolitej, najwyższej jakości obrazu nie tylko nie jest

[34] M.in. nie jest znana lokalizacja komparatora [129].

[35] Reprezentacja obrazu wejściowego pod postacią jednolitego rastra w większości systemów wizyjnych jest jedyną opcją, nie rozważa się istnienia innej.

najlepszą metodą reprezentacji, lecz co więcej jest metodą przeczącą biologicznemu pierwowzorowi.

U człowieka plamka żółta zajmuje powierzchnię[36] o średnicy około 1 mm [62][37], czyli około[38] 0.17% powierzchni siatkówki. W zaimplementowanym na potrzeby badań systemie wizyjnym powierzchnia wirtualnej żółtej plamki nie była dobierana aż tak odważnie – pierwsze testy przeprowadzano różnicując współczynnik kompresji w prostokątnym obszarze stanowiącym około 11% powierzchni obrazu wejściowego. Wersja finalna algorytmu posiada możliwość regulacji wszystkich parametrów funkcji parametru progu, ale domyślne wartości powodują, że zauważalnie gorsza jakość jest używana poza kołem o powierzchni stanowiącej około 7% obrazu.

3.2.1. Wybrane znane rozwiązania

Historia systemów wizyjnych odnotowała już kilka prób skorzystania z możliwości zróżnicowania poziomu istotności obrazu oraz wybiórczej jego analizy. Najbardziej znaną taką metodą przetwarzania danych wizualnych jest tzw. aktywna wizja (*ang. Active Vision*, w skrócie AV)

Active Vision

Fundamentalnym założeniem AV jest jedynie fragmentaryczna analiza obrazu wejściowego. Pod tym pojęciem najczęściej rozumie się jedno z trzech (podobnych do siebie) następujących rozwiązań:

1) Po wstępnej analizie całego obrazu wejściowego algorytm wybiera jego fragment (najczęściej obszar prostokątny) do analizy szczegółowej, reszta obrazu jest ignorowana. Metoda z powodzeniem używana w oprogramowaniu fotoradarów do zautomatyzowanego odczytywania numerów tablic

[36] Należy dodać, że sama plamka żółta również nie jest jednorodnie pokryta czopkami, ich największe zagęszczenie zlokalizowane jest w dołku środkowym oka (powierzchnia o średnicy ok. 0.2mm [62]). Aby dobrze odczytać proporcje, warto dodać, że cała siatkówka po rozwinięciu na płaszczyznę miałaby średnicę około 43 mm [62].

[37] W innych źródłach czasem podaje się 5mm lub 6mm, jednak te wartości powinny być używane dla rozróżnienia widzenia centralnego i widzenia peryferyjnego. Rozróżnienie to powinno być dokonywane w aspekcie „pręciki/czopki", a więc uwzględniać nie lokalizację zagęszczenia czopków, lecz pręcików (względem lokalizacji zagęszczenia czopków).

[38] Szacując według informacji zawartych w literaturze [62], cytowanych we wcześniejszych przypisach

rejestracyjnych – wstępna analiza szuka tablicy (m.in. w oparciu o położenie bryły auta lub jego reflektorów), analiza precyzyjna odczytuje znaki alfanumeryczne tablicy.

2) Obszar podlegający precyzyjnej analizie jest obszarem ruchomym, jego położenie wynika wprost z algorytmu i prostych reguł. Metoda była swego czasu używana m.in. w algorytmie rozpoznawania drzwi – system wizyjny robota mobilnego natrafiwszy na pionową krawędź „podążał wzrokiem" w górę wzdłuż krawędzi, jeżeli krawędź okazała się futryną to również w bok i w dół, co umożliwiało nie tylko wykrycie drzwi, ale również kalibrację ich położenia. Metoda ta obecnie [83] jest używana w innych zadaniach, a mianowicie przy śledzeniu obiektów ruchomych (*ang. tracking*).

3) Cały obraz jest sekwencyjnie skanowany blok po bloku (co w efekcie daje przeanalizowanie całego obrazu). Metoda pozornie mało atrakcyjna, jednak w praktyce przydatna – m.in. przy synchronizowaniu i porównywaniu obrazu z kamer tworzących kamerę stereowizyjną, lub przy klasycznym formowaniu sygnału wejściowego węzłów warstwy sensorycznej wizyjnej sieci HTM [92].

Proponowane w niniejszej pracy rozwiązanie różni się od opisanych metod, a najważniejszą różnicą (z punktu widzenia istniejących sposobów rozumienia terminu AV) jest sposób użycia sprzężenia zwrotnego, będący nierozerwalnym aspektem każdej klatki przesyłanego obrazu oraz każdego z etapów przetwarzania. W proponowanym systemie wizyjnym uwzględniono występujące w biologicznych systemach ruchy gałek ocznych pomiędzy jednym punktem fiksacji a drugim, zwane sakkadami [152] [153] [15] [46] [27]. Bezpośrednim skutkiem sakkad (zmian punktu fiksacji) jest zmiana obrazu wejściowego przechwytywanego przez receptory z obszaru plamki żółtej, a więc fragmentu obrazu o najlepszej jakości.

Region-of-Interest (JPEG 2000)

Przeprowadzona analiza literaturowa wykazała, że istnieje bardzo wiele prac opisujących rozmaite metody reprezentacji obrazu mające na celu upodobnienie działania systemu wizyjnego do ludzkiego wzroku. Najważniejszymi, wartymi wspomnienia, są metody stosujące układ współrzędnych log-polar. Ponieważ przekształcanie wszystkich

pikseli obrazu z użyciem jednostki zmiennoprzecinkowej procesora uznano za zbyt obciążające, podjęto próby wytworzenia nowej metody reprezentacji obrazu. Nowa metoda miała spełniać następujące założenia:

1) niewielka złożoność obliczeniowa algorytmu przekształcającego z/do obrazu rastrowego,
2) najwyższa możliwa jakość/rozdzielczość obrazu w okolicy punktu o zadanych współrzędnych,
3) fragmenty obrazu oddalone od ww. punku nie muszą być przekształcane metodami bezstratnymi,
4) możliwe jest stosowanie kompresji (aby zminimalizować czas transmisji między węzłami klastra) przy zachowaniu punktów 2 i 3.

We wczesnych etapach badań i eksperymentów próbowano zaproponować metodę reprezentacji obrazu ze zmienną (nierównomierną) rozdzielczością obrazu [122], jednak w trakcie badań znaleziono lepsze rozwiązanie, wywodzące się z technologii *Region-of-Interest* [138], ujętej w standardzie JPEG2000 [48].

Standard JPEG2000 został opracowany jako następca standardu JPEG. Zawiera wiele udoskonaleń, które czynią go o wiele lepszym od poprzednika, w tym najważniejsze dla niniejszej pracy:

- obraz ma zauważalnie lepszą jakość przy tej samej objętości pliku (lub mniejszy plik przy porównywalnej wizualnie jakości),
- możliwe jest użycie kompresji/konwersji bezstratnej,
- możliwe jest zdefiniowanie (prostokątnych) regionów o lepszej jakości obrazu (tzw. *Regions-of-Interest*),

JPEG2000 korzysta z dyskretnej transformaty falkowej zamiast dyskretnej transformaty kosinusowej używanej w JPEG.

Z uwagi na niesamowitą popularność standardu JPEG oraz nieco większą złożoność obliczeniową standardu JPEG2000, JPEG2000 nigdy nie zdetronizował poprzednika[39].

[39] Dotyczy obrazów dwuwymiarowych. Zgodnie ze specyfikacją DCI (Digital Cinema Initiatives), JPEG2000 jest wykorzystywany w trójwymiarowym kinie cyfrowym, jego popularność rośnie zatem wraz z popularnością kina trójwymiarowego. Dodatkowym atutem standardu są nieopisane w niniejszej pracy rozszerzenia wspomagające i kompensujące ruch.

Podstawą funkcjonowania standardu JPEG2000 [48], jak również podstawą funkcjonowania zaproponowanego w niniejszej pracy rozwiązania, jest dyskretna transformata falkowa (DWT). W standardzie JPEG2000 jakość obrazu (i zarazem wielkość pliku) kontrolowana jest w głównej mierze wartością parametru *threshold*. Parametr ten jest uwzględniany na etapie algorytmu DWT, powodując utratę części informacji[40] poprzez wyzerowanie wartości mniejszych od niego. Na potrzeby eksperymentów zaimplementowano najprostszą wersję algorytmu DWT[41] nie korzystając z żadnych metod akceleracji sprzętowej obliczeń/przekształceń[42]. Poniższy rysunek (będący wynikiem działania jednej z opracowanych implementacji) przedstawia w przekarykaturowany sposób zniekształcenia z jakimi należy się liczyć ustawiając zbyt dużą wartość parametru *threshold* (Rys. 3.2). Naturalnie w systemie wizyjnym nie należy ustawiać aż tak dużej wartości parametru *threshold*, wręcz przeciwnie – optymalnie jest korzystać z jak najmniejszych jego wartości. Poniższa tabela (Tabela 3.1) pokazuje, że nawet dla zerowej wartości parametru *threshold*, zaimplementowany prosty algorytm DWT-IDWT daje możliwość uzyskania bardzo dobrych wyników kompresji/konwersji bezstratnej.

Rys. 3.2. Obraz wejściowy (po lewej) po przetworzeniu transformatą DWT (i odwrotną – IDWT) dla wartości *threshold*=15 (w środku) oraz 45 (po prawej) (informacja o prawach autorskich do rysunku „Lena": [68])

[40] zawartej w tzw. macierzach współczynników

[41] analizuje obraz jedynie liniami poziomymi, użyta falka to falka Haar'a

[42] pozwala to podejść obiektywnie do wyników eksperymentów – otrzymane wyniki można łatwo poprawić w razie potrzeby.

Tabela 3.1. Liczba zer w bajtach pamięci przydzielonej obrazowi dla różnych wartości parametru *threshold*. Wyniki otrzymano w wyniku działania aplikacji opisanej w rozdziale 4.4.3.

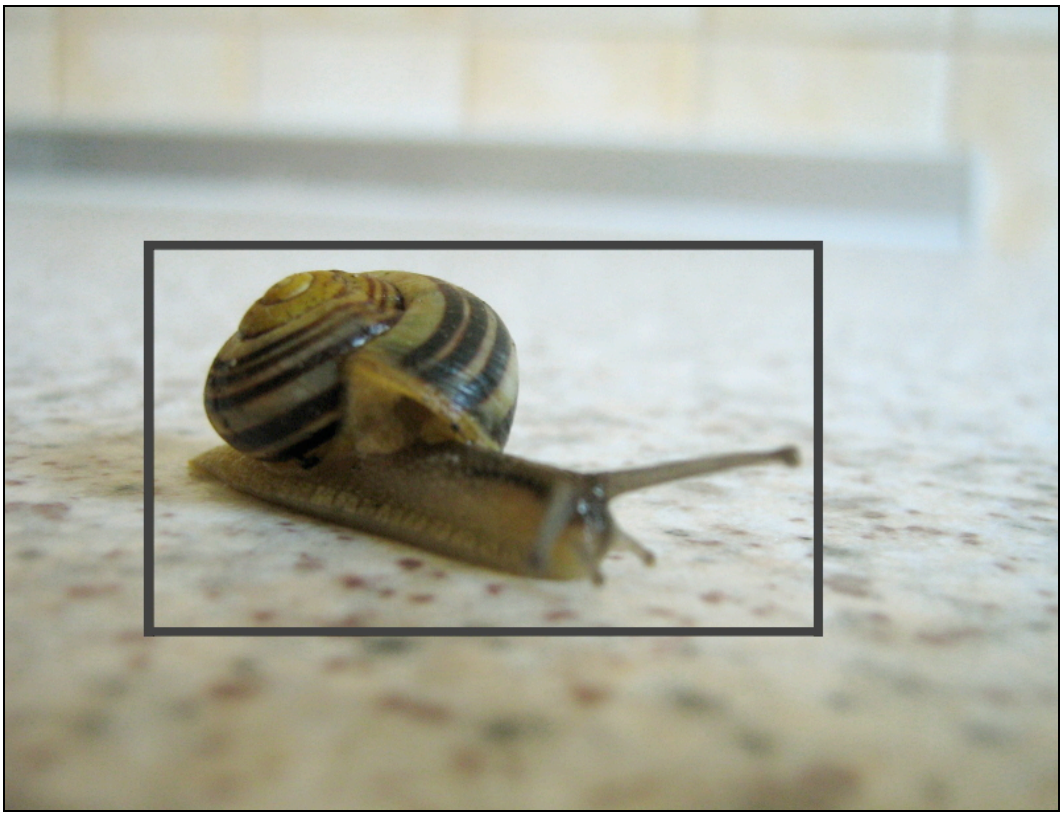
	bitmapa (Rys. 3.2 a)	DWT, *threshold*=0	DWT, *threshold*=5	DWT, *threshold*=15 (Rys. 3.2 b)	DWT, *threshold* =45 (Rys. 3.2 c)
liczba bajtów niezerowych	196608	69132	31951	8504	5771
liczba bajtów zerowych	0	127476	164657	188104	190837

Pojawienie się zer w cyfrowej reprezentacji obrazu daje możliwość zastosowania wielu prostych i bezstratnych jednocześnie algorytmów kompresji. Najprostszymi z nich są algorytmy RLE (*ang. run-length encoding*) oraz tzw. *entropy-coding*. Dla potwierdzenia skuteczności oraz celowości użycia algorytmu kompresji przeprowadzono badania z użyciem prostej implementacji – wyniki opisano w rozdziale 4.4.3.

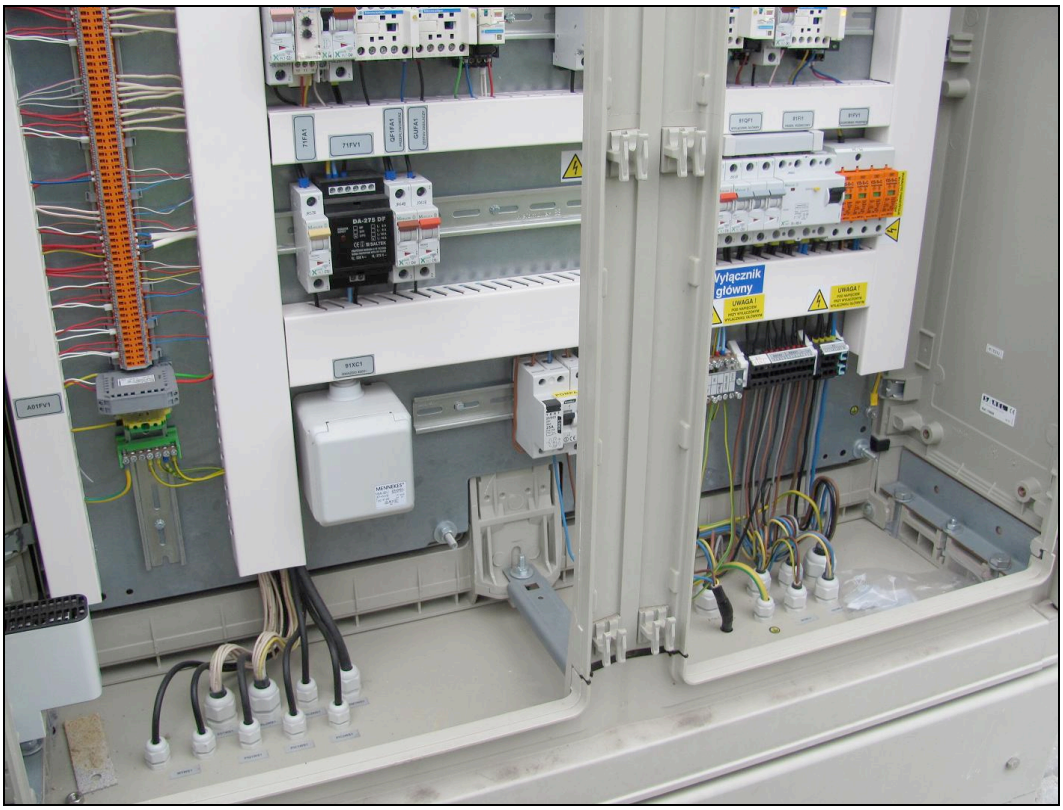
Specyfikacja standardu JPEG2000 przewiduje istnienie tzw. *Regions-of-Interest* (ROI) – prostokątnych fragmentów obrazu o innej (lepszej) jakości obrazu (Rys. 3.3). Poprawa jakości osiągana jest poprzez inną (mniejszą) wartość parametru *threshold*. W specyfikacji nie udzielono[43] wskazówek, w jaki sposób[44] algorytm (lub użytkownik) ma wybierać współrzędne tego prostokąta (Rys. 3.4) oraz o ile należy zróżnicować wartość parametru *threshold*.

[43] decyzję pozostawiono programiście

[44] Istnieją ciekawe prace dotyczące tego zagadnienia (proponujące sposób definiowania ROI), np. [14].



Rys. 3.3. Przykładowy obraz z zaznaczonym ROI (prostokątnym fragmentem bardziej istotnym, wymagającym lepszej jakości niż otoczenie)



Rys. 3.4. Przykładowy obraz, w którym trudno automatycznie określić fragment bardziej istotny lub wymagający lepszej jakości

Z braku uniwersalnych algorytmów zdolnych do automatycznego określania współrzędnych rejonów wymagających lepszej jakości[45], technologia ROI standardu JPEG2000 była przez programistów niezmiernie rzadko brana pod uwagę.

Obecnie funkcjonuje drugie, niezwiązane z JPEG2000, znaczenie terminu ROI – bywa on używany w systemach wizyjnych, lecz dopiero na etapie rozpoznawania obiektów[46]. Przykłady takich systemów [58] [154] można spotkać najczęściej w literaturze związanej z przetwarzaniem obrazu (*ang. Image Processing*).

3.2.2. Proponowane rozwiązanie – Point-of-Interest

Obierając za punkt wyjściowy technologię ROI standardu JPEG2000, podjęto działania mające na celu zaprojektowanie zarysu nowego rozwiązania, umożliwiającego użycie klastra jako mocy obliczeniowej systemu wizyjnego. W drodze analizy literaturowej oraz eksperymentów implementacyjnych udało się uściślić to bardzo ogólne określenie za pomocą szczegółowych, wypunktowanych poniżej, wymogów:

- Kontrola obszaru o lepszej jakości obrazu powinna odbywać się poprzez modyfikację współrzędnych pojedynczego punktu, wzorem biologicznego punktu fiksacji, a nie pary punktów definiującej prostokąt.
- Jakość obrazu powinna zmieniać się w sposób płynny pomiędzy obszarem o lepszej jakości a pozostałą częścią obrazu.[47]
- Współrzędne punktu fiksacji powinny być obecne nie tylko wśród wejść modułu konwertującego, ale również wśród jego wyjść – najlepiej poprzez dołączenie ich do danych wizualnych.
- Wszelkie parametry techniczne wynikające z powyższych wytycznych powinny być dostępne i modyfikowalne na zewnątrz modułu.

[45] Twórcy standardu JPEG2000 nie dysponowali gotową uniwersalną implementacją algorytmu wyznaczającego współrzędne ROI. Standard był tworzony przyszłościowo. Najbardziej popularne dziś algorytmy, takie jak algorytmy rozpoznawania twarzy, (obecnie zawarte w oprogramowaniu praktycznie każdego aparatu fotograficznego), były dopiero tworzone.

[46] Przykładowo, algorytm wyszukujący w obrazie wejściowym sylwetek ludzi, po akwizycji, filtrowaniu, przetwarzaniu i wnioskowaniu może zasugerować współrzędne prostokąta, w którym jest człowiek. Tak rozumiany ROI jest wejściem algorytmów śledzących (*ang. tracking*)

[47] W standardzie JPEG2000 podstawowym sposobem wpływu na jakość (i objętość) pliku graficznego jest wybór pożądanej wartości parametru threshold – stałej dla danego pliku. Region ROI różni się od pozostałej części obrazu tym, że posiada odmienną wartość tego parametru.

- Każda forma kompresji danych jest pożądana/dopuszczalna, o ile nie pogorszy jakości obrazu w sąsiedztwie punktu fiksacji oraz o ile nie będzie obciążająca obliczeniowo.
- Nowo tworzone rozwiązanie powinno charakteryzować się na tyle małą złożonością obliczeniową implementacji, aby możliwe (opłacalne czasowo) było przeprowadzenie konwersji oraz rozesłanie danych do kilku-kilkunastu[48] węzłów klastra.

W ten sposób usystematyzowane założenia były podstawą wytworzonego nowego rozwiązania, które nazwano *Point-of-Interest*[49], w skrócie POI.

Otrzymany z wykorzystaniem spiralnego modelu tworzenia oprogramowania prototypowy moduł reprezentacji danych graficznych wykorzystujący opracowywane rozwiązanie (POI) spełnia wszystkie wymienione powyżej wymogi. Moduł ten stanowi element aplikacji opisanej w rozdziale 4.4.3 oraz późniejszych. Działanie POI (technologii) opisano poniżej, natomiast działanie modułu (implementacji) wyjaśniono w rozdziale 4.4.3.

Rys. 3.5. Obraz wejściowy (po lewej) po przetworzeniu transformatą DWT (i odwrotną – IDWT) z użyciem POI (punkt fiksacji zdefiniowano na nosie modelki), dla wartości *threshold* określanej z wzoru (2). Obraz po środku: *a*=15, *b*=0, *c*=15; obraz po prawej: *a*=45, *b*=0, *c*=45. (informacja o prawach autorskich do rysunku „Lena": [68])

Ponieważ wartość parametru *threshold* jest wyliczana osobno dla każdego punktu (piksela) obrazu, niezwykle istotna jest dbałość o jak najniższą złożoność obliczeniową. Jednym z najprostszych sposobów spełniających postawione wymagania jest użycie odwróconej krzywej dzwonowej (1), tak jak przedstawiono na rysunku Rys. 3.6.

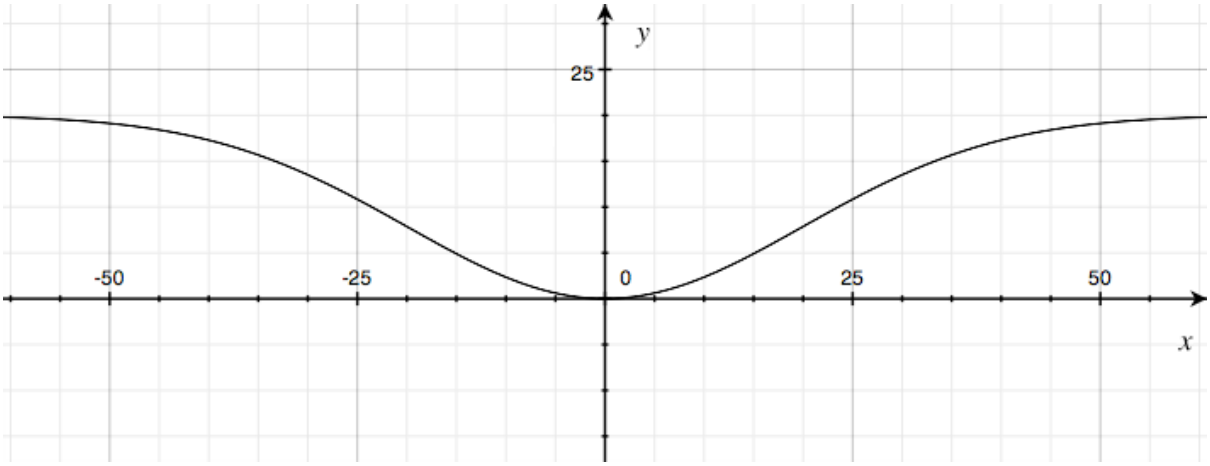
[48] Topologia klastra oraz liczba węzłów będzie rozważana pod koniec rozdziału 3.3.

[49] Ideę, z biologicznego punktu widzenia, w sposób bardziej czytelny oddawałaby nazwa „punkt fiksacji". „POI" bezpośrednio nawiązuje do genezy (ROI), jest zatem bardziej czytelne i intuicyjne dla osób pracujących z systemami wizyjnymi.

$$t = x = a - a \cdot e^{\frac{-(y-b)^2}{2 \cdot c^2}} \tag{1}$$

czyli[50] `x = (a- a* pow(M_E, (-pow((y-b),2)/(2*c*c)) ) );`

gdzie a – amplituda;
b – przesunięcie wzdłuż osi odciętych (w obliczeniach b=0);
c – skalowanie w osi odciętych („rozłożystość” funkcji)



Rys. 3.6. Przykładowy wykres wartości parametru *threshold* w funkcji odległości od punktu fiksacji.[51] Oś odciętych – odległość wyrażona w liczbie pikseli, oś rzędnych – wartość parametru *threshold*.

Wprost z równania (1) wywodzi się równanie (2), określające wartość parametru *threshold* na płaszczyźnie dwuwymiarowej obrazu (Rys. 3.7).

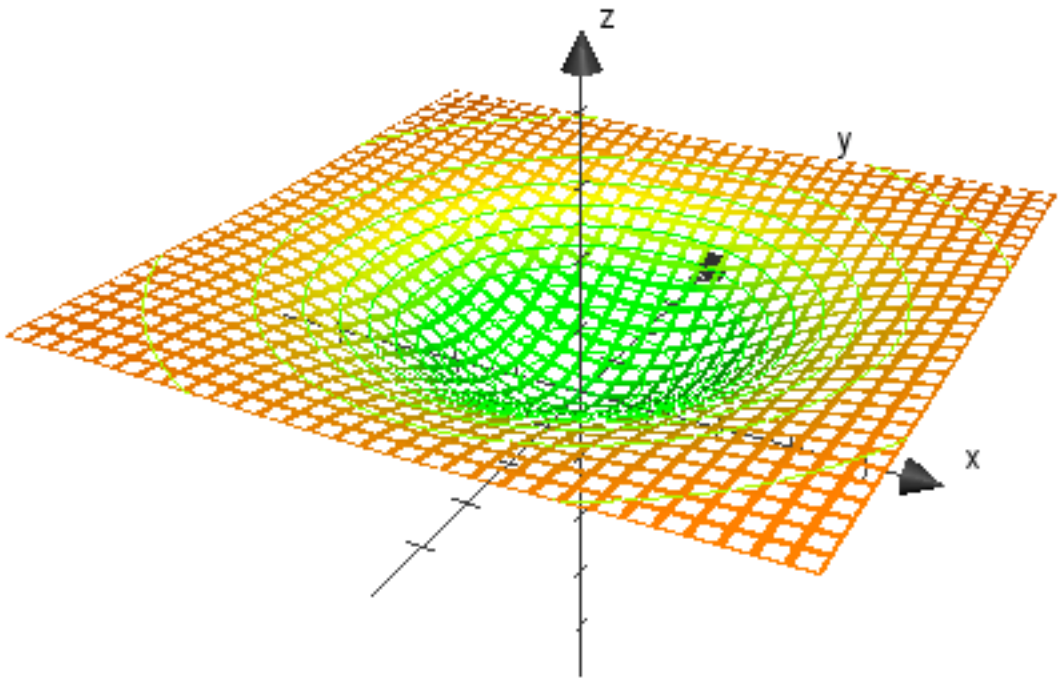
$$t = z = a - a \cdot e^{\frac{-(\sqrt{x^2+y^2}-b)^2}{2 \cdot c^2}} \tag{2}$$

czyli[50] `z = (a- a* pow(M_E, (-pow((d-b),2)/(2*c*c)) ) );`

dla `d = sqrt(x*x+y*y);`

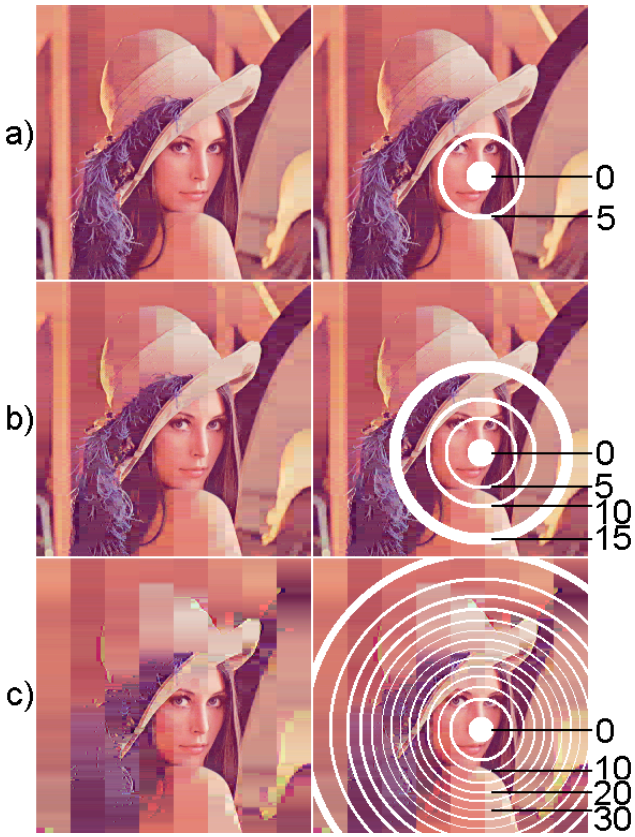
[50] Kod implementujący równanie przytoczono po to, by uniknąć nieścisłości co do sposobu, w jaki w praktyce zaimplementowano wyżej wymienione równanie teoretyczne podczas eksperymentów opisywanych w rozdziale 4.4

[51] Funkcja przedstawiona na rysunku (Rys. 3.6) i (Rys. 3.7) jest funkcją ciągłą – można przy jej użyciu wyznaczać również wartości *threshold* ze zbioru liczb rzeczywistych. W implementacjach opisanych w rozdziale 4.3 nie skorzystano jednak z tej możliwości, rzutując wyliczoną wartość do formatu int – liczb całkowitych.



Rys. 3.7. Przykładowy wykres wartości parametru *threshold* w funkcji odległości od punktu fiksacji dla dwuwymiarowego obrazu. Oś ‘z’ – wartość parametru *threshold*.

Efekt działania zaimplementowanego modułu przedstawia rysunek Rys. 3.8. Jakość obrazu na trzecim rysunku (Rys. 3.8 c) jest celowo zaniżona, aby uwypuklić działanie algorytmu. W implementacji na potrzeby systemu wizyjnego parametry funkcji *threshold* (przede wszystkim parametry *a* i *c*) zostały dobrane tak, aby zapewnić znacznie lepszą jakość obrazu (dając mniejsze wartości *threshold*).



Rys. 3.8. Działanie algorytmu POI dla trzech wariantów wartości parametrów funkcji *threshold*: a) *a*=*c*=10, b) *a*=*c*=17, c) *a*=*c*=70. Po lewej stronie – obrazy wynikowe, po prawej stronie – obrazy wynikowe wraz z liniami konturowymi z zaznaczonymi wybranymi wartościami *threshold*. (rysunek pochodzi z [45], opracowanie własne)

Jak widać na rysunku powyżej (Rys. 3.8), linie konturowe wartości *threshold* nie są rozmieszczone w równych odstępach od siebie. Odległości między nimi wynikają wprost z funkcji przedstawionej na rysunku Rys. 3.6 oraz Rys. 3.7 – najbliżej siebie położone są w punkcie przegięcia funkcji, natomiast w okolicy punktu fiksacji można zauważyć stosunkowo duży obszar, w którym obraz został przekonwertowany z użyciem względnie małego *threshold*. Oddalając się od punktu przegięcia w kierunku miejsc obrazu odległych od punktu fiksacji wartość *threshold* wzrasta coraz mniej dynamicznie, co zapobiega niepożądanej utracie jakości peryferyjnych elementów sceny.

Dzięki zachowaniu najlepszej możliwej jakości obrazu w okolicach punktu fiksacji możliwa jest jego precyzyjna analiza[52].

Dzięki zachowaniu akceptowalnej[53] jakości obrazu w rejonach peryferyjnych obrazu możliwe jest „zauważenie” nowych obiektów lub ruchu w niespodziewanym fragmencie

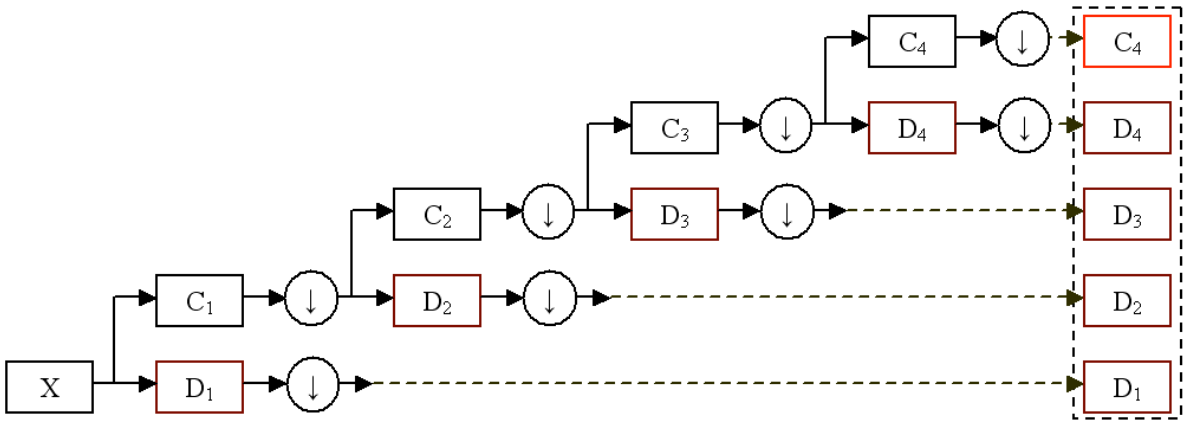
[52] Dotyczy wybranych algorytmów, na przykład podejścia opisanego w rozdziale 3.3 niniejszej pracy. Większość dziś stosowanych algorytmów nie toleruje jakiejkolwiek utraty informacji w obrazie.

sceny, jak również zgrubne określenie spodziewanej lokalizacji kolejnego punktu fiksacji. Sposób analizy danych wizualnych oraz kontrola kluczowego elementu systemu wizyjnego – współrzędnych POI – opisane są w rozdziałach następnych.

Tabela 3.2. Liczba zer w bajtach pamięci przydzielonej obrazowi dla różnych wartości parametrów funkcji *threshold*. Wyniki otrzymano w wyniku działania aplikacji opisanej w rozdziale 4.4.3

	bitmapa (Rys. 3.5 a)	DWT, *threshold*=0	DWT, POI, *threshold* dla *a*=5, *b*=0, *c*=5	DWT, POI *threshold* dla *a*=15, *b*=0, *c*=15 (Rys. 3.5 b)	DWT, POI *threshold* dla *a*=45, *b*=0, *c*=45 (Rys. 3.5 c)
liczba bajtów niezerowych	196608	69132	44377	32405	30205
liczba bajtów zerowych	0	127476	152231	164203	166403

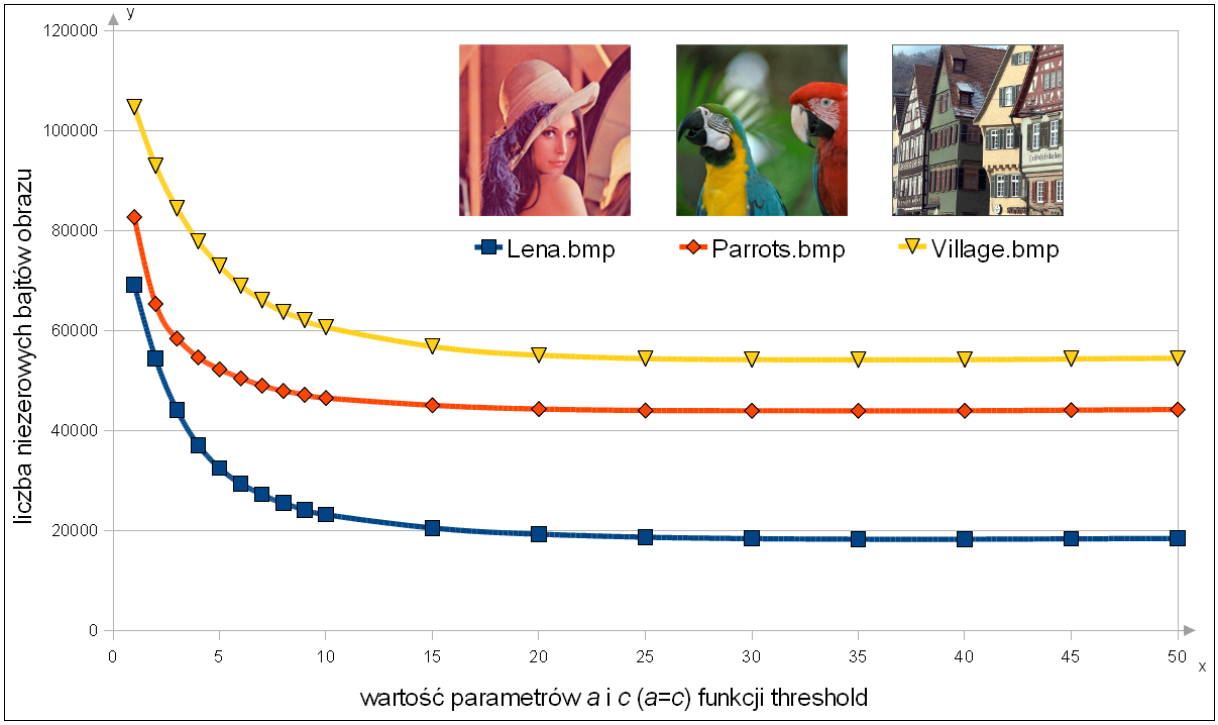
Zdecydowanie najcenniejszą korzyścią wynikającą ze zmiany reprezentacji obrazu poprzez użycie transformaty DWT jest zmiana charakteru danych opisujących ten obraz. Sedno sprawy przedstawia tabela Tabela 3.2. Obraz źródłowy (Rys. 3.5 a) w postaci bitmapy w sektorze danych zawiera informacje na temat kolorów składowych każdego piksela. Ponieważ obraz nie zawiera pikseli idealnie czerwonych, zielonych, m.in., wśród wartości nie wystąpiła wartość „0”. Po przekształceniu obrazu transformatą DWT, czyli rozdzieleniu macierzy współczynników (Rys. 3.9), w macierzach detali („*D*”) można znaleźć sporą ilość bajtów zawierających wartość „0” (Tabela 3.2, druga kolumna danych).



Rys. 3.9. Istota przekształcenia DWT [10] – dekompozycja sygnału na dwie składowe (macierz współczynników aproksymacji „*C*” i macierz współczynników detali „*D*”) za pomocą pary filtrów powtarzana hierarchicznie kilkukrotnie dla „downsamplingowanej” macierzy współczynników aproksymacji. W skrócie: Piramida Mallata [137].

[53] Definiowalnej przez użytkownika/projektanta/programistę, wyznaczanej w drodze prototypowania spiralnego (eksperymentalnie) lub w oparciu o wykres z rysunku Rys. 3.10.

Aby dodatkowo zwiększyć ilość zer w macierzach współczynników, konwersja jest przeprowadzana z użyciem wspomnianego, zmiennego, parametru *threshold*. Przy niewiele większym obciążeniu procesora można wówczas zredukować ilość znaczących bajtów jeszcze o połowę. Należy pamiętać, że w rozwiązywanym problemie – wykorzystania klastra komputerowego jako mocy obliczeniowej systemu wizyjnego – czas transmisji jest o wiele bardziej ważkim elementem niż obciążenie procesora.



Rys. 3.10. Liczba niezerowych bajtów pamięci przydzielonej obrazowi dla różnych wartości parametrów funkcji *threshold*. (informacja o prawach autorskich do rysunku „Lena": [68]) (informacja o prawach autorskich do rysunku „Parrots" i „Village": [60])

Na wykresie zamieszczonym powyżej (Rys. 3.10) na osi odciętych zaznaczono wartość parametrów *a* i *c* (mających wpływ na *threshold*), zaś na osi rzędnych – liczbę niezerowych bajtów obrazu. Wyniki otrzymano w wyniku działania aplikacji opisanej w rozdziale 4.4.2. Jak widać, od pewnego momentu (gdy parametry *a* i *c* mają wartość około 15) dalsze zwiększanie wartości parametrów *a* i *c* przestaje dawać wymierne efekty. Nie pozostaje to jednak bez wpływu na jakość obrazu – na Rys. 3.5b i Rys. 3.5c przedstawiono ten sam obraz dla $a=c=15$ i dla $a=c=45$, jak również na Rys. 3.8b i Rys. 3.8c przedstawiono ten sam obraz dla $a=c=17$ i dla $a=c=70$ – utrata jakości jest ewidentna, natomiast zysk[54] niewielki. Dlatego stosując zaproponowane rozwiązanie w systemie wizyjnym należy wziąć pod uwagę wszystkie aspekty oraz możliwości danego systemu

[54] zysk w postaci dodatkowych zer w macierzach współczynników

wizyjnego, starając się znaleźć złoty środek pomiędzy najlepszą jakością obrazu i jakością akceptowalną z punktu widzenia algorytmów przetwarzających i wnioskujących.

Jeżeli użytkownik/programista systemu wizyjnego zdecyduje się traktować zmienne *a* i *c* rozdzielnie, należy bezwzględnie pamiętać o tym, aby krzywa przedstawiona na rysunku Rys. 3.6, użyta do tworzenia powierzchni przedstawionej na rysunku Rys. 3.7, miała dobrany parametr *c* tak, aby otoczenie punktu fiksacji miało wystarczająco[55] dużą powierzchnię.

Po weryfikacji implementacyjnej na etapie wtórnego prototypowania sprecyzowano dodatkowe cechy opracowanej metodologii reprezentacji i obsługi danych wizualnych:

- Obraz wejściowy konwertowany jest za pomocą transformaty DWT – tak jak ma to miejsce w standardzie JPEG2000.
- Wartość parametru *threshold* nie jest stała w obrębie obrazu lub rejonu, lecz jest zależna od odległości od punktu fiksacji, może być opisana zmodyfikowaną krzywą dzwonową.
- Wartość parametru *threshold* w punkcie fiksacji wynosi zero, wartość maksymalną przyjmuje on w rejonach najbardziej oddalonych od punktu fiksacji.
- Wartość maksymalna parametru *threshold* powinna być dobrana tak, aby obraz był nadal (nawet pomimo ewentualnej późniejszej kompresji) użyteczny.
- Jeżeli robot ma być dedykowany zadaniom interakcji z człowiekiem i otoczeniem człowieka, warto rozważyć przeprowadzanie konwersji (i innych operacji) na reprezentacji zapisanej w modelu przestrzeni barw YUV i na takiej postaci prowadzić analizę obrazu.
- Część wymogów stawianych modułowi konwersji i kompresji obrazu będzie wynikać ze specyfiki topologii klastra oraz zaproponowanego rozdziału zadań. Wymogi te będą opisane w rozdziałach następnych.

3.3. Sieci HTM

Opisana w rozdziale poprzednim metoda reprezentacji obrazu byłaby bezużyteczna, gdyby nie istniał sposób analizy tak zapisanego obrazu i gdyby nie dało się z niego wydobyć cech (i) obiektów obecnych w scenie. Co więcej, utrata części informacji uniemożliwiałaby analizę obrazu typowymi metodami przetwarzania wizji.

[55] Przez „wystarczająco dużą powierzchnię" rozumiany jest tu zbiór pikseli otoczenia punktu fiksacji wystarczający do przeprowadzenia analizy cechy/cech fragmentu obserwowanego obiektu. Szczegóły proponowanego sposobu przetwarzania zawarte są w rozdziale 3.3

Zaproponowana metoda – POI – została zaprojektowana z myślą o wykorzystaniu w tandemie z algorytmami przetwarzania i wnioskowania obecnymi w tzw. sieciach HTM (*ang. Hierarchical Temporal Memory*). Sieci te, stworzone w oparciu o węzły z czasowo-przestrzenną pamięcią autoasocjacyjną[56], naśladują działanie kory nowej ludzkiego mózgu. Od roku 2007, gdy prace nad sieciami HTM nabrały tempa, postępy i możliwości tej technologii stały się[57] imponujące.

W tym miejscu należy zaznaczyć, że wiele spośród elementów składowych technologii HTM jest znanych nie od dziś. Sieci HTM stanowią jednak swoistą kompilację kilku istniejących zagadnień obszaru sztucznych sieci neuronowych, które rozwijane w określonym zamierzonym kierunku, pozwalają stworzyć platformę – rozwiązanie programowe (a w przyszłości sprzętowe) – pomagające w zrozumieniu (i umożliwiające badanie i wykorzystanie) sztucznej sieci działającej w sposób jeszcze bardziej podobny do ludzkiego mózgu niż wynikałoby to z samej analogii budowy biologicznego neuronu i sztucznego neuronu.

Sieci HTM, będące jedną z wielu emergentnych architektur kognitywnych [136], nie stanowią jedynej architektury, w której możliwe jest wykorzystanie zaproponowanego w pracy rozwiązania. Posiadają jednak pewne cechy implementacyjne (przedstawione w dalszej części pracy) odróżniające je od innych. Dalsza część niniejszego rozdziału zawiera krótki opis sieci HTM, uwypuklający najistotniejsze aspekty, które zaważyły na wyborze właśnie tej architektury.[58]

W pierwszej części tego rozdziału pokrótce przybliżono genezę oraz przeznaczenie sieci HTM, następnie opisano dotychczasowe trendy w stosowaniu sieci HTM w zadaniach rozumienia wizji w kontraście do proponowanego rozwiązania.

[56] Pamięć autoasocjacyjna jest takim rodzajem pamięci asocjacyjnej, w którym porównywane dane wejściowe dotyczą jednego (tego samego) wektora opisującego dane/reprezentacje (a nie różnych wektorów) [95]. Pojęcie pamięci asocjacyjnej znane jest w świecie nauki od dawna, jednak używanie jej w kontekście wzorców zarówno przestrzennych jak i czasowych nie jest podejściem popularnym. Pamięć ta znana jest również jako pamięć skojarzeniowa. Pamięć skojarzeniowa bazuje natomiast na sieciach Hopfielda i uznawana jest za najbardziej powszechny (i najlepiej funkcjonujący) rodzaj rekurencyjnych sztucznych sieci neuronowych. [134]

[57] Szczególny postęp związany był z nowościami w wersji 1.2 platformy NuPIC w roku 2007 oraz w wersji planowanej na czerwiec 2011 (jej dokumentacja już jest dostępna).

[58] Na bieżącym etapie rozważań wybór architektury nie jest najważniejszy. Sieci HTM zostały w pracy opisane, aby wskazać realną możliwość praktycznej implementacji rozproszonego (klastrowego) systemu wizyjnego korzystającego z opracowanych w pracy rozwiązań.

Geneza sieci HTM

Szacuje się, że kora nowa dorosłego człowieka ma około 30 miliardów neuronów. Pozornie jest to duża liczba, jeżeli jednak wziąć pod uwagę, że zapisane są tam wszystkie wspomnienia, doświadczenia, wiedza i uczucia, wówczas łatwiej zmienić zdanie – kora nowa zawiera „zaledwie" około 30 miliardów neuronów. W niektórych zastosowaniach sztuczna sieć neuronowa wystarczająca do otrzymania satysfakcjonującego wyniku składa się z kilku do kilkunastu neuronów – tak więc biorąc pod uwagę ilość neuronów w ludzkim mózgu[59] różnica jest oczywista.

Znana jest budowa neuronu, wiadomo jak łączyć neurony w sieć i jak później tą sieć nauczyć i odpytać, dlaczego więc nie pokusić się o zbudowanie aplikacji „dorównującej" człowiekowi – czemu nie spróbować połączyć 30 miliardów sztucznych neuronów? Czy taka aplikacja będzie wtedy mogła konkurować z możliwościami ludzkiego mózgu?

Zmysły, a dokładniej receptory, stanowią wejścia – dotyk, wzrok, słuch i pozostałe. Mózg jest natomiast zamkniętą w czaszce masą tkanki nerwowej, która bez tych wejść jest bezużyteczna. Mózg nie posiada receptorów, podobnie sztuczna sieć neuronowa – bez wejść nie znaczy nic. Sztuczna sieć neuronowa nie potrafi dorównać człowiekowi – budowane sieci są albo zbyt małe, by „zobaczyć" wycinek otaczającego świata, albo zbyt rozległe, a co za tym idzie zbyt wolne, trudne w uczeniu, skomplikowane i nieskore do wyrazistego wnioskowania. Tymczasem biorąc pod uwagę, że typowy biologiczny neuron potrafi wygenerować potencjał czynnościowy i powrócić do poprzedniego stanu w czasie 5ms, można oszacować ile neuronów bierze udział w danej operacji [66]. Załóżmy, że jakaś osoba ma do wykonania następujące zadanie: popatrzeć na zdjęcie, a jeśli na zdjęciu znajduje się Pałac Kultury, nacisnąć przycisk. Pamiętając o tym, jaki jest czas reakcji pojedynczego neuronu, można dojść do wniosku, że większość reakcji/czynności podejmowanych jest w zaledwie kilkudziesięciu – stu kilkudziesięciu krokach „algorytmu" (w literaturze [43] można spotkać tzw. „zasadę stu kroków"). Trudno powiedzieć ile czasu zajęłoby to komputerowi (i czy w ogóle byłoby to wykonalne bez specjalnego programu napisanego do tego konkretnego zadania). Niektórzy twierdzą, że ta miażdżąca przewaga mózgu nad komputerem wynika z tego, że mózg potrafi „przetwarzać" równolegle, podczas gdy komputery są maszynami sekwencyjnymi (dwa lub cztery rdzenie procesora też niewiele tu zmieniają

[59] Według literatury szacunkowo około 100 miliardów.

wobec miliardów neuronów). Należy jednak pamiętać, że po wspomnianych stu krokach pojawia się wynik – reakcja odpowiednich mięśni. Trudno sobie wyobrazić algorytm (nawet całkowicie równoległy), który potrafiłby choćby zebrać informacje z 30 miliardów neuronów i odpowiednio przedstawić wyjścia w zaledwie stu krokach. Należy pamiętać, że w tym czasie (od bodźca do reakcji) zarówno komputer, jak i mózg, musi wykonać mnóstwo operacji na danych wejściowych [102]: obraz przechwytywany (siatkówka / kamera) trafia (przez nerw wzrokowy / przez strumień wideo) do mózgu (lub komputera), gdzie neurony (w ośrodku V1 / w funkcjach i procedurach przetwarzania obrazu) reagują na określone atrybuty obrazu i tak dalej. Część czasu zostaje poświęcona przetworzeniu sygnału wejściowego, część przygotowaniu odpowiedzi na wyjściach, tak więc proces „zrozumienia" prezentowanej fotografii zajmuje jeszcze mniej niż wspomniane 100 kroków.

Jak to jest możliwe, że ewolucja potrafiła wytworzyć tak zaawansowane rozwiązanie jak mózg? Czy można zbudować maszynę, która byłaby uznana za inteligentną w naszym ludzkim, potocznym rozumieniu? Czy będzie można kiedyś nareszcie maszyny „uczyć" zamiast „programować"?

Kora nowa zbudowana jest z sześciu warstw neuronów. Wiadomo też, że ma grubość około 2 mm, że określone jej części zajmują się określonymi zadaniami. Naukowcy od lat potrafią powiedzieć, ile waży przeciętny mózg, jaką powierzchnię ma kora nowa, jak jest zbudowana. Mimo to nadal nie zbudowano maszyny prawdziwie inteligentnej. W 1978 roku Vernon Mountcastle opublikował artykuł, w którym stwierdził, że kora nowa ma „niezwykle jednolitą strukturę". Obszary zajmujące się mową, słuchem, wzrokiem, dotykiem, działaniem mięśni – wszystkie mają podobną strukturę. Oznacza to, że kora używa tego samego algorytmu do przetwarzania różnych danych wejściowych [41]. Jednak naukowcy zajęli się raczej wykorzystaniem sieci neuronowych, a nie poszukiwaniem tego „biologicznego" algorytmu. Tymczasem analizując zestawienie kilku neuronów nie można zrozumieć działania mózgu – tak samo jak eksperymentowanie kilkoma tranzystorami nie zaowocuje zrozumieniem funkcjonowania procesora.[60]

Sieci HTM noszą swoją nazwę nieprzypadkowo. Skrót ten pochodzi od angielskich słów „*hierarchical temporal memory*" i posiada następujące znaczenie [93]:

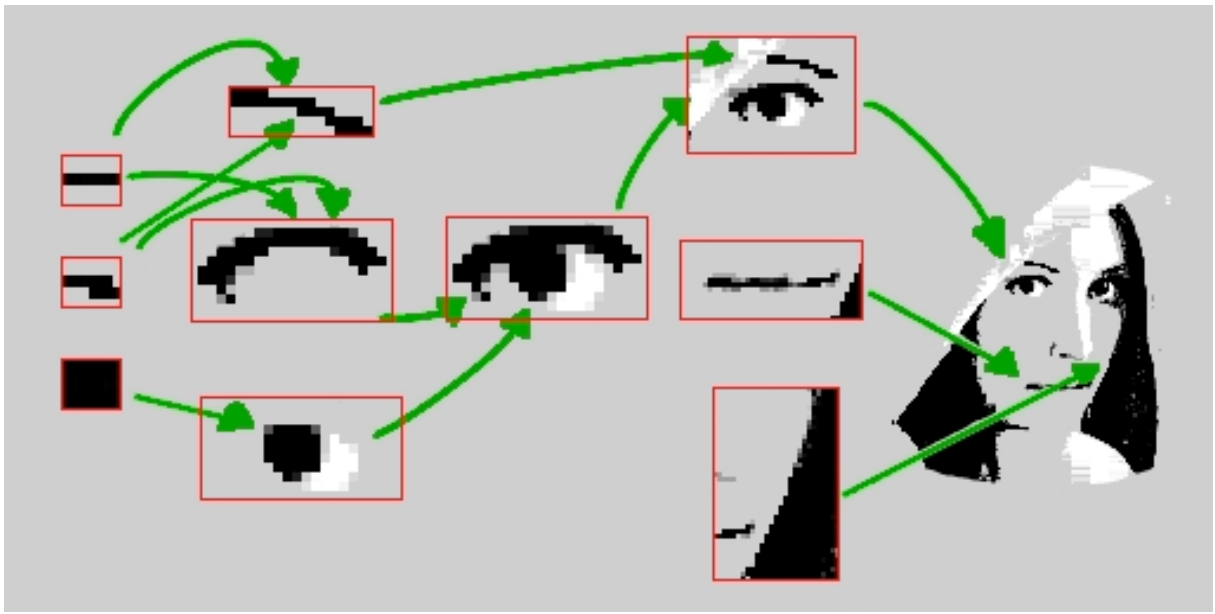
- *Hierarchical* – (hierarchiczne): Sieci HTM zbudowane są z węzłów połączonych hierarchicznie w topologii drzewiastej. Każdy z węzłów to algorytm[61], w którym

[60] Analogia do alegorii Ruperta Sheldrake'a [126] jest zamierzona.

[61] Węzeł rozumiany jako fragment sieci HTM nie ma związku z węzłem rozumianym jako fragment maszyny rozproszonej. Implementacja sprzętowa nie ma tu znaczenia ani wpływu.

zaimplementowano zarówno możliwość uczenia, jak i pamiętania danych wejściowych. Węzły niższej warstwy otrzymują dużą ilość danych wejściowych i po ich przetworzeniu wysyłają „opracowane" dane (na podstawie danych wejściowych) do warstwy wyższej (Rys. 3.11). Można więc powiedzieć, że każda warstwa sieci HTM ma zdolność tworzenia wyższego poziomu abstrakcji.

- *Temporal* – (czasowe): Podczas uczenia sieć HTM musi otrzymywać dane zmieniające się w czasie. O ile dla zmysłu słuchu lub dotyku zmiana bodźca (danych wejściowych) w czasie jest oczywista, to dla przetwarzania nieruchomego obrazu lub tabel z danymi statystycznymi należy tak przygotować dane wejściowe sieci HTM, aby były zmienne w czasie. Przykładowo nieruchomą klatkę obrazu wideo można na wejście sieci podawać w postaci przesuwającego się kolejnymi wierszami (o jeden piksel dalej) prostokąta[62] o liczbie pikseli równej liczbie wejść sieci (w ten sposób sieć analizuje kolejne fragmenty obrazu – wejścia sieci są pobudzane danymi zmiennymi w czasie).
- *Memory* – (pamięć): Węzły sieci HTM działają w dwóch trybach, które można przyrównać do ćwiczenia pamięci (uczenie się) i odpytywania pamięci (tryb wnioskowania). Każdy z węzłów nauczonej sieci HTM „pamięta" określony fragment wzorca wejściowego (w przypadku węzłów warstwy najniższej) lub określony uogólniony zestaw wzorców współistniejących czasowo lub przestrzennie na wyższym poziomie abstrakcji (w przypadku węzłów warstw wyższych).



Rys. 3.11. Hierarchia wzorców na różnych poziomach abstrakcji. (rysunek pochodzi z [45], opracowanie własne)

[62] Dokładnie w ten sposób działają wszystkie istniejące aplikacje implementujące sieci HTM do zadań rozpoznawania obrazu – wynika to z działania dostarczonego wraz z platformą przykładowego węzła wejściowego (tzw. *sensor node*) obrazów dwuwymiarowych.

HTM: Hierarchical

Pomysł, w wyniku którego powstała idea sieci HTM, polegał na możliwie jak najbardziej wiernym odwzorowaniu „algorytmu" mózgu. W korze nowej zadaniem niższych warstw jest przede wszystkim tworzenie nowych poziomów abstrakcji – jeżeli robot obserwuje coś, co ma ogon, wąsy, a obok leży mysz, to możliwe, że widzi kota – w przytoczonym przykładzie jest to po prostu[63] suma cech obserwowalnych w przestrzeni (i w czasie).

Najwyższa, czyli szósta warstwa kory nowej – tzw. warstwa kojarzeniowa – składa się ze stosunkowo niewielkiej ilości komórek, natomiast ogromnej ilości połączeń pomiędzy rozmaitymi, nieraz dość odległymi, obszarami kory. Dzięki temu m.in. istnieje możliwość łączenia informacji pochodzących z różnych zmysłów – wówczas proces wnioskowania może być znacznie bardziej skomplikowany: Załóżmy, że jeden z węzłów przedostatniej warstwy z 80%-ową pewnością rozpoznaje obiekt obserwowany przez system wizyjny i klasyfikuje go do kategorii „pies", a z 20%-ową do kategorii „kot". Jednocześnie inny węzeł tej warstwy, analizujący fale dźwiękowe, z 80%-ową pewnością rozpoznaje skrzypienie drzwi, a z 20%-ową miauknięcie kota [42]. Węzeł nadrzędny w warstwie kojarzeniowej agreguje te wyniki i może jasno określić, jaki jest ostateczny wynik wnioskowania.

Hierarchiczna organizacja węzłów w sieci HTM jest całkowicie wzorowana na najważniejszych połączeniach między warstwami występującymi w korze nowej. Najniższą warstwę tworzą obszary pierwszorzędowej kory sensorycznej, otrzymującej informacje z narządów zmysłów (dla wzroku obszar ten nazwano „V1"). Następnie informacje z V1 trafiają w górę hierarchii do innych obszarów (V2, V4, IT i do innych miejsc). Każdy z regionów przetwarza inny aspekt danych wejściowych – na innym poziomie abstrakcji. Przykładowo neurony w obszarze V4 reagują na kształty o średniej złożoności, na przykład kształty gwiazdek w kolorach niebieskim lub czerwonym [43]. Obszar nazwany MT reaguje na ruch obiektów. W wyższych warstwach kory wzrokowej (tu najważniejszy jest obszar IT) ulokowane są obszary związane z pamięcią złożonych obiektów, na przykład twarzy, zwierząt, przedmiotów. Po porównaniu informacji wizualnej z informacjami docierającymi z innych zmysłów (w warstwie kojarzeniowej) możliwe jest podjęcie reakcji. Pomimo tego, że zmysły są źródłami informacji i należy te informacje „tłumaczyć" na wyższe poziomy abstrakcji, a mięśnie są wyjściami informacji i abstrakcyjne pojęcia w korze należy

[63] Jest to pewne uproszczenie, rzeczywistość może być bardziej skomplikowana.

„upraszczać" na pojęcia mniej abstrakcyjne (aż do „rozkazów" wysyłanych poszczególnym mięśniom), to algorytm działania kory pozostaje ten sam – jedynie kierunek przesyłania jest przeciwny (w dół hierarchii). Komunikacja tak naprawdę w obu przypadkach jest dwukierunkowa, różnica polega tylko na teoretycznym określeniu, które połączenia są połączeniami „nośnymi" informacji, a które są „zwrotnymi".

Sieci HTM nie mają zdeterminowanej struktury. Do dyspozycji programisty-projektanta są węzły, które można łączyć ze sobą dowolnie, tworząc jakąkolwiek strukturę hierarchiczną. Węzły warstw wewnętrznych zazwyczaj projektuje się jako identyczne. Zdarzają się natomiast węzły, które mają odmienne wejścia lub wyjścia (m.in. wszystkie węzły pierwszej warstwy) – dla każdego rodzaju danych wejściowych powinien być wówczas stworzony odpowiedni „interfejs" wejściowy do sieci HTM – tzw. sensor. Przykładowo sensor sieci HTM przetwarzającej dźwięk mógłby zawierać algorytmy określające wysokość dźwięku i kształt fali akustycznej w strumieniu wejściowym. Następne warstwy używałyby tak skonwertowanej informacji do formowania kolejnych poziomów abstrakcji.

HTM: Temporal

Istnieje kilka aspektów współistnienia czasu jako jednego z parametrów związanych z uczeniem czy też danymi źródłowymi. Najbardziej istotnym i czytelnym spośród nich, jest fakt reprezentacji danych wejściowych jako wzorców zmiennych w czasie. Najłatwiej to przedstawić na przykładzie przetwarzania dźwięku. W pierwszorzędowej korze A1 komórki nerwowe reagują na najbardziej „fizyczne" cechy słyszanego dźwięku. Tymczasem dane wejściowe dochodzące do ośrodka A1 mogą być bardzo różne – każde słowo może być wypowiedziane z inną głośnością, szybkością, akcentem, barwą głosu. To właśnie dzięki podstawowej właściwości sieci HTM – zdolności do tworzenia wyższych poziomów abstrakcji – w wyniku określonej w czasie zmiany parametrów dźwięku możemy usłyszeć „słowo" (i skojarzyć jego znaczenie na przykład na podstawie wspomnień), „dźwięk" lub „melodię". Ciekawym przykładem jest tutaj „melodia" – jest to określona w czasie kombinacja interwałów pomiędzy występującymi po sobie dźwiękami. Nie jest istotne, czy melodię ktoś nam zaśpiewa, czy zagra na flecie, czy na pianinie – przez ośrodek słuchu kory nowej zostanie rozpoznana ściśle określona melodia. Wbrew pozorom rozpoznawanie obrazu również może (a w sieciach HTM musi) mieć swój aspekt czasowy – gdy oglądamy zdjęcie, nasz wzrok nie jest utkwiony nieruchomo w środku fotografii, nie obejmuje również całego zdjęcia naraz. W rzeczywistości gałki oczne w ciągu każdej sekundy wykonują kilka

drobnych (niezauważalnych dla świadomości) przemieszczeń zwanych sakkadami. Sakkady powodują, że na tzw. żółtą plamkę trafia coraz to inna część oglądanego zdjęcia. Można więc powiedzieć, że następny poziom abstrakcji, określający oglądany przedmiot, tworzony jest na podstawie zbioru cech z poprzedniego poziomu.

HTM: Memory

W korze nowej istnieje ogromna liczba połączeń zwrotnych między warstwami. Niektórzy naukowcy są zdania, że połączenia zwrotne stanowią więcej niż połowę wszystkich połączeń między neuronami. Po części spowodowane jest to mechanizmem uczenia/pamiętania – istnieje duże podobieństwo między „pamiętaniem" w komórkach kory nowej a typową pamięcią autoasocjacyjną. Po części natomiast wynika to z umiejętności „myślenia", czyli świadomego pobudzania neuronów w oparciu o przechowywane „wspomnienia". Jedną z zalet rozpoznawania wzorców przy użyciu pamięci autoasocjacyjnej, jest możliwość rozpoznania niekompletnych wzorców. Wynikającą z tego cechą „algorytmu" kory nowej, (związaną z istnieniem połączeń zwrotnych), jest zdolność przewidywania. Poszczególne neurony są pobudzane poprzez dane wejściowe wszystkich zmysłów – dotyczy to zarówno neuronów na poziomie sensorycznym, jak i na wyższych poziomach abstrakcji – „kompletując" widziany/słyszany/dotykany obiekt. W czasie pojawiania się kolejnych informacji o tym obiekcie, warstwa kojarzeniowa już może mieć część informacji wystarczającą do określenia, jaki to obiekt. Wystarczy usłyszeć psa, by „mieć przed oczami" jego wielkość, by określić czy jest groźny czy nie. To właśnie dzięki najwyższej warstwie – kojarzeniowej – można mówić o istnieniu tzw. niezmiennych reprezentacji [43], czyli abstrakcyjnych pojęć, określanych przez stopień pobudzenia neuronów z niższych warstw.

HTM: niezmienne reprezentacje

Neurony w najniższych warstwach – wszystko jedno, czy przetwarzają obraz, dźwięk czy dotyk – otrzymują bardzo proste informacje: natężenie/stopień (jasności/ koloru/ dźwięku/ nacisku) w określonym punkcie (wzrok: przestrzeni; dźwięk: częstotliwości; dotyk: w określonym miejscu ciała). Wyższe warstwy grupują wyjścia tych neuronów (patrz: Rys. 3.11) i reagują na bardziej złożone (abstrakcyjne) wzorce przestrzenne i/lub czasowe (wzrok: linie, zgięcia; dźwięk: sylaby, nuty). W kolejnych warstwach tworzone są jeszcze bardziej abstrakcyjne pojęcia (wzrok: proste kształty m.in. gwiazdki; słuch: określona melodia).

Podążając w górę hierarchii można na przykład (w obszarze zwanym „IT") znaleźć neurony reagujące na pojawienie się twarzy w polu widzenia. W przeciwieństwie do neuronów z najniższych warstw, które często zmieniają swój stan, neurony w wyższych warstwach wysyłają serie sygnałów tak długo, jak istnieje dany bodziec – m.in. „twarz" [43]. Pozwala to neuronom najwyższych warstw na wspomniane „skompletowanie" określonego abstrakcyjnego pojęcia (czyli niezmiennej reprezentacji). Wynik jest taki, że jeżeli widzimy psa, ale on nie szczeka, to niezmienna reprezentacja przechowywana w korze nowej podpowiada nam, że gdyby wydał z siebie jakiś dźwięk, to pewnie będzie to szczekanie. Mózg potrafi perfekcyjnie operować niezmiennymi reprezentacjami. Dla przykładu rozumienie mowy prawdopodobnie nie byłoby możliwe (albo byłoby bardzo utrudnione) gdyby nie ta umiejętność kory. Większość słów, które „docierają" do naszych uszu jest na tyle zniekształconych/zakłóconych/niekompletnych, że zrozumienie ich znaczenia możliwe jest jedynie przy użyciu niezmiennych reprezentacji. Czasem niezmienne reprezentacje mogą się przydać również na wyższym poziomie abstrakcji – gdy chcemy zrozumieć sens zdania, podczas gdy nie słyszeliśmy kilku wyrazów.

Pole zastosowań niezmiennych reprezentacji jest ogromne. Najprostszym przykładem są programy rozpoznawania tekstu (tzw. OCR). Niektóre z nich korzystają z sieci neuronowych i/lub baz wiedzy w postaci słowników. Niestety żaden program nie jest lepszy od człowieka, ponieważ w sytuacji, gdy określona litera jest nieczytelna, powstaje dylemat, jakie słowo powinno być ostatecznym wynikiem przetwarzania. Metodami słownikowymi można wykluczyć kombinacje liter nie tworzące wyrazów, ale gdy mimo to zostanie kilka możliwości, to którą wybrać? Najgroźniejszym przykładem jest sytuacja, gdy dana litera zostanie omyłkowo rozpoznana jako inna i – zupełnie przypadkowo – okaże się, że istnieje taki wyraz. Nawet procedura zaznaczania „niepewnych fragmentów" może tego nie zauważyć. Dla człowieka (lub programu korzystającego z sieci HTM) sytuacja taka nie miałaby miejsca właśnie dzięki niezmiennym reprezentacjom i mechanizmom przewidywania i uzupełniania. Gdyby w zwrocie „części mowy" literka „o" została błędnie rozpoznana jako „e" („części mewy") słownikowe sprawdzenie nie zauważyłoby żadnej nieścisłości, podczas gdy niezmienna reprezentacja związana ze słowem „mewa" określa zwierzę (które nijak nie pasuje do pierwotnego kontekstu zdania).

Między innymi dlatego można stwierdzić, że to właśnie w tym podejściu – w architekturach kognitywnych (a do tych należą sieci HTM) – leży klucz do tworzenia prawdziwie inteligentnych programów i maszyn.

HTM – zastosowania

Pomimo tego, że sieci HTM są stosunkowo świeżym pomysłem, dość szybko zaczęły zdobywać swoich zwolenników. Praktycznie w każdej branży, w której przydatne są aplikacje oparte o sieci neuronowe, sieci HTM mogą się okazać bardzo pożyteczne. Szczególnie obiecujące są zastosowania, w których pożądana jest analogia pomiędzy działaniem aplikacji i ludzkiego mózgu, ale to nie jest jedyne pole zastosowań. Najbardziej zaangażowana instytucja – firma Numenta – oferuje oprócz specjalnie przygotowanych narzędzi (dostępnych pod specjalną licencją, nieodpłatnie dla celów niekomercyjnych, w tym naukowo-badawczych) zarówno dość obszerną dokumentację swojego produktu (platformy NuPIC implementujące mechanizmy sieci HTM), jak i artykuły na temat sieci HTM i powiązanej problematyki [92]. Wśród przykładów dołączonych do dystrybucji platformy można znaleźć nie tylko zastosowania związane z wzrokiem czy słuchem, ale m.in. program analizujący tekst, czy program analizujący przepływ wody w rzece. Nie zmienia to faktu, że do najbardziej spektakularnych z pewnością należą aplikacje rozpoznające obrazy, jak słynny program „Pictures" [89] rozpoznający odręcznie narysowany przez użytkownika kształt spośród 48 wcześniej nauczonych kształtów. Ta skromna aplikacja, działająca nawet pod systemem Microsoft Windows, w prosty i dobitny sposób prezentuje przewagę sieci HTM nad sieciami neuronowymi i metodami sztucznej inteligencji – jeżeli użytkownik narysuje „kubek" z uchem po lewej stronie (podczas gdy sieć HTM uczona była wyłącznie rysunkami kubków z uchem po prawej stronie)[64], to ten rysunek i tak zostanie rozpoznany jako „kubek". Jeżeli wspomniany kubek zawsze był wielkości całego okna aplikacji, a użytkownik tym razem narysował malutki kubeczek w rogu, też zostanie prawidłowo rozpoznany. Dla sieci neuronowych czy sztucznej inteligencji te dwa proste przypadki już stanowiłyby pewnego rodzaju wyzwanie (programista musiałby przewidzieć wszystkie takie sytuacje przy projektowaniu algorytmu czy uczeniu sieci), natomiast w przypadku sieci HTM prawidłowy wynik rozpoznawania pojawia się naturalnie, dzięki jej zasadzie funkcjonowania – cechy obserwowanego obrazu łączą się w logiczne grupy, które na odpowiednich poziomach abstrakcji mają swoje reprezentacje sumujące się w niezmienną reprezentację danego obiektu (Rys. 3.11). Kot ma głowę, ogon, łapki, wąsiki – nie jest istotne, czy głowa jest z lewej strony zdjęcia, a ogon z prawej, czy odwrotnie – nadal jest to kot (tyle, że ze zmianą występowania wzorców w czasie).

[64] Trzeba tu wspomnieć, że obraz nie jest w żaden sposób modyfikowany przed poddaniem go analizie – nie ma ani filtrowania, ani skalowania, ani obracania, ani innych podobnych operacji.

Sieci HTM są dziś testowane w wielu rozmaitych zadaniach. Wysokiej aktywności naukowców i inżynierów sprzyja korzystny sposób licencjonowania rozwiązań wypracowanych przez firmę Numenta – bez opłat dla zastosowań w nauce. Dzięki temu wielu próbuje stosować sieci HTM między innymi wszędzie tam, gdzie sieci neuronowe nie dają spodziewanych wyników. Są nawet obszary, w których sieci HTM już zdobyły swoją pozycję, a od roku 2010 zaczęły się pojawiać rozwiązania komercyjne korzystające z tej technologii. Do wspomnianych zagadnień należą [90]: predykcja zachowania (kliknięć) internautów (Forbes.com), analiza, diagnoza i zarządzanie sieciami energetycznymi (EDSA), systemy wizyjne monitoringu rozróżniające sylwetkę ludzką od innych ruchomych obiektów (Vitamin D). Kolejnymi rozwijanymi cywilnymi zastosowaniami technologii są: wykrywanie oszustw finansowych wśród elektronicznych transakcji płatniczych, analiza i predykcja obciążenia infrastruktury teleinformatycznej, analiza obrazów w medycynie.

Zastosowania sieci HTM w systemach wizyjnych

Mimo tak olbrzymiej różnorodności zastosowań sieci HTM, każdy z zainteresowanych rozpoczynał przygodę z HTM od przykładów dołączonych do platformy NuPIC, gdzie najcenniejszym przykładem była aplikacja rozpoznająca obrazy. Inżynierowie firmy Numenta przy każdej okazji zastrzegali, że sieci HTM nie służą jedynie zadaniom wizji, a przykład „Pictures" [89] jedynie pokazuje zasadę działania technologii. Mimo to wielu programistów pozostało przy zagadnieniu rozpoznawania wizji, tak jak firma „Vitamin D" [146], oferująca komercyjnie pierwszy system wizyjny rozróżniający, dzięki sieciom HTM, sylwetki ludzi od innych ruchomych obiektów w kolorowym strumieniu wideo, w czasie rzeczywistym.

W przykładzie „Pictures", jak również w dokumentacji wizyjnych sensorów (węzłów wejściowych obrazu dla sieci HTM) opracowanej przez Numenta, jasno i wyraźnie widać podejście[65] wywodzące się od *Active Vision*. Ponieważ sieci HTM z założenia wymagają, aby wzorzec prezentowany sensorom był zmienny w czasie, zmienność tą uzyskuje się poprzez przesuwanie sensorów wzdłuż obrazu wejściowego – bliźniaczo podobnie do jednego ze znaczeń terminu AV, gdzie obraz wejściowy był skanowany pojedynczym blokiem w celu znalezienia określonego detalu.

[65] Sensory, będące kilkunastoelementowymi macierzami pikseli, otrzymują zmienny w czasie przestrzenny wzorzec, który powoduje (lub nie) pobudzanie wyjść niektórych z nich. Z kombinacji pobudzeń następna warstwa buduje abstrakcyjną reprezentację na kolejnym poziomie.

Pomimo ogromnych wysiłków firmy Numenta, aby całokształt funkcjonowania technologii sieci HTM był maksymalnie zbliżony do sposobu funkcjonowania kory nowej, wszystkie systemy wizyjne budowane[66] dziś w oparciu o sieci HTM korzystają z opisanego wcześniej podejścia.

Niniejsza praca wychodzi naprzeciw potrzebie zwiększenia wydajności i skuteczności systemów wizyjnych budowanych z użyciem sieci HTM. Co prawda platforma NuPIC potrafi obsługiwać sieci HTM w środowisku klastrowym, jednak z powodu opisanych w rozdziale 1.3 problemów z czasem komunikacji zadania związane z kolorowym obrazem lub jakimkolwiek strumieniem wideo były do tej pory nieimplementowalne[67]. System wizyjny robota mobilnego przeznaczonego interakcji z człowiekiem musi oferować o wiele większe możliwości, rozróżniać więcej obiektów i w krótszym czasie. Nie da się tego osiągnąć bez modyfikacji sposobu pozyskiwania obrazu przez sensory. Gdyby sensory nie skanowały całego obrazu, lecz jedynie fragmenty – okolice punktu fiksacji – wydajność[68] systemu mogłaby znacznie wzrosnąć. Aby tak się stało, należałoby zastosować reprezentację obrazu przedstawioną w rozdziale 3.2.2 oraz, co ważniejsze, zaimplementować w algorytmie sprzężenie zwrotne opisane w rozdziałach 1.4 oraz 3.6.

Znaczenie sieci HTM dla niniejszych rozważań

Bezpośrednim celem niniejszej pracy było wskazanie sposobu projektowania i implementowania takiego systemu wizyjnego robota, który będzie umożliwiał przyłączenie do algorytmu wnioskowania zaimplementowanego w środowisku rozproszonym. Zastosowanie sieci HTM jako architektury emergentnej stanowiącej „inteligencję" robota jest rozwiązaniem najbardziej intuicyjnie pasującym do charakteru danych, przepływów danych oraz również (podobnie jak proponowany POI) z uwagi na biologiczny pierwowzór.

[66] istniejące jak również dopiero tworzone systemy wizyjne

[67] Mowa tu o zadaniach związanych z funkcjonowaniem przeciętnego systemu wizyjnego – rozpoznawanie kilku obiektów, uczenie nowych obiektów, prosta interakcja. Nie dotyczy prostych zastosowań, z których najbardziej zaawansowane to aplikacja „Vitamin D Video" firmy „Vitamin D", która potrafi jedynie wykryć sylwetkę człowieka (wspomagając się wygaszaniem tła, detekcją ruchu oraz śledzeniem w celu odciążenia procesora), co stanowi zdecydowanie zbyt skromne możliwości jak na system wizyjny robota mobilnego. Analizując obciążenie procesora podczas pracy tej aplikacji z jedną kamerą, można dojść do wniosku, że rozróżnienie dwóch obiektów byłoby trudne do zaimplementowania, a kilku – niemożliwe.

[68] Wydajność nie tylko w sensie czasu przetwarzania i reakcji, ale nawet w sensie poprawności rozpoznania – algorytm decyduje, które fragmenty obrazu należy przeanalizować, a pozostałe fragmenty nie wpływają znacząco na wynik rozpoznawania (nie zakłócają).

Niestety, z powodu ograniczonych ram czasowych prac badawczych, pełna implementacja algorytmu wnioskującego (wraz z jednym z zaimplementowanych eksperymentalnych systemów wizyjnych) nie było możliwe. Integrację zaimplementowanego systemu wizyjnego z rozproszonymi sieciami HTM zaplanowano jako pierwsze zadanie badawcze po ukończeniu niniejszej pracy.

3.4. Przetwarzanie rozproszone

Algorytmy współbieżne i rozproszone, pomimo swoich ograniczeń, są dziś ochoczo stosowane w niezliczonej liczbie złożonych problemów obliczeniowych. Ich popularność wynika bezpośrednio z zalet ich stosowania:

- umożliwiają rozwiązanie zadań zbyt dużych/złożonych dla maszyny sekwencyjnej,
- umożliwiają znaczne skrócenie czasu przetwarzania poprzez równoczesne wykorzystanie wielu procesorów/węzłów,
- umożliwiają zwiększenie precyzji obliczeń przy tym samym czasie obliczeń,
- umożliwiają lepsze wykorzystanie mocy komputerów wieloprocesorowych i procesorów wielordzeniowych,
- umożliwiają zwiększenie niezawodności.

Algorytmy współbieżne mają nieoceniony udział w dzisiejszym postępie cywilizacyjnym, umożliwiają bowiem przeprowadzanie badań (symulacji/obliczeń) zagadnień tak złożonych, że ich przeprowadzenie bez współbieżności jest bardzo czasochłonne lub nawet niemożliwe. Niestety nie tylko problem zrównoleglania algorytmów sekwencyjnych stanowi nierozłączną część każdego problemu obliczeniowego, lecz co więcej nie wszystkie istniejące algorytmy da się zamienić na bardziej efektywną od sekwencyjnej postać współbieżną. Istnieje wiele czynników wpływających na opłacalność i wydajność obliczeniową algorytmu równoległego. Najistotniejszymi z nich dla niniejszej pracy są: ziarnistość zadań, topologia oraz czas komunikacji.

Podstawowe miary efektywności zrównoleglania

Zaprojektowanie oraz zaimplementowanie algorytmu współbieżnego każdorazowo wiąże się z dogłębną analizą natury rozwiązywanego problemu obliczeniowego oraz precyzyjnym określeniem wszystkich aspektów Inżynierii Oprogramowania danego

algorytmu (klas, diagramów, zależności, komunikacji, m.in.). Niezbędne jest również określenie rodzaju zastosowanego zrównoleglenia:

- zrównoleglenie procesowe (zrównoleglenie kodu)
- zrównoleglenie danych

W ramach równoległości procesowej wyróżnia się [124]:

- równoległość funkcjonalną (przypisanie odmiennych zadań obliczeniowych poszczególnym węzłom)
- równoległość geometryczną (przypisanie odmiennego fragmentu kodu zadania obliczeniowego, m.in. w zadaniach związanych z przemieszczaniem mas powietrza, teorią pola, m.in.)
- równoległość algorytmiczna (przypisanie jak najmniejszych zadań)

Zadania obliczeniowe stawiane architekturom równoległym z założenia nie są zadaniami trywialnymi (wówczas zbędne byłoby stosowanie zrównoleglenia), dlatego fazę[69] projektowania należy przeprowadzić z nadzwyczajną dbałością o optymalność[70] kodu.

Z tego powodu niezwykle istotnym zagadnieniem, nierozerwalnie związanym z algorytmami współbieżnymi, jest metodyka określania efektywności kodu równoległego. W przypadku tworzenia kodu współbieżnego na podstawie istniejącego rozwiązania/ /algorytmu sekwencyjnego (czyli tzw. zrównoleglania kodu) niezbędne jest określenie zysku/strat wynikających ze zrównoleglenia. W tym celu stosuje się tzw. miary efektywności zrównoleglenia.

Jako podstawowe miary efektywności zrównoleglenia literatura [56] podaje m.in.:

- współczynnik przyspieszenia oraz wydajność (względny współczynnik przyspieszenia),
- skalowalność i sprawność

Współczynnik przyspieszenia (równanie 3) określa się na podstawie czasu wykonania danego algorytmu dla jednego procesora (rdzenia/węzła) oraz czasu wykonania tego samego algorytmu w implementacji równoległej (wieloprocesorowej):

$$S(n,p) = \frac{T(n,1)}{T(n,p)} \tag{3}$$

gdzie S – współczynnik przyspieszenia;
T – czas wykonania zadania;

[69] Terminu użyto w znaczeniu znanym z Inżynierii Oprogramowania, dotyczy jednej z faz cyklu życia oprogramowania.

[70] Kod optymalny – poprawny i bezpieczny kod, o najmniejszej możliwej złożoności obliczeniowej, wykonujący dane zadanie w możliwie najkrótszym czasie.

n – wielkość zadania;
p – ilość procesów;

Również przy użyciu współczynnika przyspieszenia przedstawia się zależność zwaną Prawem Amdahla (równanie 4), które mówi o maksymalnym możliwym do osiągnięcia przyspieszeniu obliczeń przy przyspieszeniu jedynie części algorytmów danej aplikacji. Prawo Amdahla stosowane jest przy określaniu teoretycznego przyspieszenia algorytmów zrównoleglanych – gdzie za część przyspieszaną rozumie się część współbieżną algorytmów, w przeciwieństwie do części sekwencyjnej. Z Prawa Amdahla (4) wynika, że maksymalne teoretyczne przyspieszenie algorytmu ograniczone jest do wartości będącej odwrotnością udziału części sekwencyjnej w algorytmie [56].

$$S(n,p) \xrightarrow[p\to\infty]{} \frac{1}{\beta(n,1)} \tag{4}$$

gdzie S – współczynnik przyspieszenia;
β – udział czasu wykonania części sekwencyjnej zadania;

Literatura podaje sposób „osłabiania Prawa Amdahla" – poprzez stosowanie obliczeń asynchronicznych bez sztywnego podziału algorytmu na część sekwencyjną i równoległą [56]. Rozwiązanie prezentowane w niniejszej pracy bazuje na zrównolegleniu funkcjonalnym algorytmu, z asynchronicznym przesyłaniem danych i asynchroniczną pracą węzłów.

Inną miarą efektywności zrównoleglania jest skalowalność, określająca zależność sprawności algorytmu programu równoległego od liczby użytych procesorów (węzłów). Skalowalność definiowana jest w literaturze [56] jako własność systemu polegająca na zachowywaniu tej samej sprawności dla różnej liczby procesorów. Sprawność natomiast może być wyznaczona ze wzoru (5):

$$\eta(n,p) = \frac{1}{1+\frac{h(n,p)}{w(n)}} \tag{5}$$

gdzie η – sprawność programu równoległego;
w – liczba operacji związanych z obliczeniami;
h – narzut na komunikację;

Ogromny wpływ na skalowalność ma czas komunikacji, celowe jest zatem poszukiwanie sposobów jego zmniejszania, co również jest realizowane w niniejszej pracy.

Popularne narzędzia przetwarzania rozproszonego

W ostatniej dekadzie wzrost popularności maszyn równoległych zaskoczył nawet najbardziej optymistycznych zwolenników tego rodzaju architektur. Przed rokiem 2004 termin „maszyny równoległe" kojarzony był z wieloprocesorowymi serwerami, sieciami maszyn (klastry, gridy) i innymi specjalistycznymi architekturami. Natomiast od roku 2004, w którym producenci procesorów dotarli do bariery 4GHz [22], do grona maszyn równoległych dołączyły nawet najtańsze komputery osobiste. Pomimo od dawna istniejących rozszerzeń strumieniowych, technologia HT (*ang. HyperThreading*), uznawana za pierwszy zdecydowany krok w kierunku procesorów wielordzeniowych, nie dawała zadowalającego przyspieszenia. Dopiero pierwsze procesory wielordzeniowe[71] dały użytkownikom komfort pracy wcześniej zarezerwowany tylko dla użytkowników o wiele droższych komputerów wieloprocesorowych.

Niezwykle istotnym sposobem znacznego przyspieszenia szeroko rozumianych obliczeń macierzowych jest zastosowanie specjalistycznych kart GPGPU (*ang. General Purpose computing on Graphics Processing Units*). Stosowanie coraz potężniejszych procesorów GPU do zastosowań innych niż przetwarzanie obrazu stało się tematem szczególnie atrakcyjnym po publikacji [40] z 2005 roku, choć termin GPGPU powstał nieco wcześniej, a koncepcja – znacznie wcześniej: [149][72]. W roku 2007 firma NVIDIA rozpoczęła produkcję pierwszych dedykowanych[73] kart GPGPU serii Tesla. Obecna najnowsza[74] wersja – Tesla C2070 – oferuje moc obliczeniową do[75] 1.03 T FLOPS [94] (*ang. Floating point Operations Per Second*). Według aktualnego rankingu Top500, trzy spośród pierwszej czwórki najpotężniejszych superkomputerów na świecie, zbudowano w oparciu o karty GPGPU NVIDIA Tesla [143].

[71] Procesory wielordzeniowe (*ang. multicore*) – układy scalone zawierające dwa lub więcej pojedynczych procesorów oraz wspólny interfejs do magistrali systemowej i przełącznicę krzyżową, czasami wspólną pamięć drugiego poziomu [56]

[72] Publikację przytoczono jako przykład - profesor Wiatr w swojej monografii nie opisuje co prawda procesorów GPU, lecz „specjalizowane procesory sprzętowe o architekturze potokowej" (FPGA – *ang. Field Programmable Gate Arrays*). Programowanie dzisiejszych procesorów nierozerwalnie wiąże się z przetwarzaniem potokowym, dlatego można się dopatrywać wspólnej platformy łączącej te dwa nurty – przetwarzania potokowego wyspecjalizowanymi procesorami, mającego na celu przyspieszenie obliczeń. Przetwarzanie potokowe jest koncepcją o wiele starszą: [3].

[73] Karty zawierające szybki procesor graficzny GPU, szybką pamięć (oraz pozostałe niezbędne komponenty), nie posiadające wyjścia na monitor – służące z założenia wyłącznie do celów obliczeniowych.

[74] czerwiec 2010r.

[75] dla operacji zmiennoprzecinkowych pojedynczej precyzji

Oprócz GPGPU istnieje jeszcze wiele procesorów o innych architekturach niż te znane z komputerów osobistych. Za najbardziej reprezentatywny przykład może posłużyć procesor CELL, użyty (w wersji[76] PowerXCell 8i) w serii „Roadrunner", z których jeden był najszybszym superkomputerem na świecie w latach 2008-2009 (i jednocześnie najbardziej ekologicznym[77]) według tej samej listy [143].

Pomimo rosnącej popularności kart GPGPU, zdecydowaną większość wykorzystywanych maszyn równoległych stanowią nadal superkomputery nie wykorzystujące GPGPU oraz wirtualne (sieciowe) maszyny równoległe, czyli popularne klastry i gridy. Podział superkomputerów ze względu na ich architekturę w ciekawy i przystępny sposób opisuje [56]. Klastry i gridy są pojęciami do siebie zbliżonymi, oznaczają grupę niezależnych komputerów, połączonych siecią komputerową, z uruchomionym programem (biblioteką lub demonem) umożliwiającym wykorzystanie mocy obliczeniowej wszystkich komputerów w jednej aplikacji. To co różni klastry od gridów to zasięg (sieć) oraz architektura komputerów. Gridy łączą komputery nawet w skali sieci globalnej, połączone komputery mogą mieć różne systemy/architektury. Gridy, zwane też komputerami globalnymi, są siecią połączonych heterogenicznych maszyn, podczas gdy klastry składają się z maszyn będących w niewielkiej odległości, zazwyczaj homogenicznych.

Przetwarzanie rozproszone w niniejszej pracy

Eksperymenty implementacyjne, będące uzupełnieniem niniejszej pracy, przeprowadzano z wykorzystaniem klastrów komputerowych. Klastry komputerowe są rozwiązaniem optymalnym zarówno pod względem ekonomicznym jak i w kontekście powtarzalności (i rozwoju) przeprowadzanych badań i opracowywanych implementacji.

Szczegóły techniczne użytych narzędzi przetwarzania rozproszonego, jak również sprzętu komputerowego wchodzącego w ich skład, przedstawiono w rozdziale 4.1.

Najbardziej znaczące spośród eksperymentów implementacyjnych przeprowadzono z użyciem interfejsu MPI (*ang. Message Passing Interface*) [33], wykorzystując bibliotekę LAM/MPI [67].

[76] Asymetryczny 9-rdzieniowy procesor w architekturze Power, zawierający osiem rdzeni [56] wektorowych AltiVec

[77] Parametr ten wyznaczany jest na podstawie stosunku MFLOPS/Wat

MPI jest to specyfikacja interfejsu dla tworzonych bibliotek MPI, opisująca sposób przekazywania komunikatów [78] pomiędzy procesami w aplikacjach równoległych. Standard MPI definiuje m.in. sposoby komunikacji międzyprocesowej, synchronizacji oraz tworzenia wirtualnych topologii. Biblioteką MPI określa się każdą spośród implementacji interfejsu MPI, poczynioną wg specyfikacji. Najbardziej popularnymi bibliotekami MPI są MPICH2 i LAM/MPI.

Część eksperymentów przeprowadzono w środowisku Xgrid, stanowiącego narzędzie systemowe systemu operacyjnego MacOsX, służące tworzeniu klastrów i gridów na platformach Apple Macintosh. Konfiguracja i użytkowanie klastra Xgrid są o wiele mniej złożone niż w przypadku LAM/MPI, jednak wytworzenie aplikacji wykorzystującej klaster może się okazać trudniejsze[78].

3.5. Topologie środowisk rozproszonych

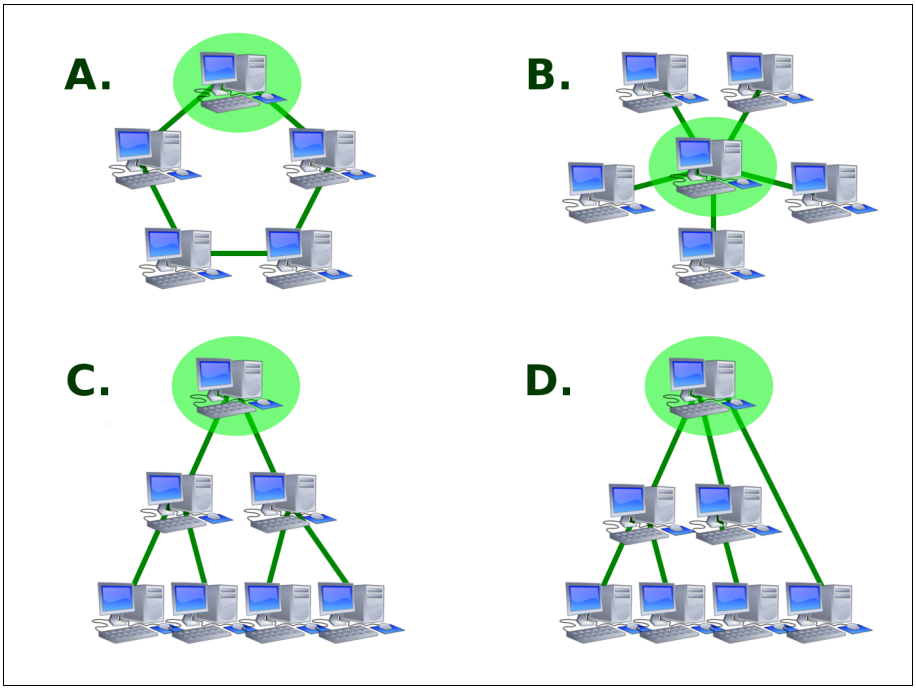
Rozważania przedstawione w niniejszym rozdziale ograniczono do środowisk klastrowych, ponieważ bezpośrednio dotyczą zastosowanego rozwiązania. Należy jednak zaznaczyć, że klastry nie stanowią ani jedynego, ani nawet typowego środowiska rozproszonego dla złożonych systemów wizyjnych. Od wielu lat z powodzeniem budowane są systemy stosujące architektury wieloprocesorowe, tak jak system CESARO-2, opisywany w [133], jednak systemy te albo przetwarzają obraz monochromatyczny o niskiej rozdzielczości, albo dysponują ograniczonymi możliwościami rozpoznawania obrazów.

System wizyjny, który mógłby być zastosowany w tandemie z architekturą emergentną do interakcji z człowiekiem, powinien oferować o wiele lepsze parametry przetwarzanego obrazu, co prowadzi do wspomnianego na wstępie pracy ograniczenia.

[78] Systemowe środowisko programistyczne – XCode – umożliwia tworzenie aplikacji w wielu językach programowania, jednak decydując się na języki najlepiej udokumentowane i generujące najbardziej efektywny kod wybór zawęża się do języków Carbon i Cocoa, które uważane są (wśród użytkowników innych platform niż Apple Macintosh) za dość egzotyczne. Najprostsze przykłady wykorzystujące XGrid zawierają kilka-kilkanaście plików i ponad tysiąc linii kodu, podczas gdy najprostsze programy MPI można przedstawić za pomocą kilku linii. Każda aplikacja dla XGrid posiada natomiast zawsze pełne możliwości w zakresie kontroli stanu (pracy) węzłów obliczeniowych, czy też bezpieczeństwa połączenia (łącznie z logowaniem szyfrowanym i/lub Kerberos).

3.5.1. Typowe topologie

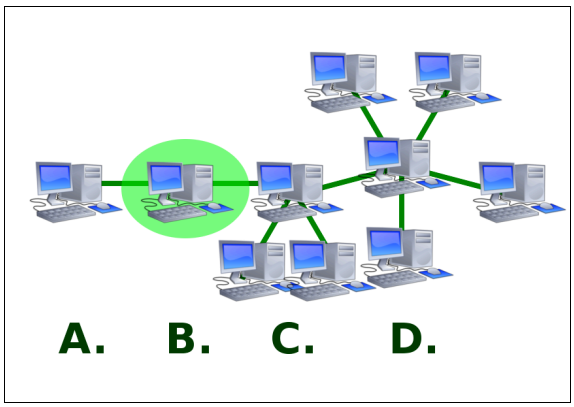
Właściwy dobór topologii klastra (Rys. 3.12) jest kluczowym zadaniem projektanta każdego rozproszonego systemu przetwarzania danych. Topologia klastra powinna w naturalny sposób wynikać ze specyfiki zaproponowanego algorytmu oraz zadania obliczeniowego, jakie ma być realizowane. Największy wpływ na ostateczny kształt algorytmu aplikacji klastrowej ma zagadnienie zrównoleglalności implementowanego algorytmu przetwarzania. Niemal każde zadanie przetwarzania danych posiada fragmenty, które można łatwo zrównoleglić, jak również fragmenty, które są z natury sekwencyjne. Nawet w zadaniach dobrze poddających się zrównolegleniu istnieją etapy komplikujące zrównoleglanie algorytmu, takie jak dystrybucja danych, synchronizacja działań i akwizycja wyników. Kolejnym, często krytycznym utrudnieniem, występującym również w rozwiązywanym problemie przetwarzania danych wizualnych, jest problem zbyt dużego czasu komunikacji. O ile w algorytmach sekwencyjnych parametr ten mógł być pomijalny, to w algorytmach przetwarzania rozproszonego często decyduje o wykonalności danego algorytmu.



Rys. 3.12. Typowe topologie klastrów. Kolorem zielonym zaznaczono węzeł nadrzędny.

Każda topologia jest kompromisem pomiędzy wizją projektanta a możliwościami technologii, niemniej jednak, prawidłowym przygotowaniem do rozwiązywanego problemu i dogłębną analizą komunikacji i relacji między projektowanymi węzłami można w znacznym

stopniu wpłynąć na stopień użyteczności finalnej implementacji. To właśnie topologia klastra, dla prawidłowo sformułowanego problemu, jest pierwszym zagadnieniem, jakie należy ustalić przy zrównoleglaniu algorytmu. Typowe topologie klastrów komputerowych przedstawiono na rysunku Rys. 3.12. Jeżeli algorytm łatwo jest podzielić (bez względu na ziarnistość) na niezależne operacje wymagające jedynie wysłania danych węzłom i odebrania wyników, najlepszym rozwiązaniem będzie topologia gwiazdy (Rys. 3.12 B). Jeżeli natomiast algorytm składa się z sekwencji rozdzielnych operacji, można rozważyć zastosowanie topologii pierścienia[79] (Rys. 3.12 A). Niestety znakomitej większości problemów obliczeniowych świata rzeczywistego nie da się sprowadzić do któregokolwiek z powyższych idealistycznych przypadków. Wynika to zazwyczaj ze specyfiki danego zadania, a dokładniej z niejednorodności problematyki przetwarzania w obrębie każdego fragmentu danego algorytmu. Gdy żadna topologia spośród klasycznych, prostych do zaimplementowania, nie spełnia oczekiwań, nie pozostaje nic innego jak zaproponować topologię specjalnie zaprojektowaną na potrzeby danego algorytmu. Przykład tego rodzaju topologii (łączącej cechy topologii typowych i dlatego zwanej hybrydową) zamieszczono na rysunkach Rys. 3.12D oraz Rys. 3.13.



Rys. 3.13. Przykład topologii hybrydowej klastra – A. Węzeł akwizycji danych; B. Węzeł nadrzędny koordynujący pracę klastra (*master node*); C. Podsystem wstępnego przetwarzania; D. Podsystem przetwarzania danych. Kolorem zielonym zaznaczono węzeł nadrzędny.

Rozsądne zaprojektowanie topologii klastra (do danego zagadnienia) wiąże się z potrzebą dekompozycji („rozbicia") całego algorytmu na fragmenty o odpowiedniej ziarnistości. Ziarnistość natomiast powinna być dobrana z uwzględnieniem zysku czasu obliczeniowego ze zrównoleglenia (podziału na drobniejsze fragmenty a więc lepiej

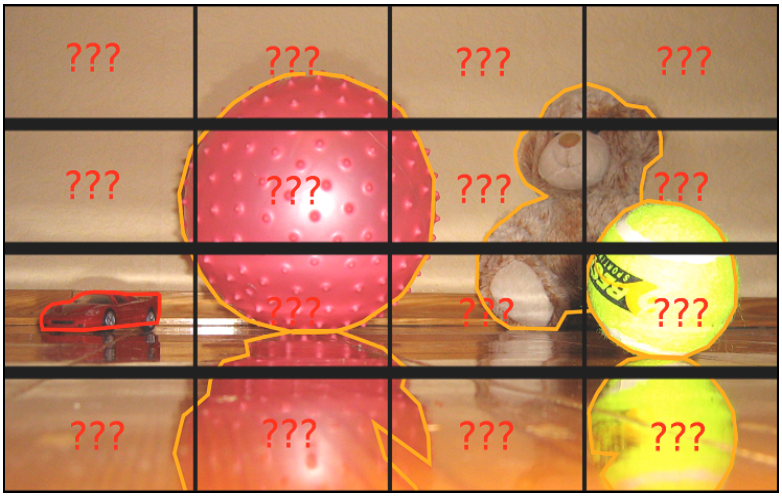
[79] W topologii pierścienia każdy węzeł zajmuje się określonym fragmentem algorytmu sekwencyjnego, przesyłając częściowo przetworzone dane kolejnemu węzłowi. Węzły mogą oczywiście pracować równocześnie nad różnymi paczkami danych.

wykorzystujące czas pracy węzłów) jak również strat spowodowanych czasem komunikacji (im więcej fragmentów, tym więcej operacji synchronizacji i przesyłania danych). Zadania łatwe do zrównoleglenia przy użyciu typowych topologii, jak na przykład obliczenia macierzowe lub wyszukiwanie danych, można bez trudu przeanalizować pod kątem doboru odpowiedniej (optymalnej[80]) ziarnistości. Natomiast zadania wymagające topologii hybrydowych najlepiej jest uprzednio rozplanować pod kątem właściwej (optymalnej[81]) komunikacji. Projektowanie algorytmu przetwarzania rozproszonego wymagającego topologii hybrydowej zazwyczaj prowadzi do zmiany sposobu dekompozycji algorytmu na tzw. dekompozycję funkcjonalną. W algorytmie z zaimplementowanym funkcjonalnym podziałem zadań klaster musi mieć ściśle określoną (przez ww. algorytm) topologię – każdy węzeł klastra zajmuje ściśle określonym zadaniem, zwykle różnym od pozostałych węzłów. Podział funkcjonalny jest również możliwy w topologiach gwiazdy czy drzewiastej, jednak najczęściej spotykany jest w topologiach hybrydowych. Algorytmy stosujące funkcjonalny podział zadań są o wiele bardziej złożone, trudniejsze w implementacji, ale bardzo często stanowią jedyny sposób rozwiązania problemu w czasie krótszym niż rozwiązywanie sekwencyjne. Doskonałym tego przykładem jest opisywany w niniejszej pracy system. Rozdział danych w topologii gwiazdy (Rys. 3.12 B) lub drzewiastej (Rys. 3.12 C) teoretycznie jest możliwy[82], ale wiąże się z potężnymi komplikacjami uniemożliwiającymi wręcz zastosowania klastra. Obok szeregu komplikacji natury technicznej, największym problemem są błędy rozpoznawania obiektów leżących w obrębie więcej niż jednego rejonu (Rys. 3.14).

[80] Optymalnej w kontekście minimalizacji sumy czasu komunikacji, obliczeń i ew. oczekiwania (synchronizacji).

[81] Optymalnej w kontekście minimalizacji czasu komunikacji.

[82] na przykład poprzez podział obrazu wejściowego na mniejsze fragmenty, tak jak przedstawiono to na rysunku Rys. 3.14



Rys. 3.14. Grafika komputerowa symbolizująca problem zrównoleglania zadań przetwarzania obrazu.[83]

Alternatywą może być zmiana koncepcji systemu wizyjnego: poszczególne węzły zajmują się nie fragmentem obrazu, lecz fragmentem wiedzy (każdy węzeł rozpoznaje inne obiekty). Pomimo kilku prób implementacji takich systemów, w literaturze nie udało się znaleźć potwierdzenia jakiegokolwiek sukcesu w tej materii. Brak sukcesów jest wynikiem zgodnym z oczekiwaniami, ponieważ w tym podejściu do węzłów rozsyłany jest „pełny" obraz wejściowy, co powoduje poważne problemy[84] z komunikacją między węzłami[85].

3.5.2. Topologia rozważana w pracy

W niniejszej pracy zaproponowano innowacyjne rozwiązanie korzystające z opisanych technologii (DWT, POI, elekcja POI), które nie posiada wspomnianej wady – obraz w proponowanej reprezentacji nie powoduje paraliżu komunikacji między węzłami, a pomimo stratnej redukcji i kompresji danych wizualnych nadal istnieje sposób (opisywany w rozdziale 3) na ich pełne wykorzystanie w procesie rozumienia wizji.

Na potrzeby weryfikacji implementacyjnej zdecydowano się na topologię hybrydową, wzorując się częściowo na biologicznym procesie rozumienia wizji, jaki ma człowiek. Jakość i wydajność użytej topologii nie jest przedmiotem badań, a użyta topologia może być

[83] Przy zastosowaniu zrównoleglenia przedstawionego na Rys. 1.2 w tym przypadku (podziału obrazu na segmenty przetwarzane w osobnych węzłach) jedynie rozpoznanie autka mogłoby zostać przeprowadzone bezproblemowo, rozpoznawanie pozostałych obiektów wymagałoby zaawansowanych mechanizmów synchronizacji (agregacji) wiedzy dot. fragmentów obiektów z poszczególnych węzłów klastra.

[84] Krytycznym parametrem jest czas komunikacji – samo rozesłanie obrazu do węzłów klastra zabiera tak dużo czasu, że nie ma żadnego zysku ze zrównoleglania algorytmu.

[85] W tym zakresie przeprowadzono wstępną analizę, wyniki dostępne w literaturze [103] [122].

zastąpiona przez dowolną inną. Są jednak powody, dla których zdecydowano się wybrać takie właśnie rozwiązanie:

- do wykrycia i analizy ruchu nie są potrzebne[86] informacje o kolorach,
- do rozpoznawania obiektów nie są niezbędne wszystkie kolejne klatki strumienia (można je pobierać asynchronicznie),
- do analizy obiektu nie jest niezbędne „przytrzymywanie" jednej klatki strumienia (sieci HTM działają dzięki stymulacji wzorcem czasowym – źródło zmienności wzorca przestrzennego nie musi pochodzić z pre-programowanego skanowania obrazu tak jak proponuje się m.in. w pracy [122], lecz może pochodzić z rzeczywistej zmienności obrazu w czasie),
- analiza informacji specyficznych dla stereoskopii – macierz głębokości – może być przetwarzana w osobnym węźle,
- mechanizmy rozumienia i pamięci mogą być zaimplementowane w innych węzłach niż przetwarzania wizji (sieci HTM nie przesyłają wewnątrz swojej struktury fragmentów obrazu, lecz stany pobudzeń grup reprezentujących niezmienne reprezentacje, tak więc ta część topologii nie będzie podlegać wspomnianym ograniczeniom komunikacji).

Z tych paru przemyśleń oraz z rozważań nad działaniem ludzkiego wzroku i specyfiką sieci HTM zaproponowano pierwowzór topologii (Rys. 3.15 1) użytej w opisanym w dalszej części pracy eksperymencie. Oto podstawowe założenia i cechy pierwszej (wczesnej) wersji tej topologii:

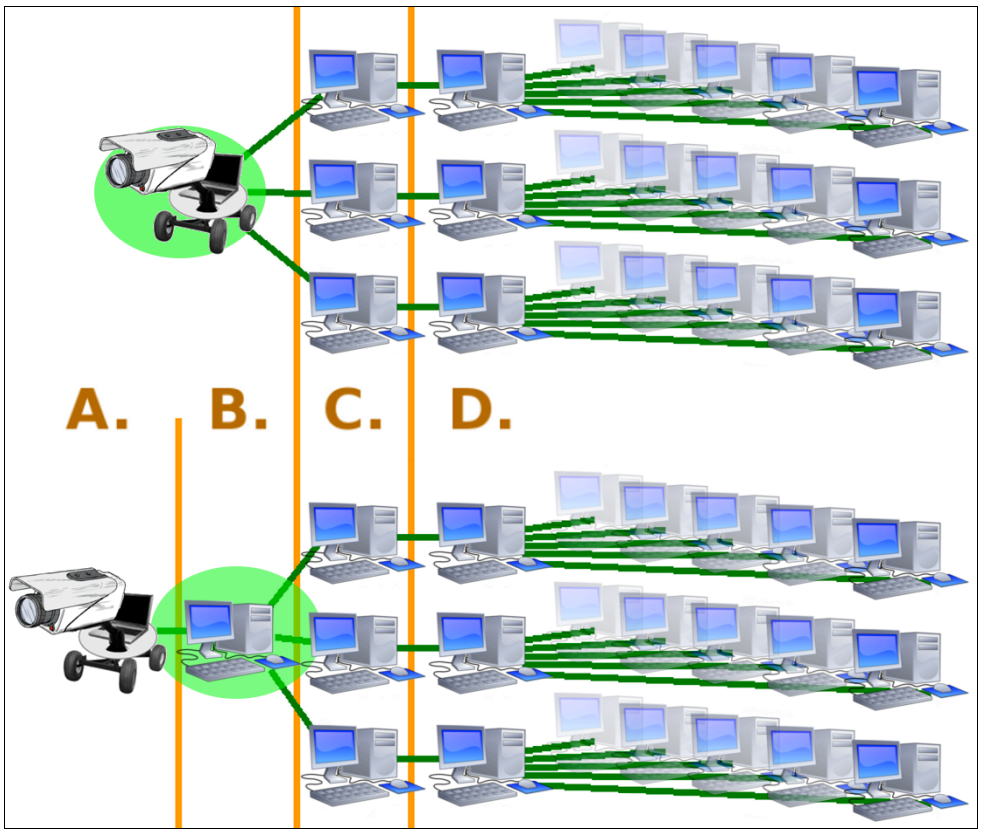
- Komputer *embedded* robota mobilnego wyposażony w kamerę stanowiącą wejście.
- Ten sam komputer jest jednocześnie[87] węzłem klastra. Ponieważ klaster dedykowany jest temu jednemu robotowi, nie ma potrzeby zastanawiać się nad metodą dynamicznego dołączania i rozłączania robotów (należałoby wówczas raczej dołączać je do węzła wejściowego zamiast integrować z nim).
- Co więcej ten sam komputer jest węzłem nadrzędnym klastra. Dopóki klaster dedykowany jest jednemu robotowi, taka sytuacja nie stwarza zagrożenia, jest natomiast dużym ułatwieniem dla programisty. Planując rozwój systemu rozumienia i wnioskowania należy rozważyć przeniesienie funkcji węzła nadrzędnego do innego węzła fizycznego (Rys. 3.15 2).
- Węzeł wejściowy przesyła klatki obrazu w postaci zgodnej z proponowaną w pracy reprezentacją (POI) do kilku węzłów, z uwzględnieniem wspomnianych wcześniej założeń, ograniczając ilość przesyłanych klatek do minimum[88].

[86] przy odpowiednim przetworzeniu wstępnym

[87] W toku prac implementacyjnych porzucono tą koncepcję – czas procesora komputera *embedded* jest zbyt cenny, aby zajmować się stosunkowo czasochłonną konwersją klatek oraz koordynacją pracy klastra. Zadania te przerzucono na osobny komputer, zgodnie z drugą częścią rysunku Rys. 3.15 oraz opisami zawartymi w rozdziałach 4.3 i 4.5

[88] Sposób wspomnianego „ograniczania danych do minimum" opisano w dalszej części pracy.

- Węzły otrzymujące obraz z węzła wejściowego analizują otrzymane dane z zachowaniem podziału funkcjonalnego – osobno przetwarzany jest ruch, osobno kolory, osobno kształty, m.in.
- Wspomniane węzły dysponują swoimi wejściami sieci HTM (rozdzielnych sieci – osobnej dla każdego węzła lub (lepiej) jednej wspólnej[89] sieci HTM).
- Sieć/sieci HTM korzysta(ją) ze wszystkich pozostałych węzłów klastra. Pomimo sztywnej topologii z podziałem funkcjonalnym zapewni to dobre skalowanie algorytmu, pozwalając na wykorzystanie każdej[90] dodatkowej mocy obliczeniowej.
- Węzły otrzymujące obraz z węzła wejściowego mają możliwość odesłania w sprzężeniu zwrotnym węzłowi wejściowemu informacji dot. preferowanej następnej lokalizacji punktu fiksacji POI (opisane w rozdziale 3.6).
- Węzeł wejściowy przy wysyłaniu kolejnej klatki obrazu uwzględni (w drodze opisanej w rozdziale 3.6 elekcji) modyfikację współrzędnych punktu.



Rys. 3.15. Przykładowa topologia do użycia z proponowanym rozwiązaniem (POI). A. Węzeł akwizycji danych (tu następuje konwersja z użyciem POI); B. Węzeł nadrzędny koordynujący pracę pozostałych węzłów klastra; C. Podsystem przetwarzania wizji z rozdziałem funkcjonalnym; D. Podsystem rozumienia i wnioskowania (sieci HTM). U góry (1) – komputer robota jest węzłem nadrzędnym klastra, u dołu (2) – komputer robota nie jest węzłem nadrzędnym klastra

[89] Rozważania nad sposobem uwspólnienia jednej rozproszonej sieci HTM pomiędzy wieloma węzłami *sensor node* są zagadnieniem na tyle złożonym, że można im poświęcić odrębną pracę.

[90] Z ograniczeniami nie większymi niż w innych aplikacjach rozproszonych.

Istnieje wiele sposobów na poprawienie topologii przedstawionej na rysunku Rys. 3.15, (niektóre dyskusyjne, niektóre oczywiste). Zdecydowano się na zaimplementowanie oraz przedstawienie w pracy właśnie takiej jej wersji, ponieważ dzięki swojej przejrzystości najlepiej przybliża koncepcję użycia klastra. Udowodniona w pracy teza[91] daje ogromne, nowe możliwości; autor nie ma możliwości przebadania wszystkich zagadnień pobocznych towarzyszących rozpatrywanej problematyce.

Topologia przedstawiona na rysunku Rys. 3.15 składa się z kilku rozdzielnych fragmentów:

A) Węzeł akwizycji danych – komputer *embedded* robota mobilnego. Najlepiej jest nie obciążać go zadaniami niezwiązanymi bezpośrednio z robotem, dlatego warto wydzielić osobną jednostkę na potrzeby komunikacji z klastrem (tak jak na rysunku Rys. 3.15_2).

B) Węzeł nadrzędny koordynujący pracę klastra – tzw. *master node*. Węzeł, który pełni kluczową dla algorytmu rolę – tu wykonywana jest konwersja obrazu wejściowego transformatą DWT przy użyciu technologii POI dla określonego punktu fiksacji. Węzeł przygotowuje kilka reprezentacji, w zależności od wymagań danego węzła z grupy „C" – węzeł rozpoznający ruch otrzymuje dane różnicowe sąsiednich klatek strumienia, węzeł rozpoznający określony kolor otrzymuje macierz pikseli zawierającą informacje tylko o tym kolorze, m.in. Dzięki tym zabiegom rozmiar każdej paczki danych jest akceptowalnie mały, a paczki mogą być dostarczane asynchronicznie, na żądanie. Zmiana trybu przesyłu danych na asynchroniczny daje dodatkową poprawę wydajności klastra (oraz zmniejszenie obciążenia komunikacji węzła wejściowego – kluczowego dla całego systemu). Węzły z grupy „C" wysyłają żądanie otrzymania kolejnej klatki, co dodatkowo likwiduje niebezpieczeństwo wystąpienia zapełnienia/ /przepełnienia bufora klatek oczekujących albo niebezpieczeństwo tylko częściowego analizowania podsyłanych danych.

C) Podsystem przetwarzania wizji z podziałem funkcjonalnym – grupa węzłów otrzymująca zadania przetwarzania z podziałem funkcjonalnym. Każdy węzeł otrzymuje inne dane[92] dotyczące obserwowanej sceny i analizuje je w innym kontekście – wykrywając i analizując ruch w scenie, wykrywając i analizując kolory

[91] patrz: strona 20.

[92] Alternatywnie, może otrzymywać te same dane, ale analizować je w innym kontekście.

lub określony kolor, wykrywając i analizując kontury, m.in.[93] Każdy z węzłów tej grupy dokonuje wstępnej analizy otrzymanej paczki i po konwersji na format optymalny dla wejść sieci HTM umieszczonych w grupie „D" przekazuje ją węzłom wejściowym sieci HTM. Dodatkowym zadaniem węzłów tej grupy jest określenie (we współpracy z siecią HTM) lokalizacji kolejnego punktu fiksacji i przesłanie tych współrzędnych w sprzężeniu zwrotnym do węzła konwertującego obraz wejściowy z wykorzystaniem POI. Liczba węzłów w tej grupie zależy od projektanta systemu i od przyjętej strategii podziału funkcjonalnego.

D) Podsystem rozumienia i wnioskowania (sieci HTM) – dla każdego[94] z węzłów grupy „C" istnieje osobna[95] sieć HTM, analizująca i wnioskująca w oparciu o dane „swojego" węzła z grupy „C". Intuicyjnym byłoby utworzyć sieć połączeń pomiędzy warstwami wszystkich tych sieci HTM[96], jednak na obecnym etapie rozwoju technologii i platformy sieci HTM jest to niemożliwe lub niezwykle złożone[97]. Można natomiast spróbować skomunikować ze sobą całe sieci HTM na etapie warstwy najwyższej (gdy sieć HTM ma już wynik rozpoznawania) – przykład takiej sytuacji opisano w rozdziale 3.3 (w części „HTM-temporal").

Ponieważ przedstawiona topologia, po osadzeniu w dzisiejszych realiach sprzętowych, nie daje gwarancji otrzymania wyniku rozpoznawania w przewidywalnym czasie, nie należy stosować tego rozwiązania jako jedynego sposobu analizy danych sensorycznych. W biologii znana jest reakcja zwana „odruchem bezwarunkowym" – podobnie należałoby zaprojektować „inteligencję" robota.

[93] Nie ma potrzeby całkowicie konsekwentnego wdrażania jedynie tych aspektów wizji, które są obecne w biologicznym pierwowzorze. Warto unikać rozszerzeń całkowicie sprzecznych z pierwowzorem, gdyż wówczas łatwiej będzie o algorytm rozumienia wizji współgrający z biologicznym odpowiednikiem (co powinno zaowocować lepszą interakcją między człowiekiem a maszyną). Nie trzeba jednak unikać takich rozszerzeń, które nie są sprzeczne z biologicznym sposobem widzenia, a jedynie stanowią jego ulepszenie (nie zmieniając sposobu akwizycji i obsługi danych) – jak na przykład zastosowanie (w formie uzupełnienia)

[94] Jedynie propozycja, możliwe, że lepsze wyniki da agregacja niektórych sieci.

[95] lub jedna wspólna, jak zaproponowano wcześniej

[96] aby umożliwić rozpoznawanie i pamiętanie obiektów złożonych z cech obserwowalnych przez różne węzły grupy „C", np. „czerwone ruchome gwiazdki"

[97] Nikomu jak na razie nie udało się łączyć warstwy rozdzielnych sieci HTM, ale technologia jest na tyle młoda, że nie znaczy to, że jest to niemożliwe.

3.6. Elekcja POI (sterowanie współrzędnymi punktu)

Dla osoby zajmującej się projektowaniem robotów, zagadnienie sterowania[98], w kontekście robota mobilnego, nie jest zagadnieniem jednoznacznym. Działanie podejmowane przez robota (lub jego wybrane komponenty/algorytmy) może być rozpatrywane jako wynikające z planowania na jednej z wielu płaszczyzn:

- planowanie[99] parametrów zadania,[100]
- planowanie parametrów trasy celem wykonania zadania,[101]
- planowanie parametrów silników celem przemieszczenia robota po trasie,[102]
- planowanie parametrów silników/położenia aktuatorów celem wykonania zadania,[103]
- planowanie parametrów silników/położenia niektórych sensorów celem modyfikacji ich wejścia,[104]
- planowanie parametrów położenia/lokalizacji punktu fiksacji systemu wizyjnego.[105]

Ostatni z wymienionych aspektów zawiera nowe[106] rozwiązanie zaproponowane przez autora niniejszej pracy.

[98] sterowania: robotem jako systemem / podsystemem robota / systemem wizyjnym / podsystemem systemu wizyjnego.

[99] Celowo użyto słowa „planowanie", ponieważ lepiej wyraża nacisk, jaki powinien być położony na kontekstową predykcję. Słowo „sterowanie" niestety często w potocznym rozumieniu jest spłycane do „bieżącej chwili wraz z historią", lub do „kolejnej chwili wraz z bieżącą i historią" w przypadku sterowania predykcyjnego. Inteligentne roboty przyszłości powinny nie tylko potrafić sterować swoim ruchem, sterować predykcyjnie – powinny potrafić planować.

[100] Algorytm planuje wykonanie zadania, czyli generuje taki graf przejść stanów, który umożliwi dokonania pożądanej zmiany w otoczeniu robota.

[101] Algorytm planuje trasę przemieszczenia robota, aby wykonać zaplanowane zadanie.

[102] Najbardziej znany w literaturze kontekst potocznego określenia „sterowanie robotem mobilnym".

[103] Najczęściej wykorzystywany w praktyce kontekst potocznego określenia „sterowanie robotem".

[104] Niektóre sensory wymagają nakierowania na analizowany wycinek rzeczywistości/ /otoczenia, na przykład laserowe systemy dalmierzowe (*ang. Laser Range Finder*) lub kamery.

[105] W odróżnieniu od poruszania kamerą (uchwytem kamery), w tym przypadku nie występuje fizyczny ruch. W biologicznym pierwowzorze występują co prawda sakkadowe ruchy mięśni, jednak radykalnie różnią się od świadomych ruchów gałkami ocznymi. Dlatego modyfikację punktu fiksacji w algorytmie POI również uznano za osobny aspekt regulacji.

[106] W literaturze, w niektórych dawnych publikacjach związanych z AV, można odnaleźć pewne nawiązania do potrzeby (lub nawet do algorytmu) przemieszczania obszaru, który dziś nazwalibyśmy ROI. Użycie takiego obszaru nie wynikało jednak z analogii między biologicznymi i cybernetycznymi systemami wizyjnymi, a jedynie z potrzeby rozdzielenia wstępnego przetwarzania od dokładnej analizy lub określenia obszaru działania algorytmów przetwarzających. Nie znaleziono w literaturze wzmianki o sterowaniu/planowaniu (sprzężeniu zwrotnym) położenia punktu fiksacji wzorowanym na ruchu sakkadowym gałki ocznej.

Na podstawie powyższego można wyszczególnić trzy najbardziej reprezentatywne i znaczące (dla pracy) grupy kontekstów sterowania robotem:

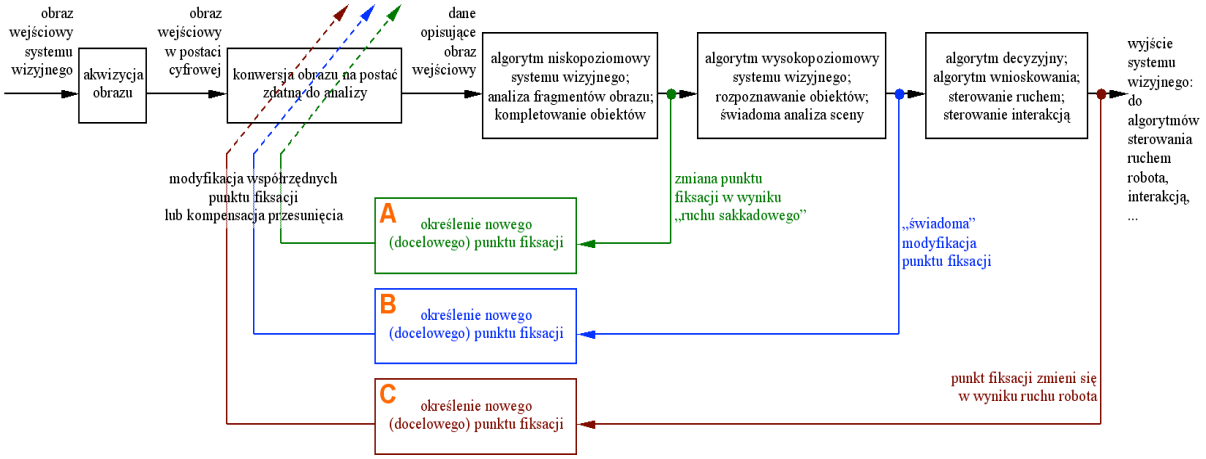
- ruch (przemieszczenie) robota[107],
- ruch (manipulowanie) uchwytem kamery,
- ruch[108] (zmiana współrzędnych na obrazie wejściowym) punktu fiksacji.

Wspomniane trzy konteksty ruchu, mimo pewnych powiązań, stanowią rozdzielne zagadnienia. W praktycznych implementacjach najczęściej robot musi się zatrzymać, aby móc manipulować uchwytem kamery – uwzględnienie obu przemieszczeń powoduje pewne utrudnienie w określaniu spodziewanych przemieszczeń na obrazie systemu wizyjnego. Tymczasem trudno sobie wyobrazić żeby człowiek musiał się zatrzymywać podczas ruchu gałek ocznych. Wprowadzając trzeci aspekt ruchu (trzeci rodzaj sterowania robotem, mającego wpływ na system wizyjny), należy już na początku podkreślić, że od projektowanych implementacji rzeczywistych wymaga się nieprzerwanej pracy systemu wizyjnego dla wszystkich trzech wspomnianych aspektów ruchu. Ogromnym ułatwieniem będzie tu natura warstwy sensorycznej sieci HTM – podczas gdy inne rozwiązania traktują ruch jako czynnik utrudniający[109] analizę obrazu/obiektów, w sieciach HTM jest on nie tylko pożądany, ale jest naturalnym sposobem stymulacji węzłów sensorycznych sieci, pobudzających procesy składowe algorytmu rozpoznawania.

[107] Najbardziej eksplorowany obecnie aspekt sterowania w robotach mobilnych, ciekawie podsumowany w publikacji [44].

[108] Warto wspomnieć o dodatkowym uzasadnieniem użycia tu słowa „ruch" – sieci HTM, otrzymujące na wejścia swoich warstw sensorycznych fragmentaryczne wejściowe dane wizualne, wymagają zmienności tych danych w czasie, co zapewniane jest poprzez przesuwanie wejść macierzy sensorów warstw sensorycznych nad macierzą obrazu wejściowego zapisanego w proponowanej reprezentacji.

[109] Większość implementacji praktycznych likwiduje ruch analizując pojedyncze klatki obrazu, uwzględniając czasami jedynie ruch jako obraz różnicowy dwóch sąsiadujących klatek i wspomagając tym sposobem algorytm wyznaczania krawędzi obiektów. Ruch nie ma w nich właściwości stymulujących.



Rys. 3.16. Schemat blokowy przedstawiający system wizyjny jako złożony system regulacji.

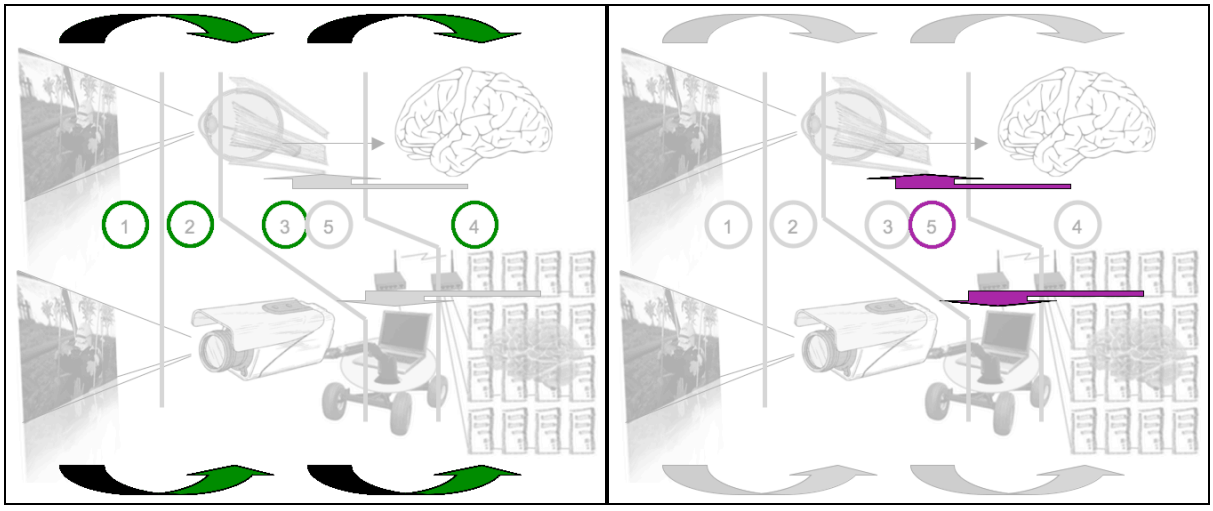
Na rysunku Rys. 3.16 pokazano wyjściową koncepcję – przedstawienia systemu wizyjnego nie jako sekwencji operacji przetwarzania obrazu, lecz jako złożonego układu regulacji. W pracy omówiono (jedynie) pętlę „A”, ponieważ jej użycie ma zasadnicze znaczenie dla wyniku prowadzonych działań badawczych – stanowi o możliwości użycia klastra komputerowego do wymienionych na wstępie zadań stawianych systemom wizyjnym.

W rozdziale 3.2.2 opisano, w jaki sposób będzie tworzona nowa reprezentacja obrazu (korzystająca z transformaty DWT, używająca technologii POI) w oparciu o określone współrzędne punktu fiksacji. Opisano również przykładową topologię klastra (Rys. 3.15), oraz uzasadniono przyjęty w niej podział funkcjonalny. W rozdziale 4.3 są opisane przykładowe implementacje (eksperymenty) weryfikujące w sposób powtarzalny implementowalność zaproponowanych innowacyjnych rozwiązań i tym samym pokazujące konkretne korzyści z ich zastosowania oraz potwierdzające tezę[110] pracy.

Niniejszy rozdział zawiera natomiast sedno rozwiązania – wspólny mianownik opisywanych technologii – algorytm modyfikacji współrzędnych punktu fiksacji. To właśnie ten algorytm spina opisane technologie w jedno rozwiązanie, w pomysł systemu wizyjnego. O ile poszczególne opisane elementy (POI, topologia, podział funkcjonalny, topologia i synchronizacja sieci HTM, m.in.) można zastąpić innymi, o tyle opisywany tu algorytm pozostaje fundamentem, na którym dobudowywane są wspomniane technologie.

W największym uproszczeniu, opisywane sterowanie można podzielić na dwie części: sterowanie w układzie otwartym (Rys. 3.17 A) oraz uzupełniające je sprzężenie zwrotne (Rys. 3.17 B), czyniące z układu sterowania układ zamknięty.

[110] patrz: strona 20.



Rys. 3.17. Kolejne etapy przesyłania informacji wizualnej (po lewej) oraz sprzężenie zwrotne wpływające na tą informację na etapie konwersji reprezentacji (POI).

Na powyższym rysunku (Rys. 3.17) przedstawiono zasadę działania proponowanego rozwiązania – po lewej stronie: przepływ informacji wizualnej (obrazu)[111] od akwizycji do wnioskowania, natomiast po prawej stronie: sprzężenie zwrotne niezbędne do określenia nowych współrzędnych punktu fiksacji w kolejnym obrazie wejściowym.

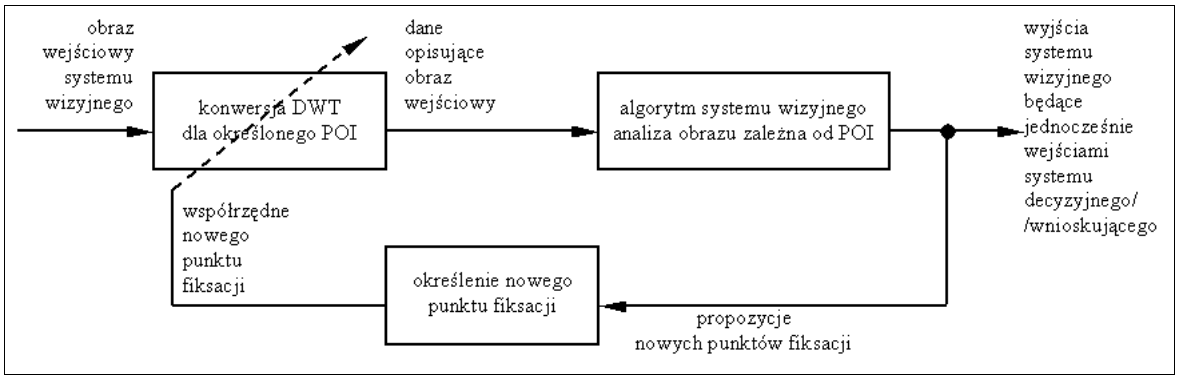
Lokalizacja punktu fiksacji pierwszej przesyłanej klatki obrazu nie jest istotna, może być wybrana losowo. Podczas przygotowywania każdej następnej klatki strumienia wideo nowe współrzędne będą już określone na podstawie algorytmu.

Każda klatka obrazu, zgodnie z zaimplementowaną topologią, będzie konwertowana z użyciem DWT i POI na kilka wersji – dla różnych węzłów grupy „podsystemu przetwarzania z podziałem funkcjonalnym", m.in.:

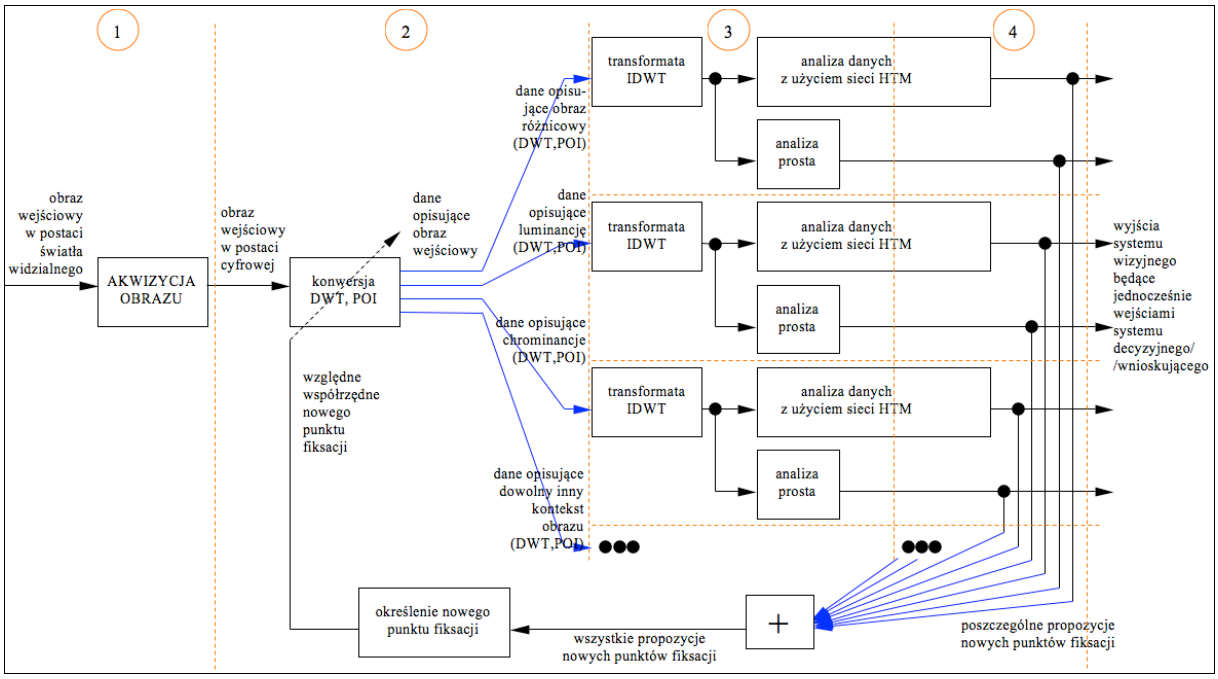
- macierz dwuwymiarowa obrazu różnicowego przedstawiająca w skali monochromatycznej ruch w scenie,
- macierz dwuwymiarowa informacji o luminancji klatki obrazu,
- macierze dwuwymiarowe zawierające inne informacje[112] o obrazie,
- macierz trójwymiarowa informacji o chrominancjach 'U' i 'V' klatki obrazu.

[111] Etapy oznaczone na rysunku cyframi wyjaśniono przy rysunku Rys. 3.1.

[112] Na przykład uproszczona reprezentacja „intensywności" koloru w formie obrazu monochromatycznego przedstawiona w [128], albo reprezentacja będąca wynikiem określania kierunku ruchu obserwowanego w scenie.



Rys. 3.18. Układ regulacji w kontekście ruchów sakkadowych w największym uproszczeniu.



Rys. 3.19. Uproszczony układ regulacji, z uwzględnieniem procesów pośrednich i klastra, w kontekście ruchów sakkadowych. 1) kamera, 2) węzeł wejściowy (i/lub komputer *embedded* robota), 3) grupa węzłów pośredniczących/koordynujących w podziale funkcjonalnym,[113] 4) rozproszona (skalowalna) część każdej z sieci HTM

Takie macierze, jak również dowolne inne, wraz ze współrzędnymi użytego przy transformacji punktu, przesyłane są do odrębnych węzłów klastra, będących lokalnymi koordynatorami przetwarzania danego kontekstu obrazu (ruchu, koloru, etc.). W węzłach tych obraz jest odzyskiwany z macierzy w wyniku transformaty odwrotnej (IDWT), oraz podany na wejście warstwy sensorycznej lokalnej[114] sieci HTM. Zaleca się, aby pierwsze warstwy

[113] „Analiza prosta" oznacza tu dowolną metodę wnioskowania, która dawałaby współrzędne nowego punktu zainteresowania (bez użycia sieci HTM), na przykład: gwałtowny ruch, jaskrawy kolor.

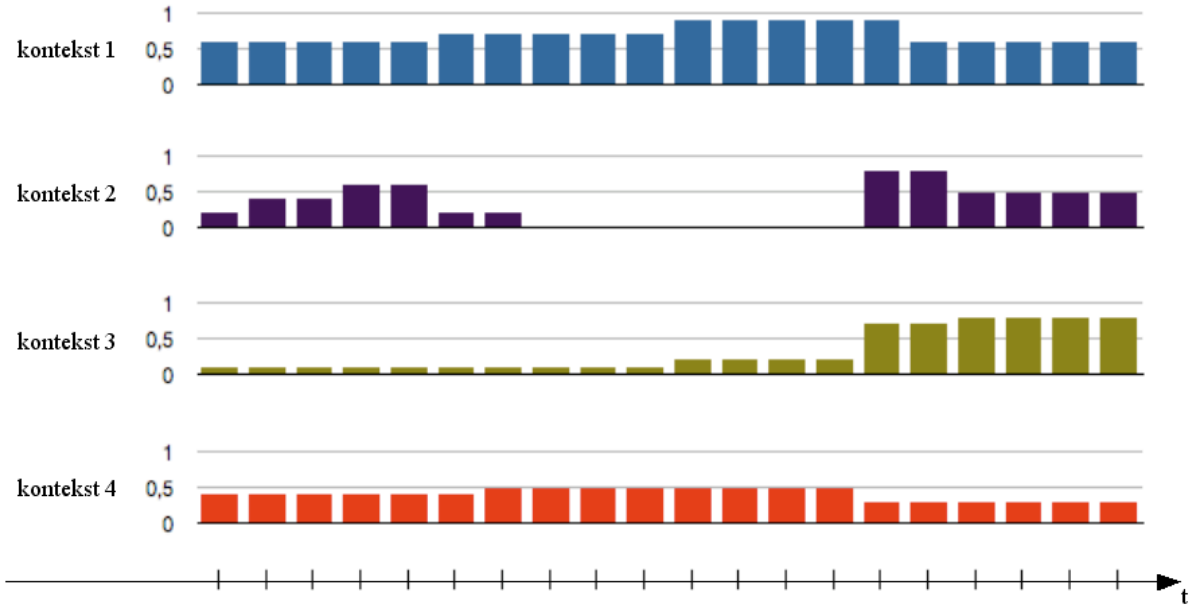
[114] Lokalna, czyli osobna dla każdego z węzłów grupy podziału funkcjonalnego. Oczywiście sieć ta może być siecią rozproszoną, korzystającą z wolnych zasobów (węzłów) klastra. Tylko wtedy możliwe będzie ich wykorzystanie oraz tym samym zapewnienie skalowalności algorytmu.

(sensoryczne) implementować w obrębie jednego fizycznego komputera/węzła, a dopiero na wyższych poziomach abstrakcji (warstwach sieci HTM) sieć była rozpraszana w klastrze. Powyższy opis zilustrowany jest rysunkami Rys. 3.18 oraz Rys. 3.19. Nowy punkt fiksacji określany jest na podstawie informacji zwrotnej przekazywanej przez węzły wnioskujące (węzły realizujące funkcje najwyższych warstw sieci HTM oraz przez węzły grupy podziału funkcjonalnego, nazwane wcześniej „lokalnymi koordynatorami przetwarzania kontekstu[115]"). Wszystkie węzły uprawnione do modyfikacji punktu fiksacji wysyłają do węzła wejściowego „swoją" propozycję nowego względnego położenia punktu fiksacji. Ponieważ każdy z kontekstów ma inny czas przetwarzania, propozycja nowych współrzędnych dla danego kontekstu podtrzymywana jest aż do momentu otrzymania nowej propozycji (Rys. 3.20). W ten sposób każdy z kontekstów, niezależnie od czasu przetwarzania, ma wpływ na lokalizację punktu fiksacji na kolejnych (następnych) przetwarzanych klatkach obrazu. Jednocześnie dla każdego kontekstu musi być określona, oprócz propozycji nowej lokalizacji, „siła głosu" określająca stopień spodziewanej atrakcyjności wnioskowanego punktu fiksacji.[116] Węzeł wejściowy, podczas konwersji transformatą DWT kolejnej klatki obrazu, uwzględnia propozycje nowych lokalizacji współrzędnych punktu fiksacji kierując się dowolnym, wybranym przez projektanta systemu, algorytmem.

Rysunek poniżej (Rys. 3.20) przedstawia cztery konteksty obrazu wizyjnego (m.in. ruch, określony znaczący kolor, głębia sceny, m.in.) oraz dyskretną wartość ich „parametru istotności" („siły głosu") propozycji nowych współrzędnych danego węzła/kontekstu. Na początku przedstawionego okresu największą siłę zgłaszał kontekst 1 (0.6), to znaczy, że nowy punkt fiksacji był wynikiem jego propozycji. Na końcu przedstawionego okresu wygrywał kontekst 3 (0.8). Kontekst 1 ma czas przetwarzania 5 jednostek czasu (m.in. przez czas pięciu przechwyconych klatek), kontekst 4 – 8 jednostek czasu. Nie ma to wpływu na działanie algorytmu elekcji punktu POI, ponieważ zgłoszone współrzędne oraz parametr „siły głosu" podtrzymywany jest aż do otrzymania informacji zwrotnej aktualizującej.

[115] Przez „kontekst obrazu" rozumiana jest tu reprezentacja klatki obrazu (oraz całokształt procedur jej przetwarzania) zawierająca (przynajmniej) dane dotyczące np.: ruchu lub jaskrawości lub nieregularności, itd.

[116] Przykładowo: jeżeli węzły kontekstu ruchu wychwycą znaczny lub „interesujący" ruch na obrzeżach sceny, „swoją" nową propozycję punktu fiksacji poprą odpowiednio dużym współczynnikiem „siły głosu".



Rys. 3.20. „Parametr istotności" propozycji nowych współrzędnych danego węzła/kontekstu (opis w tekście).

Proponowane podejście w naturalny sposób rozwiązuje wiele często spotykanych komplikacji, jak m.in. złożona analiza słabo rozpoznawalnego ruchomego obiektu. Dzięki zaproponowanemu sterowaniu współrzędnymi punktu fiksacji nie ma potrzeby implementowania operacji śledzenia obiektu (*ang. tracking*), ponieważ poruszający się obiekt będzie w naturalny sposób przyciągał uwagę systemu. Kolejną zaletą proponowanego rozwiązania jest analiza obrazu zmiennego w czasie – jeżeli w scenie istnieje słabo rozpoznawalny ale ruchomy (ruchliwy) obiekt, system wizyjny jest w stanie rozpoznać ten obiekt rejestrując kolejne jego cechy odkrywane w różnych chwilach czasu.

Zakłócenie, w przedstawionym na rysunku Rys. 3.19 układzie, należy rozumieć jako ogół czynników zaburzających proces prawidłowego wyznaczania nowych współrzędnych punktu fiksacji. W przypadku sceny nieruchomej z rozmieszczonymi w niej kolorowymi obiektami, nieruchomy[117] robot może bez trudu określić względną lokalizację kolejnego obiektu w obrazie. W wyniku działania algorytmu analizy (siecią HTM) formułowana jest nowa propozycja punktu zainteresowania, która po etapie elekcji może zostać zatwierdzona. Przy następnej klatce strumienia obraz zostanie przekonwertowany z użyciem nowych współrzędnych. Jeżeli w tym (krótkim) czasie (pomiędzy elekcją a dostarczeniem

[117] Ruch robota nie przeszkadza algorytmom rozpoznawania, aczkolwiek należy go uwzględnić m.in. w obrazach różnicowych

i zdekodowaniem kolejnej klatki) obiekt się poruszy[118], ruch ten będzie zakłóceniem, które spowoduje nieznaczną rozbieżność pomiędzy żądanymi współrzędnymi a otrzymanymi do analizy. Na szczęście graficzna reprezentacja obrazu opisana w rozdziale 3.2.2 ma na tyle spory obszar najwyższej jakości obrazu, że nie uniemożliwi to wykonania analizy obiektu (lub fragmentu obiektu). Nie bez znaczenia jest też elastyczność działania sieci HTM, dla których nie ma większego znaczenia lokalizacja, rotacja czy skala obiektu.

Największą korzyść wynikającą z zastosowania w systemie wizyjnym proponowanego sprzężenie zwrotnego, stanowi możliwość wykorzystania klastra komputerowego do szeroko rozumianej analizy wizji.

[118] Obiekt mógł być w ruchu już podczas określania nowych współrzędnych, jeszcze przed etapem elekcji czy konwersji.

4. Weryfikacja implementacyjna – środowisko eksperymentów

Metody przetwarzania rozproszonego były do tej pory terenem zakazanym dla sympatyków i zwolenników komputerowych systemów wizyjnych robotów mobilnych. Dzięki tej pracy możliwe będzie ponowne rozważenie zastosowania niebywałego potencjału klastrów, ich ogromnej mocy obliczeniowej, która była poza zasięgiem projektantów większości systemów wizyjnych. Ponieważ praca stanowi połączenie kilku rozdzielnych pomysłów, niezwykle istotna była możliwość weryfikacji praktycznej przyjmowanych założeń i stawianych tez, aby ostatecznie możliwe było określenie celowości stosowania proponowanych rozwiązań.

Zawartość tego rozdziału stanowi nie tylko opis podjętych działań, lecz również zawarte są tu konkretne wyniki otrzymane w drodze weryfikacji implementacyjnej.

Eksperymenty przeprowadzane na potrzeby pracy były realizowane na czterech różnych platformach mobilnych, przy użyciu trzech różnych rodzin systemów operacyjnych, a do komputera robota mobilnego były przyłączane cztery różne klastry komputerowe. O ile różnorodność w kontekście systemów i oprogramowania była utrudnieniem (niekompatybilne fragmenty implementacji trzeba było tworzyć od nowa), o tyle różnorodność w kontekście komputerów i klastrów miała zdecydowanie pozytywny wpływ na ostateczny kształt każdej z implementacji. Implementacje tworzone były na niewyszukanym sprzęcie, a więc z dbałością o wysoką wydajność i jakość kodu, jednak bez wykorzystywania akceleracji sprzętowej[119]. Dodatkowo przeprowadzano również weryfikacje na sprzęcie wysokiej

[119] Pominięcie w implementacjach większości ze sposobności użycia akceleracji sprzętowej było celowe, miało na celu przebadanie „najgorszej wersji". W docelowej implementacji systemu można naturalnie korzystać z wszystkich możliwości oferowanych przez sprzęt i oprogramowanie.

klasy[120], co pozwoliło mieć wgląd w niezwiązane z platformą sprzętową problemy komunikacji, topologii i skalowalności całokształtu rozwiązania.

4.1. Sprzęt użyty w eksperymentach

Na wstępie należy zaznaczyć, że praktyczna (sprzętowa) implementacja nie była celem pracy, lecz jedynie jej uzupełnieniem. Prace nad sprzętem skupiły się wokół dwóch wątków: opracowania i dokumentacji wymaganej/wystarczającej platformy robota mobilnego oraz opracowania i dokumentacji topologii i konfiguracji sprzętu komputerowego systemu wizyjnego. Wraz z postępem prac badawczych pierwszy z tych wątków porzucono, ponieważ nie wnosił nowej wartości dodanej do wyników badań. W poniższym podrozdziale zaprezentowano cztery platformy robotów mobilnych opracowane na potrzeby eksperymentów. Ponieważ opracowane w niniejszej pracy rozwiązania nie są zależne od rodzaju zastosowanych komponentów mechanicznych robota, przedstawione platformy zamieszczone są jedynie w celach poglądowych oraz w charakterze dokumentacji przeprowadzonych eksperymentów.

Platformy eksperymentalne robota mobilnego

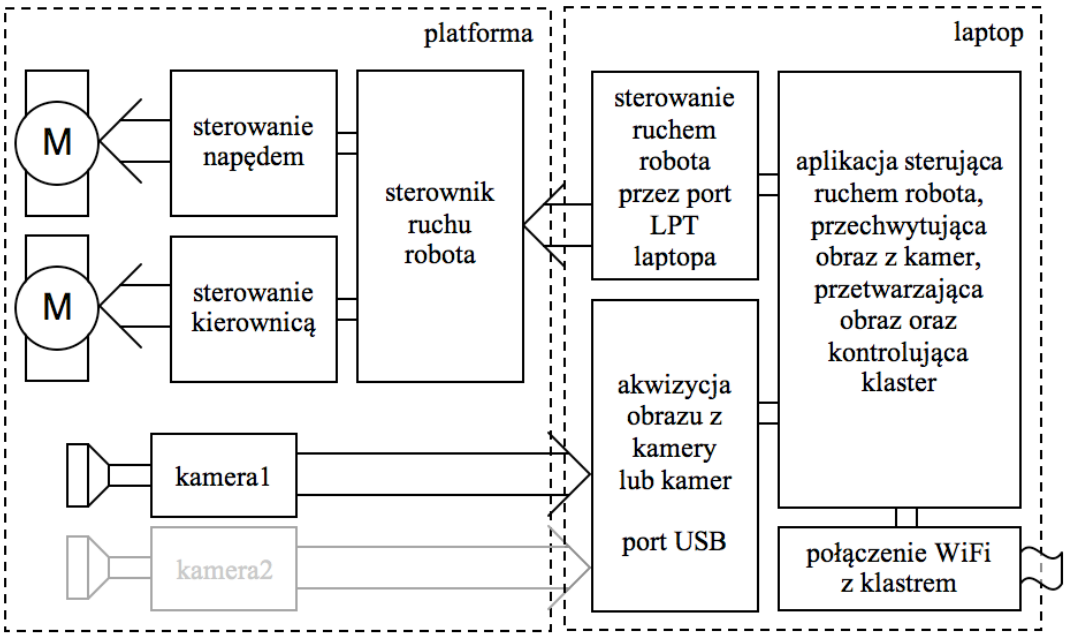
Pierwsza wersja platformy służyła jako narzędzie w eksperymentach z następujących obszarów funkcjonowania systemów wizyjnych w robotyce:

- klasyczne metody rozpoznawania obrazu (obraz różnicowy, filtry, translacje, rozpoznawanie kształtów, porównywanie z bazą),
- rozpoznawanie wzorców z użyciem sieci neuronowych (opisane w rozdziale 4.3),
- badania sposobów minimalizacji czasu komunikacji z zastosowaniem AV (4.3),
- badania nad możliwością skrócenia czasu komunikacji poprzez zastosowanie logiki rozmytej,[121]
- problematyka synchronizacji i kalibracji kamer w celu uzyskania obrazu stereoskopowego.[121]

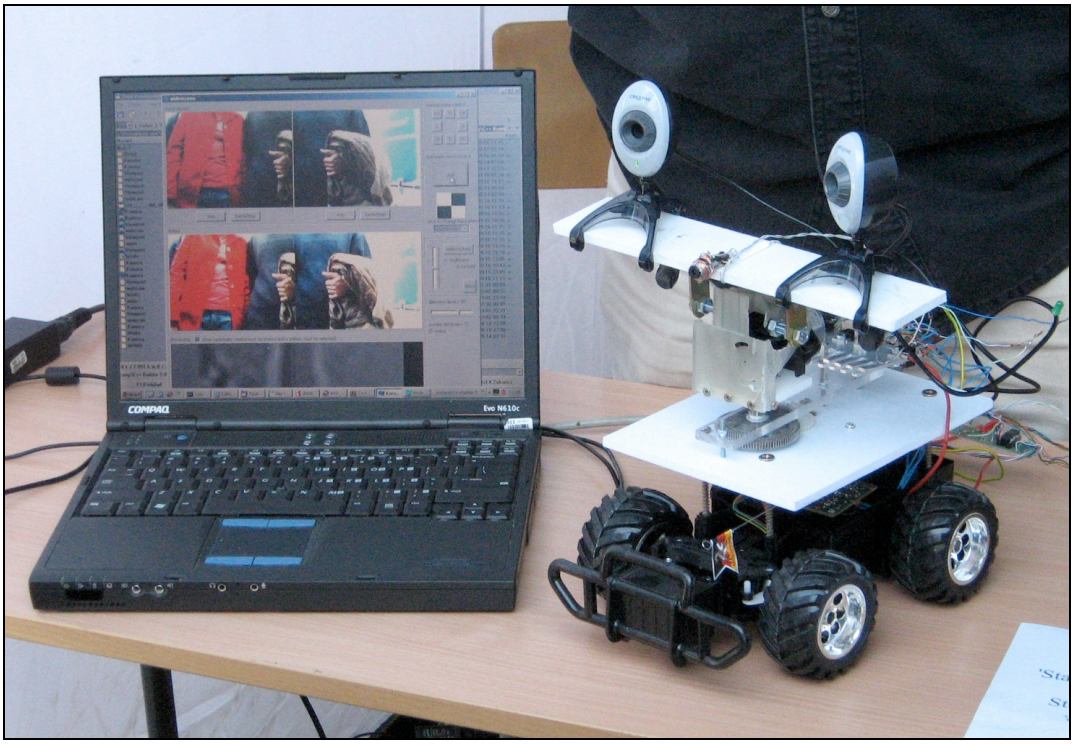
[120] m.in. na klastrze Instytutu Automatyki i Informatyki Wydziału Elektrotechniki, Automatyki i Informatyki Politechniki Opolskiej oraz na klastrze XGrid w laboratorium komputerów Apple Macintosh Zespołu Szkół Elektrycznych im.T.Kościuszki w Opolu

[121] Wątek w pracy pominięto, nie ma bezpośredniego związku z zastosowanym rozwiązaniem

System wizyjny tej wersji platformy został szczegółowo opisany w rozdziale 4.3.

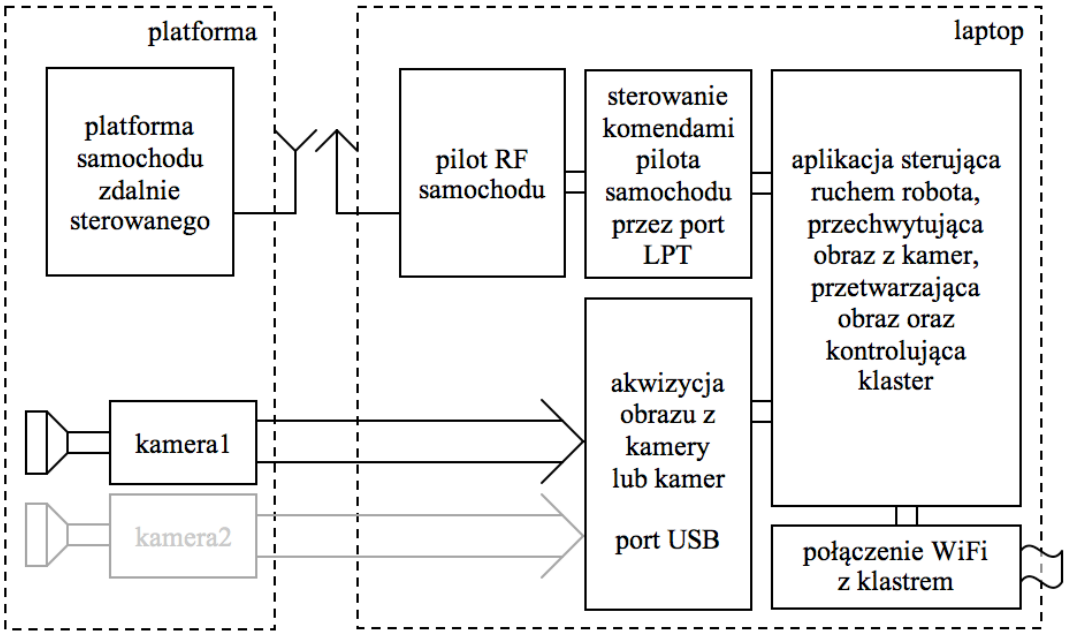


Rys. 4.1. Implementacja sprzętowa – wersja podstawowa, łączność przewodowa

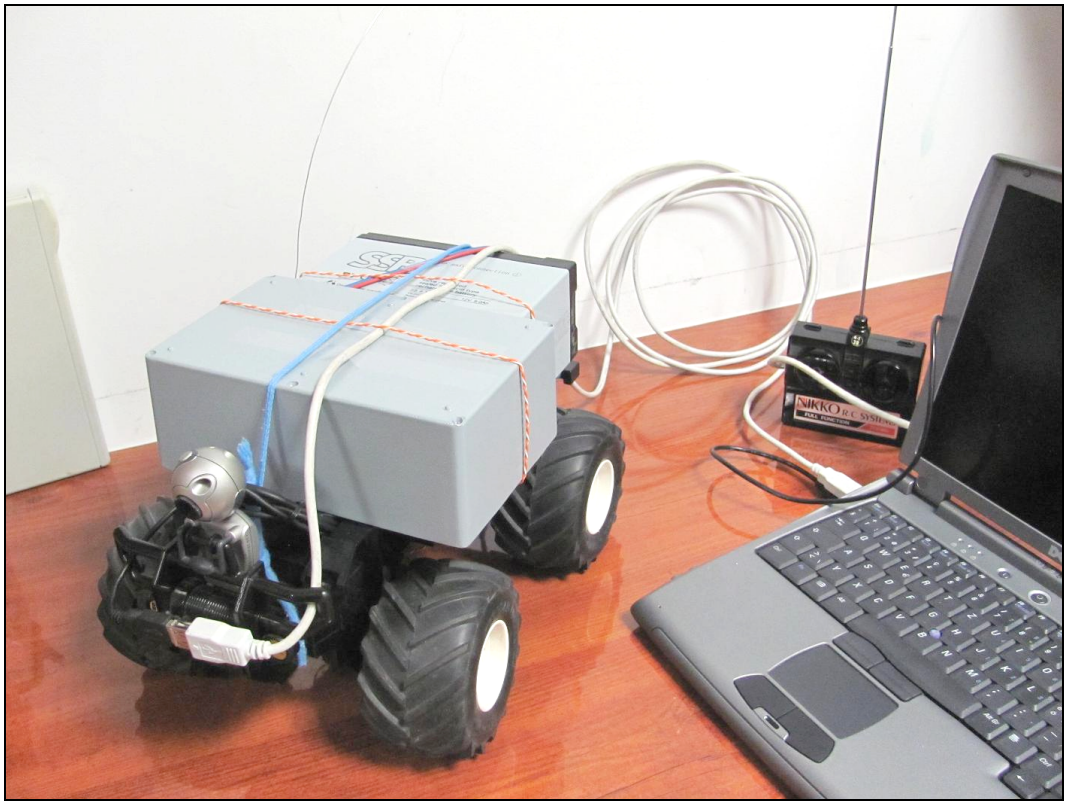


Rys. 4.2. Praktyczna realizacja platformy opisanej rysunkiem Rys. 4.1, bez połączenia z klastrem, bez sieci HTM.

Druga wersja platformy robota (Rys. 4.3, Rys. 4.4) zawierała szereg modyfikacji, które nie miały jednak znacznego wpływu na przebieg i wartość dodaną przeprowadzonych prac naukowo-badawczych.



Rys. 4.3. Implementacja sprzętowa – wersja pokazowa, platforma nie zawiera żadnych nieestetycznych kabli/elektroniki

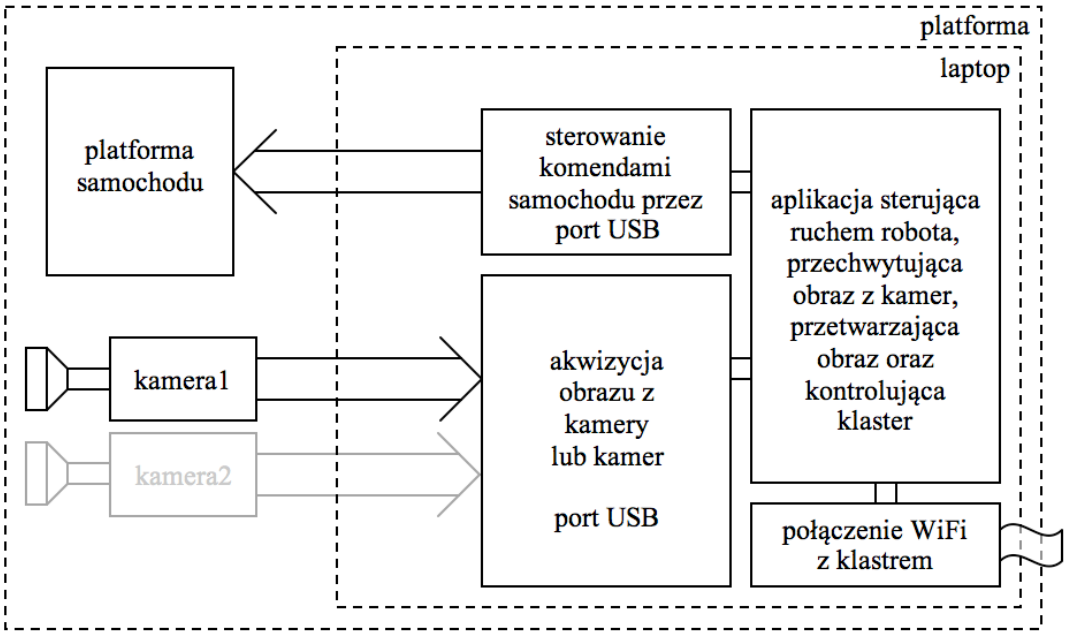


Rys. 4.4. Praktyczna realizacja platformy opisanej rysunkiem Rys. 4.3, bez połączenia z klastrem, bez sieci HTM.

Trzecia wersja platformy (prezentowana poniżej) charakteryzowała się o wiele solidniejszą konstrukcją oraz większą nośnością, co umożliwiło zamontowanie i transport

laptopa, w znacząco lepszym świetle stawiając „mobilność" tej platformy względem wcześniejszych.

Ta platforma stanowiła bazę większości[122] przeprowadzanych eksperymentów (z wyjątkiem eksperymentów opartych o technologię, sprzęt oraz klaster firmy Apple).



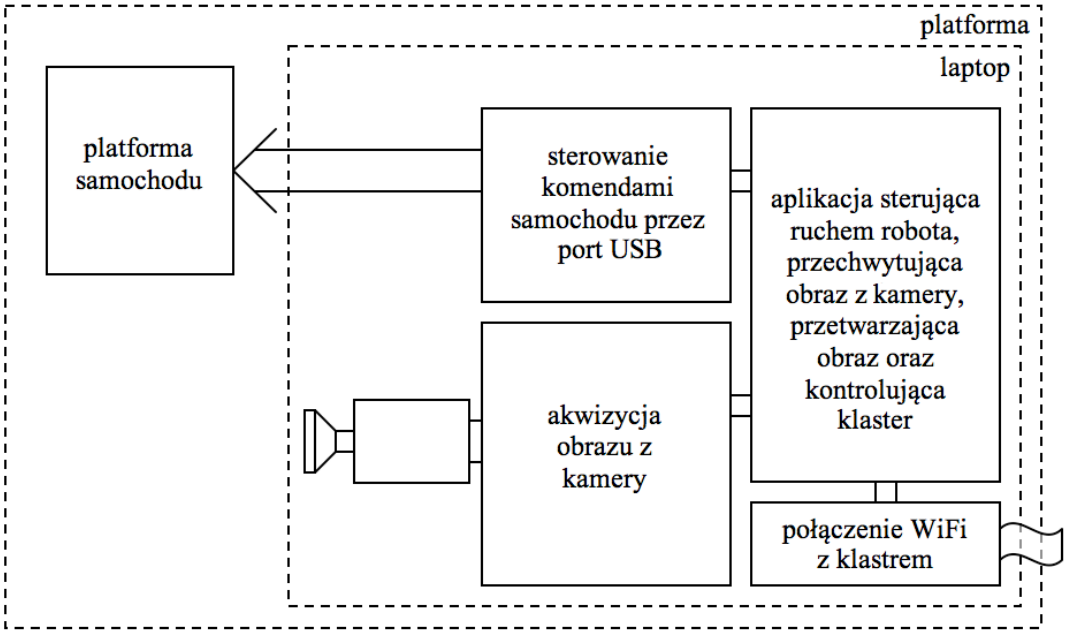
Rys. 4.5. Implementacja sprzętowa – wersja bezprzewodowa, platforma w pełni mobilna.

Czwarta wersja platformy koncepcyjnie nie różniła się od platformy trzeciej (Rys. 4.5) – różnicę stanowiło środowisko klastrowe (zamieniono klaster LAM/MPI[123] [67] na klaster Apple Xgrid [6]), co pociągnęło za sobą pewne dodatkowe modyfikacje (zmiana języka programowania na Cocoa, systemu operacyjnego na MacOsX oraz komputera robota na komputer Apple MacBook).

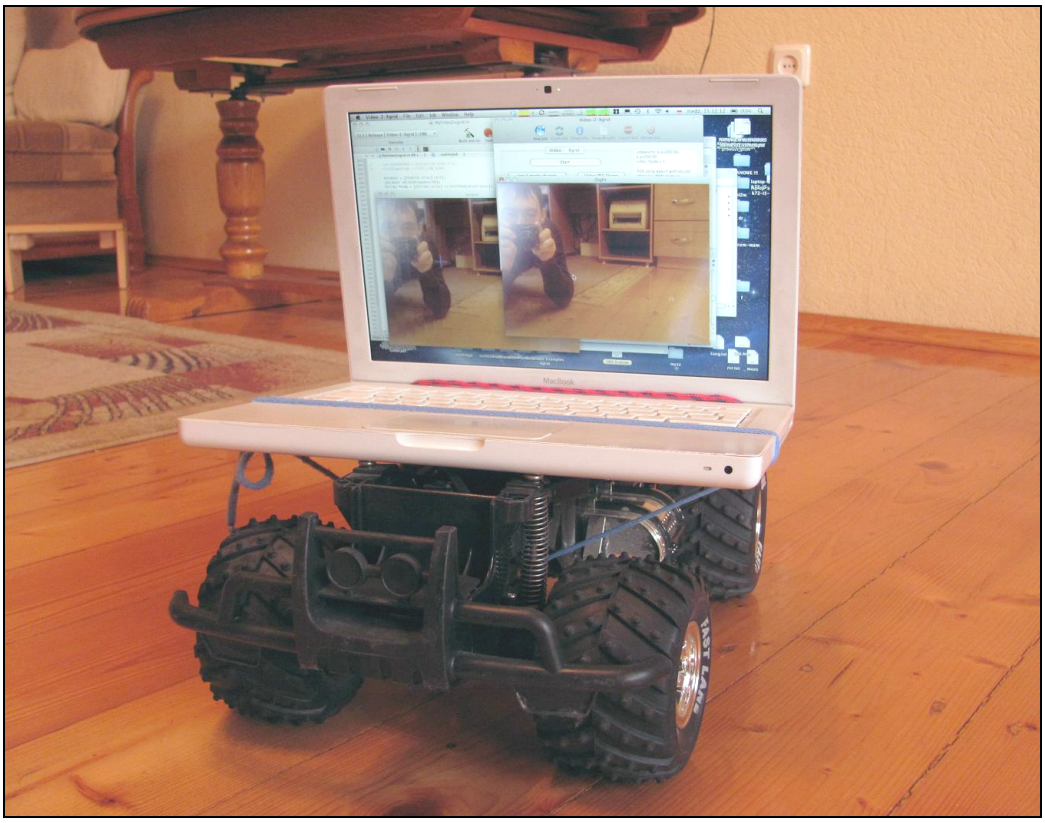
[122] Z uwagi na konieczność okresowego ładowania akumulatora, część prac przeprowadzano bez użycia platformy mobilnej, posługując się jedynie komputerem mobilnym i kamerą.

[123] LAM/MPI – (*ang. Local Area Multicomputer*) – implementacja standardu MPI (*ang. Message Passing Interface*) oparta o zasady wolnego oprogramowania. [67]

Rys. 4.6. Praktyczna realizacja platformy opisanej rysunkiem Rys. 4.5.



Rys. 4.7. Implementacja sprzętowa – wersja bezprzewodowa, platforma w pełni mobilna



Rys. 4.8. Praktyczna realizacja platformy opisanej rysunkiem Rys. 4.7.

Część eksperymentów opisanych w rozdziale 4.3 była prowadzona z wykorzystaniem powyższych platform, choć w większości do pracy wystarczał laptop z kamerą lub laptop z kamerą i klaster.

Sprzęt komputerowy

Uzyskane dzięki eksperymentom wyniki oraz spostrzeżenia łatwiej będzie odtworzyć (i uznać za powtarzalne) znając konfigurację sprzętową stanowiska testowego. Prace były prowadzone w oparciu o wiele różnych konfiguracji, które można zagregować w trzech grupach:

a) oparte o system Windows[124], bez klastra,
b) oparte o systemy rodziny Linuks, z klastrem LAM/MPI lub bez,
c) oparte o system MacOsX, z klastrem Apple Xgrid lub bez.

[124] Windows jest nazwą produktu należącą do firmy Microsoft, nazwa została użyta tylko w celu rozróżnienia produktu spośród innych, podobnie inne nazwy produktów i technologii użyte w niniejszej pracy, należące do prawowitych właścicieli.

W obrębie tych trzech grup można wyróżnić zrealizowane zadania i eksperymenty, do których użyto jednej lub dwóch konfiguracji sprzętowych:

1) Aplikacja1 (opisana w rozdziale 4.3), pozyskująca obraz z jednej lub dwóch kamer USB, sterująca platformą robota przez LPT.

- wersja stacjonarna: Komputer stacjonarny[125] PC, procesor dwurdzeniowy 2.33GHz, kamera USB, Windows
- wersja mobilna: Laptop[126], procesor jednordzeniowy 700MHz, 512MB pamięci, kamera USB, port LPT, Windows

2) Aplikacja2 (opisana w rozdziale 4.4.2), pozyskująca obraz z kamery USB

- wersja stacjonarna:
  - Komputer stacjonarny[127] PC, procesor dwurdzeniowy 2.33GHz, kamera USB, Linux
  - Klaster LAM/MPI złożony z czterech węzłów (powyższy oraz serwer[128] dwuprocesorowy IBM 300MHz i komputer[129] PC 1.1GHz)

3) Aplikacja3 (opisana w rozdziale 4.4.3)

- wersja stacjonarna: Komputer stacjonarny PC, procesor dwurdzeniowy 2.33GHz, Windows

4) Aplikacja4 (opisana w rozdziale 4.4.4), pozyskująca obraz z wbudowanej kamery USB lub zewnętrznej USB

- wersja stacjonarna ekonomiczna:
  - Komputer stacjonarny[130] PC, procesor dwurdzeniowy 2.33GHz, kamera USB, Linuks
  - Klaster LAM/MPI złożony z dwóch węzłów (powyższy oraz Apple MacBook[131] 2.1GHz)
- wersja mobilna bez klastra: Laptop[132], procesor dwurdzeniowy 3GHz, kamera USB, Linuks
- wersja mobilna ekonomiczna z klastrem:
  - Laptop, procesor dwurdzeniowy 3GHz, kamera USB, Linuks
  - Klaster LAM/MPI złożony z dwóch węzłów (powyższy oraz Apple MacBook 2.1GHz)
- wersja mobilna zaawansowana z klastrem:

[125] Procesor: Intel Core2Duo 2.33GHz, pamięć: 3GB DDR2, system WindowsXP.

[126] Procesor: Intel Pentium2 700MHz, pamięć: 512MB DDR, system WindowsXP.

[127] Procesor: Intel Core2Duo 2.33GHz, pamięć: 3GB DDR2, system Fedora3.

[128] Procesor: 2*Intel Pentium2 300MHz, pamięć: 512MB, system Fedora3.

[129] Procesor: AMD Duron 1.1GHz, pamięć: 1GB, system Fedora3.

[130] Procesor: Intel Core2Duo 2.33GHz, pamięć: 3GB DDR2, system Ubuntu 10.4.

[131] Procesor: Intel Core2Duo 2.1GHz, pamięć: 3GB DDR2, system MacOsX 10.5.8.

[132] Procesor: Intel i3 3GHz, pamięć: 6GB DDR3, SSD, system Ubuntu 10.4.

- Laptop, procesor dwurdzeniowy 3GHz, kamera USB, Linuks
- Klaster LAM/MPI Wydziału Elektrotechniki, Automatyki i Informatyki (złożony z 8 płyt[133] 2-procesorowych połączonych szybką siecią[134]), Linuks

5) Aplikacja5 (opisana w rozdziale 4.4.5), pozyskująca obraz z wbudowanej kamery USB lub zewnętrznej USB/FireWire

- wersja mobilna/stacjonarna ekonomiczna:
  - Laptop Apple MacBook, Core2Duo 2.1GHz, MacOsX 10.5
  - Klaster Apple Xgrid złożony z jednego agenta[135] (powyższy)
- druga wersja mobilna/stacjonarna ekonomiczna:
  - Laptop Apple MacBook Pro i5 2.4GHz, MacOsX 10.6
  - Klaster Apple Xgrid złożony z jednego agenta (powyższy)
- wersja mobilna/stacjonarna zaawansowana:
  - Laptop Apple MacBook, Core2Duo 2.1GHz, MacOsX 10.5
  - Klaster Apple Xgrid (Zespołu Szkół Elektrycznych im.T.Kościuszki w Opolu) złożony z 18 agentów (powyższy, 15 stanowisk[136] uczniowskich, serwer[137], stanowisko[138] nauczyciela) połączony szybką siecią[139], MacOsX 10.4

W zestawieniu pominięto aplikacje nie wnoszące nowej wartości do pracy.

Klastry

W powyższej liście pojawiają się cztery różne klastry – dwa z nich składają się ze zwykłych komputerów PC, a dwa stanowią rozwiązania profesjonalne – klastry zaprojektowane z myślą o obliczeniach równoległych i przetwarzaniu rozproszonym.

Dla eksperymentów najbardziej znaczącymi konfiguracjami były te, które korzystały z amatorskich klastrów, ponieważ wszystkie towarzyszące tworzeniu oprogramowania problemy (m.in. zbyt mała moc obliczeniowa, za wolna sieć komputerowa, zbyt duży czas

[133] Procesor: 2*Intel Pentium4Xeon(HT) 2.8GHz, pamięć: 2GB, system Linux Gentoo.

[134] Każda z 8 płyt wyposażona w dwa łącza gigabitowego Ethernet-u.

[135] Klaster Apple XGrid równomiernie rozdziela zadania węzłom – w przypadku istnienia tylko jednego węzła w klastrze zadania są kolejkowane temu jedynemu węzłowi (minimalna liczba węzłów nie jest ograniczeniem).

[136] Procesor: Intel Core2Duo 1.6GHz, pamięć: 1GB DDR2, system MacOsX 10.4.8.

[137] Procesor: 2*Intel Core2Duo 2GHz, pamięć: 4GB FBRAM, system MacOsXServer 10.4.8.

[138] Procesor: Intel Core2Duo 2GHz, pamięć: 1GB DDR2, system MacOsX 10.4.8.

[139] Komputery połączone gigabitowym Ethernet-em.

komunikacji klastra) powodowały o wiele bardziej rozważny dobór przyjętej metodologii implementacyjnej przejaskrawiając niejako skalę napotkanych problemów.

Konfiguracje korzystające z profesjonalnych klastrów obliczeniowych były cenne z zupełnie innego powodu – pozwalały nie zapomnieć o dbałości o skalowalność i wydajność implementowanych algorytmów w przypadku uruchomienia na wysokowydajnych maszynach.

4.2. Oprogramowanie użyte w eksperymentach

Oprogramowanie użyte przy eksperymentach można rozpatrywać w różnych aspektach: użytej biblioteki kontrolującej pracę klastra, użytego systemu operacyjnego, wymaganych bibliotek i/lub sterowników oraz środowiska programistycznego. W rozdziale 4.1 wymieniono wszystkie używane podczas eksperymentów konfiguracje sprzętowe. Z punktu widzenia oprogramowania, taka różnorodność nie występuje – każdy z eksperymentów wymagał spełnienia określonych warunków niezależnie od platformy sprzętowej:

1. Implementacja 1 – SSN, AV, tracking (Windows)

Aplikacja opisana bardziej szczegółowo w rozdziale 4.3, implementowana w języku C/C++ w środowisku Borland C++ Builder 6, a więc pod systemem operacyjnym Microsoft Windows[140]. Wymagania dodatkowe: Sterownik kamery (sterowniki kamer)[141], biblioteka WinIO dla sterowania ruchem platformy/robota.

2. Implementacja 2 – AV/ROI, tracking (Linux)

Aplikacja opisana bardziej szczegółowo w rozdziale 4.4.2, implementowana w języku C/C++ w środowisku Kdevelop, a więc pod systemem operacyjnym rodziny Linux[142].

[140] Testowano jej działanie pod systemami Microsoft Windows XP, Microsoft Windows 2000, Microsoft Windows 98 SE.

[141] Dowolna kamera dowolnego producenta, która istnieje w systemie operacyjnym jako urządzenie wejścia strumienia wideo.

[142] Oprogramowanie tworzono pod systemem Fedora 3, jednak można użyć dowolnego systemu umożliwiającego łatwą konfigurację klastra LAM/MPI i spełniającego wynotowane w dalszej treści wymagania.

Wymagania dodatkowe: Kamera działająca ze sterownikiem spca5xx lub inna[143] widoczna w systemie, biblioteka LAM/MPI, biblioteka SDL.

3. Implementacja 3 – DWT, ROI, POI, YUV, RLE (Windows)

Aplikacja opisana bardziej szczegółowo w rozdziale 4.4.3, implementowana w języku C/C++ w środowisku Borland C++ Builder 6, a więc pod systemem operacyjnym Microsoft Windows[144].

4. Implementacja 4 – (SSN, DWT, POI,) ROI, YUV, MPI (Linux)

Aplikacja opisana bardziej szczegółowo w rozdziale 4.4.4, implementowana w języku C/C++ w środowisku Kdevelop, a więc pod systemem operacyjnym rodziny Linux[145]. Wymagania dodatkowe: Kamera działająca ze sterownikiem spca5xx lub inna[146] widoczna w systemie, biblioteka LAM/MPI, biblioteka SDL.

5. Implementacja 5 – DWT, ROI, POI, YUV, Xgrid (MacOsX)

Aplikacja opisana bardziej szczegółowo w rozdziale 4.4.5, implementowana w języku Cocoa w środowisku Xcode 3.1.4, a więc pod systemem operacyjnym MacOsX[147].

Wszelkie szczegóły dotyczące niezbędnych bibliotek wraz z archiwum użytej wersji dostępne są na dołączonym do pracy nośniku. Tam można również znaleźć kod źródłowy, pliki uruchamialne oraz zrzuty ekranu i inną dokumentację graficzną i tekstową działania każdej z wymienionych aplikacji.

4.3. Analiza czasowa proponowanego systemu wizyjnego

Implementacja algorytmów przetwarzania danych systemu wizyjnego wiąże się z wykonywaniem wielu prostych operacji na stosunkowo dużej ilości danych. W systemach wizyjnych opartych o komputery PC procesory ogólnego przeznaczenia obciążane są dużą

[143] Może istnieć potrzeba modyfikacji kodu celem uzupełnienia listy obsługiwanych kamer.

[144] Testowano jej działanie pod systemami Microsoft Windows XP, Microsoft Windows 2000.

[145] Oprogramowanie tworzono pod systemami Fedora 3 i Ubuntu 8.10, jednak można użyć dowolnego systemu umożliwiającego łatwą konfigurację klastra LAM/MPI i spełniającego wynotowane w dalszej treści wymagania.

[146] Może istnieć potrzeba modyfikacji kodu celem uzupełnienia listy obsługiwanych kamer.

[147] Testowano jej działanie pod systemami MacOsX 10.4 oraz MacOsX 10.5.

liczbą przesłań pomiędzy pamięcią i procesorem, co konsumuje sporo czasu procesora i blokuje magistralę systemu wieloprocesorowego [148]. W przypadku zastosowania klastra jako narzędzia przetwarzania rozproszonego, dodatkowym problemem jest znaczne ograniczenie przepustowości magistrali – najczęściej sieci komputerowej. Zmniejszanie czasu potrzebnego na przesyłanie danych między węzłami (czasu komunikacji) powoduje się zmianę jakościową problemu szybkości obliczeń [148].

Pomimo asynchronicznej pracy węzłów (lub grup węzłów) przetwarzających określone konteksty strumienia wideo (ruch, kolory, kształty, m.in.), możliwe jest oszacowanie czasu opóźnienia wynikającego z pracy systemu wizyjnego. Węzeł nadrzędny klastra w każdej chwili dysponuje historycznymi (nadal aktualnymi) informacjami zwrotnymi z poszczególnych węzłów (grup węzłów), aktualizowanymi asynchronicznie po ukończeniu przetwarzania klatki obrazu w danej grupie węzłów (węźle). Pracę synchroniczną podsystemu akwizycji, przygotowania danych i wejścia klastra przedstawia rysunek Rys. 4.9. Pracę asynchroniczną węzłów klastra w podziale funkcjonalnym przedstawia rysunek Rys. 4.11. Część czasu konsumowanego przez synchroniczną część rozproszonego (klastrowego) systemu wizyjnego można określić (podobnie jak w [148]) w sposób przedstawiony na równaniu (6) poniżej. Czas części asynchronicznej można natomiast oszacować przy założeniu nieprzerwanej dostępności aktualnej informacji zwrotnej lub historycznej zastępującej aktualną (równanie (7)).

$$t_1 = t_{akw} + t_{trans1} + t_{bezkont} + t_{DWT} + t_{kompr} \tag{6}$$

gdzie t_1 – czas części synchronicznej algorytmu;
t_{akw} – czas podsystemu akwizycji;
t_{trans1} – czas transmisji do klastra;
$t_{bezkont}$ – czas operacji bezkontekstowych wstępnego przetwarzania;
t_{DWT} – czas operacji przekształcenia DWT z użyciem POI;
t_{kompr} – czas operacji kompresji danych;

$$t_2 = t_{trans2} + t_{IDWT} + t_{kont} + t_{wnios} + t_{ster} \tag{7}$$

gdzie t_2 – czas części asynchronicznej algorytmu;
t_{trans2} – czas komunikacji;
t_{IDWT} – czas operacji przekształcenia IDWT;
t_{kont} – czas operacji kontekstowych przetwarzania;
t_{wnios} – czas podstawowej pętli podsystemu wnioskowania;
t_{ster} – czas przygotowania i przesłania informacji sprzężenia zwrotnego;

Uzyskanie informacji zwrotnej w części asynchronicznej (dla dowolnego węzła klastra) trwa tyle czasu ile wynosi t_2, jednakowoż całkowity czas opóźnienia systemu wizyjnego (od zmiany w scenie do reakcji robota) powinien być wyznaczany z uwzględnieniem dodatkowych opóźnień:

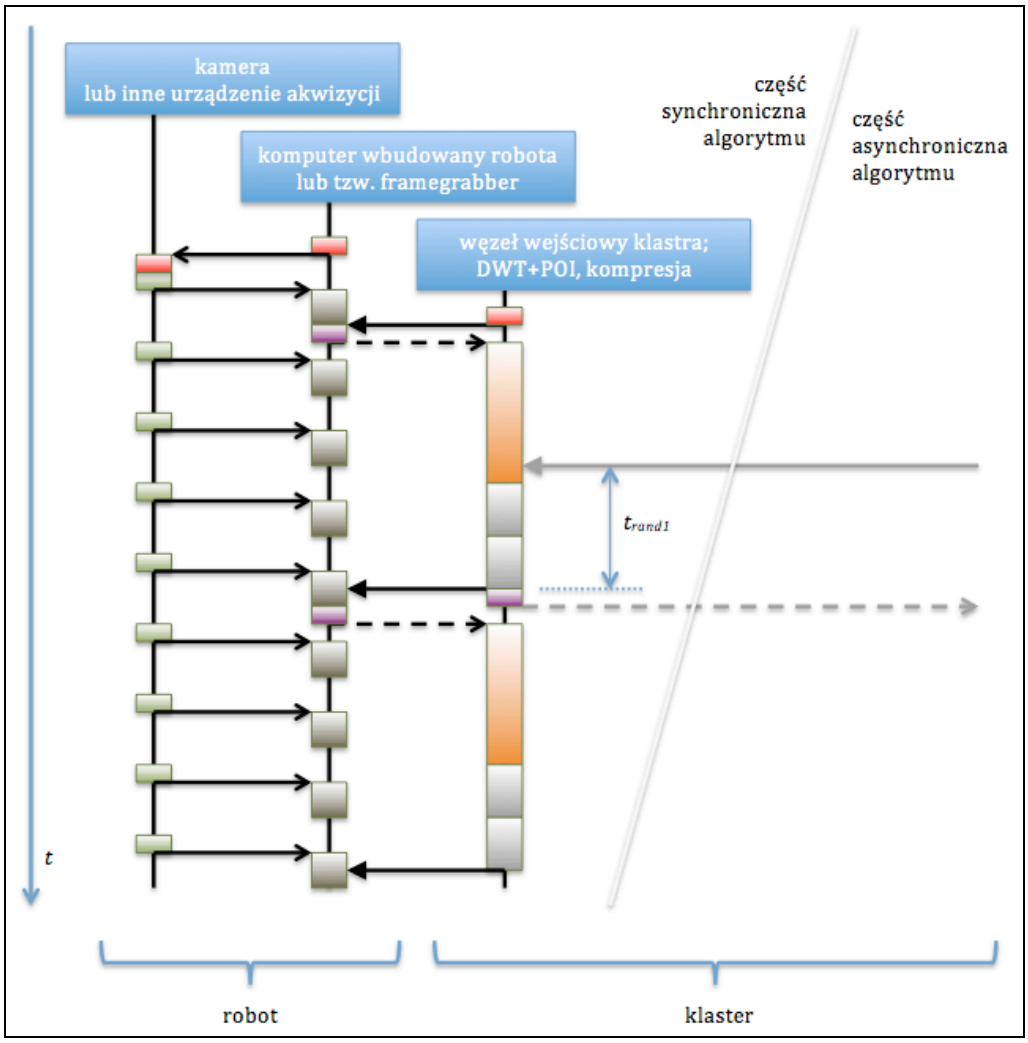
$$t_{sys} = t_1 + t_{rand1} + t_{rand2} + t_2 \tag{8}$$

gdzie t_{sys} – całkowity czas opóźnienia systemu wizyjnego (od zmiany w scenie do reakcji robota);
t_1 – czas części synchronicznej algorytmu;
t_2 – czas części asynchronicznej algorytmu;
t_{rand1} – czas (wartość przypadkowa) pomiędzy ukończeniem[148] ostatniego cyklu synchronicznego, a zainicjowaniem cyklu asynchronicznego; wartość z przedziału[149] od $0+t_{cyklu}$ do t_1, gdzie t_{cyklu} – najkrótszy czas procesora pomiędzy zakończeniem jednej operacji a rozpoczęciem drugiej;
t_{rand2} – czas (wartość przypadkowa) opóźnienia wysłania żądania kolejnej klatki wynikający z nieukończenia poprzedniego cyklu asynchronicznego; wartość z przedziału od $0+t_{cyklu}$ do t_2;

Powyższe równania wynikają po części z przeprowadzonej analizy literaturowej [148], a po części z prac projektowych wynikających z wstępnej fazy przygotowań do eksperymentów. Poniżej, na rysunkach Rys. 4.9, Rys. 4.10 i Rys. 4.11 przedstawiono szkic zachowania części synchronicznej i asynchronicznej proponowanego systemu wizyjnego. Wyniki weryfikacji implementacyjnej przedstawiono w rozdziale 4.5.

[148] Przesłanie danych przed ukończeniem przetwarzania oczywiście jest możliwe (i będzie bardzo często spotykane podczas normalnej pracy systemu), jednak skutkuje przesłaniem danych poprzednich. Aby możliwe było przetwarzanie nowych danych dotyczących najnowszego stanu sceny, należy wpierw ukończyć przetwarzanie cyklu synchronicznego.

[149] W skrajnym przypadku (najdłuższej zwłoki pomiędzy ukończeniem cyklu synchronicznego a rozpoczęciem cyklu asynchronicznego). Czas t_1 wynika z czasu przygotowania nowych danych wizualnych w cyklu synchronicznym, natomiast czas t_2 jest tu także obecny, ponieważ węzeł klastra, który nie ukończył przetwarzania cyklu asynchronicznego, nie pobierze nowych danych.

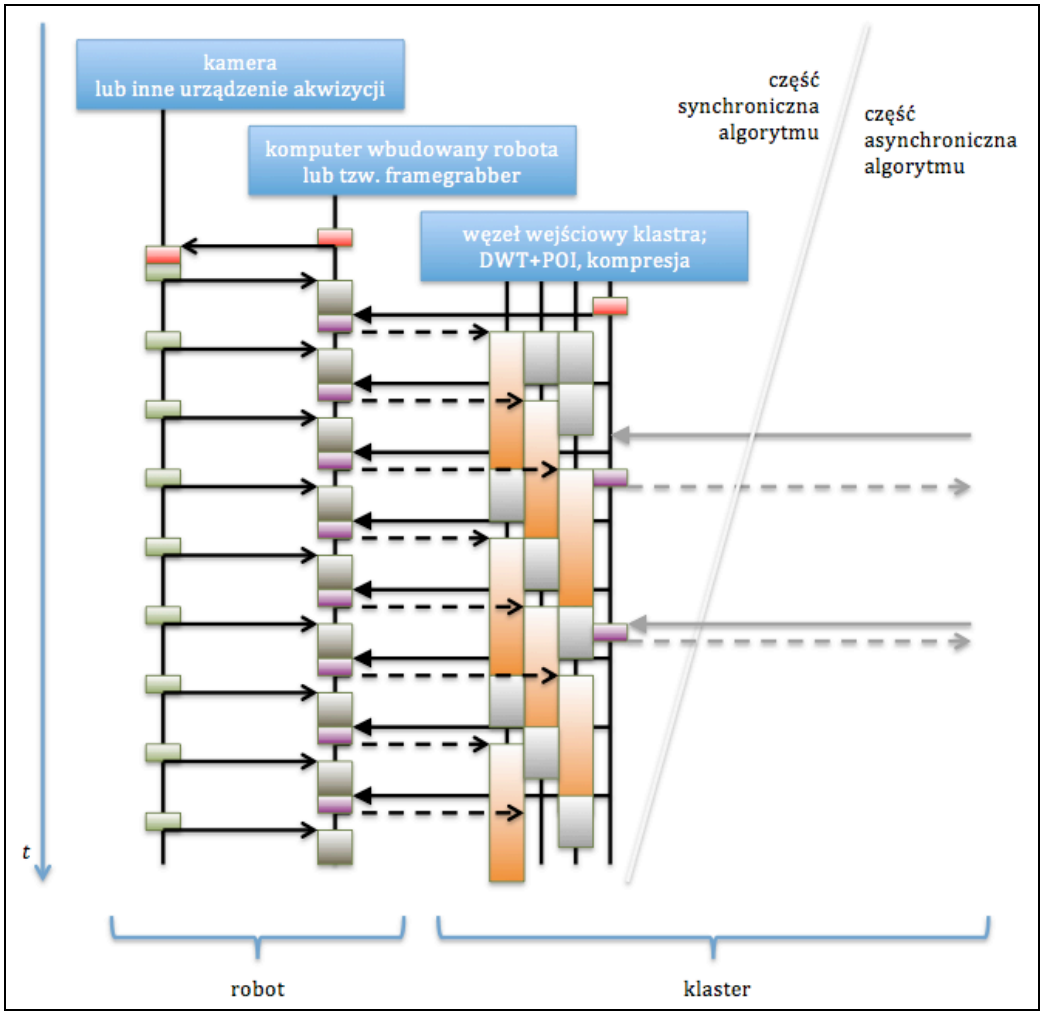


Rys. 4.9. Część synchroniczna rozproszonego (klastrowego) systemu wizyjnego, przedstawiona diagramem sekwencji. (opis w tekście)

Na rysunku powyżej (Rys. 4.9) zilustrowano część synchroniczną rozproszonego (klastrowego) systemu wizyjnego, przedstawioną diagramem sekwencji. Węzeł wejściowy klastra cyklicznie dokonuje przetwarzania wstępnego i konwersji DWT z POI oraz kompresji najnowszej klatki obrazu pobranej z robota (kolor pomarańczowy – obraz kolorowy RGB, kolor szary – macierz dwuwymiarowa – obraz monochromatyczny lub spreparowany obraz monochromatyczny danej cechy obrazu). Komputer wbudowany robota kontroluje pracę kamery, otrzymuje najnowszą klatkę obrazu, przechowuje ją i udostępnia na żądanie węzłowi wejściowemu klastra, nie wykonuje na niej żadnych innych operacji.

Komputer pokładowy robota mobilnego zwykle stanowi najsłabsze ogniwo w zakresie jego inteligencji i możliwości. Czas procesora komputera pokładowego jest bardzo cenny i z tego powodu zdecydowano się znaczną część przekształceń i operacji przenieść do węzła

wejściowego klastra komputerowego. Sprzęt komputerowy zamontowany na/w robocie przetwarza obraz jedynie w zakresie niezbędnym do jego pozyskania i odesłania do klastra. Wszystkie operacje związane ze wstępnym przygotowaniem, konwersją oraz kompresją, przeprowadzane są poza komputerem robota, w węźle wejściowym klastra (Rys. 4.9). Węzeł ten bezustannie przygotowuje kolejne wersje kolejnych klatek strumienia klatek wejściowych, a ponieważ trwa to znacznie dłużej niż czas pomiędzy kolejnymi pozyskanymi klatkami, właśnie ten komputer staje się wąskim gardłem czasu przetwarzania klatki (opóźnienia powodowanego przez system wizyjny). Aby zminimalizować skutki tej komplikacji postanowiono zrównoleglić również ten fragment algorytmu poprzez zastosowanie komputera wieloprocesorowego (lub wielordzeniowego) jako komputera węzła wejściowego klastra. Koncepcję tą zobrazowano na rysunku poniżej (Rys. 4.10).



Rys. 4.10. Część synchroniczna proponowanego rozproszonego (klastrowego) systemu wizyjnego. (opis w tekście)

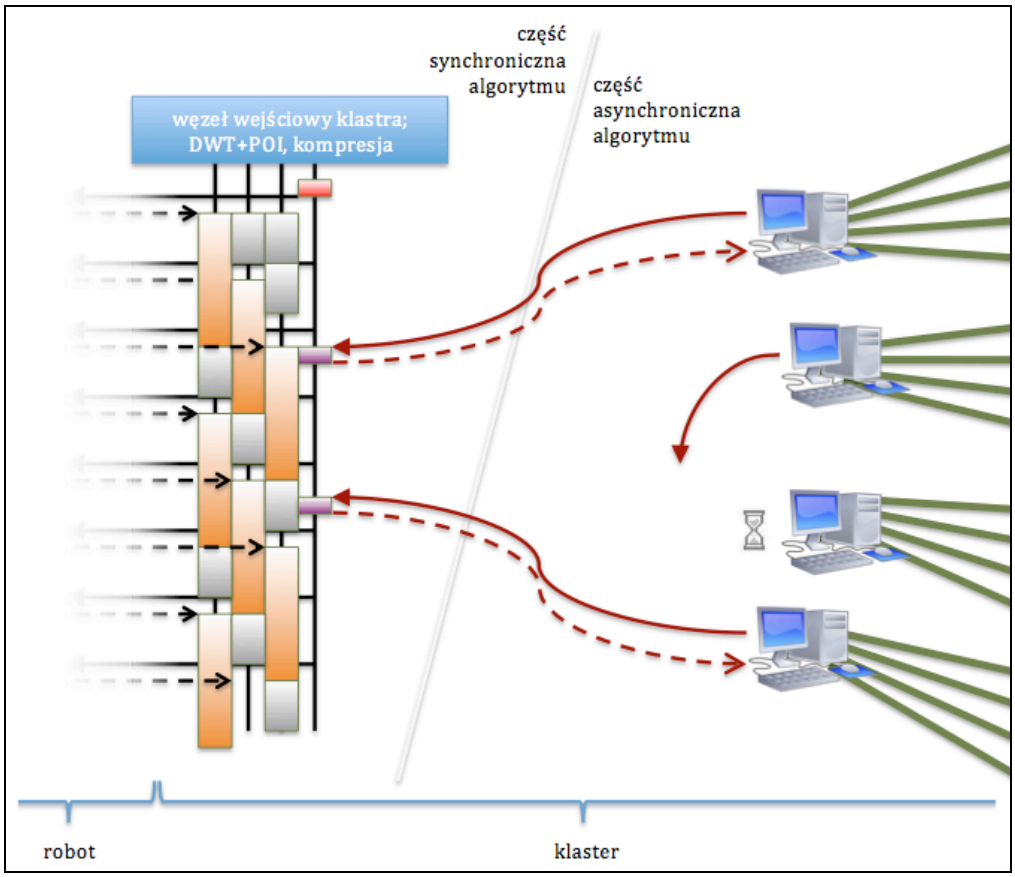
Rysunek Rys. 4.10 różni się od poprzedniego specyfikacją i algorytmem węzła wejściowego klastra – wymieniono je na wersję wielowątkową wykorzystującą współbieżność. Dzięki temu możliwe jest uruchomienie kilku operacji w tym samym czasie, na odrębnych procesorach/rdzeniach. Dodatkowo możliwe stało się wcześniejsze rozpoczęcie analizy kolejnej klatki (pomimo niezakończenia przetwarzania poprzedniej), co widoczne jest na rysunku[150] (technika ta funkcjonuje w literaturze pod nazwą *overlapping*). Wymiana algorytmu i maszyny węzła wejściowego klastra na wersję wspierającą równoległość skróciło czas obliczeń – przetwarzanie wstępne RGB odbywa się równolegle do wątków przetwarzania wstępnego monochromatycznych obrazów i macierzy cech oraz daje możliwość znaczącego zwiększenia częstotliwości pojawiania się nowych przetworzonych klatek obrazu po stronie klastra. Na rysunku zaznaczono co prawda cztery linie należące do bloku „węzeł wejściowy klastra", co mogłoby sugerować zastosowanie czterech procesorów/rdzeni, jednak liczba procesorów/rdzeni była również przedmiotem badań, których wyniki teoretyczne dostępne są poniżej, a empiryczne w rozdziale 4.5.

Począwszy od trzech współbieżnie realizowanych operacji w węźle wejściowym klastra wydzielono osobny wątek dla zadań komunikacji.

Zrównoleglenie pracy węzła wejściowego klastra oraz zastosowanie trybu asynchronicznego pomiędzy podsystemem synchronicznym przetwarzania wstępnego a klastrem komputerowym z podziałem funkcjonalnym doprowadziło do zjawiska, które w literaturze nazywane jest „osłabieniem Prawa Amdahla". W takiej sytuacji, zastosowanie asynchronicznego połączenia dwóch fragmentów algorytmu aplikacji współbieżnej zakłóca relację pomiędzy czasem trwania części sekwencyjnej zadań a przyspieszeniem.

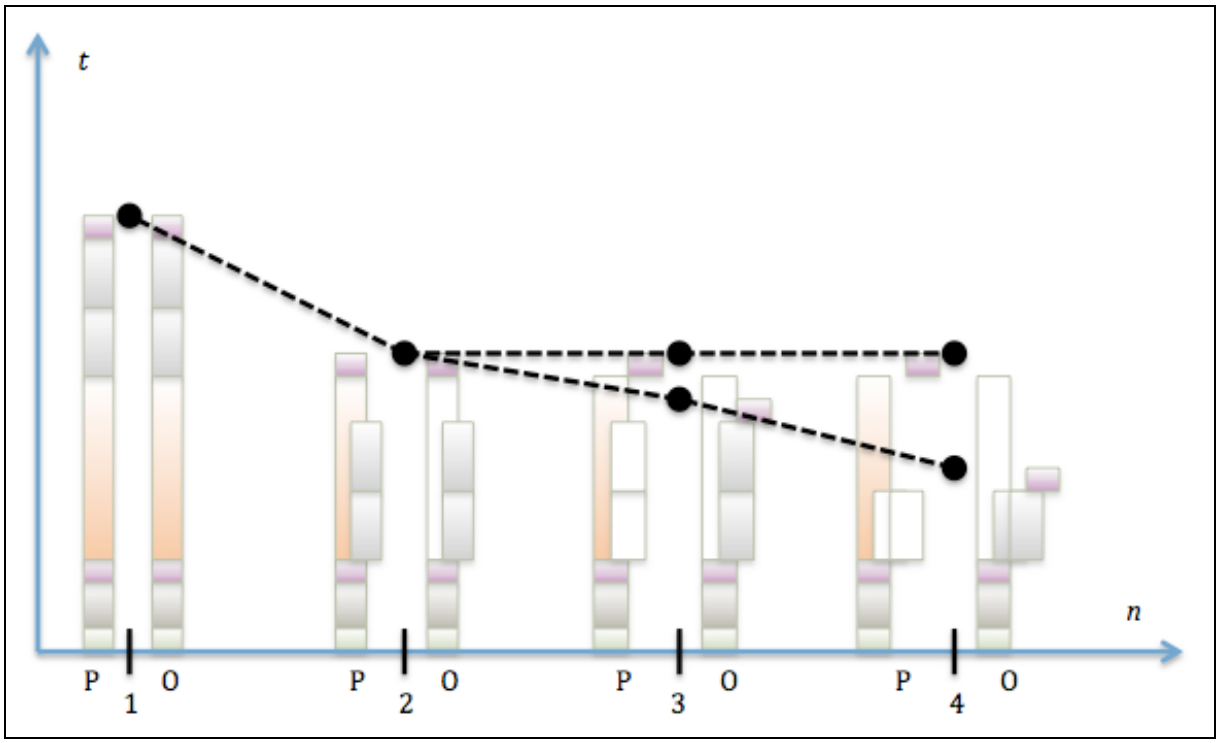
W zaproponowanym rozproszonym (klastrowym) systemie wizyjnym zdecydowano się na całkowite rozdzielenie części synchronicznej systemu od węzłów przetwarzających zadania rozumienia wizji i wnioskowania. Komunikacja pomiędzy tymi dwiema częściami systemu jest całkowicie asynchroniczna. Przewidywane korzyści z takiego rozwiązania przedstawiono na rysunkach poniżej (Rys. 4.12 i Rys. 4.13), a korzyści zaobserwowane na rzeczywistej eksperymentalnej implementacji omówiono w rozdziale 4.5.

[150] Pomarańczowe prostokąty, symbolizujące operacje przetwarzania wstępnego (bezkontekstowego, DWT z POI i kompresję) dla obrazu kolorowego, nakładają się na siebie w czasie.



Rys. 4.11. Część asynchroniczna proponowanego rozproszonego (klastrowego) systemu wizyjnego.

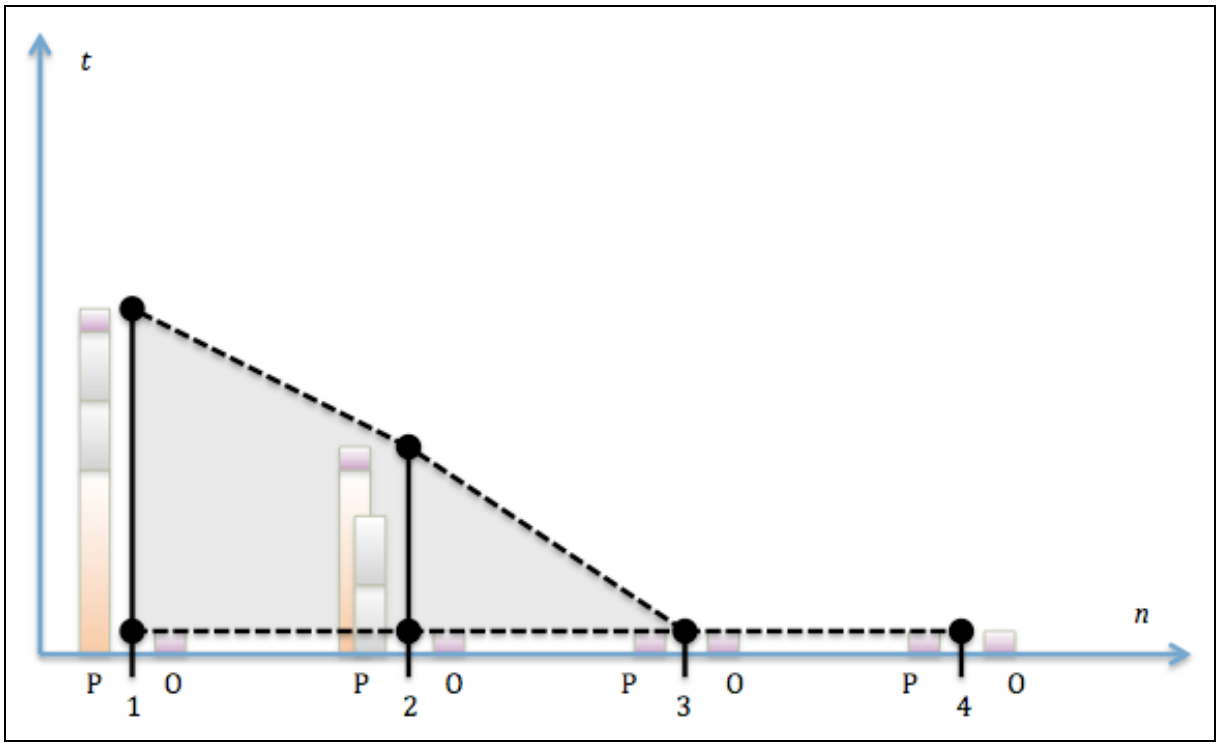
Rysunek poniżej przedstawia czas przejścia danych dotyczących klatki obrazu przez część synchroniczną systemu wizyjnego dla różnej liczby równolegle uruchomionych wątków węzła wejściowego klastra, z zaznaczeniem charakteru/rodzaju zadań i ich ziarnistości. Zwiększanie ilości wątków (/procesorów) powyżej pewnej wartości nie przynosi rezultatu, właśnie z uwagi na ziarnistość. Różne kolory oznaczają różne operacje (opisano je na wcześniejszych rysunkach). Dla jednego przypadku bywają dwa rozwiązania, gdy można rozróżnić przypadek pesymistyczny (P) – m.in. żądana jest klatka z kolorowym obrazem oraz przypadek optymistyczny (O) – m.in. dany węzeł żąda jedynie monochromatycznej reprezentacji danych danej klatki. Wartość czasu jest zależna od użytego w eksperymencie sprzętu, dlatego na osi rzędnych nie zaznaczono wartości.



Rys. 4.12. Czas przejścia danych dot. klatki obrazu przez część synchroniczną systemu wizyjnego dla różnej liczby równolegle uruchomionych wątków węzła wejściowego klastra.

Na podstawie równań (6) i (7) można określić szacowany całkowity czas opóźnienia rozproszonego (klastrowego) systemu wizyjnego (równanie (8)). Jest to czas odpowiedzi rozproszonego (klastrowego) systemu wizyjnego na zmiany obrazu znajdującego się w sąsiedztwie punktu POI (zmiana współrzędnych punktu POI również powoduje zmianę obrazu w jego sąsiedztwie). Opóźnienie jest najważniejszym spośród parametrów opisujących działanie systemu wizyjnego w kontekście czasu, ale nie jedynym. Drugim, również znaczącym aspektem jest częstotliwość, z jaką udostępniane[151] są nowe klatki obrazu systemowi wizyjnemu. Rysunek Rys. 4.13 przedstawia zależność pomiędzy ilością współbieżnych wątków wejściowego węzła klastra, wg rysunku Rys. 4.10, a czasem odpowiedzi sekwencyjnej części algorytmu na żądanie przesłania kolejnej klatki, pochodzące z dowolnego węzła klastra części asynchronicznej.

[151] Nie tylko pobierane (akwizycja), ale również wstępnie przetwarzane, konwertowane z użyciem DWT i POI, kompresowane i przygotowywane w buforze do wysłania.



Rys. 4.13. Czas odpowiedzi części synchronicznej algorytmu na żądanie przesłania jakiejkolwiek klatki (ostatniej pobranej) dla różnej liczby równolegle uruchomionych wątków węzła wejściowego klastra.

Proponowany system wizyjny ma stanowić uzupełnienie/rozszerzenie możliwości percepcyjnych („inteligencji") robota mobilnego. System ten projektowano z założeniem, że operacje krytyczne (równowaga, bezpieczeństwo, „odruchy bezwarunkowe") zostaną zaimplementowane w klasyczny sposób[152], a tylko dodatkowa funkcjonalność zostanie przeniesiona do nowego rozwiązania. W takim przypadku opóźnienie ma o wiele mniejsze znaczenie niż częstotliwość dostępności nowych (zmienionych) danych. Dlatego celowe jest zrównoleglenie algorytmu węzła wejściowego klastra, które umożliwi zastosowanie wspomnianego wcześniej *overlapping*'u. Na rysunku powyżej można zaobserwować diametralną poprawę czasu odpowiedzi wraz z zastosowaniem wydzielonego rdzenia/procesora do obsługi zadań komunikacji po stronie asynchronicznej. Ze względu na charakter i ziarnistość zadań jest to zasadne dopiero przy trzech wątkach, zatem w przypadku jednego i dwóch wątków czas odpowiedzi mieści się w opisanym w rozdziale zakresie. Wartość czasu jest zależna od użytego w eksperymencie sprzętu, dlatego na osi rzędnych ilustracji nie zaznaczono wartości. Weryfikację eksperymentalną przedstawiono w rozdziale 4.5.

[152] Tak jak to jest implementowane w istniejących robotach.

4.4. Eksperymenty implementacyjne

W tym rozdziale opisano[153] wytworzone narzędzia oraz przeprowadzane przy ich pomocy eksperymenty. Najcenniejszą wartością tego rozdziału, oprócz samej weryfikacji implementacyjnej opisywanych w rozdziale 3 pomysłów/technologii, są wyniki pomiarów czasu komunikacji wskazujące na celowość badań oraz potwierdzające tezę[154] pracy.

Przed przystąpieniem do prac implementacyjnych, w ramach przygotowań do eksperymentów, zaplanowano:

- sposób i kryteria oceny wyników → czas operacji,
- sposób wykonania pomiarów → klasyczny pomiar czasu RTC[155] najszybszym kodem dla danego języka programowania,
- sposób wykonania pomiarów transmisji do innego węzła sprzętowego → czas do powrotu potwierdzenia otrzymania kompletnych danych (pomiar czasu RTC w obrębie jednego komputera fizycznego),
- sposób mierzenia czasu operacji (sumaryczny czas wykonywania instrukcji danej operacji / rzeczywisty całkowity czas realizacji danej operacji w procesorze[156]) → rzeczywisty czas realizacji,
- sposób porównywania (1. W obrębie jednego komputera, 2. Pomiędzy odrębnymi komputerami) czasu realizacji algorytmu lub czasu komunikacji → używając tego samego kodu i tych samych procedur, różnicując działanie przy inicjalizacji operacji/transmisji.

4.4.1. Implementacja 1 – SSN, AV, tracking (Windows)

Pierwsza z prób implementacyjnych przeprowadzonych na potrzeby pracy nie zawiera żadnego z elementów opracowanej w pracy koncepcji, jest następstwem przeprowadzonego studium literatury i próbą zastosowania niektórych, najbardziej obiecujących, metod konwersji sposobu reprezentacji danych wejściowych. Wynikiem tego eksperymentu miała być odpowiedź na pytanie czy dane wejściowe w postaci (strumienia) klatek obrazu można zastąpić poprzez przesyłanie do klastra informacji dotyczących aktywności połączeń między warstwami ukrytymi sztucznej sieci neuronowej (SSN). Zarys weryfikowanej koncepcji

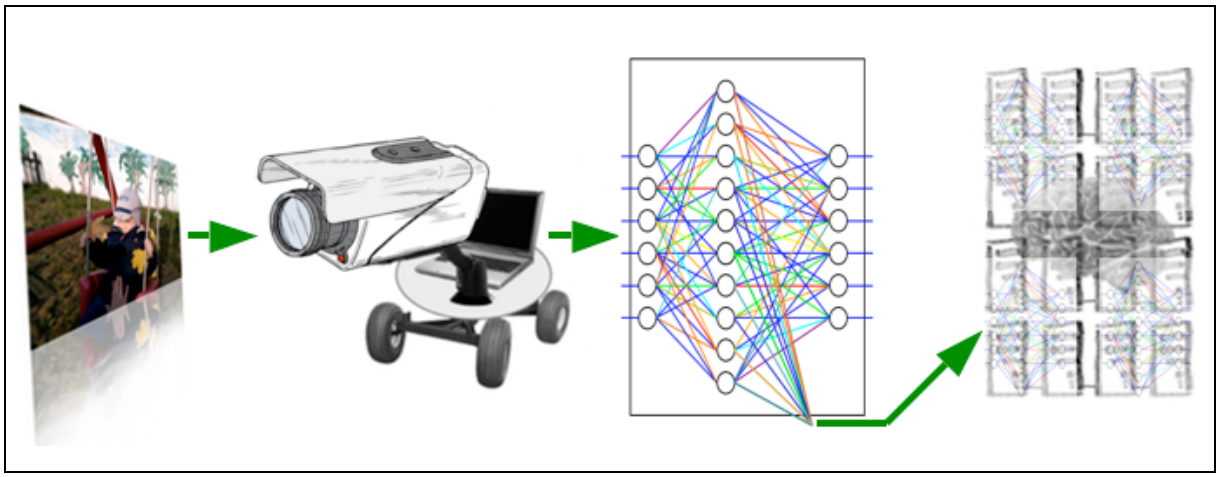
[153] w sposób jak najbardziej przystępny, a jednocześnie wystarczająco szczegółowy dla czytelnika chcącego odtworzyć przebieg i/lub wyniki eksperymentów

[154] patrz: strona 20.

[155] RTC – (*ang. Real Time Clock*) zegar czasu rzeczywistego

[156] rzeczywisty czas realizacji jest zawsze lub prawie zawsze dłuższy niż teoretyczny sumaryczny czas instrukcji, ponieważ system operacyjny obsługuje również przerwania sprzętowe

przedstawiono na rysunku Rys. 4.14. Ponieważ nie było wiadomo, jaka struktura[157] sieci jest najlepsza, postanowiono zaimplementować kod uniwersalny – umożliwiający łatwą modyfikację/wymianę sieci neuronowej.

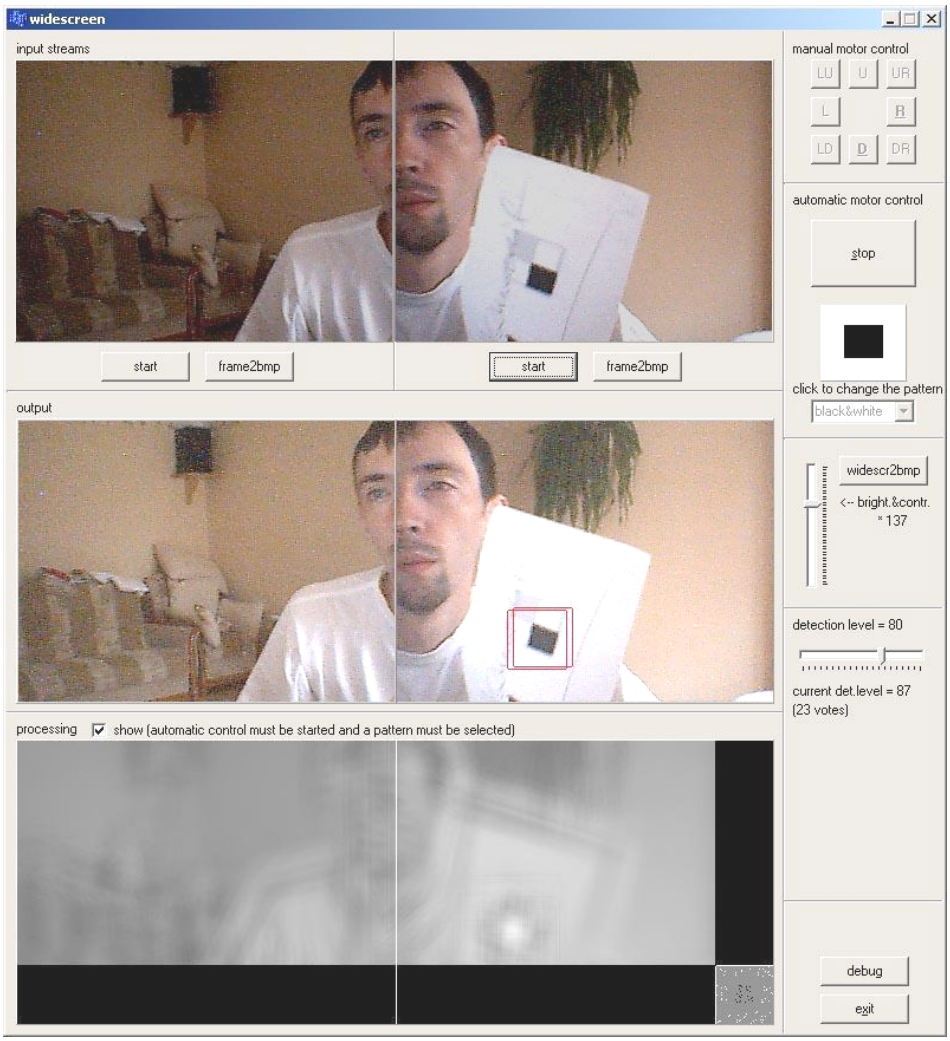


Rys. 4.14. Zarys koncepcji, do której miał prowadzić eksperyment opisywany w rozdziale 4.3.

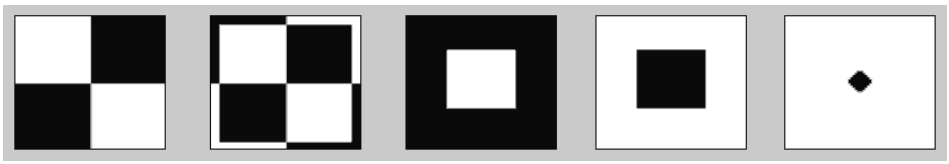
Mając na uwadze specyfikę postawionego zadania (umożliwianie swobodnego eksperymentowania ze sztucznymi sieciami neuronowymi na danych wejściowych pochodzących z systemu wizyjnego) – już sam interfejs użytkownika (Rys. 4.15) w prosty i czytelny sposób prezentuje najważniejsze aspekty przetwarzanego strumienia danych. Aplikacja umożliwia korzystanie z jednej kamery lub z dwóch, tworząc obraz panoramiczny konwertowany do postaci trójwymiarowej macierzy (640x240x3).

Formatka zawiera również predefiniowane elementy umożliwiające interakcję z implementowanym algorytmem SSN, które może dowolnie zmieniać programista. Przykładem jest tu rodzaj wzorca (wybierany spośród przedstawionych na rysunku kolejnym - Rys. 4.16) jaki ma rozpoznawać projektowana sieć neuronowa, oraz jego barwa (czarno-biały, czerwono-biały, zielono-biały, niebiesko-biały).

[157] ile warstw, ile neuronów w każdej z warstw, itd.



Rys. 4.15. Implementacja 1 – interfejs użytkownika aplikacji. (rysunek pochodzi z [114], opracowanie własne)



Rys. 4.16. Implementacja 1 – predefiniowane wzorce, które sieć może rozpoznawać w obrazie. (rysunek pochodzi z [114], opracowanie własne)

Powierzchnia ta została pomyślana jako miejsce pozyskiwania (porównywania) danych treningowych dla sieci neuronowej. Dzięki temu eksperymentowanie z własną sztuczną siecią neuronową stało się o wiele łatwiejsze – wszystko co potrzebne jest już w aplikacji przewidziane. Wystarczy wykonać następujące czynności:

1) zaprojektować i zaimplementować sieć neuronową według własnego pomysłu,
2) przy użyciu dowolnego algorytmu przeprowadzić uczenie sieci bazując na wzorcu (wzorcach) prezentowanym na panelu wzorców,

3) rozpocząć analizowanie źródła (danych przechowywanych we wspomnianej trójwymiarowej macierzy) w przewidzianym miejscu w kodzie algorytmu skanującego
4) zaprezentować wynik działania sieci

W przypadku zaimplementowanego testowo algorytmu (perceptronu jednowarstwowego prezentującego przykładowy sposób wykorzystania i działania stworzonego przez autorów narzędzia), w dolnym prawym rogu dolnej panoramicznej powierzchni (patrz: Rys. 4.15) pokazane jest aktualne rozlokowanie przestrzenne i wagi poszczególnych neuronów (kropka biała symbolizuje neuron aktywowany kolorem białym, czarna – czarnym).

Programista-projektant sieci neuronowej może w stworzonym narzędziu w rozmaity sposób prezentować wyniki działania swojego rozwiązania:

- Edytując wspomnianą wcześniej trójwymiarową macierz obrazu przetwarzanego, m.in. dorysowując ramkę, jak na rysunku Rys. 4.15, albo zaznaczając współrzędne – macierz ta jest bowiem wizualizowana na środkowym panelu każdorazowo po zakończeniu analizy kolejnej klatki obrazu.
- Może użyć dolnego panelu (całkowicie dowolnie).
- Jeżeli dysponuje nie tylko samym programem, ale również częścią mechaniczną stworzonego przez autorów[158] narzędzia, wówczas ma możliwość kontrolowania pracy silników i lasera wyjściami sieci neuronowej.

W tak przygotowanej aplikacji przeprowadzono szereg eksperymentów z sztucznymi sieciami neuronowymi o różnych topologiach i rozmiarach. Aby sieć była w stanie lokalizować w obrazie predefiniowane kształty (i rozróżniać je między sobą), należało włożyć sporo wysiłku w prawidłowe jej przygotowanie. O ile wyniki rozpoznawania są interpretowalne, o tyle trudno na tej podstawie prognozować jakość (skuteczność) rozpoznawania (przedmiotów świata rzeczywistego obecnych w scenie) w środowisku klastrowym. Dlatego postanowiono dalsze rozważania przeprowadzać przy użyciu innej aplikacji, która umożliwiłaby weryfikację algorytmu w środowisku klastrowym. Ponieważ kod implementacyjny tej aplikacji był tworzony pod system Windows, zdecydowano się na zupełnie nową implementację bazującą na systemach rodziny Linux (co umożliwi swobodne korzystanie z biblioteki LAM/MPI). Wynikiem tej decyzji jest „Implementacja 2” (rozdział 4.4.2) oraz „Implementacja 4” (rozdział 4.4.4).

[158] Część mechaniczna i elektroniczna została zaprojektowana i wykonana we współpracy ze Zbigniewem Zającem, współautorem platformy oraz publikacji [114]

„Implementacja 1” nie dostarczyła żadnych mierzalnych wyników, nie było to jej celem.

4.4.2. Implementacja 2 – AV/ROI, tracking (Linux)

Kolejna z przeprowadzonych prób implementacyjnych była tworzona ściśle na potrzeby badań. Od samego początku głównymi założeniami implementacji były: wydajny kod, sprzętowa akceleracja akwizycji, możliwość pracy jako węzeł klastra, możliwość pracy bez klastra. Z powodu licznych komplikacji (opisanych częściowo w rozdziale B.2) tylko niektóre z tych założeń udało się w tej wersji zrealizować. Wszystkie z nich spełniają dopiero implementacje „Próby implementacyjnej 4” i „Próby implementacyjnej 5”.

Wytworzona aplikacja składała się z jednego okna pokazującego obraz wejściowy, przetwarzany lub wyjściowy oraz z okna konsoli poleceń. Po uruchomieniu jej w terminalu, okno terminala stawało się konsolą poleceń zmieniających tryby pracy algorytmu i/lub okna podglądu.

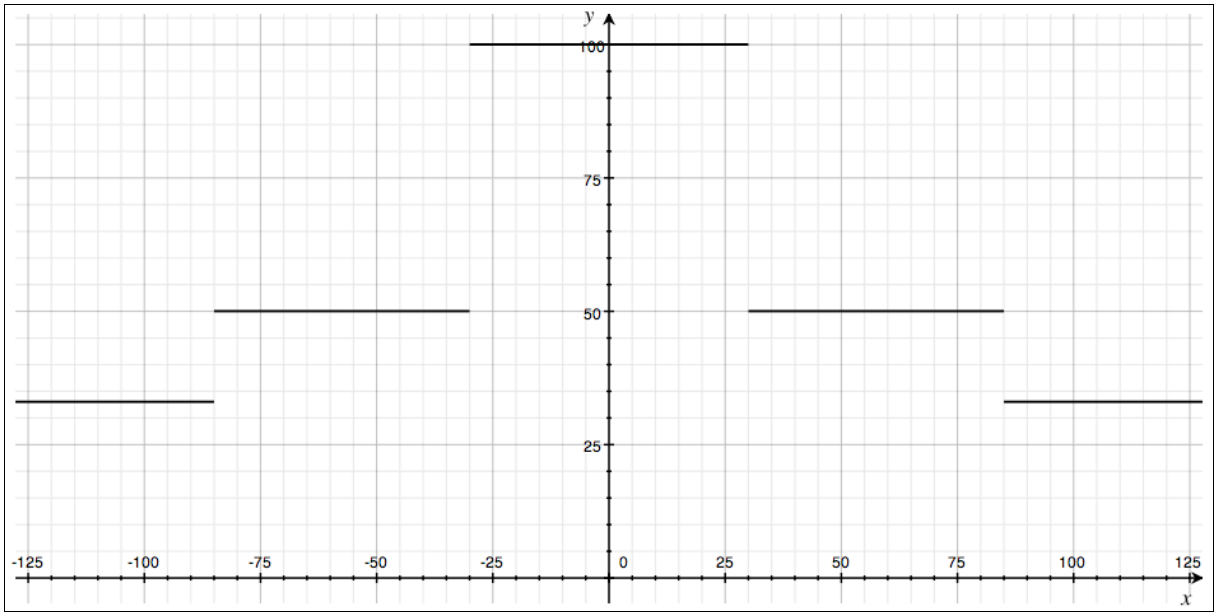
W aplikacji zaimplementowano pierwsze zwiastuny podziału funkcjonalnego zadań: niezależne fragmenty algorytmu generowały obraz różnicowy i „wykrywały kolory”[159].

Próbowano zaimplementować pokazaną na rysunkach (Rys. 4.18 i Rys. 4.17) koncepcję ROI, co udokumentowano w publikacji [122].

Umiarkowany postęp w implementowaniu utrzymywał się aż do momentu, w którym w wyniku przeprowadzanej na bieżąco analizy literaturowej postanowiono zastosować transformatę DWT jako metodę redukcji ilości danych opisujących obraz. Ponieważ autor miał wówczas o wiele większe doświadczenie w programowaniu pod systemem Windows, zdecydowano zawiesić prace nad „Próbą implementacyjną 2” na czas analizy zysku z ewentualnej implementacji oraz opracowania kompletu procedur manipulacji danymi.

[159] Pod roboczym pojęciem „wykrywania kolorów” kryje się algorytm określający odmienność danego piksela (pod względem koloru) od średniej w obrazie. Piksel o zauważalnie większej luminancji, chrominancji lub barwie podstawowej palety RGB otrzymywał odpowiednio większą wagę w zdefiniowanej w ten sposób macierzy dwuwymiarowej.

Rys. 4.17. Pierwsza z wizualizacji zastosowania koncepcji ROI do redukcji ilości danych pochodzących z systemu wizyjnego korzystającego z klastra. (rysunek pochodzi z [122], opracowanie własne)



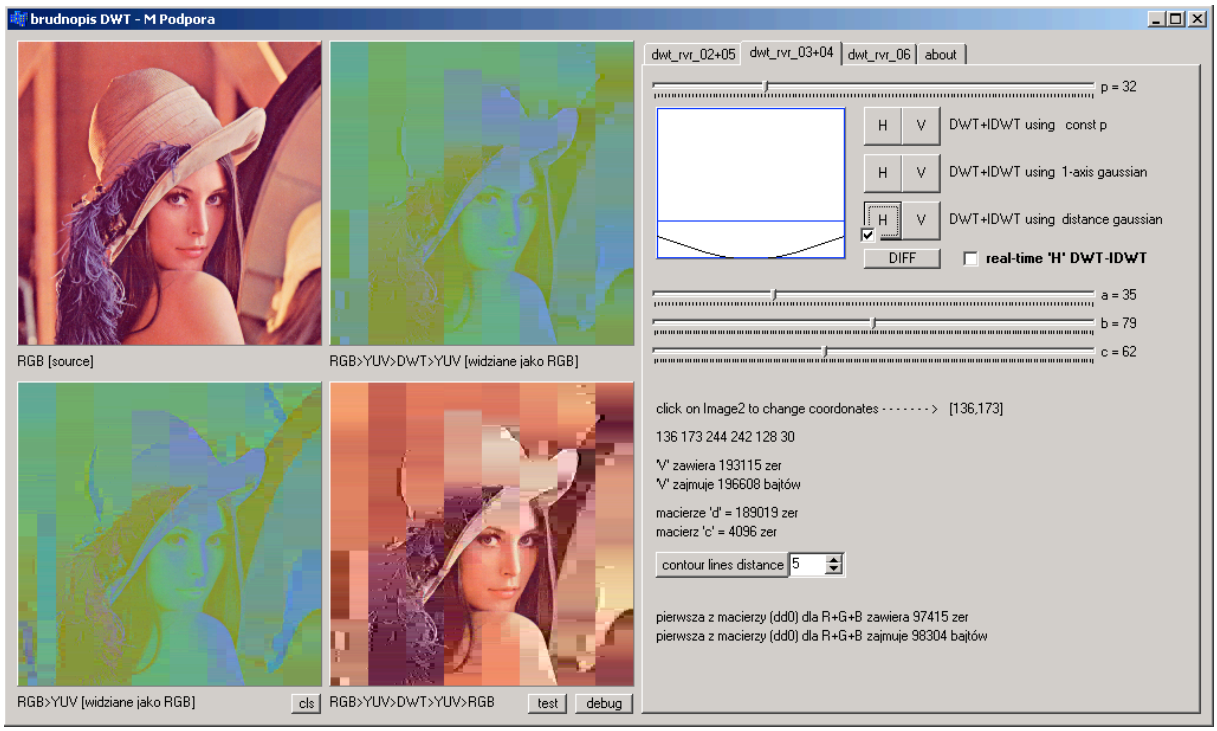
Rys. 4.18. Wykres poglądowy sposobu określania procentowego udziału pikseli obrazu wejściowego w obrazie wyjściowym – w funkcji odległości od punktu fiksacji. Oś odciętych – odległość wyrażona w liczbie pikseli, oś rzędnych – procentowy udział pikseli obrazu.

Trudno mówić o mierzalnych korzyściach z przeprowadzenia „Próby implementacyjnej 2”, należy jednak pamiętać, że ostatnia wersja jej implementacji stała się pierwszą wersją „Próby implementacyjnej 4”.

4.4.3. Implementacja 3 – DWT, ROI, POI, YUV, RLE (Windows)

Ten etap miał zdecydowanie największe znaczenie w procesie weryfikacji implementacyjnej przedstawionych w rozdziale 4.5 rozważań. Początkowo miał służyć jedynie technicznemu przyspieszeniu procesu implementacji DWT oraz ROI, z uwagi na bardziej przyjazne środowisko programistyczne, szybko okazało się jednak, że praktyczna implementacja kodu aplikacji „Próby implementacyjnej 3" będzie idealnym narzędziem testowania[160] dalszych pomysłów.

Interfejs aplikacji, przedstawiony częściowo na rysunku Rys. 4.19, posiada multum opcji, dlatego tutaj omówione zostaną tylko najistotniejsze, natomiast bardziej szczegółowy opis umieszczony jest w rozdziale B.3.



Rys. 4.19. Implementacja 3 – Jedna z zakładek graficznego interfejsu użytkownika aplikacji.

Praktyczna implementacja kodu aplikacji powstałej na potrzeby „Próby implementacyjnej 3" nie tylko była użyta do prototypowego implementowania funkcji i procedur konwersji obrazu z użyciem DWT i ROI. Była pierwszą aplikacją, w której zaimplementowano omawiany w rozdziale 3.2.2 koncept POI – na powyższym rysunku (Rys.

[160] Większość testów (nie dotyczących klastra, a dotyczących sposobu reprezentacji obrazu) przeprowadzono właśnie z użyciem implementacji kodu aplikacji „Próby implementacyjnej 3". Najważniejsze spośród nich, które z racji istotności merytorycznej oraz metodologii badań urosły do rangi eksperymentów, przedstawiono poniżej.

4.19) widoczne są m.in. suwaki modyfikujące parametry równań[161] (1) i (2), oznaczone odpowiednio: *a*,*b*,*c* oraz dwuwymiarową uproszczoną wizualizację otrzymanej z nich funkcji. Wyniki konwersji można zobaczyć na panelach po lewej stronie, klikając odpowiedni przycisk w odpowiedniej zakładce. Aplikacja umożliwia analizę i wizualizację następujących kontekstów obrazu wejściowego:

- transformata DWT z wizualizacją transformaty odwrotnej IDWT dla stałej wartości parametru *threshold*,
- transformata DWT krok po kroku z weryfikacją i wizualizacją macierzy współczynników i wyniku transformaty odwrotnej IDWT,
- kilka różnych prostych falek[162] do użycia w transformacie DWT,
- wizualizacja funkcji gęstości prawdopodobieństwa macierzy współczynników transformaty DWT dla każdego poziomu lub wszystkich razem,
- wizualizacja funkcji gęstości prawdopodobieństwa macierzy współczynników transformaty DWT dla każdego poziomu wraz z nałożoną wizualizacją dwuwymiarowej funkcji parametru *threshold*,
- wizualizacja postępu konwersji „w zwolnionym tempie”,
- wizualizacja obrazu różnicowego: przed-po konwersji (wizualizacja zniekształceń informacji),
- możliwość zmiany myszką współrzędnych punktu POI,
- możliwość zmiany parametrów funkcji parametru *threshold* z wizualizacją dwuwymiarową otrzymanej funkcji i wizualizacją zastosowania jej trójwymiarowej wersji na przetwarzanym obrazie,
- możliwość odczytania ilości zer w macierzach współczynników (szczególnie na etapie proponowania algorytmów kompresji),
- możliwość pracy na obrazie przekonwertowanym do modelu przestrzeni barw YUV,
- możliwość kompresji przekonwertowanego obrazu różnymi wersjami algorytmu RLE (*ang. Run-Length Encoding*) – wyniki opisano w publikacji [110],
- możliwość zapisania do pliku danych obrazu w postaci mapy bitowej,
- możliwość zapisania do pliku danych obrazu w postaci zrzutu pamięci macierzy współczynników transformaty DWT,
- możliwość zapisania do pliku danych obrazu w postaci zrzutu pamięci skompresowanych macierzy współczynników,
- oraz wiele innych.

[161] patrz: strona 45

[162] „prostych” w kontekście stosowania do transformaty dyskretnej – „falek” w postaci krótkich dyskretnych serii współczynników

To właśnie „Implementacja 3" umożliwiła analizę i ocenę jakościową[163] zaproponowanej metody reprezentacji danych. Praktycznie wszystkie wyniki, oprócz tych dotyczących współpracy z klastrami, zostały pozyskane z tej aplikacji.[164]

Najistotniejszymi spośród opublikowanych wyników przeprowadzanych eksperymentów są:

- tabela porównująca rozmiar pamięci jaki zajmuje obraz wejściowy i przekonwertowany z użyciem DWT i POI (uproszczona wersja tabeli umieszczonej w rozdziale 3.2.2 – Tabela 3.2) [45]
- podsumowanie i ocena wyników eksperymentów związanych z kompresją RLE obrazów konwertowanych z użyciem DWT i POI [110]
- tabela porównująca obrazy konwertowane z użyciem DWT i POI dla dwóch różnych modeli przestrzeni barw: YUV i RGB [113], z której można wysnuć jeden wniosek – wybór modelu przestrzeni barw nie ma wpływu na korzyści płynące z kompresji RLE obrazów konwertowanych z użyciem DWT i POI.

Implementacja praktyczna kodu „Próby implementacyjnej 3" przyczyniła się nie tylko do założonego przyspieszenia implementacji w ramach „Próby implementacyjnej 2" (później „Próby implementacyjnej 4"), lecz nawet po wdrożeniu zdobytego doświadczenia do wspomnianych implementacji była nadal rozwijana (równolegle, w czasie „Próby implementacyjnej 4") by stymulować jej postęp oraz dostarczać nowych inspiracji.

Wszystkie zaimplementowane i przetestowane pomysły były wdrażane (migrowane) do praktycznej implementacji „Próby implementacyjnej 4" celem sprawdzenia danego rozwiązania w środowisku klastrowym.

4.4.4. Implementacja 4 – (SSN, DWT, POI,) ROI, YUV, MPI (Linux)

„Implementacja 4" jest niemniej istotna dla pracy niż „Implementacja 3", ponieważ to właśnie w niej doświadczalny kod był uruchamiany w wersji zarówno samodzielnej jak i klastrowej, dając jasne i czytelne porównanie korzyści (lub strat) płynących z zastosowania danego rozwiązania. To również tutaj uruchamiano pierwsze próby implementacyjne opisywanego w rozdziale 3.6 układu regulacji (wirtualnego odpowiednika ruchów sakkadowych gałki ocznej).

[163] Pojęcie „analiza jakościowa" zostało tu użyte w sensie zdefiniowanym przez Bronisława Malinowskiego – badania/eksperymenty przeprowadzane są na dobrze wyselekcjonowanej próbie, miast próby dużej i przypadkowej potrzebnej przy tzw. analizie ilościowej.

[164] Czyli m.in. wyniki i porównania związane z DWT, ROI, POI, RLE.

Pierwszym z eksperymentów przeprowadzonych przy użyciu „Próby implementacyjnej 4" był pomiar czasu transmisji danych pojedynczej klatki obrazu – w obrębie jednego komputera PC oraz do innego węzła klastra. Eksperyment był przeprowadzany w warunkach rzeczywistych, na komputerach PC segmentu „ekonomicznego". Taki sposób testowania kodu[165] ma jedną przewagę nad tradycyjnymi metodami określania jakości kodu[166] - uwzględnia czynniki niezależne od samego kodu, lecz wynikające z normalnej pracy systemu komputerowego[167]. Tak uzyskane wartości w ogromnym stopniu zależą m.in. od konfiguracji danego systemu komputerowego, a co za tym idzie, mogą być trudne do odtworzenia. Mimo to „wynik eksperymentalny" jest cenny z tego względu, że pokazuje realną jakość pracy algorytmu w danym systemie komputerowym – dokładnie tak, jak będzie to widział/odczuwał użytkownik.

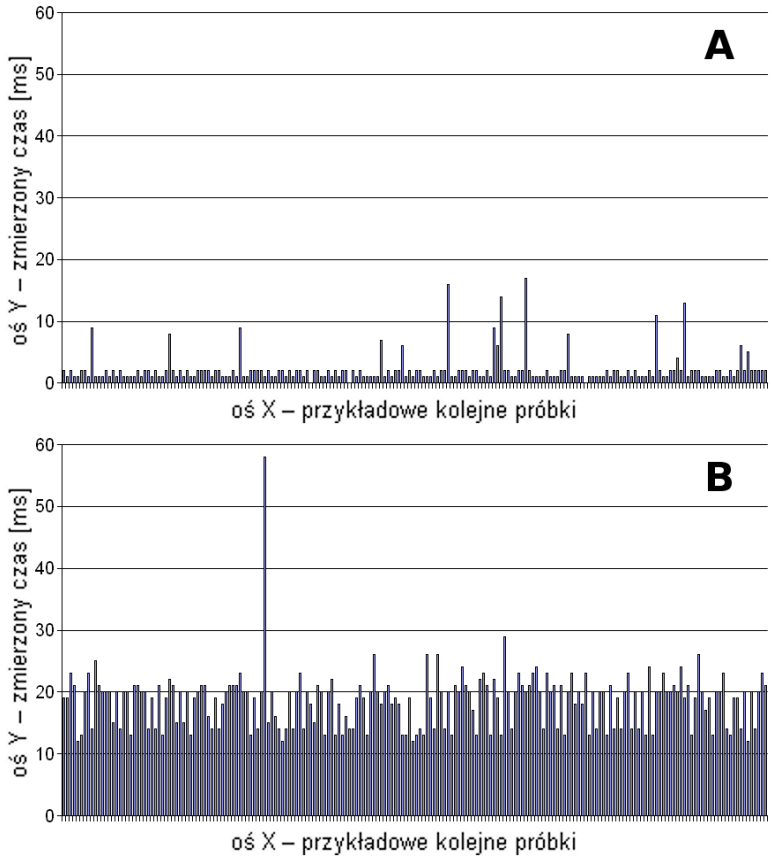
Na potrzeby eksperymentu (tego oraz kolejnych) w kodzie implementacyjnym „Próby implementacyjnej 4" zawarto kod inicjalizacji środowiska, bibliotek, podsystemu graficznego (SDL), kamery oraz klastra (LAM/MPI) uruchamiany w identyczny sposób dla każdego wariantu uruchomieniowego danego eksperymentu. Testowany fragment kodu był implementowany „w jednym miejscu", w stosownych funkcjach i procedurach, a w trakcie testu uruchamiane były jedynie odwołania do zaimplementowanych metod. Dzięki temu nawet użycie pamięci było maksymalnie zbliżone w obu weryfikowanych przypadkach.

Zadaniem **pierwszego eksperymentu** nie było określenie, który z mierzonych czasów będzie krótszy, gdyż to jest oczywiste. Istotne było określenie skali problemu, porównanie tych czasów – aby możliwe było zastanowienie się nad strategią rozwiązania problemu czasu komunikacji. Wyniki pomiarów, zasygnalizowane w publikacji [103] a omówione dokładniej w publikacji [99], prezentuje poniższy rysunek (Rys. 4.20).

[165] Testowanie „eksperymentalne" – stosowane u dzisiejszych producentów oprogramowania jako podstawowy (czasami nawet jedyny) sposób weryfikacji jakości kodu.

[166] Mowa tu o wskaźnikach optymalności kodu (i jego optymalizacji) opartych o wyznaczaną złożoność obliczeniową algorytmu.

[167] Czyli sprzętu, sterowników, systemu operacyjnego a nawet dodatkowych, nieprzewidywalnych czynników, takich jak inne programy użytkownika uruchomione w tle, czy obciążenie łącza transmisyjnego.



Rys. 4.20. Rzeczywisty czas komunikacji dla przykładowego systemu komputerowego. Przesyłane dane to jedna klatka obrazu 320x240pikseli, 24bpp. Dane przesyłano: A) do odrębnego wątka w obrębie jednego komputera (średni czas = 2,08ms) B) do odrębnego węzła klastra (średni czas = 18,60ms)

Przedstawione na powyższym rysunku pomiary to czas, jaki upłynął podczas przetwarzania komendy „MPI_Send”:

```
MPI_Send(m_xyc_out,          //message buffer
         3*320*240,          //data items
         MPI_CHAR,           //data type
         rank_i,             //dest.node no.
         1,                  //user chosen worktag message tag
         MPI_COMM_WORLD);    //default communicator
```

dla dwóch przypadków:

- gdy węzeł docelowy znajduje się w tym samym komputerze (ponieważ jest to platforma wieloprocesorowa/wielordzeniowa lub ponieważ wymuszono uruchomienie kilku węzłów na tym komputerze)
- gdy węzeł docelowy znajduje się w innym komputerze[168]

W wyniku eksperymentu otrzymano[169] odpowiednio:

- 2,08 ms

[168] Podczas tego eksperymentu komputery były połączone siecią standardu Fast Ethernet

[169] średnia z 200 pomiarów

- 18,60 ms

dla eksperymentu przedstawionego na rysunku Rys. 4.20. Dla porównania, na rysunku Rys. 1.4 zamieszczono wyniki eksperymentu powtórzonego dla innej konfiguracji sprzętowej:

- 1,96 ms
- 16,09 ms

Pierwszy eksperyment bardzo wyraźnie zakreślił sedno problemu – przesyłając 25 klatek obrazu w ciągu jednej sekundy samo przesyłanie danych (do tylko jednego węzła) zajęłoby prawie połowę czasu komputera. Oczywiście „wynik" ten można poprawić na wiele sposobów: używając nowoczesnego sprzętu, optymalizując kod, używając kompresji obrazu, m.in. Niestety żadne z tych rozwiązań nie da efektów tak dobrych, by możliwe było rozesłanie obrazu wysokiej jakości do kilku/kilkunastu węzłów klastra.[170] Należy pamiętać, że komputer robota mobilnego musi obsługiwać szereg funkcji i zapewnić wszystkie przewidziane funkcjonalności robota – nie można przeznaczać połowy czasu procesora na rozsyłanie obrazu. Oczywiste stało się, że to właśnie tu zlokalizowana jest jedna z barier rozwoju rozproszonych (klastrowych) systemów[171] wizyjnych. Konsekwencją tego eksperymentu były dalsze prace zmierzające do rozwiązania tego problemu.

Przez pewien okres próbowano zaimplementować sztuczne sieci neuronowe w węzłach klastra, jednak z uwagi na brak sukcesów[172] wątek ten porzucono.

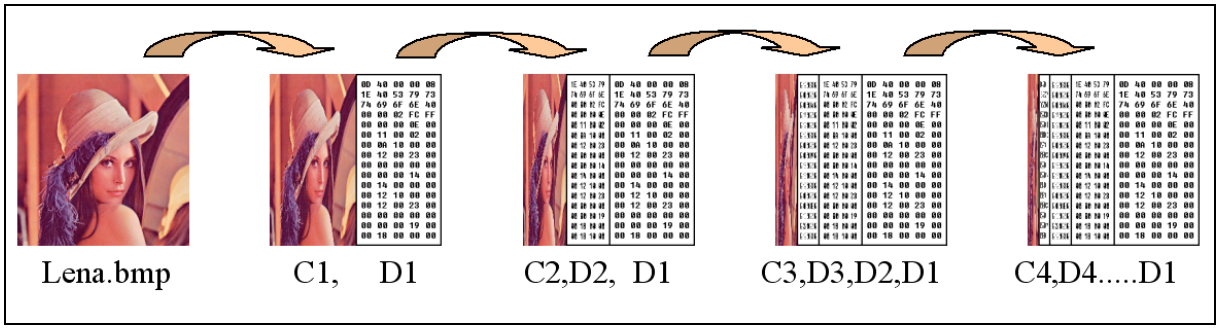
Po wstępnym przetestowaniu algorytmów konwersji obrazu z użyciem DWT i ROI (w aplikacji wytworzonej w ramach „Próby implementacyjnej 3") przeprowadzono **drugi eksperyment**, jednak jego wyników nie utrwalono w żadnej publikacji, ponieważ były na tyle obiecujące, że natychmiast przystąpiono do implementacji nowoopracowanego algorytmu POI. Jak pokazuje rysunek Rys. 4.21, samo użycie algorytmu DWT nie implikuje bezpośredniego zysku w postaci zmniejszenia ilości danych opisujących obraz.

[170] Dotyczy zastosowań innych niż przemysłowe, wymagających sporej ilości rozpoznawalnych obiektów na złożonym (niejednolitym, zmiennym) tle.

[171] Użytego tutaj terminu „rozproszony (klastrowy) system wizyjny" nie należy mylić z terminem „rozproszony system wizyjny" (*Distributed Vision System, DVS*) (system rozproszonej wizji). W systemach DVS używa się wielu kamer obserwujących scenę z różnych lokalizacji [24]. Termin DVS nie dotyczy komputerów klastrowych. W publikacji [82] w przystępny sposób podsumowano zalety i wady dziś projektowanych systemów DVS (wielokamerowych) próbując rozwiązać niektóre z problemów.

[172] SSN otrzymywały na swoje wejścia piksele fragmentu obrazu różnicowego, niestety nie udało się uzyskać satysfakcjonującej skuteczności rozpoznawania jakiegokolwiek obiektu – zapewne z powodu zbyt dużych sieci oraz stosunkowo trudnych (rzeczywistych) danych wejściowych.

Dekompozycja obrazu na macierze współczynników w najlepszym wypadku[173] zajmuje tyle samo bajtów co obraz wejściowy.



Rys. 4.21. Macierze współczynników podczas algorytmu DWT.

Aby implementacja algorytmu DWT przyniosła korzyść w postaci zmniejszenia ilości danych opisujących obraz, niezbędna jest zmiana reprezentacji i/lub użycie kompresji. Najbardziej naturalną modyfikacją zawartości macierzy współczynników jest użycie parametru *threshold*[174]. Wartości w macierzy mniejsze od *threshold* są zerowane (co powoduje utratę części informacji, dlatego według wytycznych standardu JPEG2000 nie jest to etap obowiązkowy). Cechą charakterystyczną macierzy współczynników transformaty DWT jest o wiele większa ilość zer niż w pliku obrazu wejściowego (porównaj: rozdział 3.2.2). Umożliwia to zastosowanie rozmaitych metod kompresji (stratnej lub bezstratnej). **Trzeci eksperyment** miał na celu weryfikację jednego z kluczowych dla niniejszej pracy kwestii: Czy stosując opisane tu przekształcenia (DWT z POI) możliwe będzie przesyłanie danych wejściowych rozproszonemu (klastrowemu) systemowi wizyjnemu? Przeprowadzono testy porównawcze czasu transmisji pojedynczych klatek obrazu w trzech wersjach/ /postaciach:

- w postaci macierzy obrazu wejściowego 320x240x3 (R,G,B),
- w postaci przekonwertowanej z użyciem DWT z POI i skompresowanej RLE,
- w postaci przekonwertowanej z użyciem DWT z POI i skompresowanej RLE oraz *entropy-coding*.

[173] Przy konwersji samych linii poziomych (tak jak to przedstawiono na rysunku Rys. 4.21) lub pionowych. W implementacji docelowej można pozostawić taki sposób konwersji z uwagi na bardzo małe użyte wartości lokalne threshold, można też rozbudować algorytm do konwersji dwu- lub trzyetapowej (w poziomie, pionie oraz 45º – taki sposób konwersji DWT znany jest m.in. z pakietu „Wavelet toolbox" środowiska „Matlab")

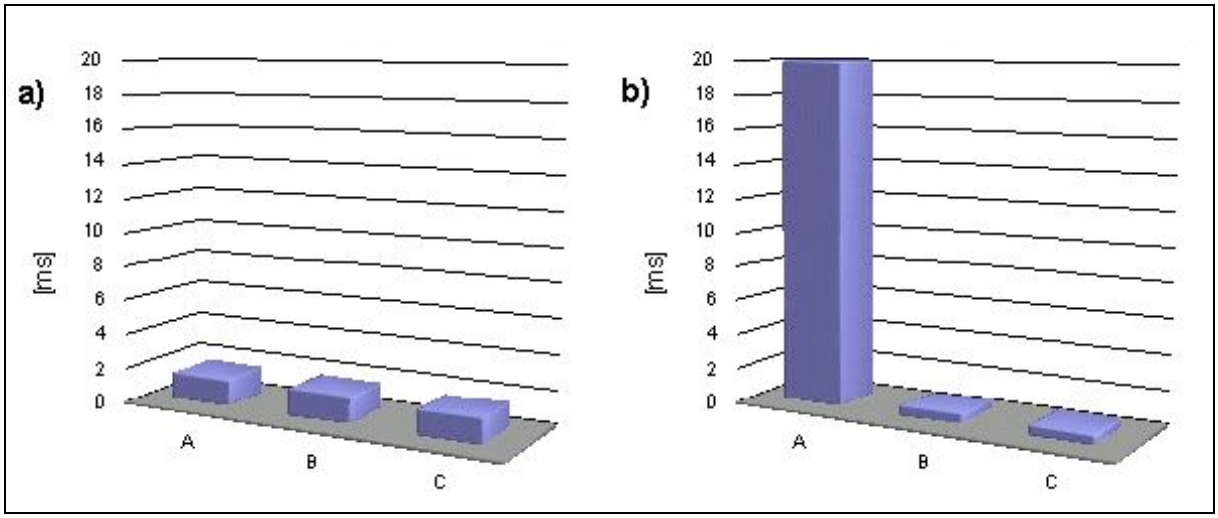
[174] W tradycyjnym podejściu jest to wartość stała dla całego obrazu (DWT), stała dla obrazu i inna, również stała dla jego predefiniowanego fragmentu (DWT z ROI). W proponowanym podejściu wartość parametru threshold definiowana jest opisaną wcześniej funkcją, zależną od odległości od punktu o dowolnych współrzędnych definiowanych osobno dla każdej klatki obrazu.

Kompresję RLE zaimplementowano zgodnie z definicją, natomiast kompresję *entropy-coding* zaimplementowano w najbardziej uproszczonym zakresie – z użyciem jednobitowego prefiksu, który oznaczał:

0) zakodowany (zastąpiony) bajt (lub bajty[175]) mają wartość zero
1) zakodowany (określony po prefiksie) bajt (lub bajty[175]) mają wartość różną od zera

Ten sposób kodowania został na etapie projektowania opisany w publikacji [110], lecz zabrakło[176] tam wyników porównania czasu komunikacji. Eksperymenty przeprowadzone z użyciem aplikacji opracowanej na potrzeby „Próby implementacyjnej 4" dostarczyły wyników przedstawionych na poniższych rysunkach (Rys. 4.22 i Rys. 4.23), opublikowanych w [99].

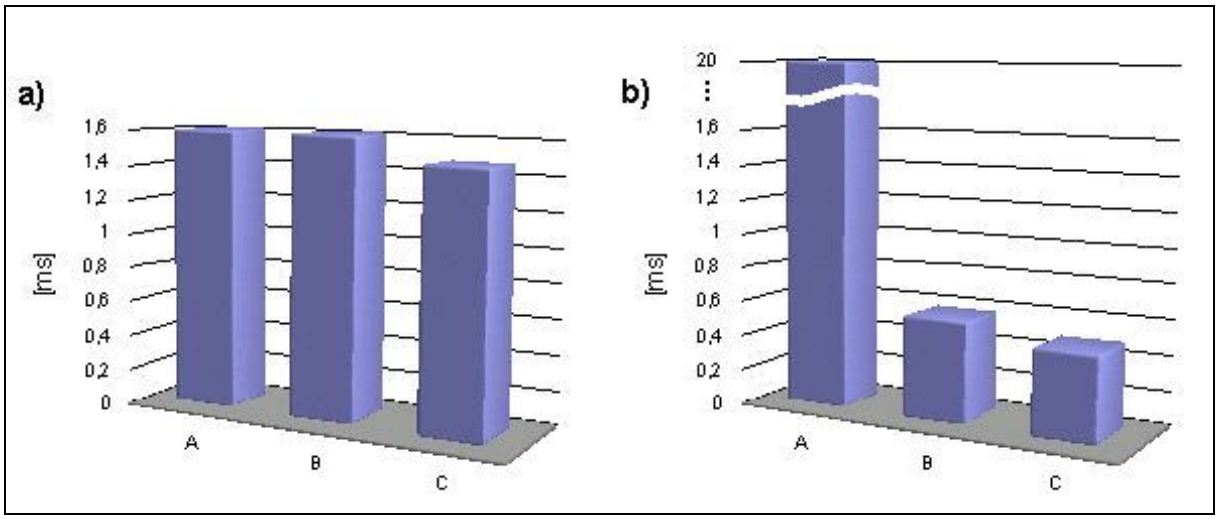
Podobnie jak w poprzednich eksperymentach przeprowadzonych z użyciem „Próby implementacyjnej 4", tak również i tu eksperyment był przeprowadzany przy pomocy tej samej konfiguracji i tego samego kodu – zarówno dla testów lokalnych jak i klastrowych.



Rys. 4.22. Uśredniony czas przesyłania macierzy danych opisujących jedną klatkę obrazu (średnia z 250 pomiarów): a) w obrębie jednego komputera, b) do innego węzła klastra; A- obraz nieskompresowany, B- obraz przekonwertowany DWT z kompresją RLE, C- obraz przekonwertowany DWT z kompresją RLE oraz *entropy-coding*. (rysunek pochodzi z [99], opracowanie własne)

[175] Kompresja RLE może rozszerzyć zakres prefiksu na więcej niż jeden bajt

[176] Publikacja [110] była tworzona na etapie prac z aplikacją „Próby implementacyjnej 3", która nie obsługiwała klastra. Dopiero praca z aplikacją „Próby implementacyjnej 4" dostarczyła wyników, które opublikowano w [99].



Rys. 4.23. Wartości pokazane na rysunku poprzednim, po przeskalowaniu umożliwiającym dokładne porównanie różnic; a) A=1,60 ms , B=1,61 ms , C=1,47 ms ; b) A=19,97 ms , B=0,59 ms , C=0,49 ms (rysunek pochodzi z [99], opracowanie własne)

Przesłanie do innego węzła klastra jednej klatki obrazu wejściowego systemu wizyjnego w niekonwertowanej postaci macierzy 320x240x3 trwa[177] średnio około 20 ms (Rys. 4.22 „b)A"). Zestawienie tego wyniku z wynikami widocznymi po lewej stronie rysunku (Rys. 4.22 „a)") przedstawia przyczynę niepowodzeń w odnotowanych w historii [21] próbach użycia klastra w systemach wizyjnych.

Przesłanie do innego węzła klastra jednej klatki obrazu przekonwertowanego według opisanych w pracy i publikacjach wytycznych powoduje skrócenie tego czasu do[177] średnio około 0,5 ms (Rys. 4.23 „b)C"). Jest to wynik bardzo zachęcający – przy tak krótkim czasie przesyłania danych rozesłanie ich do kilku węzłów klastra nie jest problemem.[178]

Ponieważ wyniki eksperymentów przeprowadzonych przy użyciu aplikacji opracowanej w ramach „Próby implementacyjnej 4" mają największe[179] znaczenie dla niniejszej pracy, podsumowanie i wnioski umiejscowiono w rozdziale 4.5.

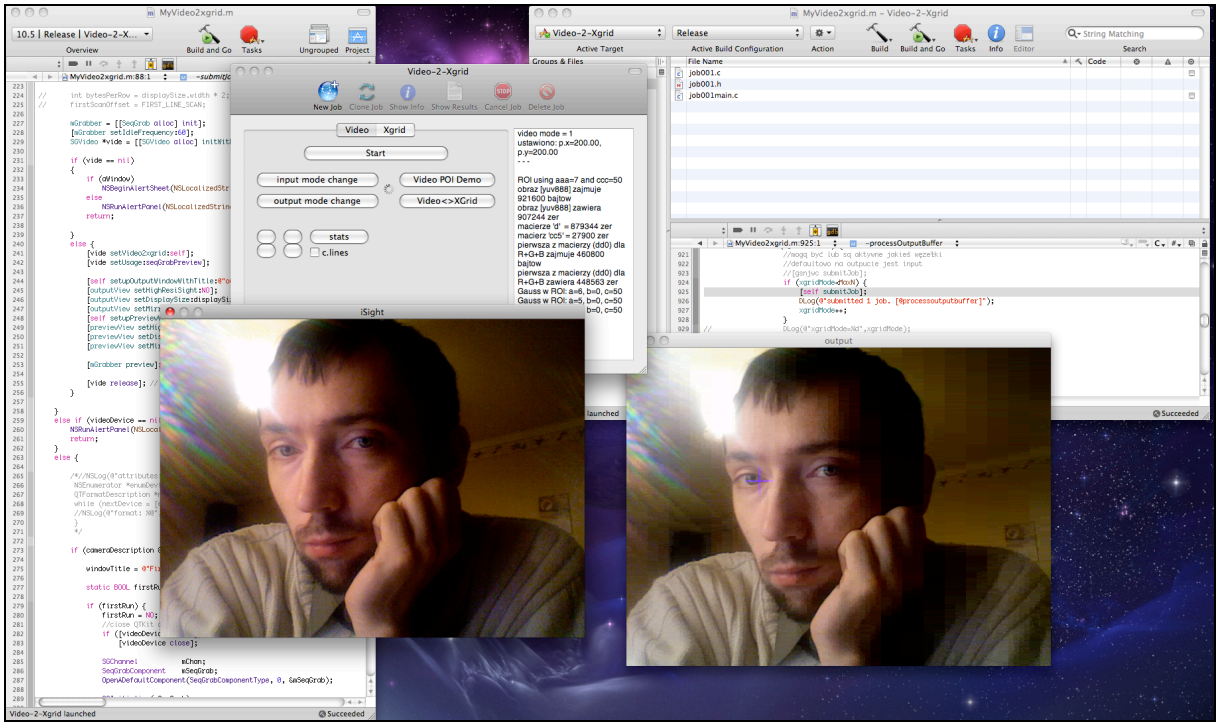
[177] Wynik zależny od konfiguracji sprzętowej, systemu operacyjnego i zainstalowanego oprogramowania. Podana/otrzymana wartość zawiera w sobie również m.in. czas systemu operacyjnego i przerwań sprzętowych.

[178] Należy jednak pamiętać, że konwersji DWT użyto z kompresją stratną, zatem nie wszystkie metody przetwarzania obrazu mogą być stosowane. Z pewnością można stosować opisaną w rozdziale 3.3 koncepcję sieci HTM.

[179] w porównaniu do innych przeprowadzonych eksperymentów i innych stworzonych implementacji

4.4.5. Implementacja 5 – DWT, ROI, POI, YUV, Xgrid (MacOsX)

Ostatnia spośród wytworzonych aplikacji miała za zadanie umożliwić przetestowanie algorytmów w całkowicie odmiennym środowisku, również klastrowym. W tym celu zaimplementowano całkowicie nowy interfejs i aplikację – w języku Cocoa – korzystając z natywnego środowiska dostępnego użytkownikom komputerów z systemem MacOsX. Zdecydowaną zaletą tego rozwiązania w stosunku do aplikacji „Próby implementacyjnej 4" jest niezwykle prosta przenaszalność kodu i kompatybilność sprzętu – aplikacja „Próby implementacyjnej 5" zadziała na praktycznie każdym[180] komputerze Macintosh.



Rys. 4.24. Aplikacja5 – interfejs użytkownika.

Aplikacja częściowo spełniła swoją rolę, zdecydowaną większość algorytmów udało się przetestować w nowej implementacji, jednak z powodu ograniczeń czasowych w aplikacji „Próby implementacyjnej 5" nie zaimplementowano rozwiązań lepszych ani nowszych niż w aplikacji „Próby implementacyjnej 4".

[180] z zainstalowanym systemem MacOsX w wersji 10.4 lub nowszym (czyli wszystkie wyprodukowane po 2006 roku)

Praktyczna implementacja aplikacji „Próby implementacyjnej 5" była wykorzystywana do testów porównawczych czasu przetwarzania poszczególnych fragmentów kodu[181] ostatecznej wersji rozwiązań.

4.5. Ocena wyników

W rozdziale 3.6 opisano propozycję sprzężenia zwrotnego (wirtualnych ruchów sakkadowych), będącą bazą i spoiwem serii zastosowanych rozwiązań technicznych opisanych w rozdziałach 3.2 (DWT, ROI, POI), 3.3 (HTM) i 3.5.2 (topologia klastra). Na metodę weryfikacji wybrano weryfikację eksperymentalną, ponieważ wyniki teoretyczne[182] mogą nie uwzględniać wielu czynników[183] mających na nie ogromny wpływ. Należy pamiętać, że opracowane rozwiązanie będzie użyteczne wyłącznie wówczas, gdy będzie działać w praktyce.

Przeprowadzone eksperymenty na wytworzonych aplikacjach będących weryfikacją implementacyjną opracowywanych rozwiązań składały się z kilku etapów wynikających z przyjętego harmonogramu prac:

1) określenie „skali problemu" – jaki jest czas komunikacji przy przesyłaniu obrazu w niezmienionej formie
2) opracowanie metody reprezentacji obrazu umożliwiającej użycie klastra komputerowego, porównanie rozmiaru danych opisujących obraz
3) określenie właściwej topologii klastra
4) określenie właściwej metodologii przetwarzania obrazu i/lub wnioskowania
5) testowa implementacja rozproszonego (klastrowego) systemu wizyjnego, porównanie czasów komunikacji
6) określenie osiągniętych celów poprzez interpretację zmierzonych czasów poszczególnych etapów działania algorytmu

[181] takich jak czas konwersji DWT, czas konwersji IDWT, czas konwersji palet barw, itd.

[182] np. czas komunikacji wyliczony na podstawie teoretycznego czasu trwania operacji systemowych wchodzących w skład procedury przesyłania danych

[183] Temat wstępnie opisany został w rozdziale 4.4.4 ; mowa tu m.in. o trudnym do oszacowania opóźnieniu pakietów w sieci, o trudnym do przewidzenia zachowaniu systemu operacyjnego itd.

Wszystkie etapy zostały zrealizowane. Eksperymenty przeprowadzono posługując się implementacjami opisanymi w rozdziale 4.3 (omówionymi bardziej szczegółowo[184] w dodatku Dodatek B). Otrzymane wyniki oraz związana z nimi część sprawozdawcza pracy naukowo-badawczej zostały zasygnalizowane wcześniej[185], tutaj zostaną omówione w kontekście ich znaczenia dla pracy.

Chronologicznie pierwszym eksperymentem była próba zgrubnego określenia czasu komunikacji algorytmu przesyłającego obraz wejściowy w postaci niemodyfikowanej macierzy 320x240x3. Jak nadmieniono w rozdziale 4.4.4, wynik był nieznacznie zależny od platformy sprzętowej – dla różnych konfiguracji[186] otrzymano następujące wyniki – od 16,09 ms (Rys. 1.4), przez 18,60 ms (Rys. 4.20), do 19,97 ms (Rys. 4.22). Dzięki eksperymentowi otrzymano czas potrzebny na przesłanie jednej klatki obrazu. Otrzymany wynik – około 18 ms – można interpretować dwojako:

Z jednej strony jest to czas wystarczająco krótki, aby wykonać kilka operacji przesłania w ciągu sekundy i nadal dysponować czasem procesora do wykorzystania dla pozostałych zadań komputera pokładowego robota mobilnego. Niestety węzły odbierające poświęcą tyle samo czasu na odebranie danych, a jeżeli będą przesyłały te dane do innych węzłów, czas komunikacji w danym węźle odpowiednio się wydłuży.[187] W każdym wypadku możliwe jest użycie co najwyżej kilku węzłów klastra przy niskiej rozdzielczości przesyłanego obrazu.

Z drugiej strony otrzymany w drodze eksperymentu czas przesyłania niekonwertowanej klatki obrazu jest na tyle długi, aby zasadne było poszukiwanie metody jego skrócenia.

[184] Dodatek B różni się od opisów umieszczonych w rozdziale 4.3 nie tylko poziomem szczegółowości. Opis w rozdziale 4.3 stanowi informację o korzyści uzyskanej z danej implementacji oraz o sposobie i metodzie jej uzyskania, natomiast opis w dodatku Dodatek B zawiera informacje techniczne niezbędne do obsługi danej aplikacji, zrozumienia zasad jej funkcjonowania i docelowo uzyskania możliwości odtworzenia wyników eksperymentów na niej przeprowadzanych.

[185] pod koniec każdego z rozdziałów 4.3, 4.4.2, 4.4.3, 4.4.4 oraz 4.4.5, zależnie od tego, z której aplikacji pochodziły dane wyniki

[186] W podanych trzech wynikach eksperymentu zmieniano sprzęt i systemy operacyjne, nie zmieniano natomiast parametrów technicznych obrazu wejściowego – rozdzielczość 320x240 pikseli, głębia 24bpp.

[187] Należy również pamiętać o ograniczeniach wynikających z zastosowanej technologii łącza fizycznego węzłów klastra. Ruch generowany „równocześnie" przez różne węzły wpływa na czas transmisji, a więc i czas komunikacji każdego z nich.

Na tym etapie sformułowano tezę[188] przedstawioną w rozdziale 1.4 oraz zintensyfikowano studium literaturowe, co zaowocowało eksperymentami z transformatą DWT oraz ze zdefiniowanym w standardzie JPEG2000 rozszerzeniem ROI. Testowa implementacja pod systemem Windows (patrz: rozdział 4.4.3) dała na tyle obiecujące wyniki[189], że postanowiono kontynuować badania bez weryfikacji w środowisku klastrowym[190]. Postanowiono wymienić klasyczne podejście AV (skanowanie obrazu obszarem przetwarzania precyzyjnego) na koncepcję bliższą biologicznym odpowiednikom komputerowych systemów wizyjnych poprzez zaimplementowanie algorytmu sterowania ruchem obszaru ROI w zależności od decyzji algorytmów wnioskujących systemu wizyjnego (a więc w sprzężeniu zwrotnym) na wzór ruchów gałki ocznej. Opracowywane i na bieżąco testowane rozwiązania szybko ewoluowały w kierunku ostatecznego rozwiązania, opisywanego w niniejszej pracy. Najważniejsze zmiany dotyczyły następujących zagadnień:

- algorytm konwersji reprezentacji obrazu (zawierający w sobie transformatę DWT) nie korzysta z rozszerzenia ROI, lecz z opisanego w rozdziale 3.2.2 POI,
- sterowanie ruchem, o którym mowa wyżej (oraz w rozdziale 3.6), nie powinno być utożsamiane z biologicznym świadomym ruchem gałek ocznych, lecz raczej z ruchami sakkadowymi, które są bardzo podobne w założeniach do implementowanego algorytmu:
  - Podczas czytania tekstu (lub w innych podobnych zadaniach) gałka oczna wykonuje ruchy sakkadowe bez potrzeby świadomego ich powodowania. Podobnie w proponowanym rozwiązaniu ruchy sakkadowe będą niezbędne do wytworzenia zmienności czasowej wzorców wejściowych węzłów sensorycznych – wirtualne ruchy sakkadowe nie będą „wynikiem" działania algorytmu wnioskującego, lecz będą stanowiły niezbędny element generowania danych potrzebnych dla funkcjonowania algorytmu systemu wizyjnego.
  - Podczas oglądania sceny (lub w innych podobnych zadaniach) gałka oczna wykonuje serię ruchów sakkadowych mających na celu zebranie detali obiektów widocznych na scenie[191]. Zupełnie odmienną sytuacją jest „kierowanie wzroku w stronę" określonego obiektu, jednak nie ma to wpływu na i tak wykonywane mimowolne „skanowanie" detali obiektu przy użyciu ruchów sakkadowych gałki ocznej.

[188] patrz: strona 20.

[189] Wyniki odtworzono w celu zamieszczenia w niniejszej rozprawie – patrz: strona 45.

[190] Rozwiązania wypracowane pod systemem Windows mogły być zweryfikowane w środowisku klastrowym tylko po migracji kodu, co może się okazać procesem czasochłonnym (zależnie od ilości i rodzaju napotkanych rozbieżności implementacyjnych).

[191] Doskonale pokazuje to film [26], dostępny również off-line na załączonej płycie CD.

Sposób definiowania[192] samego POI przybliżono w publikacji [45] oraz niektórych późniejszych, stanowiących niejako dokumentację opracowywanej technologii. Zalety i zastrzeżenia dotyczące otrzymywanego po przekształceniach obrazu przedstawiono w rozdziale 3.2 oraz 5. Wyniki eksperymentów cząstkowych z tego etapu prac nie dotyczyły czasu komunikacji czy też nawet rozmiaru struktury opisującej obraz w pamięci komputera, lecz jedynie ilości zer pojawiających się w macierzach współczynników przekształcenia DWT oraz subiektywnie ocenianej jakości/przydatności obrazu.[193] Ten sposób oceny tworzonych algorytmów został przyjęty ze świadomością, że blok danych zawierający stosunkowo dużą ilość zer stanie się bardzo wdzięcznym obiektem etapu kompresji. Dalsze prace implementacyjne, opisane m.in. w [110], potwierdziły to przypuszczenie. Każdy, nawet najprostszy algorytm kompresji okazywał się niezwykle skuteczny[194].

Celem pracy nie jest określenie optymalnego[195] sposobu kompresji[196]/reprezentacji obrazu, lecz udowodnienie[197] sformułowanej w rozdziale 1.4 tezy, dlatego eksperymenty przeprowadzano z użyciem zwykłego RLE i tak uzyskane wyniki poddawano dalszym rozważaniom.

Na rysunkach Rys. 4.22 i Rys. 4.23, zamieszczonych w rozdziale 4.4.4, przedstawiono wyniki najważniejszego dla pracy eksperymentu, na podstawie którego możliwe jest wyciągnięcie wniosków dotyczących tezy pracy. Otrzymane wartości omówiono już w rozdziale 4.4.4, ale do prawidłowej interpretacji widocznego na wykresach wyniku należy jeszcze naszkicować propozycję sposobu wykorzystania omówionych rozwiązań.

---

[192] Funkcję przedstawioną wcześniej na rysunkach Rys. 3.6 i Rys. 3.7 w rozdziale 3.2.2.

[193] Z tego okresu pochodzi m.in. kolejna tabela, zamieszczona na stronie 45.

[194] Warto tu dodać, że oprócz znacznej ilości zer wśród bajtów opisujących obraz, zmienia się również charakter przechowywanej informacji. Zamiast wartości jednego z kanałów barw jednego z pikseli, każdy z bajtów macierzy współczynników „D" przechowuje różnicę pomiędzy określonymi sąsiednimi pikselami poprzedniego poziomu dekompozycji. Jest to przyczyną zupełnie innego rozkładu funkcji gęstości prawdopodobieństwa zbioru przechowywanych w macierzy wartości. Jeden z rysunków zamieszczonych w dodatku Dodatek B – rysunek Rys. B.9 – zawiera m.in. szkic funkcji gęstości prawdopodobieństwa dla jednej spośród macierzy „D". Zbiór danych reprezentujących obraz, zawartych w macierzach współczynników transformaty DWT, najłatwiej (prze)analizować korzystając z pakietu „Wavelet Toolbox" środowiska Matlab.

[195] optymalnego w znaczeniu któregokolwiek z aspektów nowej reprezentacji obrazu

[196] Zagadnienie doboru metody kompresji nie jest w tym przypadku bynajmniej zagadnieniem trywialnym i wymagałoby gruntownego przeanalizowania wszystkich korzyści i komplikacji związanych z zastosowaniem danego (każdego proponowanego) sposobu kompresji.

[197] W tym celu wystarczyło wskazać dowolną metodę kompresji która umożliwi potwierdzenie tezy. Algorytmy kompresji są zagadnieniem niezwykle obszernym oraz – przede wszystkim – niezwykle dyskusyjnym z uwagi na mnogość kryteriów jakości, które mogłyby definiować „optymalny" algorytm kompresji.

Płaszczyzną spajającą opracowane na potrzeby pracy rozwiązania jest pomysł „wirtualnych ruchów sakkadowych", sprowadzający się de facto do specyficznego[198] sprzężenia zwrotnego implementowanego w torze przetwarzania wstępnego danych wizualnych. Na bieżącym etapie rozważań sprzężenie to przedstawiono jako sprzężenie zwrotne przenoszące informacje o żądanych nowych współrzędnych POI, pochodzące z węzłów klastra. Wejściem[199] układu regulacji jest strumień danych systemu wizyjnego[200], zależny od współrzędnych punktu POI. Właśnie współrzędne punktu POI są przekazywane jako sygnał sprzężenia zwrotnego[201] wpływając na dane poddawane dalszej obróbce. Sygnał wejściowy zastępowany[201] jest jego nową reprezentacją w wyniku działania procedur konwersji DWT[202], dając nowy (inny) sygnał[203], zdatny do obróbki/analizy w klastrze komputerowym. Po rozesłaniu[204] go do węzłów klastra, obraz jest wstępnie analizowany w węzłach[205]. Węzły w wyniku działania zaimplementowanych procedur przetwarzania oprócz sygnału wyjściowego formują propozycje nowych współrzędnych punktu POI, w sposób opisany w rozdziale 3.6[206]. Wyjściem opisywanego układu regulacji jest sygnał w postaci danych przesyłanych z rozproszonego podsystemu rozpoznawania do systemu decyzyjnego/wnioskującego niezwiązanego z „wirtualnymi ruchami sakkadowymi". Jeżeli system wnioskujący również[207] będzie miał kontrolę nad ruchem kamer(y), ruch ten można rozpatrywać jako świadomie podjętą akcję, a więc niepodlegający definicji ruchu sakkadowego.

Powyższy opis zawiera kilka wymogów, jakie muszą spełniać stosowane wraz z proponowanym w pracy rozwiązaniem algorytmy rozumienia wizji, aby możliwe było

[198] Układ regulacji, o którym mowa, z uwagi na użycie klastra zawiera w sobie przedstawione na rysunku (Rys. 3.19, na stronie 80) zrównoleglenie. Sygnał w postaci wstępnie przetworzonego obrazu trafia do kilku/wielu węzłów klastra, jak również opisywane sprzężenie zwrotne ma swój początek w tych węzłach.

[199] (Rys. 3.19, „(1)")

[200] w reprezentacji zgodnej z opisywaną w pracy, czyli o najlepszej jakości dla określonego punktu obrazu oraz skompresowany dowolnym algorytmem kompresji (nawet stratnej) dla obszarów odległych od wspomnianego punktu

[201] (Rys. 3.19, „(2)")

[202] przy użyciu wspomnianych współrzędnych punktu POI jako kluczowego parametru procedury konwersji

[203] w postaci macierzy współczynników (oraz współrzędnych punktu POI)

[204] w podziale funkcjonalnym

[205] np. w sposób przedstawiony na rysunku (Rys. 3.19, „(3)"), m.in. przy użyciu lokalnych lub rozproszonych sieci HTM

[206] poprzez wysyłanie wszystkich propozycji do węzła wejściowego, a tam wyłonienie nowych współrzędnych spośród przesłanych propozycji

[207] oprócz omawianego „wirtualnego ruchu sakkadowego"

praktyczne wykorzystanie wyników ostatniego eksperymentu. Nie każdy system wizyjny będzie mógł użyć klastra komputerowego poprzez samo zastosowanie się do zawartych w pracy wskazówek. Z pewnością przedstawiona w pracy wizja rozproszonego (klastrowego) systemu wizyjnego nie będzie ułatwieniem w algorytmach przetwarzania wyczulonych na zakłócenia[208], natomiast wydaje się być warta wypróbowania w większości algorytmów opartych na AV. W podsumowaniu omówiono bardziej szczegółowo ograniczenia proponowanych rozwiązań.

Komputerowy system wizyjny, którego część algorytmu przetwarzania (lub przetwarzania i wnioskowania) przeniesiono do środowiska rozproszonego (klastra komputerowego), może skorzystać z przedstawionych w pracy rozwiązań, jeżeli:

- istnieje możliwość użycia podziału/rozdziału funkcjonalnego zadań rozpoznawania,
- istnieje możliwość precyzyjnej analizy jedynie fragmentu obrazu (AV),
- istnieje możliwość implementacji sprzężenia zwrotnego, o którym mowa w rozdziale 3.6, czyli możliwość zgłaszania propozycji nowych współrzędnych punktu POI,
- i jeżeli żadne z ograniczeń ujętych w rozdziale 5 nie stanowi przeciwwskazania w zastosowaniu proponowanego rozwiązania.

Rozproszony (klastrowy) system wizyjny z zaimplementowanymi wirtualnymi ruchami sakkadowymi spełniający powyższe wymagania jest w stanie spożytkować przedstawioną na rysunku Rys. 4.22 różnicę czasu komunikacji[209].

Oceniając otrzymane wyniki należy również mieć na uwadze to, że samo skrócenie czasu komunikacji o ponad 97% nie oznacza proporcjonalnej korzyści w postaci „wolnego" czasu „do zagospodarowania". Otrzymanie wspomnianego skrócenia czasu komunikacji jest rezultatem szeregu czynności przeprowadzonych na danych wizualnych. Czas węzła wejściowego[210] poświęcony zadaniom wizji, mający znaczenie dla całokształtu funkcjonalności robota, wcale nie ulegnie skróceniu. Poniższy rysunek (Rys. 4.25) przedstawia czas, jaki musi poświęcić węzeł wejściowy na wysłanie jednej klatki obrazu do innego (jednego) węzła (kolor czerwony) oraz na uprzednie przygotowanie danych do

[208] proponowane rozwiązanie zawiera kompresję stratną wprowadzającą zniekształcenia w przetwarzanym obrazie

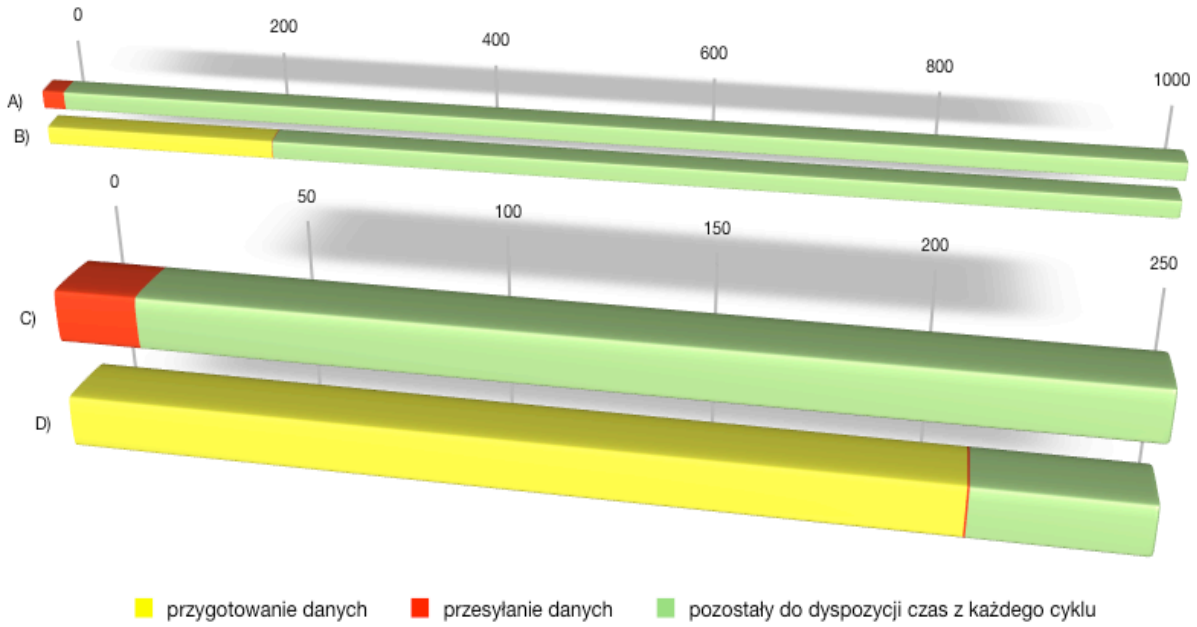
[209] różnicę pomiędzy czasem komunikacji przesyłania klatki niekonwertowanej a klatki przekonwertowanej proponowaną metodą

[210] komputera pokładowego (*embedded*) robota mobilnego

przesłania (kolor żółty) w każdej sekundzie (słupki A i B) lub ¼ sekundy (słupki C i D) swojego czasu. Ilustracja została wykonana przy następujących założeniach:

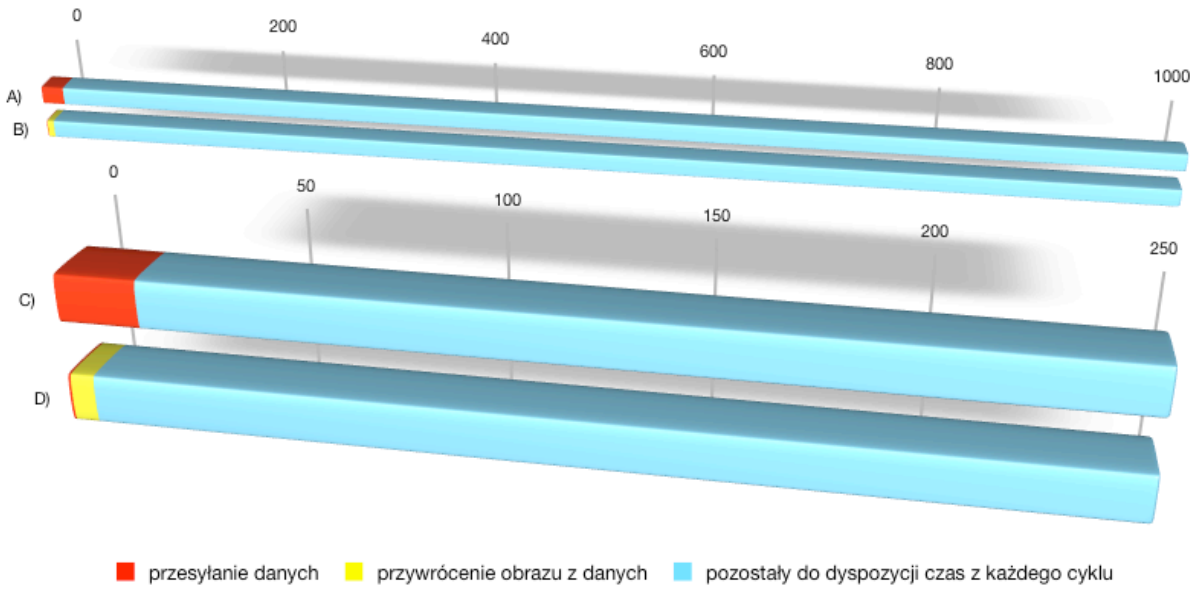
- Założenie1 (niekorzystne): węzeł wejściowy przetwarza jednowątkowo,
- założenie2 (niekorzystne): przygotowanie danych realizowane jest kodem powstałym z języka wysokiego poziomu, bez jakiejkolwiek akceleracji operacji macierzowych.

Słupki A i C przedstawiają przesyłanie niekonwertowanych klatek (19,97 ms), słupki B i D – konwersję (209,72 ms)[211] i przesyłanie skonwertowanej postaci (0,59 ms).



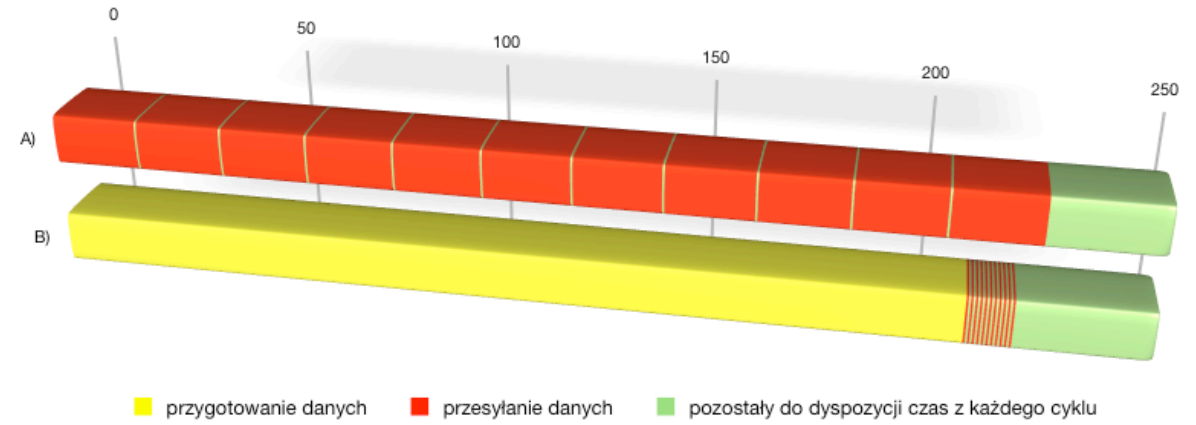
Rys. 4.25. Czas przygotowania i przesłania danych do węzła klastra: A/C) bez proponowanego rozwiązania; B/D) z wykorzystaniem proponowanego rozwiązania. (opis w tekście)

[211] średnia z 200 pomiarów



Rys. 4.26. Czas, jaki algorytm węzła odbierającego musi poświęcić na odebranie jednej klatki obrazu. Założenia identyczne jak w rysunku poprzednim. A i C – klatki niekonwertowane (19,97 ms), B i D – po konwersji (0,59 ms) i odtworzenie obrazu (IDWT) (5,84 ms).

Jak widać na rysunku Rys. 4.25, konwertowanie klatki obrazu w celu jej przesłania do węzła klastra wcale nie stanowi oszczędności czasu. Dopiero w przypadku bardziej realnym, gdy klatka ma być rozesłana do wielu węzłów (miast jednego), okazuje się, że różnica w czasie komunikacji (zaznaczony na rysunkach na czerwono) ma ogromne znaczenie. Poniższy rysunek (Rys. 4.27) przedstawia czas węzła wejściowego, który rozsyła sekwencyjnie tą samą klatkę do wielu węzłów.[212]

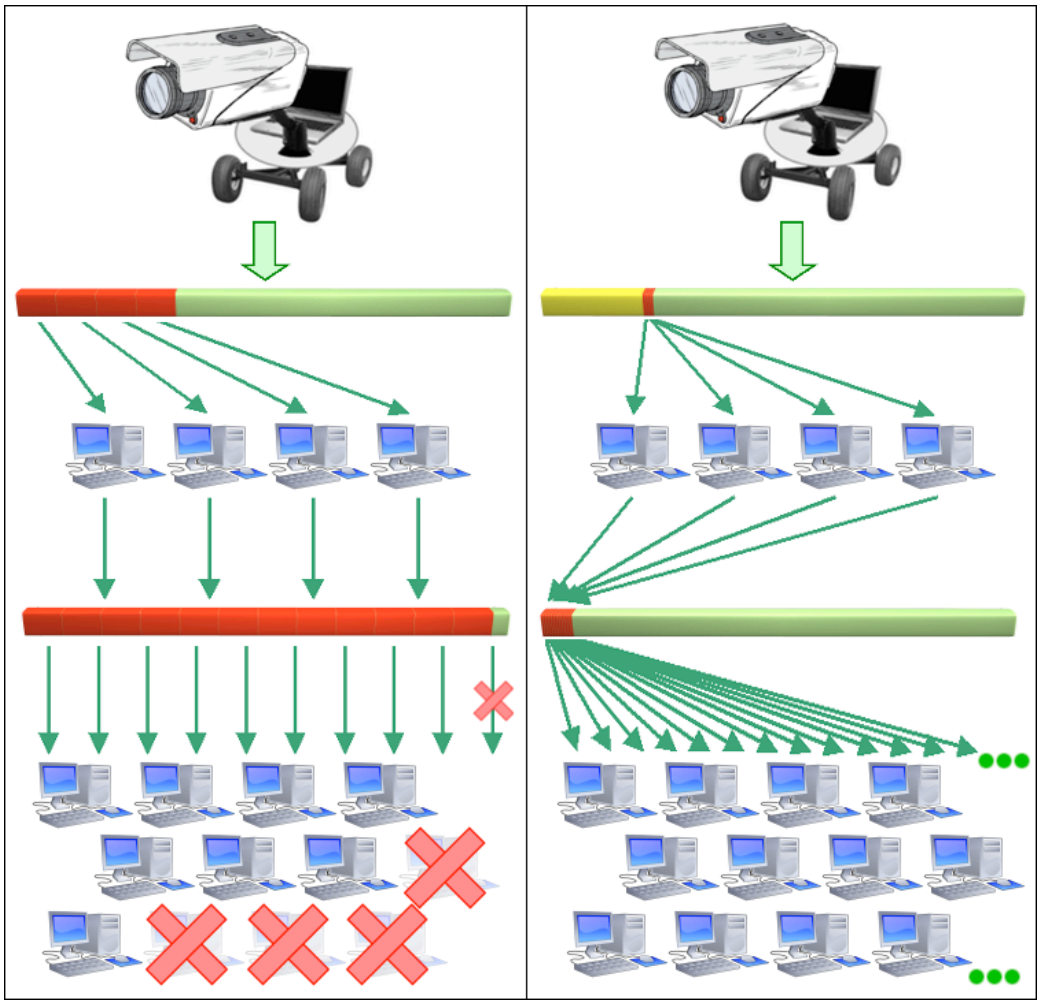


Rys. 4.27. Czas, jaki algorytm węzła wejściowego musiałby poświęcić na przesłanie jednej klatki obrazu do wielu węzłów.

[212] sytuacja teoretyczna, w praktyce między kolejnymi transferami należałoby obsłużyć pozostałe funkcje robota, co „rozsunęłoby" czerwone fragmenty

Pozornie powyższy rysunek (Rys. 4.27) nie wnosi żadnej korzyści w całym przedsięwzięciu. W rzeczywistości jednak nie węzeł wejściowy jest problemem (wąskim gardłem dalszego rozwoju „inteligencji" robotów). Problemem jest brak mocy obliczeniowej – brak możliwości zastosowania klastra komputerowego – w zagadnieniach dotyczących wizji. Najważniejszym fragmentem rysunku Rys. 4.27 są cieniutkie czerwone paski na dolnym słupku, ale odczytywane nie w kontekście czasu węzła wejściowego, lecz w kontekście zagadnień transmisji międzywęzłowej klastra. W dotychczasowych realizacjach klaster realizujący rozproszony algorytm przetwarzania klatek strumienia wideo „zatykał się" z powodu zbyt masywnych danych przesyłanych między węzłami (Rys. 4.28, po lewej). Konieczne było używanie szybkiej sieci komputerowej oraz ograniczanie rozmiarów klastra do kilku węzłów. Organizacja węzłów w różne topologie nic nie zmieniała w przypadku problemów z przepustowością sieci.

Opracowane rozwiązanie, tak jak to przedstawiono na rysunku (Rys. 4.28, po prawej), umożliwia użycie klastra, ponieważ czas potrzebny na przesłanie danych dotyczących obrazu między węzłami jest znacznie krótszy.



Rys. 4.28. Przetwarzanie danych wizualnych robota mobilnego w systemie rozproszonym. Górne słupki – czas jednosekundowego cyklu[213] przetwarzania i komunikacji węzła wejściowego, dolne słupki – jednosekundowy wycinek czasu komunikacji sieci klastra. Po lewej – rozsyłanie niekonwertowanych klatek obrazu „zapycha" klaster, po prawej – czas zainwestowany w konwersję obrazu na proponowaną postać[214] daje korzyść w postaci krótkiego czasu komunikacji umożliwiając zastosowanie klastra.

Wyniki otrzymane w drodze weryfikacji implementacyjnej, podsumowane powyżej, potwierdzają prawdziwość postawionej w pracy tezy[215] dla wybranych[216] algorytmów przetwarzania wizji. Aby możliwe było odzwierciedlenie przydatności proponowanego rozwiązania w zastosowaniach praktycznych, według rozważań z rozdziału 4.3, na aplikacji opisanej w załączniku B.5 przeprowadzono pomiary poszczególnych etapów elementarnych,

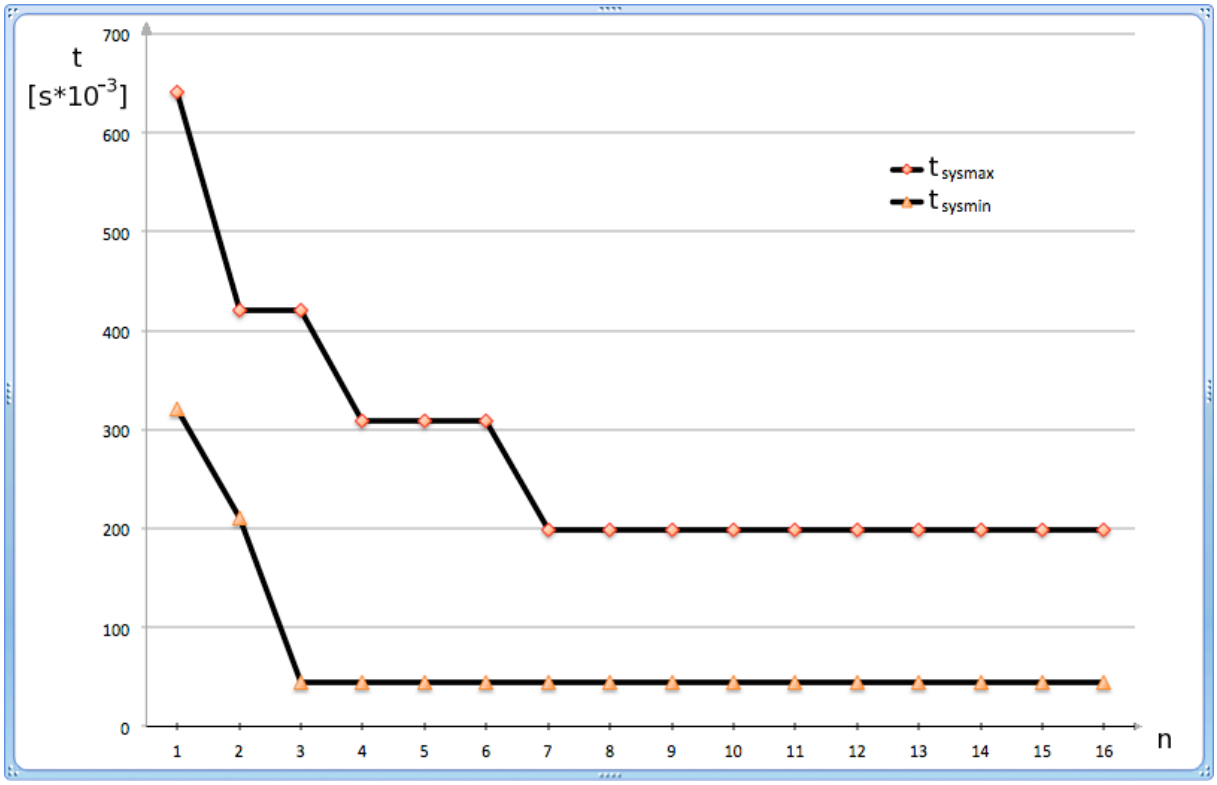
[213] Jest to rysunek umowny – w rzeczywistości niewiele spośród zdarzeń ma charakter cykliczny/okresowy. Przesyłanie nowego obrazu można jednak traktować umownie jako zadanie wykonywane cyklicznie, w każdej sekundzie czasu pracy węzła wejściowego.

[214] oznaczony kolorem żółtym

[215] patrz: strona 20.

[216] Dla algorytmów spełniających opisane w niniejszym rozdziale wymagania, z których najważniejsze dotyczy obszaru precyzyjnej analizy (podejścia zgodnego z AV)

określając czas opóźnienia danych systemu wizyjnego (od kamery do algorytmu wnioskującego) oraz czas odpowiedzi części synchronicznej algorytmu na żądanie klatki przesłane przez dowolny węzeł części asynchronicznej. Wyniki przedstawiono na poniższych rysunkach.



Rys. 4.29. Czas opóźnienia proponowanego systemu wizyjnego bez podsystemu wnioskowania. (opis w treści)

Rysunek Rys. 4.29 przedstawia czas opóźnienia[217] systemu wizyjnego (bez podsystemu wnioskowania) dla różnej ilości współbieżnych wątków węzła wejściowego klastra, w zależności od konfiguracji komputera będącego węzłem wejściowym klastra, w rozumieniu schematu widocznego na rysunku Rys. 4.10 (strona 97). Liczbę współbieżnych wątków oznaczono przez *n*, na osi rzędnych zaznaczono zsumowany czas wykonywania operacji. Wykres przedstawia dwa skrajne przypadki, odległe o wartość maksymalną zmiennej t_{rand}, wspomnianej w rozdziale 4.3.

Wartości użyte do narysowania niniejszego wykresu (jak również następnych – Rys. 4.30, Rys. 4.31) są wartościami rzeczywistymi, zmierzonymi[218] z wykorzystaniem kodu

[217] Opóźnienie jest tu rozumiane jako czas jaki upływa pomiędzy momentem zaistnienia zmiany w obserwowanej scenie a rozpoczęciem wnioskowania na podstawie klatki w węzłach klastra.

[218] Do pomiaru użyto klastra Apple XGrid z jednym zdalnym agentem w postaci komputera Apple MacBook Core2Duo 2.1GHz, z wykorzystaniem łączności przewodowej poprzez sieć Fast Ethernet (100Mbps)

aplikacji opisanej w rozdziałach 4.4.5 i B.5, na wspomnianym w rozdziale 4.1 laptopie MacBook Pro[219].

Poniższa tabela (Tabela 4.1) przedstawia wartości czasów operacji wchodzących w skład wartości umieszczonych na rysunku.

Tabela 4.1. Wartości poszczególnych czasów składowych występujących w równaniach (6), (7) i (8), zmierzone z wykorzystaniem kodu aplikacji opisanej w rozdziale 4.4.5 i B.5, na wyżej opisanym sprzęcie.

Nazwa	t_{akw}	t_{trans1}	$t_{bezkont}$	$t_{dwt} + t_{kompr}$	t_{trans2}	t_{idwt}	t_{kont}	t_{wnios}	t_{ster}
wartość zmierzona [$s \cdot 10^{-3}$]	0.0024	34.33	4.2702	166.1343	0.59	4.5624	0	0	0

Na rysunku Rys. 4.29 pokazano każdorazowo dwie skrajne wartości czasu upływającego od zaistnienia zmiany w obserwowanej scenie a rozpoczęciem wnioskowania na podstawie klatki w węzłach klastra (czyli: $t_{sys} - (t_{wnios} + t_{ster})$). Wartość obserwowana w rzeczywistości mieści[220] się w tym zakresie. Precyzyjne określenie położenia pojedynczego punktu/pomiaru wewnątrz zakresu nie jest możliwe, ponieważ zależy od wartości t_{rand1} i t_{rand2}, których znaczenie objaśniono pod równaniem (8).

Czas, oznaczony na powyższym rysunku przez t_{sysmin}, w analizowanym systemie ulega znacznemu skróceniu (do wartości równej t_{trans2}), dla trzech i więcej wątków węzła wejściowego. Wynika to głównie z tego, że w zastosowanym algorytmie właśnie przy trzech współbieżnych wątkach wydzielono osobny wątek do obsługi komunikacji.

Czas oznaczony przez t_{sysmax} zależny jest od częstotliwości z jaką część synchroniczna systemu wizyjnego kończy przygotowywanie nowej klatki obrazu. Częstotliwość ta wynika z wydajności węzła wejściowego klastra (węzeł ten dokonuje operacji bezkontekstowych przetwarzania wstępnego oraz transformaty dyskretnej i kompresji) oraz ze sposobu rozdziału zadań między wątki węzła, a więc w ograniczony sposób również z liczebności współbieżnie obsługiwanych wątków.

[219] Apple MacBook Pro i5 2.4GHz z systemem MacOsX 10.6

[220] podczas normalnej pracy algorytmu/aplikacji oraz przy stabilnej i przewidywalnej pracy systemu operacyjnego i sieci łączącej komputery

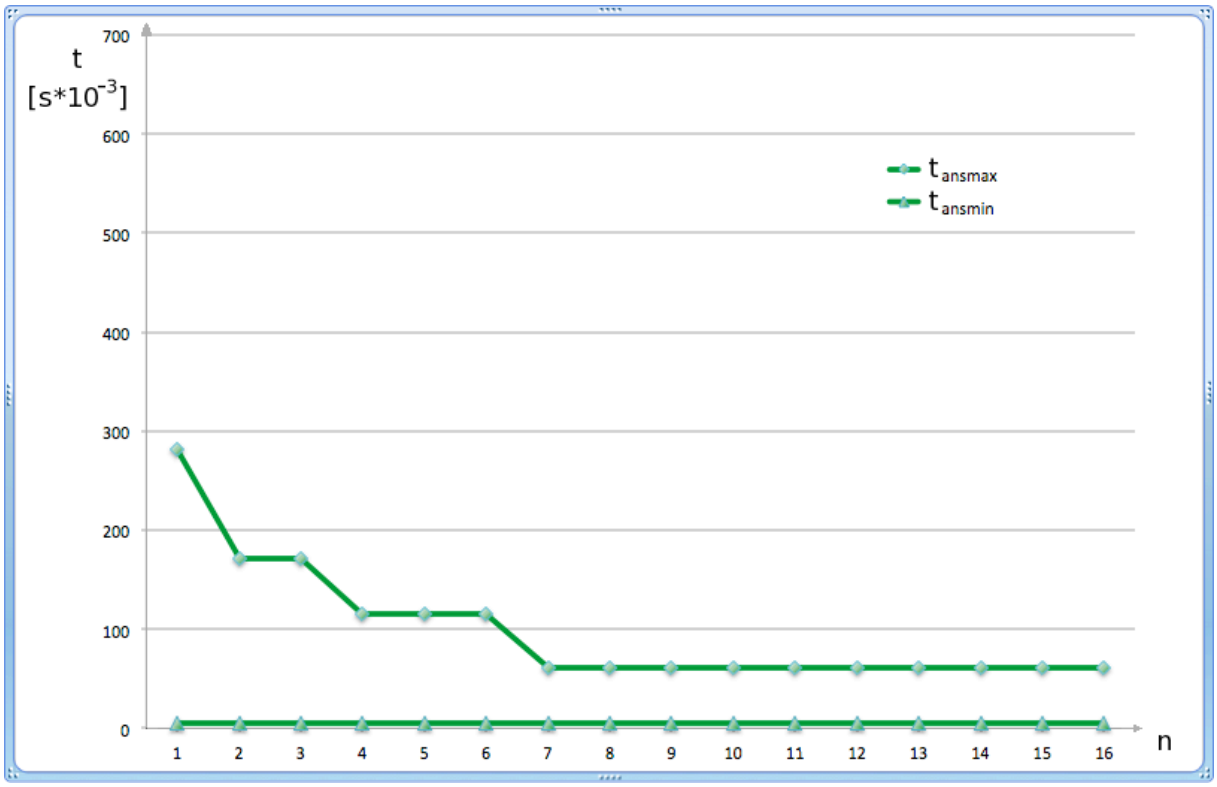
Zgodnie z rysunkiem Rys. 4.29, zaimplementowanie algorytmu węzła wejściowego klastra proponowanego systemu wizyjnego jako jednowątkowego sekwencyjnego algorytmu jest najmniej efektywnym rozwiązaniem – czas opóźnienia systemu wizyjnego (od zmian w scenie do uruchomienia algorytmu wnioskującego) mieści[221] się w przedziale od około[222] 321 ms do około 642 ms. Zrównoleglanie gruboziarniste daje wymierne korzyści (widoczne dla n=2), jednak dopiero wydzielenie osobnego wątku do obsługi zadań komunikacji powoduje ostateczne skrócenie czasu t_{sysmin} do poziomu około 44 ms. Czas t_{sysmax} również maleje, jednak dopiero przy 7 wątkach osiąga wartość minimalną. Czas t_{sysmax} na rysunku powyżej (Rys. 4.29), podobnie jak czas t_{ansmin} na rysunku Rys. 4.30, maleje w podobny, specyficzny sposób, wynikający z ziarnistości zadań.

Możliwe jest osiągnięcie jeszcze lepszych rezultatów (skrócenie czasów widocznych na rysunkach) poprzez zrównoleglenie działania algorytmu samej transformaty DWT, w obrębie jednego komputera fizycznego – węzła wejściowego klastra. W niniejszej pracy pominięto ten sposób skrócenia czasu obliczeń.

Rysunek Rys. 4.30 przedstawia czas odpowiedzi części synchronicznej algorytmu proponowanego systemu wizyjnego nową klatką (inną niż przechowywana w buforze węzła wejściowego). Ponieważ projektowany system wizyjny z założenia nie będzie służył robotowi w zadaniach krytycznych (związanych z bezpieczeństwem robota lub innymi zadaniami czasu rzeczywistego), czas odpowiedzi t_{ans} można uważać za bardziej istotny niż czas opóźnienia t_{sys}.

[221] dla sprzętu i oprogramowania użytego w eksperymencie ; dla innego sprzętu wyniki mogą różnić się w pewnym uzasadnionym zakresie

[222] dokładne zmierzone i wyliczone wartości zamieszczono w załączniku – Dodatku B.6

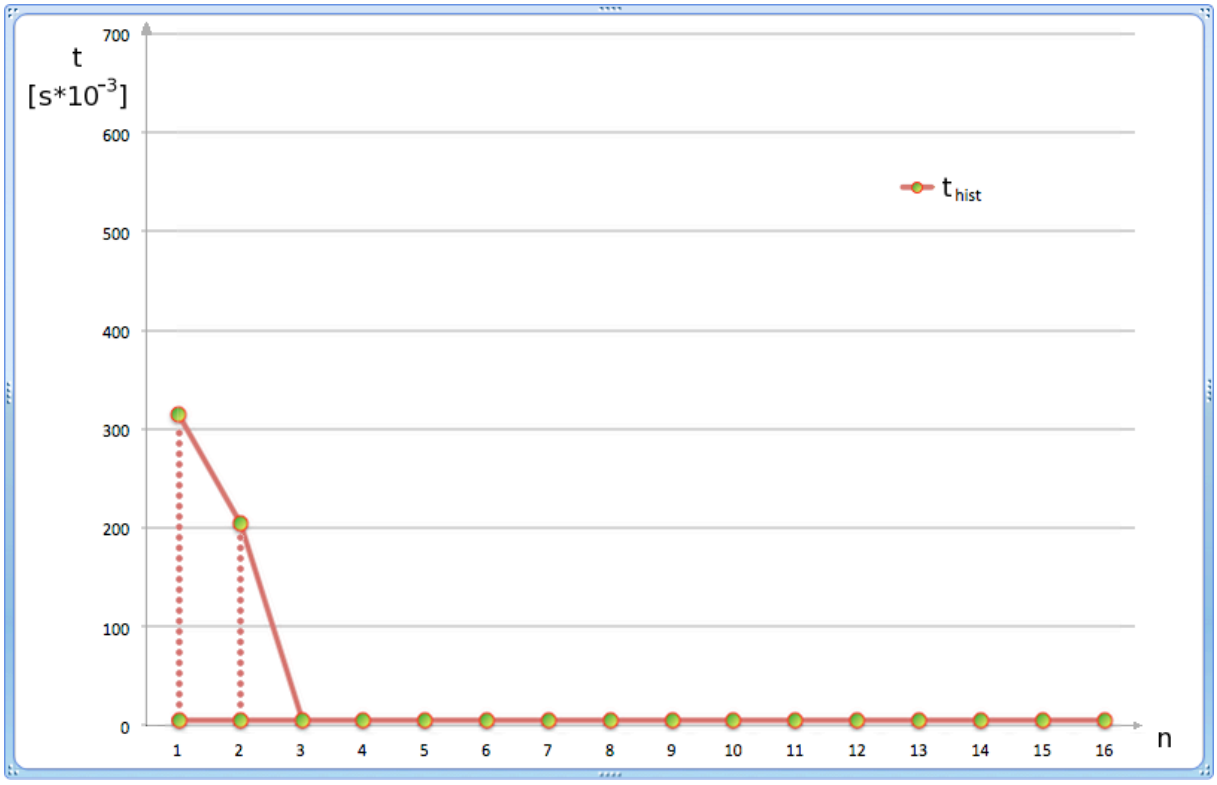


Rys. 4.30. Czas odpowiedzi części synchronicznej algorytmu proponowanego systemu wizyjnego nową klatką. (opis w treści)

Na rysunku Rys. 4.30 przedstawiono czas odpowiedzi systemu wizyjnego (węzłowi klastra wnioskującego) nową klatką (czas jaki upływa pomiędzy żądaniem nowej klatki obrazu, innej niż historyczna przechowywana w buforze, a rozpoczęciem wnioskowania na podstawie przesłanej klatki), dla różnych wersji komputera będącego węzłem wejściowym klastra, w rozumieniu schematu widocznego na rysunku Rys. 4.10 (strona 97). Liczbę współbieżnych wątków oznaczono przez *n*, na osi rzędnych zaznaczono zsumowany czas wykonywania operacji. Wykres przedstawia dwa skrajne przypadki czasu oczekiwania na nowszą klatkę niż znajdująca się w buforze. Czas ten zależy od częstotliwości, z jaką wątki obliczeniowe węzła wejściowego klastra kończą przetwarzanie poszczególnych klatek. Widoczny na rysunku brak zysku powyżej określonej liczebności wątków wynika z ziarnistości współbieżnych zadań.

Przypadek widoczny na rysunku powyżej można nieco uprościć – czas trwania operacji algorytmów wnioskujących, które będą zaimplementowane w węzłach klastra, z pewnością będzie dłuższy niż czas aktualności danej klatki obrazu. Z tego powodu prawdopodobieństwo, że wnioskowanie zostanie ukończone w czasie aktualności nadal tej samej klatki obrazu i klatka zostanie ponownie pobrana do wnioskowania jest pomijalnie

małe. Co więcej, sytuacja taka nie ma negatywnego wpływu na działanie algorytmu. W rozważaniach można zatem uwzględnić nie czas odpowiedzi nową klatką, lecz czas odpowiedzi (jakąkolwiek) klatką przechowywaną w buforze (pamięci). Rysunek Rys. 4.31 przedstawia czas odpowiedzi systemu wizyjnego klatką obrazu przechowywaną w buforze.



Rys. 4.31. Czas odpowiedzi części synchronicznej algorytmu proponowanego systemu wizyjnego aktualną klatką. (opis w treści)

Na rysunku Rys. 4.31 zilustrowano czas odpowiedzi części synchronicznej algorytmu proponowanego systemu wizyjnego aktualną klatką (czas jaki upływa pomiędzy żądaniem aktualnej klatki obrazu, najnowszej lub historycznej przechowywanej w buforze, a rozpoczęciem wnioskowania), dla różnych wersji komputera będącego węzłem wejściowym klastra, w rozumieniu schematu widocznego na rysunku Rys. 4.10 (strona 97). Liczbę współbieżnych wątków oznaczono przez *n*, na osi rzędnych zaznaczono zsumowany czas wykonywania operacji. Widoczne na rysunku wyniki zostały osiągnięte dzięki użyciu bufora oraz rozłączeniu zależności sekwencyjnej poprzez wykorzystanie transmisji asynchronicznej klatek. Rysunek sporządzono na podstawie czasów zmierzonych w aplikacji zaimplementowanej w oparciu o algorytm o architekturze opisanej w rozdziale 4.3. Dla $n>=3$ widoczne są skutki uruchomienia osobnego wątka dedykowanego zadaniom komunikacji.

Rysunek powyżej w dobry sposób wizualizuje elastyczność i użyteczność synchronicznej części proponowanego systemu wizyjnego, umożliwiającej – poprzez proponowane rozwiązanie – użycie klastra komputerowego. Węzeł klastra może otrzymać nowe dane wizualne niezwłocznie po przesłaniu żądania, a czas pomiędzy wysłaniem żądania a otrzymaniem kompletnych danych wynika jedynie z czasu transmisji spreparowanych i skompresowanych danych.

W tym świetle można potwierdzić, że wypracowana koncepcja rozproszonego (klastrowego) systemu wizyjnego, korzystającego z opisanych w niniejszej pracy algorytmów, rozumianego jako złożony układ regulacji, stanowi dobrze rokującą platformę do eksperymentów z rozproszonymi implementacjami zaawansowanych metod sztucznej inteligencji, również architektur emergentnych.

5. Podsumowanie

Wnioski z toku badań

Niniejsza praca stanowi próbę wskazania jednej z możliwych dróg ewolucji robotów inteligentnych, próbę pogodzenia emergentnych architektur kognitywnych i systemów wizyjnych robotów mobilnych. Architektury emergentne mają zazwyczaj bardzo wyrazistą hierarchiczność kategoryzowanych pojęć, często zapisaną lingwistycznie, natomiast systemy wizyjne (mimo że niejednokrotnie niemniej złożone) nie potrafią samodzielnie określić przynależności wykrytych obiektów do poszczególnych grup pojęć. Określanie dodatkowych cech obiektów uzyskiwane jest w drodze dodatkowej analizy wykrytych w scenie obiektów pod kątem istnienia określonych cech. To z kolei wydłuża i komplikuje działanie systemu wizyjnego. Hierarchiczna przynależność niezmiennych reprezentacji różnych poziomów abstrakcji w naturalny sposób prowadzi od detali do pojęcia – ludzki mózg nie musi uzyskiwać detali obiektów w wyniku post-processingu obserwowanego obiektu. W systemach wizyjnych projektowanych dla architektur emergentnych to nie obiekt składa się z detali, lecz detale składają się na obiekt. Taki jest też podstawowy sens projektowania algorytmów hierarchicznego rozumienia wizji – aktywne cechy na jednym poziomie abstrakcji aktywują niektóre spośród cech wyższego poziomu abstrakcji.

Opisana w pracy propozycja rozproszonego systemu wizyjnego wkomponowuje się w wizję badań i rozwoju zrobotyzowanych inteligentnych systemów usługowych przedstawioną w Propozycji Strategicznego Programu Badawczego Komitetu Automatyki i Robotyki Polskiej Akademii Nauk [34]. Koncepcja wydzielenia części procesów poznawczych do oddzielnej usługi klastrowej poza robotem jest w pełni kompatybilna z zaproponowanym Rozszerzeniem Internetu, dzięki usadowieniu w warstwie realizacji nie tylko agentów upostaciowionych (robotów) ale i agentów wirtualnych (nazwanych w dokumencie „aplikacjami"). Ponieważ w warstwie wyższej możliwa jest (nie jest wykluczona) komunikacja pomiędzy agentami (za pośrednictwem interfejsów usług) nie ma przeszkód, aby agent wirtualny świadczył usługi agentom upostaciowionym.

W pracy zasugerowano odświeżenie definicji „aktywnej wizji" poprzez rewalidację znaczenia członu „aktywna" – do tej pory oznaczającego skanowanie obrazu obszarem ROI lub jego przemieszczanie wg pre-programowanej sekwencji. Takie jego rozumienie jest niewystarczające. Przedstawione w pracy sterowanie wirtualnymi ruchami sakkadowymi stanowi oryginalną próbę włączenia problemu sterowania (na tym poziomie –kontekście– ruchu) w mechanizmy funkcjonowania systemów wizyjnych. To właśnie dzięki wzbogaceniu algorytmów rozumienia wizji o możliwość autonomicznej modyfikacji współrzędnych punktu fiksacji (celem agregacji i akwizycji detali obserwowanych obiektów), możliwe jest zbudowanie interfejsu pomiędzy komputerem obsługującym zadania akwizycji obrazu (oraz zadania kontrolowania ruchów i interakcji robota z otoczeniem) a architekturą emergentną, zaimplementowaną w środowisku rozproszonym.

Najistotniejsze problemy

Opracowane, przedstawione w pracy rozwiązanie, nie jest pozbawione ograniczeń wymuszających użycie określonych zabiegów technicznych i zabezpieczeń w algorytmach robota.

Najważniejszym z tych ograniczeń, wynikającym z przeniesienia części algorytmu decyzyjnego „na zewnątrz" robota, jest możliwość wystąpienia zakłóceń, opóźnień lub nawet zerwania łączności z częścią rozproszoną algorytmu. Implikuje to konieczność zaimplementowania podstawowej funkcjonalność robota[223] lokalnie.

Drugie ograniczenie stanowi asynchroniczność pracy poszczególnych grup węzłów podziału funkcjonalnego. Algorytm węzła robota odpowiedzialny za komunikację z klastrem powinien umożliwiać odebranie wyniku oraz przesłanie nowej klatki w dowolnym momencie, ponieważ w niektórych przypadkach bardzo trudne może się okazać określenie/przewidzenie momentu, w którym pojawi się wynik pochodzący z części klastra.

Trzecie ograniczenie wynika bezpośrednio z faktu zastosowania technologii POI, wywodzącej się z technik AV – proponowane w pracy rozwiązanie nie jest przewidziane dla systemów wizyjnych korzystających z całego obszaru obrazu wejściowego jako obszaru o najwyższej jakości obrazu. Algorytmy analizujące cały obraz mogą działać gorzej z powodu

[223] takie zadania jak m.in. procedury bezpieczeństwa – awaryjne zatrzymywanie / awaryjne sekwencje ruchów („odruchy bezwarunkowe")

stratnej konwersji obrazu, która ma miejsce podczas normalnej pracy. Można modyfikować parametry funkcji[224], aby znaleźć kompromis[225].

Sposób działania algorytmu wnioskowania musi być dostosowany do zaproponowanej reprezentacji obrazu. Algorytm musi być w stanie wykorzystać opisaną w rozdziale 3.2.2 reprezentację obrazu i opisany w rozdziale 3.6 mechanizm sterowania wirtualnymi ruchami sakkadowymi. Dopiero wówczas algorytm taki będzie miał sens, będzie mógł przynosić korzyści. W przypadku użycia opisanych w rozdziale 3.3 sieci HTM, może się okazać konieczne zaimplementowanie nowego węzła wejściowego sieci HTM (tzw. *sensor node*) – istniejące (w dokumentacji i przykładach implementacji) rozwiązania co prawda działają zgodnie z założeniami AV, ale skanują całą powierzchnię obrazu.

Kierunki dalszych prac

Badania opisane w niniejszej rozprawie będą kontynuowane celem wytworzenia systemu wizyjnego korzystającego z wypracowanych rozwiązań oraz implementującego sieci HTM w rozproszonym algorytmie wnioskującym. Aby było to możliwe, konieczne są następujące działania:

- implementacja algorytmu węzła wejściowego sieci HTM (*sensor node*) współpracującego z POI (zamiast węzła skanującego cały obraz),
- zaprojektowanie topologii i przygotowanie sieci HTM,
- zaprojektowanie i wytworzenie rozproszonego (klastrowego) systemu wizyjnego,
- wykonanie serii eksperymentów mających na celu określenie możliwości wytworzonego rozproszonego (klastrowego) systemu wizyjnego.

W czerwcu 2011 odbyła się konferencja[226], na której miała być ogłoszona nowsza[227] wersja podstawowej platformy[228] sieci HTM, niestety nowa wersja nie została jeszcze udostępniona.

[224] opisanej wzorem (1) na stronie 45

[225] Naturalnie spowoduje to zwiększenie ilości niezerowych bajtów w macierzach współczynników, skutkując wydłużeniem czasu komunikacji, czyli zmniejszając zysk z zastosowania rozwiązania.

[226] Konferencja „HTM Workshop", odbywająca się corocznie w Kalifornii, USA.

[227] Nowa dystrybucja zawiera wiele bardzo istotnych innowacji, które znacząco wpływają na użyteczność i poręczność środowiska oraz projektowanych przy jego użyciu sieci HTM, jak np. m.in. nowe algorytmy uczące

Kolejnym krokiem będzie implementacja algorytmów rozpoznawania i wnioskowania bazujących na HTM. Sieci HTM już są od pewnego czasu wykorzystywane w systemach wizyjnych [146] i w rozpoznawaniu obrazów [89], nie będzie to więc stanowiło nowości. Różnica polega na tym, że istniejące sieci HTM do zadań wizji używają zwykle[229] pojedynczego komputera. Jest to podstawowym źródłem ograniczeń skalowalności algorytmów rozumienia wizji.[230]

Wartą rozważenia opcją jest modyfikacja kodu konwersji transformaty DWT na wersję opisaną w [140], wykorzystującą akcelerację GPU. W niniejszej pracy pominięto ten wątek, gdyż projektowane rozwiązanie jest przeznaczone dla robotów mobilnych[231]. Nic nie stoi na przeszkodzie, aby wykorzystać ten pomysł dla skrócenia czasu konwersji w implementacjach wykorzystujących komputery stacjonarne[232].

Proponowane rozwiązanie nie stanowi jedynie alternatywy dla istniejących metod rozumienia wizji. Jest raczej brakującym ogniwem pomiędzy realiami technicznych ograniczeń robotów mobilnych a naszymi oczekiwaniami wobec inteligencji otaczających nas maszyn. Niniejsza praca pojawia się w bardzo dobrym momencie, bowiem wielu naukowców w ostatnim czasie pracuje nad rozwiązaniami, które w wizjach pisarzy tzw. fantastyki naukowej od dawna są rzeczywistością. W styczniu 2011 pojawił się artykuł [35] sugerujący[233] wykorzystanie chmur obliczeniowych (*ang. Cloud Computing*) jako sposobu na

(*HTM Cortical Learning Algorithms*) [91] (uczenie sieci to obecnie najsłabszy punkt technologii HTM). W świetle już ogłoszonych modyfikacji, samodzielne implementowanie potrzebnych (a dostępnych w tej modyfikacji) funkcji jest bezcelowe.

[228] Numenta NuPIC (Numenta Platform for Intelligent Computing)

[229] Platforma NuPIC sieci HTM posiada możliwość uruchomienia w środowisku klastrowym, jednak z powodu koncentracji przetwarzania obrazu w jednym węźle rozproszony algorytm nie daje spodziewanych korzyści (daje inne korzyści niż spodziewane – pozwala w pewnym stopniu wykorzystać klaster do tworzenia abstrakcyjnych reprezentacji, ale bez uprzedniego rozdziału funkcjonalnego nie da możliwości elastycznego zaimplementowania sprzężenia zwrotnego, które jest kluczowym elementem całego rozwiązania, oraz poważnie ograniczy skalowalność rozwiązania).

[230] Korzystanie z dowolnej innej architektury kognitywnej również nie rozwiąże problemu, ponieważ problem nie leży po stronie architektury, lecz wynika z charakteru danych wejściowych (i ilości rozpoznawanych obiektów).

[231] Uznano, że komputer pokładowy robota mobilnego powinien być możliwie najprostszym rozwiązaniem – zastosowanie potężnego GPU z pewnością do takich nie należy.

[232] Alternatywnie, można zaimplementować algorytm komputera robota mobilnego w architekturze klient-serwer połączonego z komputerem stacjonarnym jako węzłem konwertującym i koordynującym.

[233] Wypowiedź prof. James'a Kuffner'a z Carnegie Mellon University

poszerzenie w przyszłości możliwości robotów oraz przeniesienie części obliczeń, natomiast w lutym 2011 – artykuł [147] podsumowujący ponad roczną współpracę 30 naukowców kilku uczelni i firm nad otwartą platformą wymiany informacji operacyjnych robotów mobilnych RoboEarth[234].

Zastosowanie sieci HTM w połączeniu z wizją – najważniejszym i najbardziej bogatym źródłem informacji o otoczeniu człowieka – pozwala na nowo odzyskać nadzieję i entuzjazm w nieustannych poszukiwaniach recepty na przezroczysty interfejs pomiędzy światem maszyn i ludzi.

Świat informatyki zdecydowanym krokiem zbliża się do nowego rozdziału w swojej historii – przetwarzania w chmurach obliczeniowych. Świat robotyki nie pierwszy raz pozostaje nieco z tyłu tego wspólnego postępu. W przeszłości udawało się przezwyciężać trudności techniczne i implementacyjne istniejące na sprzęgu pozornie niekompatybilnych technologii, tak więc może i obecny impas jest sytuacją chwilową i przejściową. Może i robotyka będzie potrafiła wkrótce „sięgnąć chmur".

Podsumowanie prac implementacyjnych (eksperymentów)

Pierwszym etapem prac było określenie problemu badawczego – określenie przyczyny i ewentualnego sposobu eliminacji „wąskiego gardła" ewolucji systemów wizyjnych robotów przeznaczonych do interakcji z człowiekiem. Ustalenia z przeprowadzonego rozpoznania, po wstępnej konfrontacji z literaturą przedmiotu, pozwoliły na określenie i sprecyzowanie problemu badawczego, tezy, celów oraz kierunków poszukiwań. Podczas prac badawczych autor wspomagał się eksperymentalnymi implementacjami, dzięki którym możliwa była ocena zarówno skali problemu jak również jakości wypracowywanych rozwiązań. To właśnie dzięki weryfikacji implementacyjnej zrezygnowano z implementacji skalowalnej wielowarstwowej sieci neuronowej[235], również dzięki eksperymentom potwierdzono trafność prognoz (rozdział 4.3) dotyczących czasu oczekiwania na wynik wnioskowania węzła

[234] RoboEarth ma działać dla robotów podobnie jak Wikipedia dla ludzi. Jeżeli robot wykona (z pomocą człowieka lub w innych okolicznościach) jakieś nowe zadanie, wysyła „przepis na wykonanie" do Internetu. Gdy inny robot napotka na podobny problem, będzie go umiał pokonać dzięki tym wiadomościom pobranym z RoboEarth.

[235] Zgodnie z literaturą okazała się zbyt trudna w uczeniu i zbyt „ostrożna" w podejmowaniu decyzji (wnioskowaniu).

rozproszonego (klastrowego) systemu wizyjnego w stosunku do wartości tego czasu zaobserwowanego w zaimplementowanym kodzie (rozdział 4.5). Dzięki eksperymentom trafnie określono problem i jego przyczynę (rozdział 4.4.4, Rys. 4.20) oraz zweryfikowano zaproponowane rozwiązanie (rozdział 4.5, Rys. 4.25 – Rys. 4.31). Praktyczne realizacje platform robotów mobilnych (przedstawione w rozdziale 4.1) okazały się niepotrzebnym wątkiem – nie wniosły wiele do rozważań, ponieważ rozpatrywana (Rys. 3.16) pętla sprzężenia zwrotnego definiującego współrzędne POI nie wymagała fizycznego poruszania ani kamerą ani robotem.

Do rezultatów naukowych pracy należą:

- opracowanie nowej metody reprezentacji danych wejściowych (obrazu) dedykowanej algorytmom AV i pokrewnym (rozdział 3.2),
- opracowanie metody wykorzystania ww. reprezentacji z użyciem na bieżąco aktualizowanych współrzędnych punktu POI (rozdział 3.6).

Do rezultatów praktycznych pracy należy m.in.:

- opracowanie szeregu aplikacji implementujących ww. metody, umożliwiających przeprowadzanie eksperymentów, potwierdzających użyteczność zaproponowanych rozwiązań (rozdział 4).

Weryfikacja tezy

Wyniki otrzymane na drodze weryfikacji implementacyjnej, podsumowane w rozdziale 4.5, potwierdzają prawdziwość postawionej w pracy tezy dzięki zastosowaniu zaproponowanych w pracy metod reprezentacji, konwersji i manipulacji danymi wizualnymi.

Zaproponowany układ regulacji (rozdział 3.6, Rys. 3.16), stanowiący bazę zaproponowanego rozproszonego (klastrowego) systemu wizyjnego (rozdział 3.1, Rys. 3.1), dzięki współbieżności węzła wejściowego (rozdział 4.3, Rys. 4.10) oraz asynchroniczności żądań węzłów klastra (rozdział 4.3, Rys. 4.11), umożliwia przekazanie klatki obrazu do węzłów algorytmów wnioskowania w relatywnie krótkim czasie (rozdział 4.5, Rys. 4.31) oraz skutecznie obsługuje wiele współbieżnych zapytań nie powodując problemów z komunikacją (rozdział 4.5, Rys. 4.28).

Bibliografia i materiały uzupełniające

- Pozycje oznaczone tym symbolem zostały zarchiwizowane na załączonej płycie CD.
- Pozycje zawierające kod umożliwiający użycie łącza internetowego wprost z druku, na telefonie lub innym urządzeniu mobilnym.

[1] *A 13th Century Programmable Robot*, University of Scheffield, dostępne on-line (2010-07-07): http://www.shef.ac.uk/marcoms/eview/articles58/robot.html

[2] *A History of Automata*, Dodane 2009-07-23, dostępne on-line (2010-06-20): http://www.handworx.com.au/gearworx/history/ancient.html

[3] Agrawal D.P., Jain R., *A Pipelined Pseudoparallel System Architecture for Real-Time Dynamic Scene Analysis*, IEEE Transactions on Computers, Nr 5, 1988

[4] Al-Hassani S., *Al-Jazari: The Mechanical Genius*, dostępne on-line (2010-07-07): http://muslimheritage.com/topics/default.cfm?ArticleID=188

[5] Almasi G.S., Gottlieb A., *Highly Parallel Computing*, Benjamin-Cummings publishers, Redwood City, 1989

[6] Apple Inc., *Xgrid – the simple solution for distributed computing*, dostępne on-line (2011-01-22): http://www.apple.com/server/macosx/technology/xgrid.html

[7] Asimo.pl, *Asimo – historia*, dostępne on-line (2010-06-04): http://asimo.honda.com/AsimoHistory.aspx

[8] Asimo.pl, *Kalendarium Robotyki*, dostępne on-line (2010-06-10): http://www.asimo.pl/historia.php →

[9] Bagnell J. A., Bradley D., Silver D., Sofman B., Stentz A., *Learning for Autonomous Navigation*, IEEE Robotics and Automation, Nr 2, str.74-84, 2010

[10] Berowski P., *Transformata Falkowa*, Instytut Elektrotechniki, Warszawa, dostępne on-line (2010-12-29): http://bambus.iel.waw.pl/pliki/ogolne/ /studia%20doktoranckie/wyklady/wyklad_falki.pdf

[11] Bubnicki Z., *Teoria i algorytmy sterowania*, PWN, ISBN: 83-01-14414-9, Warszawa, 2005

[12] Buratowski T., *Teoria Robotyki*, AGH, dostępne on-line (2010-07-20): http://www.robotyka.com/teoria_spis.php →

[13] Cassin B., Solomon S., *Dictionary of Eye Terminology*, wydanie piąte, Triad Publishing Company, ISBN: 978-0-93740468-3, Gainsville, 2006

[14] Chih-Chang Chen Ch., *Region of interest determined by picture contents in JPEG 2000*, Proceedings of the 2003 International Symposium on Circuits and Systems, ISCAS '03, Nr 2, str.II-868 – II-871, ISBN: 0-7803-7761-3, 2003

[15] Christensen H., *Active Vision*, prezentacja, Kungl Tekniska Hoegskolan, dostępne on-line (2011-02-11): http://www.cas.kth.se/~hic/active-vision.pdf

[16] Davis L., *Foundations of Image Understanding*, The Springer International Series in Engineering and Computer Science – Vol. 628, Kluwer Academic Publishers, Boston, ISBN: 978-0-7923-7457-2, 2001

[17] Eccerobot, Materiał wideo: *ECCEROBOT – Embodied Cognition in a Compliantly Engineered Robot*, dodane 2009-08-03, dostępne on-line (2010-11-26): http://www.youtube.com/watch?v=cI9H4FoA0b4

[18] Eccerobot.org, Oficjalna strona projektu ECCEROBOT, *Overview*, dostępne on-line (2010-11-26): http://eccerobot.org/home/robot

[19] *Elektro – wyniki wyszukiwania*, dostępne on-line (2010-06-11): http://www.youtube.com/ /results?search_query=electro+robot+-dance&aq=f

[20] European Commission ICT Research, *ICT Challenge 2: Cognitive Systems, Interaction, Robotics – overview*, dostępne on-line (2011-06-01): http://cordis.europa.eu/fp7/ict/programme/overview2_en.html

[21] Fisher R., *Cvonline: The Evolving, Distributed, Non-Proprietary, On-Line Compendium of Computer Vision*, aktualizacja 2010-08-20, dostępne on-line (2010-09-26): http://homepages.inf.ed.ac.uk/rbf/Cvonline/

[22] Flynn M.J., Hung P., *Computer Architecture and Technology – Some Thoughts on the Road Ahead*, IEEE Micro, nr 3, str.16-31, 2005

[23] Foster I., *Designing and Building Parallel* Programs, Addison-Wesley & Argonne National Labolatory, ISBN: 0201575949, 1995, dostępne on-line (2011-05-01): http://www.mcs.anl.gov/~itf/dbpp/text/book.html

[24] Gaspar J., Winters N., Santos-Victor J., *Vision-based navigation and environmental representations with an omnidirectional camera*, IEEE Transactions on Robotics and Automation, Nr 16, 2000

[25] Gawrysiak M., *Robot jako system komputerowy*, Politechnika Białostocka, dodane 2006, dostępne on-line (2010-10-31): http://pbc.biaman.pl/Content/551/Robot+jako+system+komputerowy.pdf

[26] Gazetracker.com, Materiał wideo: *GazeTracker.avi* , dostępny on-line (2007-03-20): http://ftp.seeingmachines.com/pub/videos/gazetracker.avi

[27] Gazetracker.com, *Strona internetowa projektu GazeTracker*, dostępna on-line (2010-11-25): http://www.gazetracker.com/?page_id=10

[28] German Aerospace Center, Institute of Robotics and Mechatronics, *Robotic Systems*, dostępne on-line (2010-11-26): http://www.dlr.de/rm/en/desktopdefault.aspx/tabid-3758/

[29] Ghanem B., Resendiz E., Ahuja N., *Segmentation-based Perceptual Image Quality Assessment (SPIQA)*, International Conference on Image Processing, San Diego, 2008, dostępne on-line (2011-09-04): http://vision.ai.uiuc.edu/~bghanem2/Files/SPIQA_ICIP08.pdf

[30] González R.C., Woods R.E., *Digital Image Processing*, trzecia edycja, Prentice Hall, ISBN: 978-0-13-168728-8, 2008

[31] Grama A., Gupta A., Karypis G., Kumar V., *Introduction to Parallel Computing*, Pearson Education, ISBN: 0201648652, 2003

[32] Graves R., *Mity Greckie*, vis-à-vis Etiuda, 2009, ISBN: 978-83-61516-23-1

[33] Gropp W., Lusk E., *The Message Passing Interface (MPI) standard*, Argonne National Laboratory, dostępny on-line (2011-05-08): http://www.mcs.anl.gov/research/projects/mpi/

[34] Grupa Robocza Komitetu Automatyki i Robotyki Polskiej Akademii Nauk, *Strategiczny Program Badawczy: Rozszerzenie Internetu – Zrobotyzowane inteligentne systemy usługowe wspomagające człowieka*, dostępne on-line (2011-06-01): http://www.kair.pan.pl/index.php?option=com_content&view=article&id=66&Itemid=49

[35] Guizzo E., *Cloud Robotics:Connected to the Cloud, Robots Get Smarter*, IEEE Spectrum On-line, dodane 2011-01-24, dostępne on-line (2011-02-09): http://spectrum.ieee.org/automaton/robotics/robotics-software/cloud-robotics/ →

[36] Guizzo E., *Hubo II Humanoid Robot Is Lighter and Faster, Makes His Creator Proud*, IEEE Spestrum On-line, dodane 2010-03-30, dostępne on-line (2010-11-26): http://spectrum.ieee.org/automaton/robotics/humanoids/033010-hubo-ii-humanoid-robot-is-lighter-and-faster

[37] Guizzo E., *Humanoid Robot Mahru Mimics a Person's Movements in Real Time*, IEEE Spectrum On-line, dodane: 2010-04-27, dostępne on-line (2010-11-26): http://spectrum.ieee.org/automaton/robotics/humanoids/042710-humanoid-robot-mahru-real-time-teleoperation

[38] Guizzo E., *Humanoid Robots Rise. Now, Can They Walk?* IEEE Spectrum On-line, dodane 2010-10-04, dostępne on-line (2010-11-26): http://spectrum.ieee.org/automaton/robotics/humanoids/humanoid-robots-rise

[39] Guizzo E., *Iran's Humanoid Robot Surena 2 Walks, Stands on One Leg (Video)*, IEEE Spectrum On-line, dodane: 2010-08-16, dostępne on-line (2010-11-26): http://spectrum.ieee.org/automaton/robotics/humanoids/iran-humanoid-robot-surena-2-walks-stands-on-one-leg

[40] Harris M., *Mapping Computational Concepts to GPUs*, SIGGRAPH'05 Proceedings, ACM Digital Library, New York, 2005

[41] Hawkins J., *Learn Like a Human*, IEEE Spectrum, kwiecień 2007, dostępne on-line (2010-12-31): http://spectrum.ieee.org/computing/hardware/learn-like-a-human/ →

[42] Hawkins J., Dileep G., *Hierarchical Temporal Memory – Concepts, Theory and Terminology*, dodane: 2007, dostępne on-line (2010-12-31): http://www.numenta.com/htm-overview/education/Numenta_HTM_Concepts.pdf →

[43] Hawkins J., Blakeslee S., *On Intelligence*, Times Books, 2004

[44] Hendzel Z., Żylski W., Burghardt A., *Autonomiczne mobilne roboty kołowe – Mechatroniczne projektowanie i sterowanie*, Oficyna Wydawnicza Politechniki Rzeszowskiej, Rzeszów, ISBN: 978-83-7199-524-8, 2008

[45] red.: Hippe Z., Kulikowski J., *Human-Computer Systems Interaction – Backgrounds and Applications*, rozdział IV – *Robots and Training Systems* ; **Podpora M**., Sadecki J., *Biologically reasoned point-of-interest image compression for mobile robots*, ISBN: 978-3-642-03201-1, DOI: 10.1007/978-3-642-03202-8_29, Springer, str.389-401, 2009

[46] Hunziker H.W., *Im Auge des Lesers: vom Buchstabieren zur Leserfreude; foveale und periphere Wahrnehmung*, Transmedia Zurich, ISBN: 978-3-7266-0068-6, 2006

[47] IDG News, *Panasonic Has Big Plans for Robots*, dodane 2009-10-16, dostępne on-line (2010-10-04): http://www.pcworld.com/article/173788/panasonic_has_big_plans_for_robots.html?tk=rss_news , http://www.youtube.com/v/5l9dkS7MevM

[48] ISO / IEC FCD 15444-1 : 2000, "*JPEG2000 image coding system*", Annex H, dostępny on-line (2009-03-30): http://www.jpeg.org/public/fcd15444-1.pdf

[49] ITT Visual Information Solutions, *Image Processing*, wydanie: IDL.ver.6.7, NASA IDL Astronomy User's Library, 2007, dostępne on-line (2011-09-04): http://idlastro.gsfc.nasa.gov/idl_html_help/Image_Processinga.html

[50] Iwanowski M., *Metody morfologiczne w przetwarzaniu obrazów cyfrowych*, Akademicka Oficyna Wydawnicza EXIT, Warszawa, ISBN: 978-83-60434-66-6, 2009

[51] Johns K., Taylor T., *Professional Microsoft® Robotics Developer Studio*, Wrox, ISBN: 978-0-470-14107-6, 2008

[52] Kaczorek T., *Teoria sterowania i systemów*, PWN, ISBN: 83-01-12-072-X, Warszawa, 1996

[53] Karakuri ninyo – wyniki wyszukiwania, dostępne on-line (2010-07-06): http://www.youtube.com/results?search_query=karakuri+ningyo&aq=f

[54] Karakuri.info, dodane 2008-01, dostępne on-line (2010-07-06): http://www.karakuri.info/

[55] Karakuriya, dodane 2010-01-20, dostępne on-line (2010-07-06): http://karakuriya.com/english/index.htm

[56] red. Karbowski A., Niewiadomska-Szynkiewicz E., *Programowanie równoległe i rozproszone*, Oficyna Wydawnicza Politechniki Warszawskiej, ISBN: 978-83-7207-803-2, Warszawa 2009

[57] KAWADA Industries, *HRP-4*, dodane 2010-09-15, dostępne on-line (2010-11-26): http://www.plasticpals.com/?p=24521

[58] Khan S. M., Shah M., *Tracking Multiple Occluding People by Localizing on Multiple Scene Plates*, IEEE Transactions on Pattern and Machine Intelligence, Nr 3, str.505-519, 2009

[59] Kimura, T., *Actroid-F Makes Appearance at AIST Lab Fair, Wows World*, Akihabara News, dodane: 2010-10-28, dostępne on-line (2010-11-27): http://en.akihabaranews.com/68574/robot/actroid-f-makes-appearance-at-aist-lab-fair-wows-world →

[60] Kodak, *Informacja dot. Praw autorskich do pliku „Parrots" oraz „Village"*, True-color Kodak Test Images, dostępne on-line (2011-04-12): http://r0k.us/graphics/kodak/

[61] Koetsier T., *On the prehistory of programmable machines: musical automata, looms, calculators*, Mechanism and Machine theory, 36/2001, str.590-591

[62] Kolb H., Fernandez E., Nelson R., *Simple Anatomy of Retina*, John Moran Eye Centre, Universtity of Utah, dostępne on-line (2010-12-08): http://webvision.med.utah.edu/sretina.html#foveal

[63] Kosek W., *Metody analiz widmowych filtracji i prognozowania (wykłady)*, Space Research Center, Polish Academy of Sciences, Warszawa, dostępne on-line (2011-06-01): www.cbk.waw.pl/~kosek/tsa/WYKLADENG16pol.pdf

[64] Körner E., *Elements of Brain-like intelligence for ASIMO - Learning visually guided autonomous interaction*

[65] Kubiak Z., *Mitologia Greków i Rzymian*, 2003, ISBN: 8324701257

[66] Kurzweil R., *The Age of Spiritual Machines: When Computer Exceed Human Intelligence*, Penguin Books, 2000

[67] LAM/MPI Parallel Computing, dostępne on-line (2011-01-22): http://www.lam-mpi.org/

[68] Lena.org, *Informacja dot. Praw autorskich do pliku „Lena"*, dostępne on-line (2011-04-12): http://www.cs.cmu.edu/~chuck/lennapg/

[69] Ludwiczak B., *Mitologia Greków i Rzymian*, 2008, ISBN: 978-83-7517-082-5

[70] Markowska W., *Mity Greków i Rzymian*, 2002, ISBN: 8324701257

[71] Materiał wideo: *ASIMOs new artificial intelligence – ASIMO is learning*, stanowi fragment całości: BBC & Open University, *James May's Big Ideas – Man-machine*, 2008.
plik dodano 2009-04-14, dostępny on-line (2010-06-26): http://www.youtube.com/watch?v=P9ByGQGiVMg →

[72] Materiał wideo: The History Channel – *Ancient Discoveries – Robots*, Wild Dream Films, 2007-04-17, dostępne on-line (2010-06-20): http://www.youtube.com/watch?v=cbW_xKRBt_M

[73] Materiał wideo: The History Channel – *Ancient Discoveries – Machines of gods*, Wild Dream Films, 2007-01-06, dostępne on-line (2010-06-25): http://www.youtube.com/watch?v=GrI1FsH4UH8

[74] Materiał wideo: The History Channel – *Ancient Discoveries – Islamic science*, Wild Dream Films, 2008-12-15, dostępne on-line (2010-07-01): http://www.youtube.com/watch?v=v2HcjanNWFM

[75] Materiał wideo: *Home Exploring Robot Butler – several tasks.mpg*, Quality of Life Center – Carnegie Mellon University and Intel Labs Pittsburgh, dodane 2010-05-18, dostępne on-line (2010-10-04): http://www.youtube.com/watch?v=NsTdFF19fQQ

[76] McCulloch W., Pitts W., *A logical calculus of the ideas immanent in nervous activity*, Bulletin of Mathematical Biophysics, nr 5, str.115-133, 1943

[77] Merriam-Webster, *Merriam-Webster online medical dictionary*, dostępny on-line (2011-09-01): http://www.merriam-webster.com/medical/

[78] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard, Version 2.2*, High Performance Computing Center Stuttgart (HLRS), 2009, EAN-ISBN: 1114444410022, dostępny on-line (2011-05-08): http://www.mpi-forum.org/docs/mpi-2.2/mpi22-report.pdf

[79] Minor D., Rippa S., *GRAPE – An Industrial Distributed System for Computer Vision*, Materiały konferencyjne IPDPS, 2005, streszczenie dostępne on-line (2011-02-02): http://arnetminer.org/viewpub.do?pid=410032

[80] red. Morecki A., Knapczyk J., *Podstawy Robotyki – Teoria i elementy manipulatorów i robotów*, WNT, 1999, ISBN: 83-204-2331-7

[81] Morse B.S., Image Understanding, dodane: 2000, dostępne on-line (2011-09-04): http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MORSE/iu.pdf

[82] red. Nakashima H., Aghajan H., Augusto J. C., *Handbook of Ambient Intelligence and Smart Environments* ; Radke R. J., *A Survey of Distributed Computer Vision Algorithms*, Springer 2010, ISBN: 978-0-387-93807-3, rozdział dostępny on-line (2011-02-14): www.ecse.rpi.edu/~rjradke/papers/jaise08-radke.pdf

[83] Nałęcz M., Duch W., Korbicz J., Rutkowski L., Tadeusiewicz R., *Biocybernetyka i Inżynieria Biomedyczna 2000 –Tom 6 –Sieci Neuronowe*, Akademicka Oficyna Wydawnicza Exit, ISBN: 83-87674-18-4, Warszawa 2000

[84] National Geographic, Materiał wideo: National Geographic – *Frontlines of Construction – Construction Robots*, National Geographic Channel, 2000

[85] National Institute of Advanced Industrial Science and Technology, 究成果 - プレスリリース [*Informacje Prasowe nt. Aktualnych Badań*], dodane 2009-03-16, dostępne (2010-11-26): http://www.aist.go.jp/aist_j/press_release/pr2009/pr20090316/pr20090316.html

[86] red. Nawrot A., *Encyklopedia – Biologia*, Wydawnictwo GREG, ISBN: 978-83-7327-756-4, 2010

[87] Nobelprize.org, *The Nobel Prize in Physiology or Medicine 1967 (Granit R., Hartline H. K., Wald G.)*, dostępne on-line (2011-02-18): http://nobelprize.org/nobel_prizes/medicine/laureates/1967/

[88] Nobelprize.org, *The Nobel Prize in Physiology or Medicine 1981 (Hubel D. H., Wiesel T. N.)*, dostępne on-line (2011-02-18): http://nobelprize.org/nobel_prizes/medicine/laureates/1981/

[89] Numenta, *Aplikacja „Pictures”*, dostępna on-line (2007-03): http://www.numenta.com/about-numenta/technology/pictures-demo.php

[90] Numenta, dostępne on-line (2010-12-31): http://numenta.com/about-numenta/customers.php

[91] Numenta, *Hierarchical Temporal Memory including HTM Cortical Learning Algorithms*, dodane 2010-12-10, dostępne on-line (2011-02-09): http://www.numenta.com/htm-overview/education/HTM_CorticalLearningAlgorithms.pdf →

[92] Numenta, *Problems that Fit HTMs whitepaper*, dodane: 2007, dostępne on-line (2010-12-31): http://www.numenta.com/htm-overview/education/ /ProblemsThatFitHTMs.pdf

[93] Numenta, *What's in a Name: Hierarchical Temporal Memory*, dostępne on-line (2007-03): http://www.numenta.com/about-numenta/numenta-technology.php

[94] NVIDIA, *NVIDIA Tesla C2050/C2070 Datasheet*, dodane 2010, dostępne on-line (2011-05-03): http://www.nvidia.pl/docs/IO/43395/NV_DS_Tesla_C2050_C2070_jul10_lores.pdf

[95] Osowski S., *Sieci Neuronowe*, Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa, ISBN: 83-86569-01-8, 1996

[96] Ostwald D., Lam J.M., Li S., Kourtzi Z., *Neural Coding of Global Form in the Human Visual Cortex*, Journal of Neurophysiology, 2008, dostępne on-line (2011-09-04): http://jn.physiology.org/content/99/5/2456.full

[97] Parandowski J., *Mitologia*, PULS, 2009, ISBN: 0-907-58785-2

[98] Petrosino A., Tarantino E., *Parallel image understanding algorithms on MIMD multicomputers*, Computing, Springer-Verlag, vol.60, no.2, DOI: 10.1007/BF02684359, dostępny on-line (2011-09-04): http://www.springerlink.com/content/t5851k5285r17777/

[99] **Podpora M.**, *Biologically reasoned machine vision: RLE vs. Entropy-coding compression of DWT-transformed images*, Proceedings of the 14th Conference Student EEICT 2008 vol.4, ISBN: 978-80-214-3617-6, str.457-460, Brno 2008

[100] **Podpora M.**, *Computer vision in parallel computing*, ISTET'07 Book of abstracts, ISBN 978-83-74-57-039-8, Szczecin 2007

[101] **Podpora M.**, *Computer vision in parallel computing – reducing communication time*, Proceedings of the 13th Conference Student EEICT 2007 vol.4, ISBN 978-80-214-3410-3, Brno 2007

[102] **Podpora M.**, *Computer vision in parallel computing*, Przegląd Elektrotechniczny, ISSN 0033-2097, no.11/2007, dostępne on-line (2010-12-30): http://www.sigma-not.pl/publikacja-31274-computer-vision-in-parallel-computing-1564-przeglad-elektrotechniczny-2007-11.html

[103] **Podpora M.**, *Decentralizacja sterowania robota mobilnego przy wykorzystaniu klastra komputerowego*, materiały konferencyjne I SSNE 07, Opole 2007

[104] **Podpora M.**, *Distributed Computer Vision System – The Theory and an Implementation*, materiały konferencyjne V Środowiskowych Warsztatów Doktorantów Politechniki Opolskiej, Pokrzywna 2011, Oficyna Wydawnicza Politechniki Opolskiej, ISSN 1429-1533, Nr 342/2011, zeszyt 66, Opole 2011

[105] **Podpora M.**, *Dynamic re-definition of Region-of-Interest in Vision System's Feedback*, Proceedings of the 2nd International Students Conference on Electrodynamic and Mechatronics, ISBN 978-1-4244-3898-3, Gora Sw. Anny 2009

[106] **Podpora M.**, *Dynamic re-definition of Region-of-Interest in Vision System's Feedback*, Proceedings of the 2nd International Students Conference on Electrodynamic and Mechatronics, ISBN 978-1-4244-3897-6, IEEE eXplore, DOI:10.1109/ISCON.2009.5156101, INSPEC:10749736, 2009

[107] **Podpora M.**, *HTM – nowa era w historii sztucznych sieci neuronowych?*, Zeszyty Naukowe –Informatyka, Oficyna Wydawnicza Politechniki Opolskiej, przyjęte do publikacji

[108] **Podpora M.**, *HTM Networks – the future of computer vision systems?*, proceedings of the 8th annual workshop WOFEX 2010, VŠB – Technical University of Ostrava – FEECS, ISBN 978-80-248-2276-1, Ostrava 2010

[109] **Podpora M.**, *O przyspieszaniu klastra sterującego robotem mobilnym*, materiały konferencyjne I Warsztatów Środowiskowych Doktorantów Politechniki Opolskiej, Jarnołtówek 2007, Oficyna Wydawnicza Politechniki Opolskiej, ISSN 1429-1533, Nr 322/2008, zeszyt 59, Opole 2008

[110] **Podpora M.**, *Selected compression algorithms for mobile robots' vision systems*, InterTech – I International Interdisciplinary Technical Conference of Young Scientists, ISBN: 978-83-926896-0-7, Poznań 2008

[111] **Podpora M.**, *Task parallelization of image recognition procedures in a computer cluster*, materiały konferencyjne IV Środowiskowych Warsztatów Doktorantów Politechniki Opolskiej, Pokrzywna 2010, Oficyna Wydawnicza Politechniki Opolskiej, ISSN 1429-1533, Nr 335/2010, zeszyt 63, Opole 2010

[112] **Podpora M.**, *Wykorzystanie dyskretnej transformaty falkowej przy kompresji obrazu systemu wizyjnego robota mobilnego*, materiały konferencyjne II Warsztatów Środowiskowych Doktorantów Politechniki Opolskiej, Jarnołtówek 2008, Oficyna Wydawnicza Politechniki Opolskiej, ISSN 1429-1533, Nr 326/2008, zeszyt 61, Opole 2008

[113] **Podpora M.**, *YUV vs. RGB – A comparison of lossy compressions for human-oriented man-machine interfaces*, materiały konferencyjne III Warsztatów Środowiskowych Doktorantów Politechniki Opolskiej, Głuchołazy 2009, Oficyna Wydawnicza Politechniki Opolskiej, ISSN 1429-1533, Nr 329/2009, zeszyt 62, Opole 2009

[114] **Podpora M.**, Zając Z., *'Starter kit' sieci neuronowych dla robotyka: sterowanie ruchem systemu wizyjnego wraz z przykładową implementacją*, Zeszyty Naukowe –Informatyka, Oficyna Wydawnicza Politechniki Opolskiej, przyjęte do publikacji

[115] **Podpora M.**, materiał wideo – *Aplikacja4 podczas uruchomienia*, utworzone 2007, dostępne on-line (2011-02-14): http://www.youtube.com/watch?v=GkzzHIAXNJU →

[116] Polpitiya A. D., Dayawansa W. P., Martin C. F., Ghosh B. K., *Geometry and Control of Human Eye Movements, IEEE Transactions on Automatic Control*, Nr 2, str.170-180, 2007, dostępne on-line (2011-02-12): http://netra.math.ttu.edu/lab_papers_books/open_content/open_papers/smcb/POLPITIYA_DAYAWANSA_MARTIN_GHOSH_2007.pdf

[117] Prasanna-Kumar V.K., *Parallel Architectures and Algorithms for Image Understanding*, Academic Press, ISBN: 978-0125640404, 1991

[118] Rao S., Tron R., Vidal R., Ma Y., *Motion Segmentation in the Presense of Outlying, Incomplete, or Corrupted Trajectories*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Nr 32, str.1832-1845, 2010

[119] red. Rojek R., *Zastosowanie sztucznych sieci neuronowych i logiki rozmytej w automatyce*, Skrypt Nr 234, Oficyna Wydawnicza Politechniki Opolskiej, ISSN 1427-9932, 2000

[120] Romig III P.R., Samal A., DeViouS: A Distributed Environment For Computer Vision, Software–Practice and Experience, Nr 25, 1995, dostępne on-line (2011-02-02): http://www.cs.ubc.ca/local/reading/proceedings/spe91-95/spe/vol25/issue1/spe930.pdf

[121] Rosheim M.E., *Robot evolution: the development of anthrobotics*, John Wiley & Sons Inc., 1994, ISBN: 0-471-02622-0

[122] red. Rutkowski L., Malinowski K., *Sterowanie i automatyzacja: Aktualne problemy i ich rozwiązania*, rozdział XI – *Sztuczna Inteligencja* ; **Podpora M.**, *Czy zmniejszanie ilości danych pochodzących z systemu wizyjnego humanoida może zwiększyć jego możliwości?*, ISBN: 978-83-60434-42-0, Oficyna Wydawnicza EXIT, str.680-687, 2008

[123] Rutkowski L., *Metody i techniki sztucznej inteligencji*, PWN, Warszawa, ISBN: 978-83-01-15731-9, 2005, 2009

[124] Sadecki J., *Algorytmy równoległe optymalizacji i badanie ich efektywności; systemy równoległe z rozproszoną pamięcią*, Studia i Monografie, zeszyt 126, Oficyna Wydawnicza Politechniki Opolskiej, ISSN 1429-6063, Opole 2001

[125] red.: Shapiro S.C.; Tsotos J.K., *Encyclopedia of Artificial Intelligence*, Wiley & Sons, New York, 1987,

[126] Sheldrake R., *Mind, Memory, and Archetype: Morphic Resonance and the Collective Unconscious*, CiderPress, dostępne on-line (2011-02-17): http://ciderpress.wordpress.com/media/rupert-sheldrake/

[127] Simple DirectMedia Library, dostępna on-line (2011-01-22): http://www.libsdl.org/

[128] Song M., Tao D., Chen C., Li X., Chen C.W., *Color to Gray: Visual Cue Preservation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Nr 32, str.1537-1552, 2010

[129] Sparks D. L., *The Brainstem Control of Saccadic Eye Movements*, Nature Reviews Neuroscience, Nr 12, str.952-964, 2002, ISSN: 1471-003X, doi: 10.1038/nrn986, dostępne on-line (2011-02-13): www.dbsparks.com/research/html/SNRN02.pdf

[130] Szondy D., *Tales of Future Past – Electro*, dodane 2009-10-16, dostępne on-line: http://davidszondy.com/future/robot/elektro1.htm →

[131] Szynkiewicz W., prezentacja do wykładu pt. *Wstęp do Robotyki, cz.1*, 2006, dodane on-line 2007-01-09, dostępne on-line (2010-06-19): home.elka.pw.edu.pl/~mlula/download/Roboty/WR/WR_cz1.pdf

[132] Taddei M., *Leonardo da Vinci's robots (I robot di Leonardo)*, Ed.Leonardo3, 2007, ISBN: 978-88-6048-008-8, fragmenty dostępne on-line (2010-07-12): http://www.leonardo3.net/leonardo/books%20I%20robot%20di%20Leonardo%20-%20Taddei%20Mario%20-%20english%20Leonardo%20robots%201.html →

[133] Tadeusiewicz R., *Systemy wizyjne robotów przemysłowych*, Wydawnictwo Naukowo-Techniczne, Warszawa, ISBN: 83-204-1531-4, 1992

[134] Tadeusiewicz R., Gąciarz T., Borowik B., Leper B., *Odkrywanie własności sieci neuronowych przy użyciu programów w języku C#*, Polska Akademia Umiejętności, Kraków, ISBN: 978-83-60183-53-3, 2007

[135] Tadeusiewicz R., Kohorda P., *Komputerowa analiza i przetwarzanie obrazów*, Wydawnictwo Fundacji Postępu Telekomunikacji, Kraków, ISBN: 83-8647615-X, 1997

[136] red. Tadeusiewicz R., *Neurocybernetyka Teoretyczna* ; rozdział 14: Duch W., *Architektury kognitywne, czyli jak zbudować sztuczny umysł*, Wydawnictwa Uniwersytetu Warszawskiego, Warszawa, ISBN: 978-83-235-0479-5, 2009

[137] Tariov A., Majorkowska-Mech D., *Wielopoziomowa dekompozycja i rekonstrukcja sygnału za pomocą pakietów falkowopodobnych przy zastosowaniu bazy kosinusowej*, Metody Informatyki Stosowanej, nr 2/2007, Kwartalnik Komisji Informatyki Polskiej Akademii Nauk Oddział w Gdańsku, dostępne on-line (2010-12-29): http://pan.wi.ps.pl/pliki/MetInfStos/200702/ /MetInfStos_2007_02_Art_15.pdf

[138] Taubman D., Marcellin M., *JPEG2000: Image Compression Fundamentals, Standards and Practice*, Kluwer Academic Publishers, str.415–416 oraz 442–449, 2004

[139] Taylor M., *The REEM-B humanoid robot*, IEEE Spestrum On-line, dodane 2008-09-17, dostępne on-line (2010-11-26): http://spectrum.ieee.org/automaton/robotics/robotics-software/the_reemb_humanoid_robot

[140] Tenllado C., Setoain J., Prieto M., Piñuel L., Tirado F., *Parallel Implementation of the 2D Discrete Wavelet Transform on Graphics Processing Units: Filter Bank versus Lifting*, IEEE Transactions on Parallel and Distributed Systems, Nr 3, 2008

[141] The Tech Museum, *The History and Workings of Robotics*, dostępne on-line (2010-06-10): http://www.thetech.org/robotics/

[142] *Timeline of Robotics*, dodane 2007-11-19, dostępne on-line (2010-06-14): http://www.thocp.net/reference/robotics/robotics.html

[143] Top500, *Top500 List – November 2010*, dostępny on-line (2011-05-04): http://www.top500.org/list/2010/11/100

[144] Turing A.M., *Computing machinery and intelligence*, Mind, nr 59, str.433-460, 1950

[145] Virginia Tech, Robotics and Mechanisms Laboratory, *CHARLI: Cognitive Humanoid Autonomous Robot with Learning Intelligence*, aktualizowane 2010-04-27, dostępne on-line (2010-11-26): http://www.romela.org/main/CHARLI:_Cognitive_Humanoid_Autonomous_Robot_with_Learning_Intelligence

[146] Vitamin D Inc., oficjalna strona projektu, dostępne on-line (2010-12-31): http://www.vitamindinc.com/

[147] Waibel M., *RoboEarth: A World Wide Web for Robots*, IEEE Spectrum On-line, dodane 2011-02-05, dostępne on-line (2011-02-09): http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/roboearth-a-world-wide-web-for-robots/

[148] Wiatr K., *Akceleracja obliczeń w systemach wizyjnych*, Uczelniane Wydawnictwa Naukowo-Dydaktyczne AGH, Kraków, 2003

[149] Wiatr K., *Architektura potokowa specjalizowanych procesorów sprzętowych do wstępnego przetwarzania obrazów w systemach wizyjnych czasu rzeczywistego*, Uczelniane Wydawnictwa Naukowo-Dydaktyczne AGH, Kraków, 1998

[150] Winkler Z., Dlouhy M., *ASIMO in Prague*, dodane 2003-08-26, dostępne on-line (2010-06-11): http://robotika.cz/articles/asimo/en

[151] Wróbel Z., Koprowski R., *Praktyka przetwarzania obrazów w programie MATLAB*, Akademicka Oficyna Wydawnicza EXIT, Warszawa, ISBN: 83-87674-76-1, 2004

[152] Yarbus A. L., *Eye Movements and Vision*, Plenum, New York, 1967

[153] Yarbus A. L., *visitors*, grafika pochodząca z publikacji [152], dostępne on-line (2011-02-11): http://scasher.nfshost.com/wp-content/uploads/2010/05/YarbusVisitorEdit.png

[154] Zhao T., Nevatia R., Wu B., *Segmentation and Tracking of Multiple Humans in Crowded Environments*, IEEE Transactions on Pattern and Machine Intelligence, Nr 7, str.1198-1211, 2008

[155] Zhong N., Liu J., *Intelligent Technologies for Information Analysis*, Springer, ISBN: 3-540-40677-8, 1998

Dorobek naukowy doktoranta

Rozdziały w monografiach:

1. red. Rutkowski L., Malinowski K., *Sterowanie i automatyzacja: Aktualne problemy i ich rozwiązania*, rozdział XI – *Sztuczna Inteligencja* ;
   Podpora M., *Czy zmniejszanie ilości danych pochodzących z systemu wizyjnego humanoida może zwiększyć jego możliwości*? ; KKA'08 ; 2008 [122]
2. red.: Hippe Z., Kulikowski J., *Human-Computer Systems Interaction – Backgrounds and Applications*, rozdział IV – *Robots and Training Systems* ;
   Podpora M., Sadecki J., *Biologically reasoned point-of-interest image compression for mobile robots* ; HSI'09 ; 2009 [45]

Artykuły w czasopismach i materiałach konferencyjnych:

3. Podpora M., *Computer vision in parallel computing* (short); ISTET'07 ; 2007 [100]
4. Podpora M., *Computer vision in parallel computing* (full paper); Przegląd Elektrotechniczny ; 2007 [102]
5. Podpora M., *Computer vision in parallel computing – reducing communication time*, EEICT'07 ; 2007 [101]
6. Podpora M., *HTM – nowa era w historii sztucznych sieci neuronowych?* ; Zeszyty Naukowe –Informatyka ; przyjęte do publikacji (2007) 2011 [107]
7. Podpora M., Zając Z., *'Starter kit' sieci neuronowych dla robotyka: sterowanie ruchem systemu wizyjnego wraz z przykładową implementacją* ; Zeszyty Naukowe – Informatyka ; przyjęte do publikacji (2007) 2011 [114]
8. Podpora M., *Decentralizacja sterowania robota mobilnego przy wykorzystaniu klastra komputerowego* ; I SSNE'07 ; 2007 [103]
9. Podpora M., *Wykorzystanie dyskretnej transformaty falkowej przy kompresji obrazu systemu wizyjnego robota mobilnego* ; II SWD ; 2008 [112]

10. Podpora M., *Biologically reasoned machine vision: RLE vs. Entropy-coding compression of DWT-transformed images* ; EEICT'08 ; 2008 [99]
11. Podpora M., *Selected compression algorithms for mobile robots' vision systems* ; InterTech ; 2008 [110]
12. Podpora M., *O przyspieszaniu klastra sterującego robotem mobilnym* ; I SWD ; 2008 [109]
13. Podpora M., *Dynamic re-definition of Region-of-Interest in Vision System's Feedback* (short); II SCE ; 2009 [105]
14. Podpora M., *Dynamic re-definition of Region-of-Interest in Vision System's Feedback* (full paper); IEEE eXplore ; 2009 [106]
15. Podpora M., *YUV vs. RGB – A comparison of lossy compressions for human-oriented man-machine interfaces* ; III SWD ; 2009 [113]
16. Podpora M., *Task parallelization of image recognition procedures in a computer cluster* ; IV SWD ; 2010 [111]
17. Podpora M., *HTM Networks – the future of computer vision systems?* ; WOFEX'2010 ; 2010 [108]
18. Podpora M., *Distributed Computer Vision System – The Theory and an Implementation* ; V SWD ; 2011 [104]

Więcej na *http://podpora.opole.pl*

Dodatek A. Tło rozważań – ewolucja robotów inteligentnych

Trudno precyzyjnie określić, kiedy robotyka pojawiła się w historii naszej cywilizacji. Według wielu naukowców nastąpiło to wraz z pierwszym użyciem słowa „robot" [155][141], jednak niektóre źródła podają, że o wiele wcześniej, wraz z pierwszymi automatami [121][8]. Niezależnie od tego, początki dzisiejszej robotyki są niezaprzeczalnie powiązane ze światem fantastyki i to właśnie literatura w znacznym stopniu zdefiniowała sposób, w jaki społeczeństwo rozumie pojęcie „robotyka". W latach trzydziestych XX wieku robotem potocznie nazywano maszynę mechaniczną o człekokształtnej budowie, zaprojektowaną, by wspomagać człowieka w jego codziennych czynnościach. W 1939 roku na targach światowych w Nowym Jorku, podczas prezentacji takiego właśnie „robota" powiedziano, że do końca wieku każdy będzie posiadał swojego własnego robota-pomocnika [84][19]. Po wojnie rozwój techniki znacznie przyspieszył, a roboty faktycznie zawitały pod strzechy. Nie były to jednak tak niezwykłe i inteligentne maszyny, o jakich mówiono wcześniej. Zmianie uległo podejście społeczeństwa do zagadnienia i definicji robota. Dziś pod tym pojęciem kryje się jedynie mechaniczne urządzenie, które automatycznie wykonuje określone czynności. Co kilka lat mass media pokazują kolejnego robota, który prezentowany jest jako szczyt techniki, z komentarzem bliźniaczo podobnym do wspomnianego [130] z targów w 1939 roku. Mimo to ludzkość nie porzuciła nadziei na to, że roboty kiedyś staną się bardziej uniwersalne, wszechstronne, a przede wszystkim bardziej inteligentne – w naszym ludzkim rozumieniu.

Wielu uważa, że największym wkładem badawczym w rozwój humanoidów może się pochwalić firma Honda, od niemal 25 lat [7] pracująca nad ich rozwojem. Najnowszy robot – Honda Asimo – potrafi naprawdę wiele. Wśród obserwatorów zdarzają się jednak sceptycy [150], którzy wbrew powszechnemu zachwytowi wytykają niedociągnięcia. Przyglądają się, czy na podłodze są znaki kalibracyjne, czy robotowi przeszkadzają flesze aparatów, oceniają, na ile robot jest autonomiczny.

Dzisiejsze roboty nie są w stanie rozumieć otaczającego świata w stopniu wystarczającym do swobodnej interakcji z człowiekiem. Nie posiadają aktuatorów porównywalnych z biologicznymi odpowiednikami, ich sensory nie potrafią wykorzystać bogactwa informacji świata otaczającego. Skoro ich ewolucja przez dłuższy już okres nie może znaleźć wyjścia z impasu, może być potrzebna bardziej odważna zmiana podejścia do zagadnień przetwarzania i wnioskowania.

Pomysł skonstruowania urządzenia potrafiącego działać, widzieć, a nawet rozumieć tak jak ludzie nie jest tematem nowym. Słyszymy o tym od wielu lat i na różnych płaszczyznach – od literackich wizji po naukowe artykuły.

Pierwsze mechanizmy

Według niektórych źródeł [142], początków marzeń o robotach można się dopatrywać już w starożytnym Egipcie czy mitologii[236]. Można tam bez trudu natrafić na mity związane z tworzeniem nowych istot z materii nieożywionej, lub tworzeniem maszyn. W literaturze [142] czasy te datowane są na około 3500-3000 rok p.n.e., może być to jednak pewna przesada[237]. Również w starożytnym Egipcie trudno określić precyzyjnie stan wiedzy i umiejętności inżynierskie w tym okresie[238]. Równolegle do obu tych cywilizacji rozwijała się cywilizacja Dalekiego Wschodu, gdzie przykładowo około roku 2600 p.n.e. Ma Jun skonstruował Rydwan Pokazujący Północ. Na osi czasu tych cywilizacji da się zaznaczyć kilka ważnych momentów związanych z maszynami lub pracami, które przetrwały do dnia dzisiejszego. Często nie były to automaty w pełnym tego słowa znaczeniu, lecz proste mechanizmy lub praktyczne implementacje wykorzystania określonych sił natury, tak jak w przypadku Kolosów Memnona[239] (Egipt, 1370r.p.n.e) [73][74]. Te wszystkie wynalazki

[236] Mitologia grecka: Hefajstos – bóg, kowal – stworzenie kobiety/Pandory, skonstruowanie łoża Afrodyty [97], stworzenie dwóch mechanicznych posągów-dziewcząt służebnych, które umiały się same poruszać, podpierały niepełnosprawnego stwórcę, potrafiły mówić, pomagały w kuźni [70][32], wstawienie sztucznej łopatki Pelopsowi [65] ; Mitologia grecka: Pigmalion – król, rzeźbiarz – stworzenie Galatei [69] ; Mitologia grecka: Dedal – artysta, wynalazca, rzeźbiarz, architekt, kowal – skonstruowanie jałówki dla Pazyfae, skrzydeł dla Ikara [32][65][69] ; Mitologia grecka: wojownicy wyrastający ze smoczych zębów zasianych przez Jazona, mit o Argonautach [32] ; Mitologia grecka: Talos – potwór z Krety/Sardynii z jedną żyłą, twór Hefajstosa [65]-s.495, [32]-s.290 (jest jednocześnie symbolem techniki odlewu [32] „na wosk tracony")

[237] Część mitologii związana z Iliadą datuje się na 13 wiek p.n.e., natomiast okres około roku 3000 p.n.e. dotyczy tzw. hipotezy o cywilizacji Atlantydy.

[238] Niektóre źródła podają szacunkowo rok 3000 p.n.e. [131] choć pierwsze piramidy powstały dopiero po roku 2650 p.n.e

[239] 1370 p.n.e. – jeden z Kolosów Memnona (posągów faraona Amenhotepa III) wydawał dźwięki o wschodzie słońca (prawdopodobnie dzięki zmyślnemu wykorzystaniu północno-wschodniego wiatru) [73]

i konstrukcje służyły rozmaitym celom – od wzmacniania pozycji kapłanów i bogów [72][2][73], przez mechanizmy i maszyny wykonujące określone zadania życia codziennego [74], aż po zegary [74] i rozmaite urządzenia tworzone jedynie ku rozrywce władców [72][2].

Maszyny, pierwsze automaty

Stopniowy rozwój szeroko rozumianej inżynierii owocował coraz bardziej istotnymi wynalazkami, wśród których wymienić można konstrukcje wykorzystujące sprężystość powietrza autorstwa Ktesibiosa (IIIw.p.n.e), mechanizm z Antykithiry (ok.150r.p.n.e) przez wielu zwany pierwszym komputerem, wynalazki Herona z Aleksandrii (Iw.n.e), pierwsze automaty opisane przez braci Banu Musa w „Księdze Pomysłowych Urządzeń" (IXw.n.e.) [61] oraz maszyny Al–Jazari'ego[240] (ok.1200r.n.e.) korzystające z wielu nowatorskich rozwiązań, jak wał rozrządu czy wał korbowy [74]. Postęp w rozumieniu zasad mechaniki, pneumatyki i hydrauliki umożliwił budowanie coraz bardziej złożonych mechanizmów. Al–Jazari w swojej „Księdze Wiedzy o Pomysłowych Urządzeniach Mechanicznych" opisał (oprócz innych maszyn i automatów) programowalne[241] automaty humanoidalne [4] – czterech muzyków na pływającej łodzi. Jest to jeden z najstarszych przykładów stworzenia automatu z założenia imitującego człowieka, choć do niedawna uważano, że dopiero „Rycerz w zbroi" (1495r.) autorstwa Leonarda da Vinci był pierwszym projektem[242] tego rodzaju urządzenia [8]. Najbardziej dynamicznym okresem rozwoju automatów był XVI i XVII w.n.e. – ponowne odkrywanie kultury Greków podczas Renesansu w znaczący sposób wpłynęło na rozwój nie tylko filozofii, astronomii, matematyki czy geometrii, ale przyspieszyło postęp technologiczny. Ponowne pojawienie się dzieł m.in. Ktesibiosa, Herona z Aleksandrii czy też Filona z Bizancjum wywarło znaczny wpływ na naukowców z tamtych czasów. Również na dalekim wschodzie automaty miały swój okres rozkwitu. W XVII wieku zaczęły się pojawiać m.in. tzw. Karakuri Ningyo [53][54][55] – automaty w postaci mechanicznych lalek, wykonujące określone sekwencje ruchów o zadziwiającej złożoności.

[240] Znany w dzisiejszym świecie jako Al-Jazari, lecz po prawdzie zwał się Badi' al-Zaman Abu-'l-'Izz Ibn Isma'il Ibn al-Razzaz al-Jazari [4].

[241] Programowalność polegała na możliwości zmiany położenia kołków na cylindrze sterującym, co powodowało zmianę np. rytmu wygrywanego na bębnie [1].

[242] Projekt robota odkryto dopiero w 1957r. [132], co więcej nic nie wskazuje na to by robot kiedykolwiek istniał, dopiero w roku 2002 został zbudowany na potrzeby filmu dokumentalnego dla stacji BBC [132]

Era Robotyki

Narodziny robotyki datowane są różnie. Niektóre źródła odnoszą się do muzyków Al–Jazari'ego [8] lub do innych urządzeń, ale najczęściej wymienia się słynną sztukę „R.U.R.", której autorem jest Karel Čapek, ponieważ w niej pierwszy raz pojawia się słowo „robot" użyte w konkretnym kontekście [12][155][141]. Choć różnica między dzisiejszymi definicjami automatu i robota jest bardzo ścisła, techniczna, to w tamtym czasie była bardziej emocjonalna, związana z nadziejami na inne jutro. Ludzkość marzyła o maszynach sobie podobnych, potrafiących współdziałać, pomagać i wyręczać w codziennych zadaniach. Rzeczywista ścieżka rozwoju robotyki była jednak nieco inna od roztaczanych wówczas wizji. Prawdziwy rozkwit robotyki miał miejsce nie w salonach, lecz w fabrykach.

Literatura [80](-rozdz.14.3) wymienia pięć okresów rozwoju robotyki:

- przedelektroniczny (od mitologii przez Aleksandrię aż do około 1950 roku),
- startowy / początkowy (około 20 lat, pierwsze konstrukcje, zastosowania przemysłowe oraz pierwsze badania w ośrodkach naukowych),
- euforii / entuzjazmu (do około 1985r., ponad 500 firm zajmuje się produkcją robotów, zapotrzebowanie nieustannie rośnie, nawet o 35% rocznie),
- rozczarowania i analizy (druga połowa lat osiemdziesiątych, prognozy dalszego wzrostu zapotrzebowania nie sprawdziły się, ujawniając nasycenie rynku),
- ponownego ożywienia / wzrostu (od około 1993r.).

W tzw. okresie euforii robotyka zyskiwała sprzymierzeńców i zwolenników jak nigdy wcześniej, mimo to o rozwoju robotyki w tym okresie należy mówić bardzo ostrożnie. Dopiero w tzw. okresie rozczarowania udało się odkryć jaki był powód nagłego nasycenia rynku. Produkowane automaty i roboty były przewidziane do ściśle określonych zadań, nie było możliwe zastosowanie ich do innych zadań, a co za tym idzie nie stosowano ich do nowych, coraz bardziej skomplikowanych technologicznie zadań. Koniec lat 80-tych poświęcono wiele uwagi badaniom nad możliwościami zwiększenia funkcjonalności robotów, m.in. przez odpowiednie układy sensoryczne, czy przez rozwój możliwości programowania robotów.

Humanoidy

Choć początków robotów humanoidalnych można doszukiwać się w Robocie Leonarda (1495r.) czy też w automacie Torriano[243] niemieckiego konstruktora Hansa

[243] automat w ludzkiej formie, przedstawiający kobietę grającą na lutni [142]

Bullmanna (1525r.) [142], to należy pamiętać, że tym konstrukcjom brakuje najważniejszych atrybutów robota humanoidalnego – rozbudowanej sensoryki oraz sztucznej inteligencji. Od pierwszych automatów humanoidalnych do współczesnych robotów humanoidalnych poczyniono olbrzymi postęp. Częściowo wynika to z podejścia, jakie swego czasu prezentowało kilka firm (wśród nich Honda, Toyota, Sony) – wkład w rozwój humanoidów dawał namacalny zysk wizerunku firmy. Społeczeństwo bardzo przychylnie przyglądało się poczynaniom firm, media chętnie relacjonowały. W ludziach odżyła nadzieja na posiadanie cybernetycznego pomocnika. Honda zasłynęła całą serią prototypów, w tym słynny P3 oraz ASIMO. Toyota zaistniała dzięki swoim automatycznym muzykom, wiele innych firm również próbowało swych sił. Jednak w pewnym momencie osiągnięto być może granice ówczesnej technologii – zabrakło bowiem spektakularnych sukcesów, zainteresowanie słabło, tematyka spowszedniała. Ikona robotyki – ASIMO – choć nadal nazywany robotem – coraz częściej był kategoryzowany przez ludzi jako automat (pomimo rozbudowanej sensoryki i algorytmów decyzyjnych), ponieważ na pokazach wykonywał preprogramowane zadania. Inną strategię wybrała firma Sony, prezentując zabawkę – mechanicznego psa AIBO. Mechanicznych zabawek było w tamtych czasach wiele, ale ta miała w sobie coś wyjątkowego (jej nazwa pochodzi od Artificial Intelligence roBOt) – była zdolna nie tylko do interakcji z otoczeniem, ale również pamiętania i uczenia się.

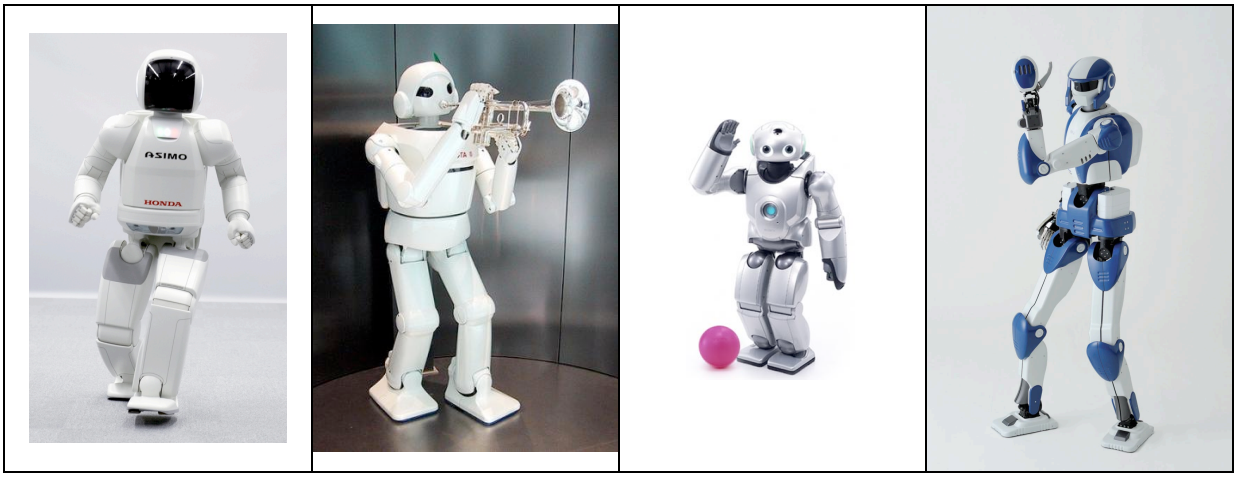
Humanoidy, człekokształtne roboty inteligentne, podobnie jak wcześniej roboty przemysłowe, trafiły na swój okres rozczarowania i analizy (choć w przeciwieństwie do przemysłowych nie były w okresie entuzjazmu tak licznie produkowane). Historia robotów przemysłowych uczy, że teraz może nadejść czas ponownego ożywienia, jeżeli tylko prawidłowo zostaną zinterpretowane słabe strony robotów inteligentnych.

Ostatnie dwa lata, a w szczególności miniony rok 2010 to zdecydowanie najbardziej aktywny okres ożywienia, również w kręgach akademickich. Do liderów robotyki humanoidalnej (oprócz firm Honda, Toyota i Sony) dołączyły m.in. [38]:

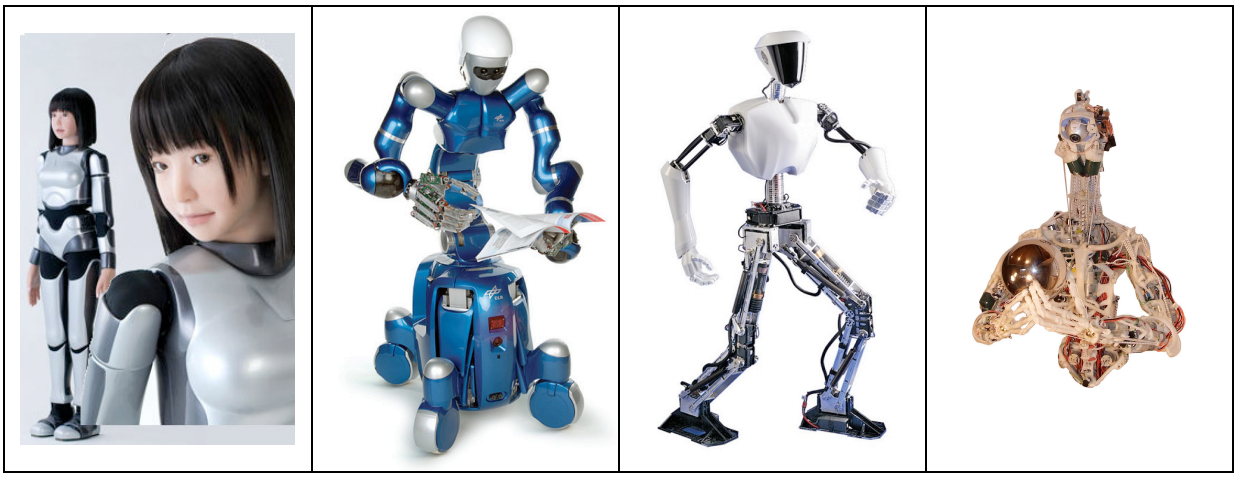
- Japan National Institute of Advanced Industrial Science and Technology oraz KAWADA Industries Inc. – z humanoidem HRP 4 [57] oraz androidem[244] HRP 4c [85],
- Korea Institute of Science and Technology, Humanoid Robot Research Center – humanoid HUBO 2 [36],

[244] androidem w nieortodoksyjnym znaczeniu tego słowa, czyli humanoidem upodobnionym wizualnie do człowieka

- Korea Institute of Science and Technology oraz Samsung Electronics – humanoid Mahru Z [37],
- Pal Robotics (Hiszpania, Zjednoczone Emiraty Arabskie) – humanoid REEM-B [38][139],
- German Aerospace Center, Institute of Robotics and Mechatronics – modularny humanoid Justin [28],
- Virginia Tech's Robotics and Mechanisms Laboratory – humanoid CHARLI [145],
- University of Tehran, Center for Advanced Vehicles, Faculty of Mechanical Engineering (Iran) – humanoid Surena [39],
- Konsorcjum złożone z pięciu partnerów: University of Sussex, University of Zurich (Artificial Intelligence Lab), Univerzitet u Beogradu (Elektrotehnicki Fakultet), Technische Universität München (Robotics and Embedded Systems), The Robot Studio – antropomorfizowany robot ECCEROBOT [17][18]



Rys. A.1. Cztery najbardziej znane roboty humanoidalne: od lewej: ASiMO (Honda), Partner (Toyota), Qrio (SONY), HRP-4 (KAWADA&AIST)



Rys. A.2. Cztery kolejne, najbardziej zaawansowane roboty humanoidalne: od lewej: HRP-4C, JUSTIN, CHARLI, ECCEROBOT

Wielu inżynierów uważa, że kolejnym etapem historii humanoidów są tzw. aktroidy – roboty antropomorficzne wizualnie upodabniane do człowieka w celu m.in. umożliwienia ekspresji za pomocą mimiki. Wspaniałym przykładem jest Actroid-F, efekt współpracy firmy Kokoro oraz Osaka University ([71] -materiał wideo). W niniejszej pracy aktroidy nie będą punktem odniesienia, ponieważ dzisiejsze aktroidy zwykle są robotami działającymi jedynie w trybie teleprezencji. Wśród aktroidów wyposażonych w możliwość interakcji z człowiekiem/ /otoczeniem zazwyczaj stosuje się interakcję opartą o komendy głosowe. Systemy wizyjne dzisiejszych aktroidów posiadają minimalną funkcjonalność.

A.1. Inteligencja robotów

Biorąc pod uwagę dynamikę rozwoju cywilizacyjnego oraz starzejące się społeczeństwo krajów wysoko rozwiniętych można stwierdzić, że istnieje paląca potrzeba stworzenia maszyny będącej w stanie wejść w aktywną, niezdeterminowaną interakcję z człowiekiem, również z człowiekiem starszym, niedołężnym. Na wagę złota byłby robot, który mógłby pomóc w codziennym życiu, w prostych czynnościach, a w razie zagrożenia podjąć decyzję o wezwaniu pomocy lub choćby o zainicjowaniu połączenia wideorozmowy. Pomoc osoby trzeciej mogłaby być udzielona natychmiast - przez tego samego robota, w trybie teleoperacji. Dzisiejsze humanoidy, uogólniając: roboty inteligentne, nie są jeszcze na to gotowe. Brakuje im właśnie inteligencji.

Istnieje wiele definicji i miar inteligencji. Jednak w przypadku maszyn inteligencja jest niesamowicie trudnym zagadnieniem do zdefiniowania. Po dziś dzień wysoko cenione jest intuicyjne podejście zaproponowane w 1950 roku przez Alana Turinga [144] – aby inteligencję maszyny oceniać nie na podstawie definicji, lecz za pomocą gry (testu). Słynny test Turinga zyskał swoich zwolenników i przetrwał do dziś między innymi dlatego, że jedną z najważniejszych cech maszyny inteligentnej powinna być właśnie zdolność swobodnej interakcji i komunikacji z człowiekiem.

Sztuczna inteligencja

Jedna z definicji terminu „sztuczna inteligencja” określa to pojęcie jako inteligencję maszyny uzyskaną w procesie inżynieryjnym (w przeciwieństwie do naturalnego). Jest to

najbardziej pierwotna i mimo wszystko dość wąska definicja. Większość maszyn (lub aplikacji), o których potocznie mówi się, że „mają zaimplementowane metody sztucznej inteligencji", faktycznie podlega tej rdzennej definicji. Aby uwypuklić drobną różnicę między takim rozumieniem sztucznej inteligencji, a innymi przypadkami, można o tych pozostałych przypadkach powiedzieć: „Nie jest ważne jak zaprojektowany jest sztuczny system, o ile tylko zachowuje się on tak jak człowiek" [43]. W tym podejściu istnieje analogia między obliczeniami a myśleniem. Mózg operuje abstrakcyjnymi symbolami bazując na stanie wejść tak samo jak każdy komputer – mózg jest więc rodzajem komputera [43]. W 1943 roku McCulloch i Pitts opublikowali pracę [76], w której opisali możliwość funkcjonowania pojedynczego neuronu jako bramki logicznej, co utwierdziło zwolenników porównywania mózgu z komputerem w przekonaniu, że badania są na właściwej ścieżce. W czasach powojennych, gdy komputery stawały się coraz bardziej dostępne, zaczęły pojawiać się nowe zastosowania i nowe implementacje. Cały świat nauki patrzył z zainteresowaniem na rozkwit „sztucznej inteligencji" prorokując rychłe dorównanie możliwościom intelektualnym człowieka, a nawet ich przewyższenie. Tak się jednak nie stało. Zarówno tworzone maszyny, jak i pisane aplikacje, potrafiły rozwiązywać tylko takie zadania, jakie przewidział programista. Program był pisany pod kątem danego zadania – albo do tłumaczenia tekstu, albo do szachów, albo innego ściśle określonego celu. Program, który w maju 1997 roku wygrał z mistrzem świata Garrim Kasparovem w szachy, nie potrafi grać w warcaby [43]. Wygrał z mistrzem nie z powodu inteligencji, lecz dlatego, że był o wiele szybszy. Człowiek, gdy popatrzy na planszę, od razu widzi, w którym miejscu rozegra się walka; komputer natomiast analizuje wszystkie możliwe kombinacje ruchów. Komputer potrafi wygrać w szachy z mistrzem „nie rozumiejąc" ich, podobnie jak kalkulator wykonuje działania „nie znając" matematyki [43]. Nawet najsłynniejszy program wśród miłośników sztucznej inteligencji – Eliza – który nota bene w swoich czasach był niesamowicie blisko przejścia testu Turinga – działa także w oparciu o bardzo prosty algorytm[245]. W każdym przypadku jednak, algorytm danego programu postawiony w nowych, nieznanych warunkach, nie zadziała prawidłowo.

Nieco młodsze narzędzie stanowią systemy eksperckie, które oparciu o predefiniowaną bazę danych z faktami, potrafią udzielać odpowiedzi na stawiane pytania.

[245] wyszukuje w zdaniu słowa kluczowe i w oparciu o bazę zwrotów odpowiada wstawiając tylko słowa kluczowe w odpowiednim miejscu

Systemy eksperckie okazały się bardzo przydatnym narzędziem i są stosowane z powodzeniem po dziś dzień, ale niestety nie przybliżają sztucznej inteligencji do ludzkiej.

Sieci neuronowe

Sieci neuronowe pojawiły się jako alternatywa i konkurencja dla zwolenników sztucznej inteligencji. W latach 80-tych dobra passa sztucznej inteligencji zaczęła się mieć ku końcowi[246] – niewykluczone, że w związku z brakiem jakichś spektakularnych osiągnięć, nowości. Zyskali na tym zwolennicy sieci neuronowych, a z biegiem czasu sieci neuronowe dowiodły swojej wartości okazując się bardzo szeroko stosowaną techniką.

Dziś oba te nurty współistnieją uzupełniając się a nawet łącząc w niektórych implementacjach.

Sieci neuronowe składają się z połączonych (w określoną strukturę) pojedynczych sztucznych neuronów, których sposób funkcjonowania jest całkowicie zaczerpnięty ze świata biologii. Eksperymenty ze sztucznymi sieciami neuronowymi pokazały, że dzięki odpowiednim połączeniom między neuronami, cała sieć jest w stanie rozwiązywać niektóre problemy nierozwiązywalne dla sztucznej inteligencji, głównie dzięki zdolnościom aproksymacyjnym i generalizacyjnym [119]. Do podstawowych zalet modeli behawioralnych tworzonych przy pomocy tego narzędzia należy brak konieczności odwoływania się do przyczynowości, co bezpośrednio prowadzi do możliwości otrzymywania prawidłowych odpowiedzi w zadaniach, w których nie jest znana zasada/reguła wyznaczania odpowiedzi. Czasami sieci neuronowe stosowane są dla oszczędności – gdy prosty model behawioralny na tyle dobrze odzwierciedla zachowanie modelowanego systemu, aby zrezygnować z trudnego i czasochłonnego tworzenia modelu dokładnego.

Budowa sieci neuronowej z założenia jest w pewnym zakresie wzorowana na budowie mózgu – jedno i drugie tworzą odpowiednio połączone neurony, ponadto sieć neuronowa nie ma wydzielonego obszaru danych i listy instrukcji (jak jest w przypadku klasycznego programu komputerowego, a więc i algorytmu sztucznej inteligencji).

W niektórych zastosowaniach sieć wystarczająca do otrzymania satysfakcjonującego wyniku składa się z kilku do kilkunastu neuronów (tymczasem kora nowa dorosłego człowieka zawiera szacunkowo około 30 miliardów neuronów, a według innych źródeł cały

[246] Sztuczna inteligencja jest zagadnieniem nadal zgłębianym i eksplorowanym, nie można już jednak twierdzić, że sztuczna inteligencja jest bardziej popularna lub bardziej efektywna niż sieci neuronowe.

mózg zawiera około 100 miliardów neuronów[247]). Stąd widać, że zbudowanie sztucznej sieci neuronowej porównywalnej z korą nową człowieka jest nie lada wyzwaniem.

Dzisiaj sztuczne sieci neuronowe są nadal bardzo powszechnym narzędziem, często łączone są z systemami eksperckimi oraz logiką rozmytą[248]. Logika rozmyta łagodzi wymogi precyzji oraz umożliwia użycie operatorów lingwistycznych i reguł wnioskowania [119].

Inteligencja maszyn w niniejszej pracy

W niniejszej pracy nie zastosowano żadnej spośród omówionych powyżej rdzennych koncepcji inteligencji maszyn. Sztuczna inteligencja, rozumiana jako zachowanie pre−programowane, jest sprzeczna z zasadą elastyczności i uniwersalności ludzkiego umysłu. Sztuczne sieci neuronowe zostały natomiast odrzucone w toku badań, jako zbyt złożone w przypadku rozbudowanych sieci. Pierwsze eksperymenty przeprowadzano z użyciem najprostszych sieci neuronowych, jednak wycofano się z powodu ogromu pracy potrzebnej do opracowania sieci neuronowej zdolnej do rozpoznawania przy użyciu rzeczywistych obrazów (oraz bardzo małego prawdopodobieństwa sukcesu opracowania takiej sieci i jej implementacji). Zamiast tego uwaga autora skierowała się ku rozwiązaniom o wiele bardziej złożonym (choć de facto wywodzącym się z omówionych wcześniej rdzennych nurtów) – ku architekturom emergentnym, a dokładniej sieciom HTM (omówiono je w rozdziale 3.3). Węzły sieci HTM zbudowane są w oparciu o pamięć autoasocjacyjną, co czyni tą technologię ściśle powiązaną z sieciami neuronowymi.

A.2. Cel horyzontalny badań

Pięcioletnie dziecko rozumie i samodzielnie posługuje się językiem mówionym, potrafi odróżnić psa od kota, potrafi bawić się piłką – te trzy umiejętności (tak jak i wiele innych) są proste dla ludzi, a jednocześnie niewykonalne dla robotów [93]. Po części wynika z tego, że naukowcy od pół wieku starają się tworzyć inteligentne maszyny w oparciu

[247] podane wartości – 30 i 100 mld – są szacunkowe a ponadto pochodzą z różnych źródeł literaturowych nie powinny zatem być tu ze sobą konfrontowane.

[248] Co wcale nie oznacza, że logika rozmyta jest podejściem zarezerwowanym wyłącznie dla sztucznych sieci neuronowych – jest z powodzeniem stosowana równie intensywnie w zadaniach sztucznej inteligencji. Wspaniałą lekturę w tym zakresie zapewnia [123]

o logikę komputera. Profesor filozofii John Searle, aby udowodnić, że komputery nie mogą być inteligentne, zaproponował w 1980 roku eksperyment myślowy zwany „chińskim pokojem" [43] – osoba mówiąca wyłącznie po angielsku, zamknięta w pokoju z biurkiem, otrzymuje opowiadanie i dotyczące go pytania, wszystko napisane po chińsku. Człowiek ten ma do dyspozycji stosy kartek papieru, ołówki i księgę z instrukcjami. Czasem instrukcje każą napisać pewne znaki na kartkach, czasem zamienić ich kolejność lub wymazać niektóre z nich. Kiedy instrukcje stwierdzą, że to już koniec, na kartce znajduje się jakiś zestaw znaków, który jest odpowiedzią na pytania. Osoba, która zna chiński widzi, że wszystkie odpowiedzi są poprawne, a niektóre nawet błyskotliwe. Tylko czy poprawność i jakość tych odpowiedzi wynika z inteligencji osoby, która odpowiadała na pytania? „Chiński pokój" wywołał prawdziwą burzę między zwolennikami i przeciwnikami sztucznej inteligencji – trudno zdefiniować kryteria, kiedy dany system jest inteligentny, a kiedy nie. Jeżeli natomiast za miarę inteligencji maszyny będzie wzięty stopień podobieństwa jej algorytmów przetwarzania do procesów myślowych człowieka, wówczas można śmiało powiedzieć, że inteligentne maszyny jeszcze nie istnieją.

Świat opisywany w powieściach Isaaca Asimova był, jest i prawdopodobnie jeszcze długo będzie fantastyką. Nikt nie wie, czy sieci HTM okażą się ważnym krokiem w stronę stworzenia inteligentnych maszyn – ale jest to następny krok. Roboty już dziś stanowią bardzo cenioną siłę roboczą, może kiedyś staną się czymś więcej niż narzędziami czy zabawkami. *Historia rozwoju robotyki dopiero się zaczyna.* [84]

Dodatek B. Szczegóły implementacyjne eksperymentów

Ten rozdział zawiera skondensowane informacje dodatkowe dotyczące praktycznych implementacji wytworzonych aplikacji napisanych na potrzeby eksperymentów. Informacje tu zawarte zostały uznane za zbyt szczegółowe by ująć je w treści rozdziału 4.3, ale na tyle ciekawe bądź istotne by ich nie pomijać.

B.1. Implementacja 1

Wejście i interfejs

Dane wejściowe aplikacji stanowi strumień wideo pochodzący z dwóch kamer podłączonych do komputera i obecnych w systemie operacyjnym. Możliwe jest korzystanie z pojedynczej kamery, wówczas drugie źródło (bez zainicjalizowanej kamery) będzie przekazywało „czarny ekran". Obraz z kamer/kamery przekazywany jest na powierzchnię (Tpanel→Canvas) panelu pierwszego z użyciem tzw. uchwytu (*ang. handle*) – jest to najszybszy istniejący programowy sposób prezentacji źródła wideo na formatce. Od tego momentu poszczególne klatki obrazów przetwarzane są cyklicznie, w odstępach czasowych wynikających z wydajności systemu. W pierwszym kroku zawartość powierzchni paneli kopiowana jest na wspólną globalnie zdefiniowaną wirtualną powierzchnię – ponieważ dla przyspieszenia obliczeń rozdzielczość obrazu pozyskiwanego z kamer predefiniowano na 320x240 pikseli, wspólna (panoramiczna) powierzchnia ma rozmiary 640x240 pikseli. W drugim kroku powierzchnia ta jest konwertowana na postać trójwymiarowej macierzy bajtów o rozmiarze 640x240x3 (gdzie 3 oznacza kolory składowe: czerwony, niebieski, zielony). Można przyspieszyć działanie aplikacji rezygnując z informacji o kolorze, tworząc monochromatyczną (dwuwymiarową) macierz, ale w ten sposób część informacji zostanie

utracona a więc rozpoznawanie zadanych obiektów może być znacznie trudniejsze – często informacja o kolorze jest informacją kluczową (jak w przypadku sygnalizacji świetlnej na skrzyżowaniu), tak więc kojarzenie wartości poszczególnych bajtów obrazu monochromatycznego z wartością luminancji odpowiadających im pikseli nie jest rozwiązaniem ani jedynym ani najlepszym.

Podczas wspomnianych konwersji ma miejsce dodatkowa korekta jasności i kontrastu – aby poprawić wyrazistość obrazu. Zabieg ten jest wymuszony między innymi tym, że sprzętowa korekta automatycznego balansu bieli ma pewne swoje ograniczenia – gdy kamera przechwytuje obraz bardzo jasny, wówczas automatyczny balans bieli obniża poziom jasności, niestety na obrazie wynikowym obiekty o kolorze czarnym są zbyt jasne, również obiektów o kolorach jasnych (białych) jest zbyt wiele. Podobnie jest w przypadku przechwytywania obrazu niedoświetlonej scenerii – wszystko jest zbyt ciemne, białe obiekty są żółte lub różowe zamiast białe.

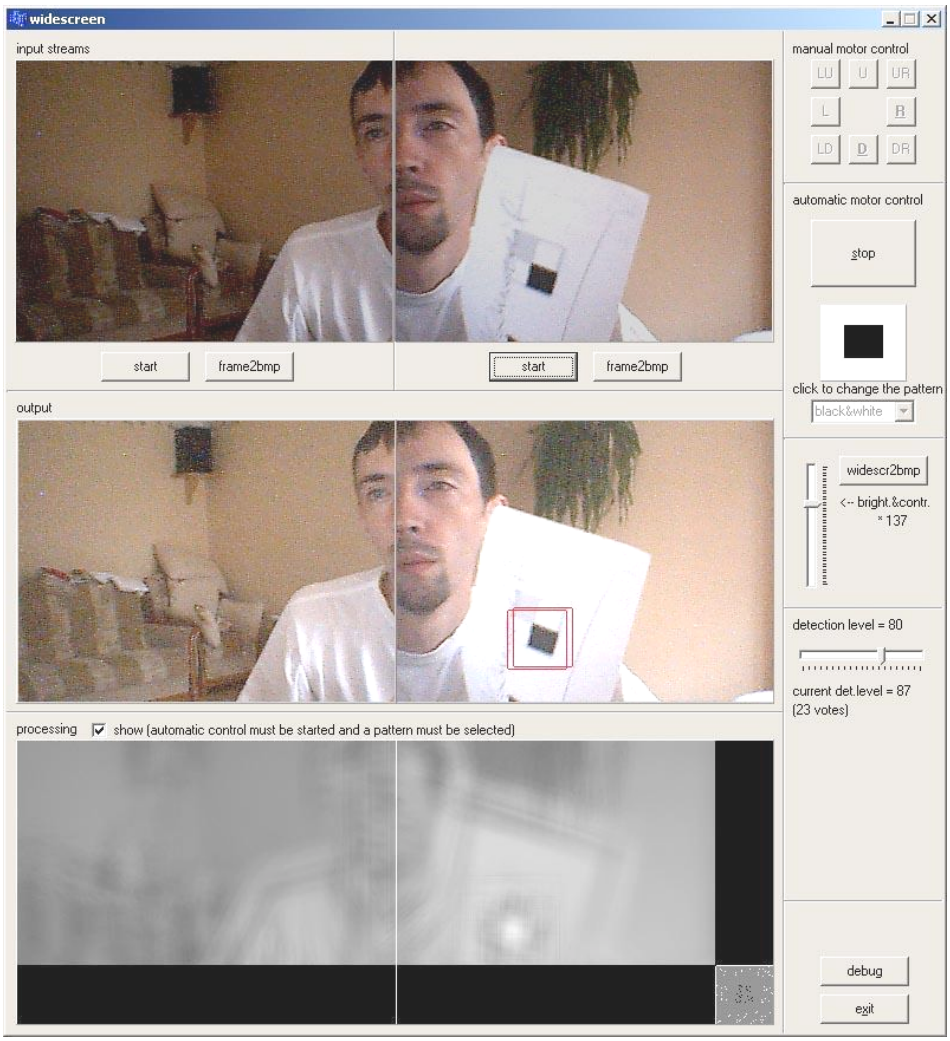
Trzeci panel, również panoramiczny 640x240 pikseli, pozostaje do dyspozycji projektanta-programisty (eksperymentującego z sieciami neuronowymi).

Po prawej stronie interfejsu użytkownika dostępnych jest kilka sekcji umożliwiających ręczne lub automatyczne (z wykorzystaniem algorytmu decyzyjnego) sterowanie pracą silników, jak również kilka komponentów umożliwiających zmianę niektórych parametrów algorytmu. Przyciski ręcznego sterowania silników (nieaktywne w trybie sterowania automatycznego) pokazują w którą stronę powinien przesuwać się obraz w wyniku pracy silników – symbol literowy umieszczony na danym przycisku staje się pogrubiony i podkreślony podczas pracy danego silnika w określonym kierunku. Na rysunku (Rys. B.1) w sekcji „automatic motor control" widać przycisk „stop" co znaczy, że aktualnie aplikacja steruje silnikami w trybie autonomicznym – w (nieaktywnej z tego powodu) sekcji „manual motor control" widać pogrubione etykiety „R" i „D", co oznacza, że w tych kierunkach obraca się platforma z kamerami. Jest to wynikiem działania jedynej[249] procedury (oprócz wizualizacji) obecnie wykorzystującej wiedzę wynikową sieci neuronowej:

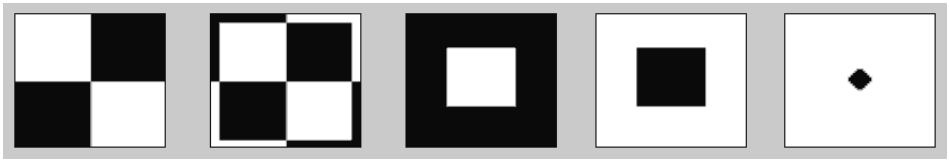
```
void celuj(int x, int y);
```

Funkcja ta steruje ruchem silników tak, aby wykryty wzorzec znalazł się w linii promienia lasera.

[249] w obecnej wersji aplikacji



Rys. B.1. Aplikacja1 – interfejs użytkownika. (rysunek pochodzi z [114], opracowanie własne)



Rys. B.2. Aplikacja1 – predefiniowane wzorce, które sieć może rozpoznawać w obrazie. (rysunek pochodzi z [114], opracowanie własne)

Projektant-programista sieci neuronowych może zmieniać następujące parametry algorytmu: rodzaj wzorca (Rys. B.2) jaki ma rozpoznawać projektowana sieć neuronowa oraz jego barwy (czarno-biały, czerwono-biały, zielono-biały, niebiesko-biały). Rodzaj wzorca, wybierany za pomocą myszki, wyświetlany jest na powierzchni komponentu Image3→Canvas. Powierzchnia ta została pomyślana jako miejsce pozyskiwania (porównywania) danych treningowych dla sieci neuronowej.

Możliwa jest również modyfikacja automatycznie dobranego wspólnego[250] współczynnika korekcji jasności i kontrastu (jeżeli zdaniem programisty zostały dobrane nieprawidłowo, lub zmieniły się znacząco warunki oświetlenia sceny lub wzorca).

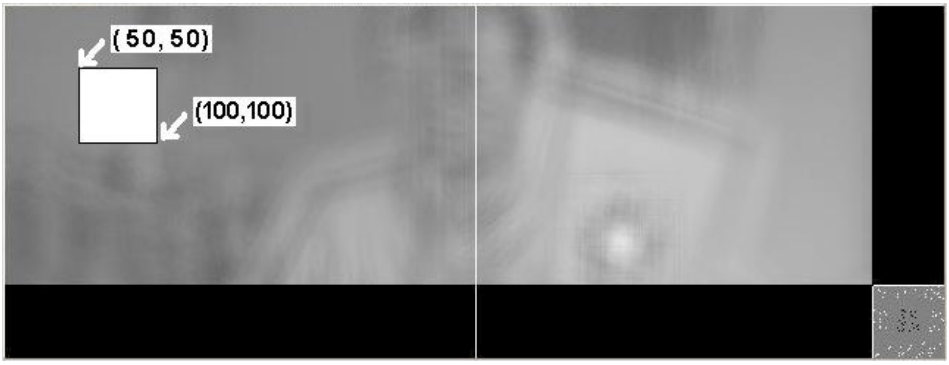
Przetwarzanie

Klatki obrazu pobierane są ze strumienia wejściowego na żądanie, dzięki czemu obraz wejściowy widoczny na górnych panelach jest odświeżany z częstotliwością jaką potrafi obsłużyć sterownik kamery, a obraz wyjściowy widoczny na panelu środkowym jest uaktualniany cyklicznie po wykonaniu wszystkich innych poleceń pętli głównej programu.

W każdym z cykli pętli głównej algorytmu, zgodnie z jedną z koncepcji AV opisanych w literaturze [83], obraz wejściowy jest skanowany w poziomie. Obszar będący obszarem roboczym wejść sieci neuronowej (prostokąt o wysokości 50 pikseli i szerokości 50 pikseli) przesuwany jest wzdłuż kolejnych linii poziomych obrazu prezentując kolejne kombinacje wartości na wejściach sieci.

Aplikacja została pomyślana jako narzędzie o wymiennej/edytowalnej strukturze sieci neuronowej. Można ją jednak uruchomić nie implementując sieci – posiada zaimplementowany testowo algorytm (perceptron jednowarstwowy prezentujący przykładowy sposób wykorzystania i działania stworzonego przez autorów narzędzia). W dolnym prawym rogu dolnej panoramicznej powierzchni dolnego panelu (patrz: Rys. B.1, Rys. B.3) pokazane jest aktualne rozlokowanie przestrzenne i wagi poszczególnych neuronów (kropka biała symbolizuje neuron aktywowany kolorem białym, czarna – czarnym), a cała pozostała część tej powierzchni reprezentuje odpowiedź sieci dla danego punktu w źródle. Przykładowo (Rys. B.3), w punkcie o współrzędnych (50,50) na trzecim panelu postawiony będzie punkt biały, jeżeli obszar prostokątny (jak na rysunku Rys. B.3) będzie idealnie zgodny z aktywnym wzorcem, natomiast czarny jeżeli obszar będzie całkowitym przeciwieństwem (negatywem) danego wzorca. Oczywiście te dwa przypadki są niezmiernie rzadko spotykane, dlatego należy zadać pewną wartość progową określającą czy w danym punkcie wzorzec rozpoznano, czy nie. To zadanie klasyfikacji może być również wykonywane przez dowolną inną od predefiniowanej, samodzielnie zaprojektowaną, sieć neuronową.

[250] wspólnego dla fragmentów pochodzących od obu kamer



Rys. B.3. Aplikacja1 – przykładowe wykorzystanie dolnego panelu (do podglądu działania algorytmu rozpoznającego). (rysunek pochodzi z [114], opracowanie własne)

Wyjście

Programista-projektant sieci neuronowej może w stworzonym narzędziu w rozmaity sposób prezentować wyniki działania swojego rozwiązania:

- Edytując wspomnianą wcześniej trójwymiarową macierz obrazu przetwarzanego, np. dorysowując ramkę, jak na rysunku Rys. B.1, albo zaznaczając współrzędne – macierz ta jest bowiem wizualizowana na środkowym panelu każdorazowo po zakończeniu analizy kolejnej klatki obrazu.
- Może użyć dolnego panelu (całkowicie dowolnie).
- Jeżeli dysponuje nie tylko samym programem, ale również częścią mechaniczną stworzonego przez autorów[251] narzędzia, wówczas ma możliwość kontrolowania pracy silników i lasera wyjściami sieci neuronowej.

Aplikacja udostępnia bardzo prosty interfejs sterowania programowego platformą – wystarczy uruchomić funkcję:

```
void Rotate(bool góra, bool lewo, bool dół, bool prawo, int liczba_kroków);
```

Odpowiednie parametry funkcji oznaczają możliwe kierunki obrotu platformy, ostatni parametr oznacza liczbę kroków silnika krokowego lub – jeśli będzie tu wartość 0 – tryb pracy ciągłej. Przykładowo po uruchomieniu funkcji „Rotate” w następujący sposób:

```
Rotate(false, true, false, false, 10);
```

jeden z silników wykona 10 kroków i zatrzyma się, tak że kamery systemu wizyjnego zostaną obrócone o pewien kąt w lewą stronę (kąt ten zależy od silnika krokowego i zastosowanych przekładni zębatych). Jeżeli przed wykonaniem wszystkich 10 kroków zostanie zakończona analiza kolejnej klatki obrazu źródłowego, kolejnym wywołaniem funkcji „Rotate” można zresetować poprzedni rozkaz. Zatrzymanie silników następuje

[251] Część mechaniczna i elektroniczna została zaprojektowana i wykonana we współpracy ze Zbigniewem Zającem, współautorem rozwiązania [114]

w wyniku zakończenia wykonywania sekwencji z komendy „Rotate” (i braku kolejnej komendy) lub w wyniku wywołania funkcji „Rotate” z następującymi parametrami:

```
Rotate(false, false, false, false, 0);
```

Do dyspozycji jest również wskaźnik laserowy – jeżeli zadaniem aplikacji jest wskazać jakiś obiekt/wzorzec w świecie rzeczywistym. Załączanie jak i wyłączanie lasera zaimplementowano funkcją:

```
void Laser(bool włączony);
```

Wytworzone narzędzie – w postaci aplikacji i mechaniki – doskonale spełnia stawiane mu pierwotnie wymagania. Osoba chętna do eksperymentowania ze sztucznymi sieciami neuronowymi może w tym narzędziu odnaleźć bardzo dobry punkt wyjściowy do tworzenia sieci. Aplikacja stwarza ku temu sprzyjające warunki oferując przejrzyste, czytelne i „przyjazne użytkownikowi” struktury i funkcje. Dzięki temu podejściu nie tylko „podłączanie” wejść i wyjść tworzonej sieci jest łatwe, ale również postać struktur i funkcji nie determinuje rozwiązania jakie musiałby przyjąć użytkownik. Sposób pozyskania danych dla sieci (jak i wizualizacji stanu jej wyjść) jest przy tym bardzo elastyczny – programista ma tutaj całkowitą swobodę.

Istotny jest również fakt, że aplikacja została stworzona w taki sposób, aby możliwe było jej używanie w przypadku, gdy użytkownik nie dysponuje kompletnym sprzętem. Jeżeli do komputera nie podłączy silników, aplikacja będzie działać zupełnie normalnie. Jeżeli nie dysponuje dwiema kamerami internetowymi, aplikacja równie dobrze zadziała na jednej.

B.2. Implementacja 2

Wejście

Dane wejściowe stanowi strumień wideo pochodzący z podłączonej do komputera kamery. Ponieważ żaden z posiadanych testowanych systemów rodziny Linux nie obsługiwał żadnej z posiadanych kamer internetowych, niezbędne było napisanie odpowiednich sterowników kamery. Współpraca aplikacji z dedykowaną, nietypową kamerą może być znacznym utrudnieniem w ewentualnych próbach późniejszego odtworzenia eksperymentu, dlatego zdecydowano się tworzyć nowy sterownik w oparciu o jeden z wówczas

największych pakietów sterowników kamer internetowych – bibliotekę sterowników „spca50x.c", grupującą wiele urządzeń w 19 podstawowych grupach kamer:

```
static struct cam_list clist[] = {
        {   0, "Unknown" },
        {   1, "Intel PC Camera Pro" },
        {   2, "Intel Create and Share" },
        {   3, "Grandtec V.cap" },
        {   4, "ViewQuest M318B" },
        {   5, "ViewQuest VQ110" },
        {   6, "Kodak DVC-325" },
        {   7, "Mustek gSmart mini 2" },
        {   8, "Mustek gSmart mini 3" },
        {   9, "Creative PC-CAM 300" },
        {  10, "D-Link DSC-350" },
        {  11, "Creative PC-CAM 600" },
        {  12, "Intel Pocket PC Camera" },
        {  13, "Intel Easy PC Camera" },
        {  14, "3Com Home Connect Lite"},
        {  15, "Kodak EZ200"},
        {  16, "Maxell Max Pocket LEdit. 1.3 MPixels" },
        {  17, "Aiptek Mini PenCam  2 MPixels" },
        {  18, "Hama USB Sightcam 100" },
        {  -1, NULL }
};
```

Po drobnych modyfikacjach[252] kodu sterownika kamera została rozpoznana i przyporządkowana do jednej z grup.

Niezaprzeczalną zaletą użycia wspomnianego sterownika była jego natywna obsługa biblioteki SDL (*ang. Simple DirectMedia Library*, [127]), która wykorzystuje akceleracje sprzętową dla grafiki 3D i 2D.

Algorytm, SSN

„Aplikacja2" z założenia miała służyć jako środowisko eksperymentów pozwalających ocenić opracowywane w pracy rozwiązania. Nie było celem stworzenie systemu rozpoznającego czy klasyfikującego. Podczas migracji fragmentów kodu „Aplikacji1" przeniesiono również (i nieco poprawiono) procedury obsługi sztucznych sieci neuronowych i wizualizacji aktywności. Nie przeprowadzano natomiast, nie na potrzeby niniejszej pracy, eksperymentów wykorzystujących którekolwiek z funkcjonalności związanych z SSN zaimplementowanych w „Aplikacji2".

Rozwój „Aplikacji2" został przerwany, a później kontynuowany w „Aplikacji4". Funkcjonalności związanej z SSN nie rozwijano z przyczyn opisanych w pracy. Zamiast tego

[252] Szczegóły przeprowadzonych w 2006 przez autora modyfikacji można odszukać w załączonym na płycie CD pliku oryginalnym i zmodyfikowanym sterownika.

zasugerowano zastosowanie jednego z zaawansowanych rozwiązań – platformy NuPIC, zaliczanej do emergentnych architektur kognitywnych [136].

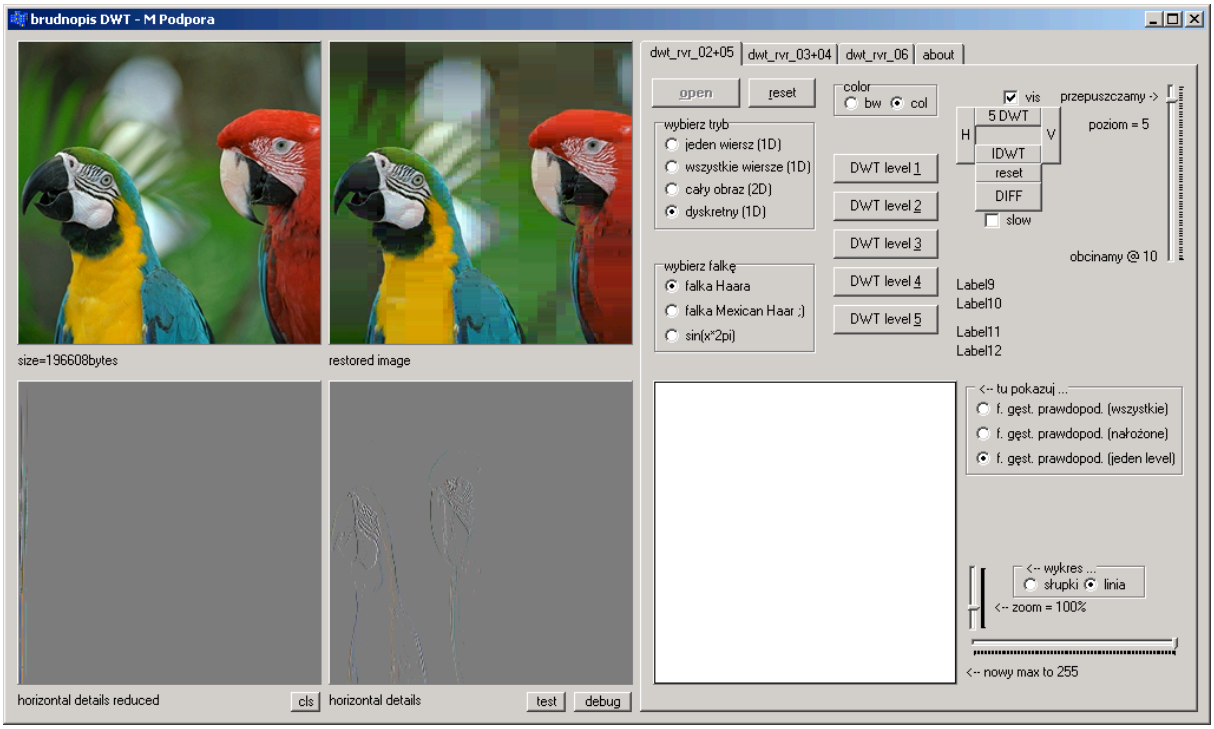
B.3. Implementacja 3

Wejście

Aplikacja operuje na plikach graficznych zapisanych w formacie bmp. Plik musi[253] zawierać obraz o rozmiarach 256x256 pikseli oraz posiadać głębię koloru równą 24 bpp[254]. Aby rozpocząć pracę z aplikacją, należy wczytać obraz (przycisk: „open”).

Interfejs, funkcjonalności, wyjście

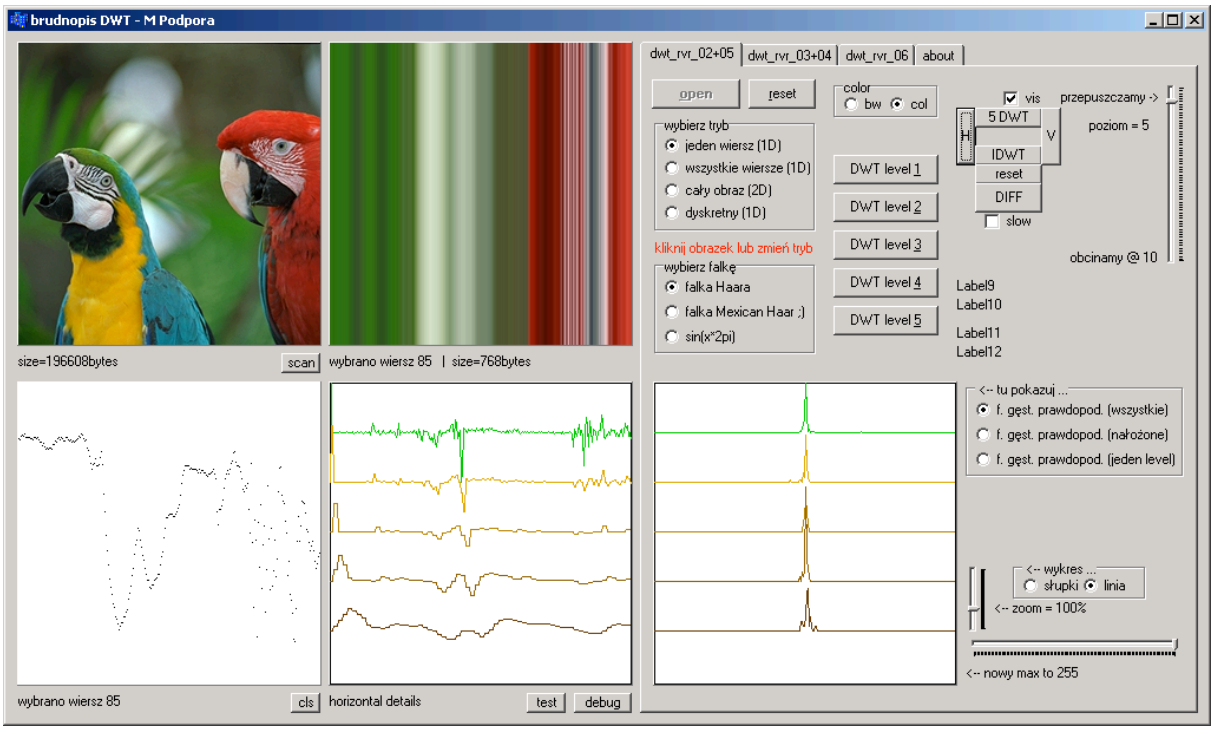
Rys. B.4 przedstawia graficzny interfejs użytkownika z widokiem pierwszej zakładki („dwt_rvr_02+05”) po wczytaniu przykładowego obrazu przyciskiem „open”.



Rys. B.4. Aplikacja3 – graficzny interfejs użytkownika, zakładka1, po wczytaniu pliku. (informacja o prawach autorskich do rysunku „Parrots”: [60])

[253] Pliki zawierające obraz o większych rozmiarach zostaną wczytane poprawnie, ale aplikacja będzie przetwarzać tylko ich fragment – kwadrat o boku 256 pikseli.

[254] bpp – bitów na piksel (*ang. bits per pixel*)



Rys. B.5. Aplikacja3 – graficzny interfejs użytkownika, zakładka1, DWT przeprowadzona jednowymiarowo (pole wyboru: ”wybierz tryb”->”1D”, przycisk: ”H”).

Na powyższym rysunku (Rys. B.5) pokazano jedną z dostępnych opcji podglądu/ /kontroli działania kodu realizującego transformatę DWT. Na panelu pierwszym wybrano myszką linię poziomą (przebiegającą przez oko czerwonej papugi), panel drugi pokazuje tą linię, panel czwarty – zawartość macierzy detali, a panel piąty – funkcje gęstości prawdopodobieństwa dla danych zgromadzonych w tych macierzach. Zakładka ta dokumentuje postęp prac; o wiele większe znaczenie ma zakładka kolejna („dwt_rvr_03+04”) (Rys. B.6).

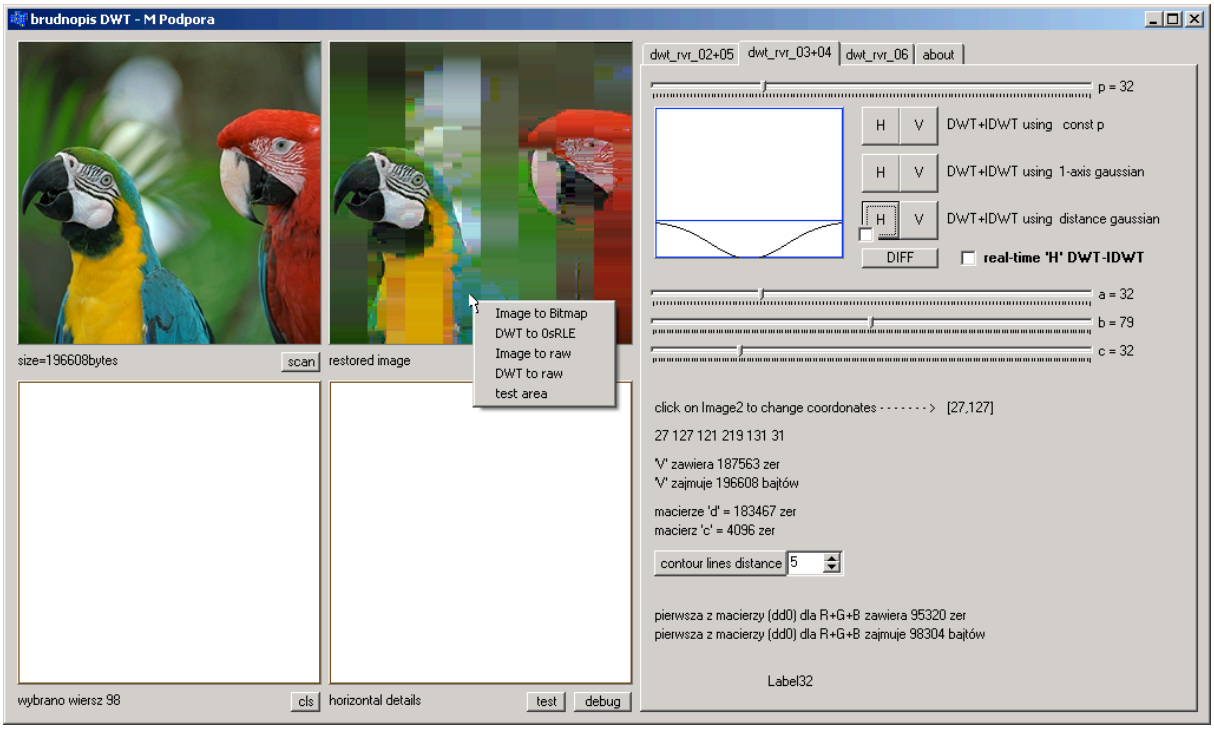
Druga zakładka „Aplikacji3” zawiera najistotniejsze opcje, umożliwiające wstępne określenie przydatności opisanej w rozdziale 3.2.2 koncepcji POI. Po załadowaniu obrazu[255], kolejne przyciski oznaczone „H” wykonują konwersję DWT (i odwrotną IDWT) obrazu z panelu 1 na panel 2 dla:

- stałej wartości parametru threshold dla całego obrazu (jego wartość kontrolowana suwakiem „p”),
- zmiennej jednowymiarowo, zgodnie ze wzorem[256] (1), wartości parametru threshold (wartości parametrów opisującego go równania można zmieniać suwakami „a”, „b” i „c”),

[255] w zakładce pierwszej

[256] patrz: strona 45

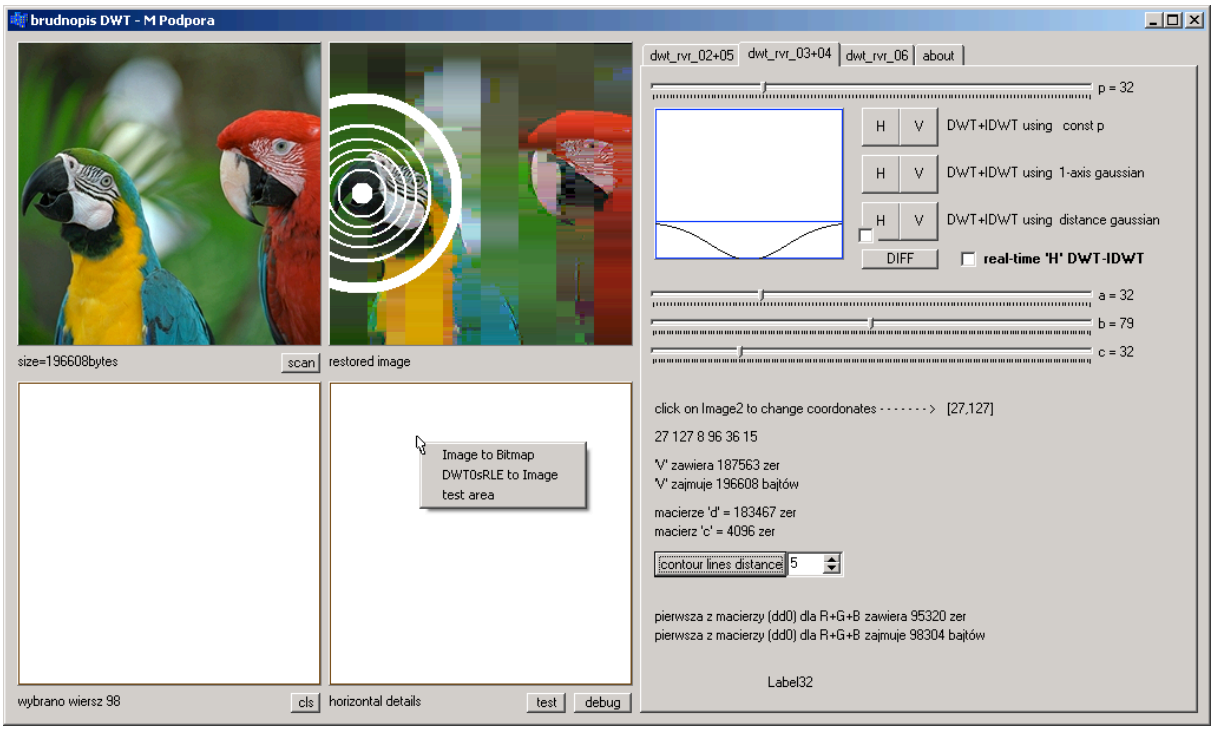
- zmiennej dwuwymiarowo, zgodnie ze wzorem[257] (2), wartości parametru threshold (wartości parametrów opisującego go równania można zmieniać suwakami „a”, „b” i „c”).



Rys. B.6. Aplikacja3 – graficzny interfejs użytkownika, zakładka2, DWT przeprowadzona dwuwymiarowo (kliknięcie dowolnego punktu na panelu2, trzeci przycisk: ”H”).

Etykiety poniżej suwaków zawierają najważniejsze informacje o zawartości macierzy współczynników, uaktualniane są przy każdej operacji powodującej zmianę obrazu wynikowego. Przycisk „contour lines distance” powoduje pokazanie nad obrazem wynikowym izolinii wartości parametru threshold (Rys. B.7). Dodatkowo, pod prawym przyciskiem myszy, na panelu 2 (Rys. B.6), dostępne są opcje eksportu do pliku danych dotyczących bieżącego sposobu przetwarzania obrazu wejściowego. Na panelu 4 (Rys. B.7) dostępne są opcje importu wyeksportowanych plików.

[257] patrz: strona 45

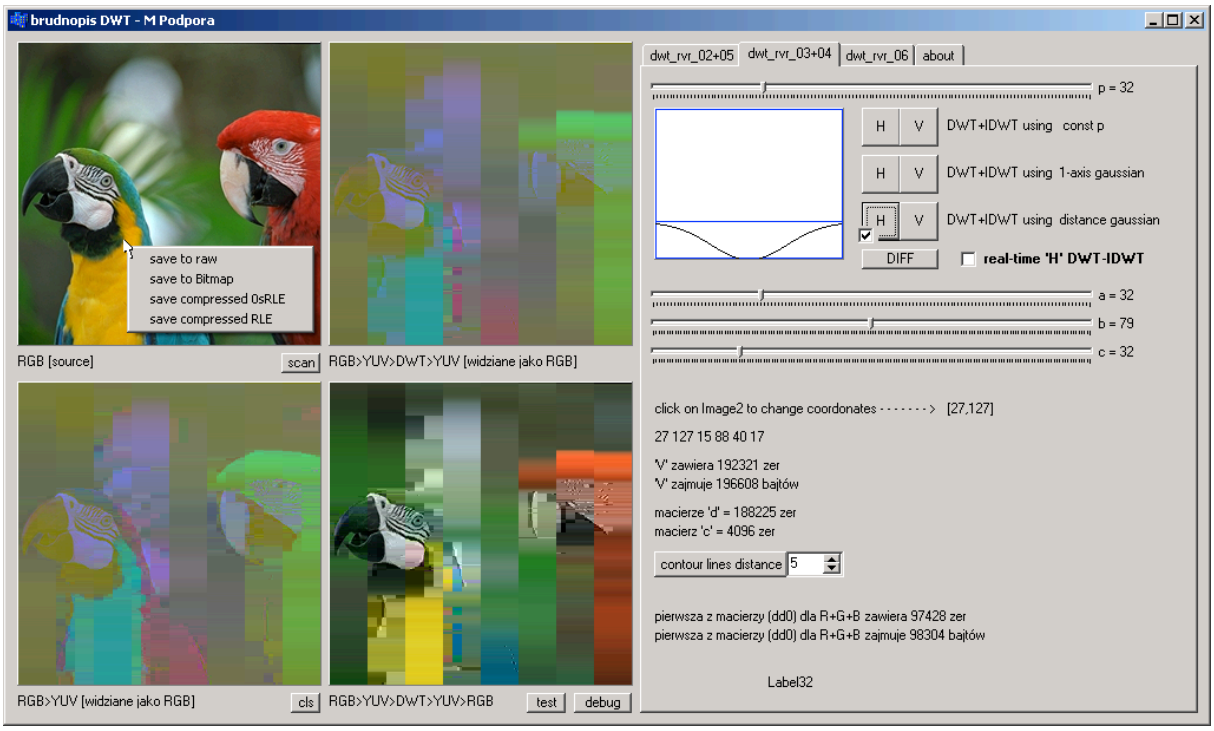


Rys. B.7. Aplikacja3 – graficzny interfejs użytkownika, zakładka2, DWT przeprowadzona dwuwymiarowo (kliknięcie dowolnego punktu na panelu2, trzeci przycisk: ”H”, przycisk ”contour lines distance”).

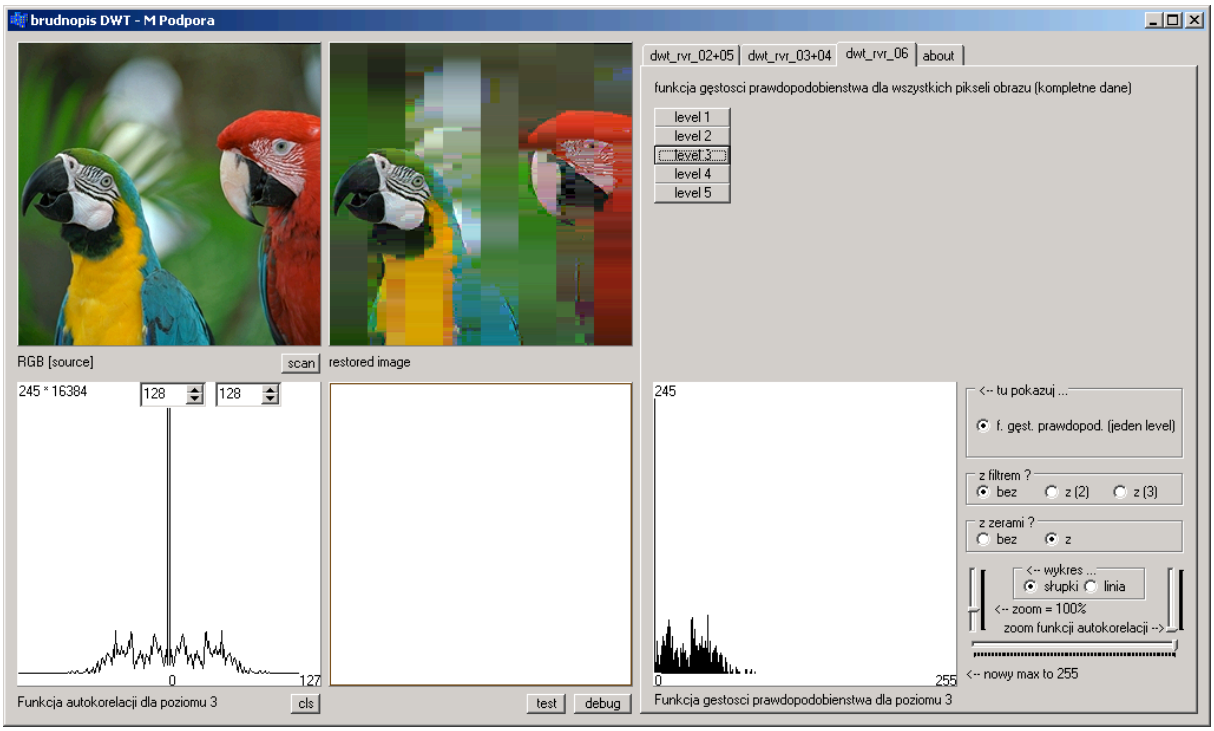
Zaznaczając pole wyboru widoczne przy trzecim przycisku „H”, do algorytmu przetwarzania dołączane są procedury konwersji pomiędzy modelami przestrzeni barw – transformata DWT przeprowadzana jest na obrazie zapisanym w formacie YUV844 zamiast RGB 24bpp (Rys. B.8). Widoczna na rysunku różnica barw (panel 1 a panel 4) wynika z zapisu informacji o kolorze poszczególnych składowych piksela w formacie liczb całkowitych[258].

Trzecia zakładka „Aplikacji3” („dwt_rvr_06”) (Rys. B.9) daje możliwość prostego wygenerowania wykresu funkcji autokorelacji i funkcji gęstości prawdopodobieństwa dla danych zawartych w dowolnej spośród macierzy współczynników transformaty DWT uruchomionej przy ostatniej konwersji.

[258] Skalowanie kolorów jest operacją w pełni odwracalną, jednak na obu etapach niezbędne jest zaokrąglanie do liczb całkowitych, co powoduje efekt widoczny na rysunku Rys. B.8.



Rys. B.8. Aplikacja3 – graficzny interfejs użytkownika, zakładka2, DWT przeprowadzona jednowymiarowo (kliknięcie dowolnego punktu na panelu2, zaznaczenie pola wyboru przy trzecim przycisku ”H”, kliknięcie trzeciego przycisku ”H”).



Rys. B.9. Aplikacja3 – graficzny interfejs użytkownika, zakładka3, DWT przeprowadzona jednowymiarowo (kliknięcie dowolnego punktu na panelu2, kliknięcie trzeciego przycisku ”H” zakładki2, kliknięcie przycisków aż do odpowiadającego żądanej macierzy współczynników DWT, tu: ”level3”).

B.4. Implementacja 4

„Aplikacja4" jest najbardziej istotną w pracy implementacją, ponieważ umożliwiła przeprowadzenie najistotniejszych eksperymentów związanych z określaniem czasu komunikacji wersji klastrowej (oraz jednostanowiskowej) algorytmu. Implementacja rozrosła się do niemal 5000 linii kodu[259], dlatego bezcelowe byłoby cytowanie jego większych fragmentów.

Implementacja nie zawiera również interaktywnego graficznego interfejsu użytkownika – polecenia wydawane są poprzez okno konsoli, dodatkowe okno graficzne pokazuje obraz w jednym z dostępnych trybów (omówionych poniżej). (Rys. B.10)

Wejście

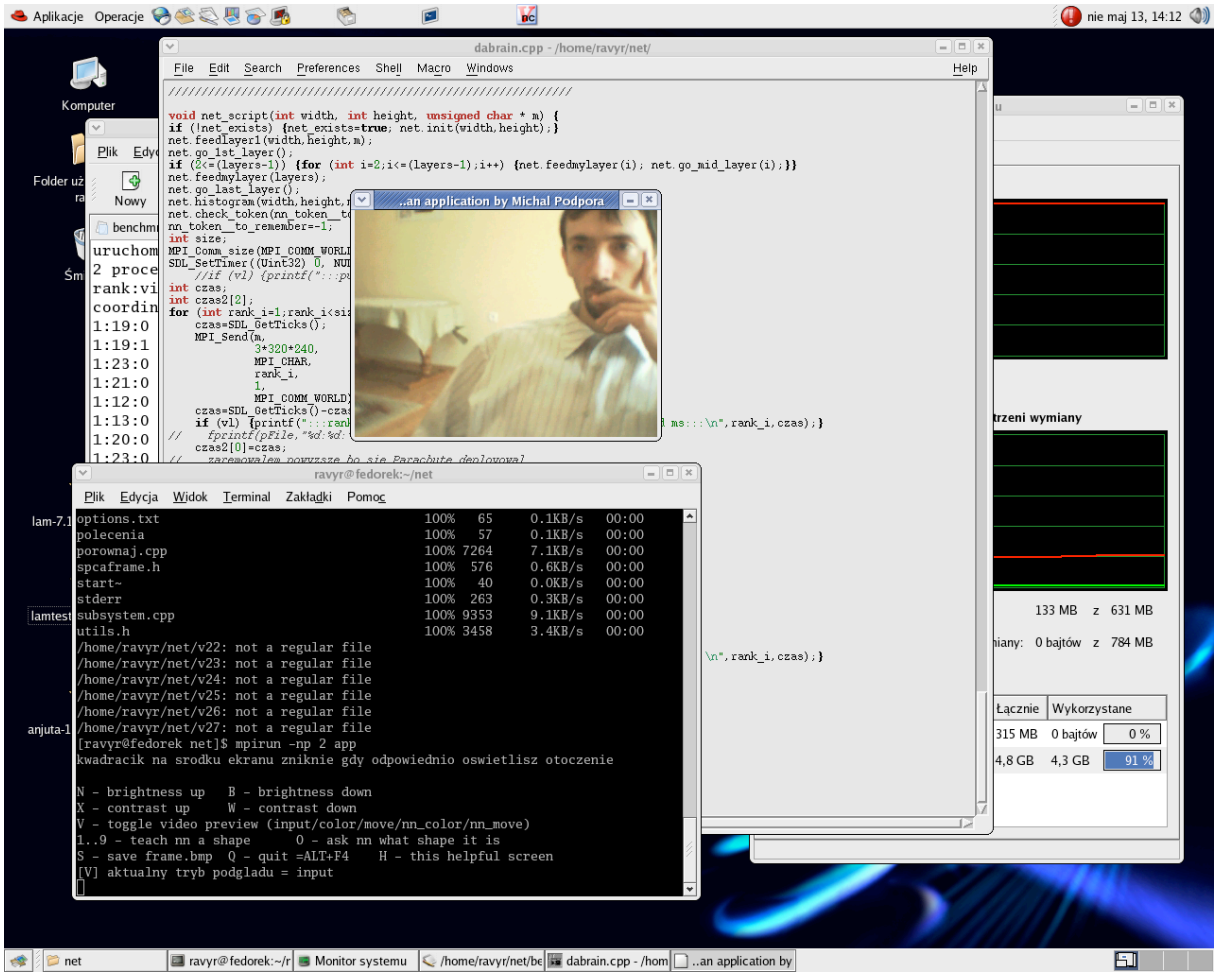
Wejściem w znaczeniu wejścia systemu wizyjnego jest strumieniowe urządzenie przechwytywania wizji, czyli kamera internetowa. Dostęp do kolejnych klatek obrazu realizowany jest mechanizmem „*frame grabber*". Klatki wejściowe pierwszej z procedur wchodzących w skład opracowanego algorytmu mają format RGB 320x240 pikseli, 24 bpp.

Wejściem w znaczeniu sposobu ingerowania w działanie algorytmu jest klawiatura i okno konsoli pozostającej po uruchomieniu aplikacji w terminalu. Po samym uruchomieniu wyświetlana jest krótka instrukcja dotycząca podstawowej funkcjonalności:

```
printf ("N - brightness up   B - brightness down\n");
printf ("X - contrast up     W - contrast down\n");
printf ("V - toggle video preview (input/color/move/nn_color/nn_move)\n");//1/2/3/4
printf ("1..9 - teach nn a shape      0 - ask nn what shape it is\n");
printf ("S - save frame.bmp  Q - quit =ALT+F4    H - this helpful screen\n");
```

Niezależnie od akcji użytkownika (lub braku akcji ze strony użytkownika) uruchamiany jest również podgląd widoku kamery w trybie obrazu wejściowego oraz inicjowany jest skrypt automatycznej korekcji kontrastu i jasności, przeprowadzany w oparciu o histogramy wyliczone z obrazu wejściowego. Po zakończeniu działania tego skryptu (zwykle trwa to około 5-25 klatek) aplikacja przechodzi do normalnego trybu uruchomieniowego.

[259] Nie licząc dołączanych bibliotek SDL, MPI oraz umożliwiających wykorzystanie urządzenia *frame grabber*.



Rys. B.10. Aplikacja4 – okno kodu, okno terminala/konsoli, okno podglądu – po uruchomieniu.

Przetwarzanie

W normalnym trybie uruchomieniowym możliwe są dwa sposoby działania aplikacji – zależnie od tego czy aplikację uruchomiono w środowisku klastrowym czy nie. Ponieważ w przypadku biblioteki LAM/MPI decyzja ta ma wpływ na wybór kompilatora, program jest skompilowany z użyciem kompilatora obsługującego polecenia MPI, przerzucając moment decyzji na samo uruchomienie aplikacji. Aplikację można uruchomić na klastrze lub na jedno-węzłowym klastrze[260] – w tym drugim przypadku biblioteka MPI jest również ładowana i inicjowana, ale algorytm nie używa klastra, lecz transmisję wykonuje do innego wątka w tym samym komputerze/systemie.

Czas przetwarzania poszczególnych funkcji (wraz z czasem komunikacji ale i czasem systemu operacyjnego) mierzony jest za pomocą polecenia z biblioteki SDL:

[260] Zmiana liczby węzłów nie stanowi komplikacji w posługiwaniu się aplikacją, gdyż zgodnie z zasadami uruchamiania programów pisanych dla LAM/MPI, liczbę aktywowanych węzłów można narzucić sztucznie podczas uruchamiania.

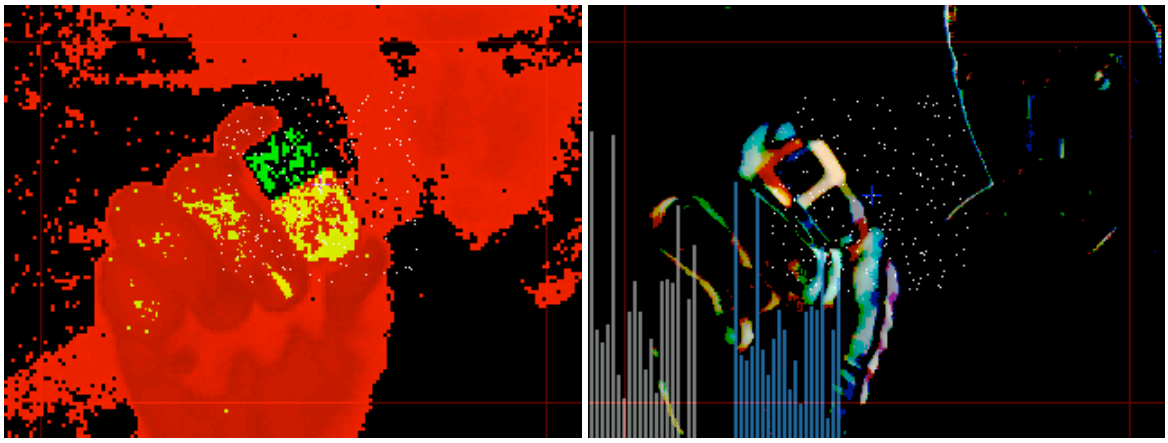
```
time_ = SDL_GetTicks();
```

Fragmenty kodu odpowiedzialne za mierzenie czasu i wyświetlanie wyniku do konsoli[261] umieszczane są w odpowiednich (możliwie najwęższych) miejscach kodu osobno dla każdego analizowanego fragmentu.

Podczas uruchomienia programu dostępne są następujące tryby:

```
if (videomode==1) printf ("[V] aktualny tryb podgladu = input\n");
if (videomode==2) printf ("[V] aktualny tryb podgladu = color\n");
if (videomode==3) printf ("[V] aktualny tryb podgladu = move\n");
if (videomode==4) printf ("[V] aktualny tryb podgladu = nn@color\n");
if (videomode==5) printf ("[V] aktualny tryb podgladu = nn@move\n");
if (videomode==6) printf ("[V] aktualny tryb podgladu = yellowspot\n");
```

Wymienione tryby dotyczą okna graficznego SDL podglądu obrazu, a przy pomocy klawisza „v” możliwe jest zmienianie widzianej reprezentacji obrazu. W każdym przypadku przeprowadzany jest pełny cykl wszystkich procedur, przeprowadzanych sekwencyjnie, cyklicznie i synchronicznie (dla asynchronicznie wybieranej[262] klatki obrazu).



Rys. B.11. Aplikacja4 – zrzut okna podglądu w trybie 2 (po lewej) i 5 (po prawej). Białe kropki symbolizują przestrzenne rozmieszczenie wejść sieci neuronowej. Siwe i niebieskie paski oznaczają aktywność neuronów odpowiednich warstw. Po lewej – obraz wynikowy węzła klasyfikującego kolor dominujący, po prawej – obraz wynikowy węzła tworzącego warstwy różnicowe (wykrywającego ruch).

Wyjście

Wyjściem programu jest:

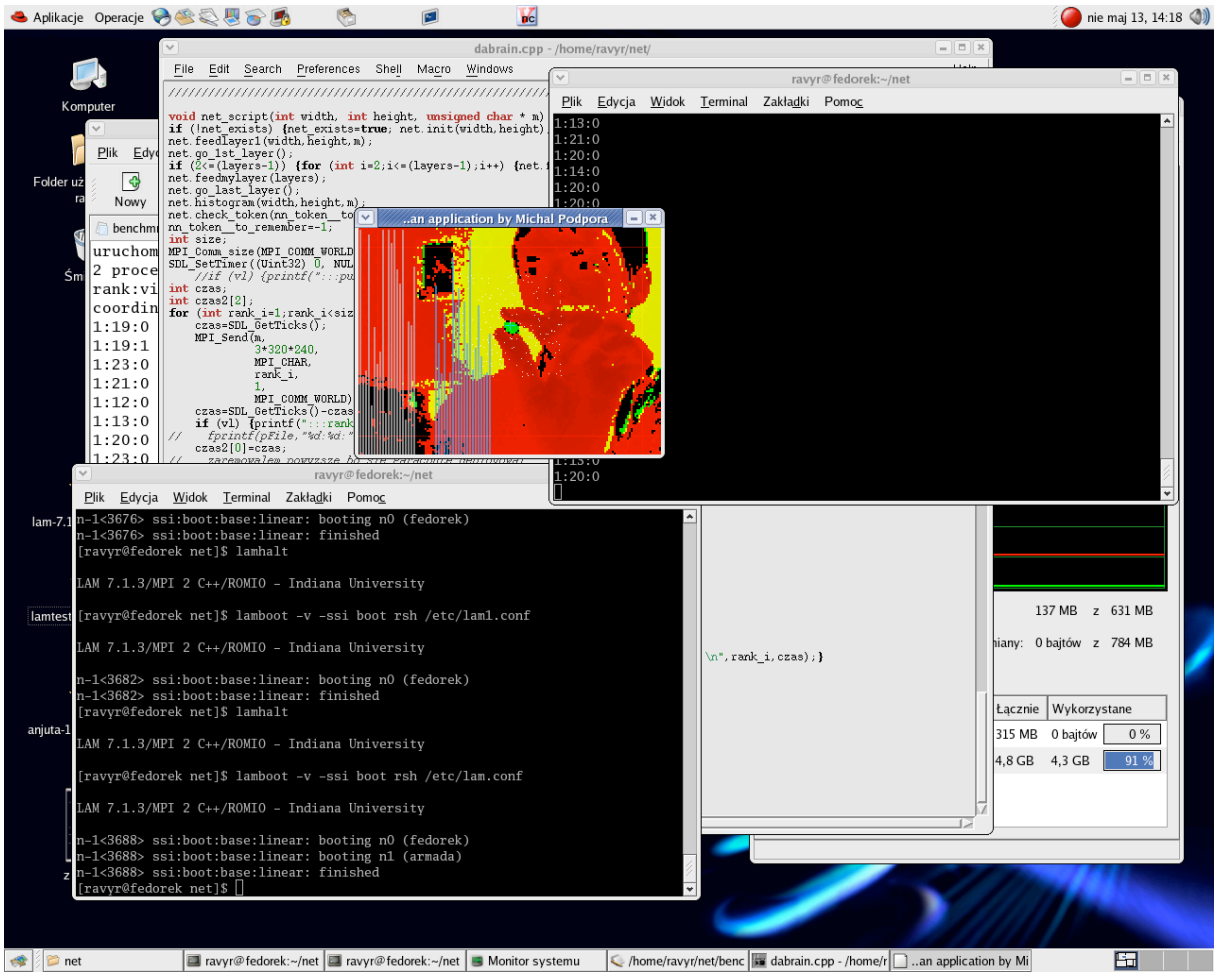
1) okno podglądu,
2) konsola zawierająca informację zwrotną o działaniu systemu,

[261] W formacie ułatwiającym dalsze przetwarzanie danych – nagłówek oraz numer pomiaru, czas przed, czas po, czas trwania (różnica), rozdzielane znakiem tabulatora. Taka forma zapisu ułatwia analizę danych w dowolnym arkuszu kalkulacyjnym lub środowisku obliczeniowym.

[262] Gdy ukończony zostaje pełny cykl przetwarzania i przesyłania jednej klatki obrazu, brana jest nowa klatka (co nie znaczy „kolejna”, ponieważ czas przetwarzania jest dłuższy niż czas oczekiwania na kolejną ramkę).

3) konsola zawierająca wyniki pomiarów czasu wykonania określonego fragmentu kodu (jeżeli przeprowadzany jest jakiś pomiar czasu),

4) pliki *.bmp (na żądanie) zawierające zrzuty obrazu widocznego w oknie podglądu.



Rys. B.12. Aplikacja4 – w czasie działania, w lewym dolnym rogu okno terminala, w którym uruchomiono klaster (*lamboot*), w prawym górnym rogu w tle okno konsoli aplikacji wyświetlające wartości przynależności, będące próbą[263] działania sieci neuronowej.

„Aplikację4" cechował stosunkowo wydajny kod[264], co owocowało dość szybkim działaniem aplikacji. Na dołączonej płycie CD oraz w Internecie [115] umieszczono film będący zrzutem jednej z wczesnych wersji aplikacji (jeszcze przed zaimplementowaniem algorytmów DWT), na którym widać podczas przełączania trybów podglądu, że klatki przetwarzane są na tyle szybko, iż obraz w oknie podglądu sprawia wrażenie płynnego filmu.

[263] Będące nieudaną próbą działania sieci neuronowej – na ówczesnym etapie działania aplikacji (na zdjęciu wersja v27 z maja 2007) sieć była zbyt duża, wejść miała zbyt mało, a przede wszystkim wejścia były wartościami pojedynczych pikseli (na rysunku Rys. B.11 oznaczone białymi kropkami) co wprowadzało dodatkową losowość pochodzącą z zakłóceń klasyfikacji

[264] w porównaniu do kolejnej implementacji, „Aplikacji5"

B.5. Implementacja 5

Celem „Aplikacji5" była gruntowna weryfikacja wyników otrzymywanych w drodze wcześniej opisanych eksperymentów poprzez ponowne ich zaimplementowanie w zupełnie odmiennym środowisku. Dodatkową wartością „Aplikacji5" jest jej uniwersalność – podczas gdy uruchomienie „Aplikacji4" i odtworzenie eksperymentu wymaga doinstalowywania bibliotek[265], „Aplikacja5" zadziała bez modyfikacji na praktycznie każdym komputerze Macintosh. Nie bez znaczenia dla podjęcia decyzji o implementacji były również graficzne (wizualne) środowisko programistyczne oraz względnie prosta konfiguracja klastra[266].

Niestety nie wszystkie zalety „Aplikacji4" udało się pomyślnie migrować do „Aplikacji5". Głównym ograniczeniem był brak biegłości autora w programowaniu w dość egzotycznym[267] jak na polskie warunki języku Cocoa.

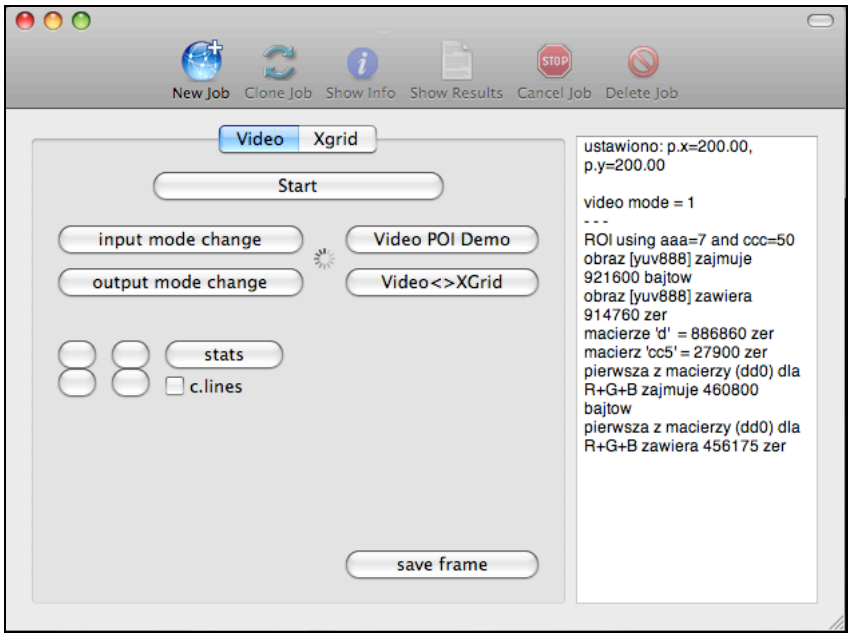
Wejście

Wejściem „Aplikacji5" jest wbudowana kamera iSight lub dowolna inna kamera wybrana jako domyślna dla systemu operacyjnego. Parametry algorytmu można modyfikować poprzez prosty i przejrzysty interfejs przedstawiony na rysunku Rys. 4.24 oraz na poniższych rysunkach.

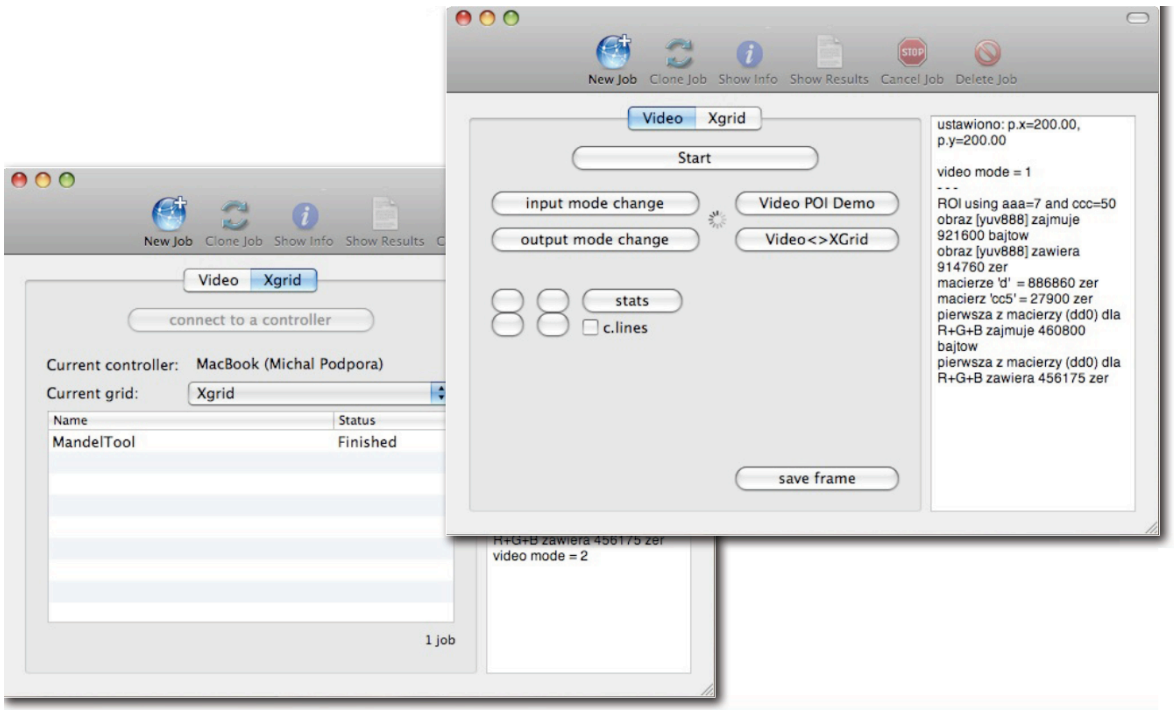
[265] a w skrajnym przypadku może nawet wymagać edycji kodu sterowników

[266] Wystarczy zaznaczyć jedno pole w preferencjach systemu operacyjnego, aby dany komputer współtworzył lokalny klaster XGrid.

[267] Język programowania Cocoa należy do grupy najbardziej popularnych (a więc i najlepiej wspieranych) języków dla systemu MacOsX. Niestety komputery firmy Apple nie są na razie w Polsce ta popularne jak za granicą, tym bardziej trudno o znalezienie fachowej pomocy przy programowaniu.



Rys. B.13. Aplikacja5 – okno podstawowe ustawień, zakładka „Video”.



Rys. B.14. Aplikacja5 – okno podstawowe ustawień, po lewej zakładka „XGrid” po połączeniu z kontrolerem klastra.

Warto wspomnieć, że w „Aplikacji5” jako nominalną rozdzielczość pracy kamer i systemu wizyjnego przyjęto 640x480 pikseli. Wbrew oczekiwaniom nie spowodowało to znaczącego spadku wydajności aplikacji.

Przetwarzanie

W chwili uruchomienia aplikacji pojawia się okno ustawień (okno interfejsu) przedstawione na rysunku Rys. B.14. Typową czynnością jest kliknięcie przycisku start powodującego start procedur obsługi wejścia i wyjścia systemu wizyjnego.

Pojawiają się wówczas dwa dodatkowe okna:

- okno podglądu wejścia (odświeżane sprzętowo, zgodnie z częstotliwością pracy kamery),
- okno podglądu wyjścia lub dowolnego etapu pośredniego (odświeżane synchronicznie z algorytmem cyklicznie przetwarzającym pobierane klatki).

Okno ustawień otwarte jest na zakładce „Video", przechwytywanie klatek jest uruchomione, okno wejścia pokazuje obraz z kamery, algorytm przetwarzania jest zatrzymany, okno wyjścia ma kolor ciemnozielony.

Możliwe jest wypróbowanie procedur i algorytmów bez łączenia z klastrem – w takim przypadku wystarczy nacisnąć przycisk „Video POI demo". Aktywowany w ten sposób algorytm będzie lokalnie generował obraz wyjściowy. Przyciski po lewej stronie umożliwiają podgląd obrazu na poszczególnych etapach przetwarzania.

Aby skorzystać z klastra, należy w dowolnym momencie przejść do zakładki „XGrid" i tam kliknąć przycisk „connect to a controller". Po wybraniu kontrolera spośród automatycznie wyszukanych w sieci lokalnej kontrolerów klastrów może pojawić się monit o autentykację. W zależności od zastosowanego w danym kontrolerze sposobu, można wyróżnić trzy przypadki:

- autoryzacja przez Kerberos
- autoryzacja za pomocą hasła
- autoryzacja wyłączona

Po połączeniu z kontrolerem klastra, w dolnej części wyświetlane są aktualne zadania jakie realizuje klaster, ich status oraz postęp. Może się zdarzyć, że ta lista będzie pusta, oznacza to, że klaster nie jest zajęty żadnym zadaniem.

Po powrocie do zakładki „Video" (i uruchomieniu przechwytywania jeżeli jeszcze nie zostało uruchomione) można kliknąć przycisk „Video-2-Xgrid" przełączający tryb przetwarzania na zdalny – w węzłach klastra.

Przy pomocy małych nieoznakowanych przycisków można sztucznie zmieniać parametry funkcji definiującej *threshold*.

Wyjście

W „Aplikacji5” wyjścia można rozumieć dwojako:

- jako okno podglądu zawierające obraz będący efektem (końcowym lub cząstkowym) algorytmu przetwarzania,
- jako prawa część okna ustawień, zawierająca panel tekstowy pokazujący wszystkie informacje zwrotne i komunikaty informacyjne pochodzące z aplikacji.

B.6. Weryfikacja eksperymentalna przeprowadzonej analizy czasowej

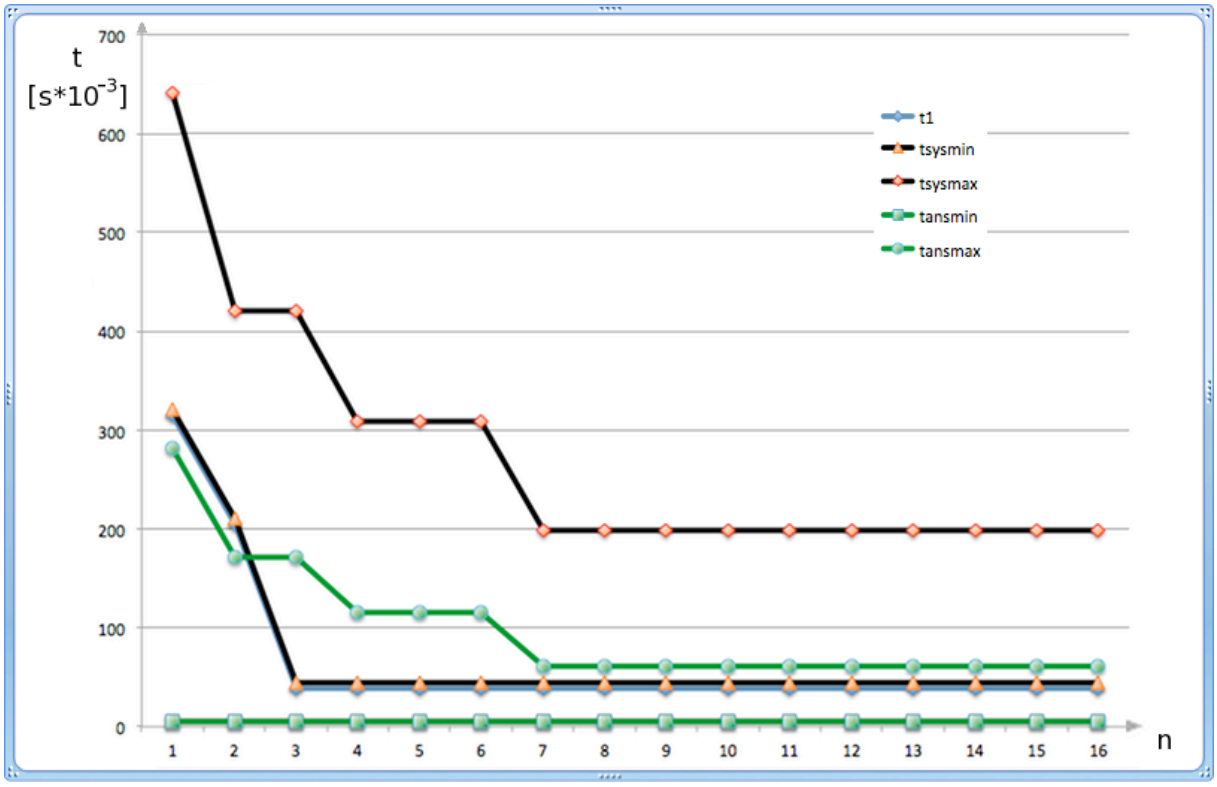
W niniejszym dodatku zawarto wyniki pomiarów oraz wyliczeń pozwalających na wykonanie analiz i kalkulacji zgodnie z równaniami zawartymi w rozdziale 4.3 pracy, prowadzącymi do wykresów z rozdziału 4.5.

Tabela B.1. Zmierzony czas rzeczywisty wykonywania poszczególnych etapów systemu wizyjnego.

parametr	czas [$s \cdot 10^{-3}$]
t_{akw} =	38,7126
t_{trans1} =	5,1524
t_{bezkon} =	43,8651
t_{dwt} =	198,4862
t_{trans2} =	5,1524
t_{idwt} =	60,5305
t_{kont} =	0
t_{wnios} =	0
t_{ster} =	0

Tabela B.2. Wyliczony czas wykonywania poszczególnych połączonych etapów systemu wizyjnego. Czas wyliczony przy pomocy wykonanej aplikacji obliczeniowej, na podstawie serii ponad dwustu pomiarów każdego z czasów etapów składowych.

n	Czy osobny wątek dla komunikacji?	t_1 [s·10^{-3}]	t_2 [s·10^{-3}]	t_{sysmin} [s·10^{-3}]	t_{sysmax} [s·10^{-3}]	t_{ansmin} [s·10^{-3}]	t_{ansmax} [s·10^{-3}]
1	Nie	315,6031	5,1524	320,7556	641,511	5,1524	282,0429
2	Nie	204,8469	5,1524	209,9994	419,9986	5,1524	171,2867
3	Tak	38,7126	5,1524	43,8651	419,9986	5,1524	171,2867
4	Tak	38,7126	5,1524	43,8651	309,2424	5,1524	115,9086
5	Tak	38,7126	5,1524	43,8651	309,2424	5,1524	115,9086
6	Tak	38,7126	5,1524	43,8651	309,2424	5,1524	115,9086
7	Tak	38,7126	5,1524	43,8651	198,4862	5,1524	60,5305
8	Tak	38,7126	5,1524	43,8651	198,4862	5,1524	60,5305
9	Tak	38,7126	5,1524	43,8651	198,4862	5,1524	60,5305
10	Tak	38,7126	5,1524	43,8651	198,4862	5,1524	60,5305
11	Tak	38,7126	5,1524	43,8651	198,4862	5,1524	60,5305
12	Tak	38,7126	5,1524	43,8651	198,4862	5,1524	60,5305
13	Tak	38,7126	5,1524	43,8651	198,4862	5,1524	60,5305
14	Tak	38,7126	5,1524	43,8651	198,4862	5,1524	60,5305
15	Tak	38,7126	5,1524	43,8651	198,4862	5,1524	60,5305
16	Tak	38,7126	5,1524	43,8651	198,4862	5,1524	60,5305



Rys. B.15. Wartości z tabeli powyżej (Tabela B.2), ujęte na wspólnym wykresie. Szczegółowo omówione w rozdziale 4.5.

Dodatek C. Zawartość płyty CD

- aplikacje
  - Aplikacja1
    - plik uruchamialny `kamera.exe`
    - kompletny projekt Borland C++ Builder 6 z kodem źródłowym
    - wszelkie wymagane dodatkowe biblioteki (`winio.vxd`, `viewport.h`, ..)
    - dokument `patterns.doc` zawierający plansze do prezentowania przed kamerą systemu wizyjnego
  - Aplikacja2
    - plik uruchamialny `main`
    - kod źródłowy (`main.cpp`, `subsystem.cpp`) wraz z archiwum wersji
    - repozytoria z wszelkimi wymaganymi dodatkowymi bibliotekami:
      - spca5xx-20060501.tar.gz
      - spcaview-20051212.tar.gz
      - SDL-1.2.11-1.i386.rpm
      - SDL-devel-1.2.11-1.i386.rpm
      - lam-7.1.3-2.i586.rpm
    - zrzuty ekranu aplikacji podczas uruchomienia
  - Aplikacja3
    - plik uruchamialny `project1.exe`
    - kompletny projekt Borland C++ Builder 6 z kodem źródłowym
    - przykładowe wynikowe pliki graficzne
  - Aplikacja4
    - plik uruchamialny `app`
    - kod źródłowy (`main.cpp`, `subsystem.cpp`, `porownaj.cpp`, `dabrain.cpp`) wraz z archiwum wersji
    - repozytoria z wszelkimi wymaganymi dodatkowymi bibliotekami:
      - spca5xx-20060501.tar.gz

        - spcaview-20051212.tar.gz
        - SDL-1.2.11-1.i386.rpm
        - SDL-devel-1.2.11-1.i386.rpm
        - lam-7.1.3-2.i586.rpm
    - zrzuty ekranu aplikacji podczas uruchomienia
    - zrzut ekranu w postaci filmu (plik `MPodpora_avi.avi`) prezentującego nieimplementowany tracking
    - dokument `MPodpora_pdf.pdf` zawierający prezentację opisującą tą aplikację
- Aplikacja5
    - kompletny uruchamialny projekt XCode w języku Cocoa z kodem źródłowym oraz archiwami wersji (plik początkowy projektu to `\aplikacja5\Video-2-Xgrid17\Sample\MyVideo2xgrid.m`)
    - zrzuty ekranu aplikacji podczas uruchomienia
    - dokument `MPodpora_100112_seminariumEFS.pdf` zawierający prezentację opisującą tą aplikację

➢ wykresy
- pliki źródłowe `*.gcx` wzorów dla oprogramowania (Grapher dla MacOsX) przy pomocy którego wygenerowano wykresy funkcji, m.in. Rys. 3.6 i Rys. 3.7
- zrzuty ekranu
- wyeksportowane pliki graficzne i animacje

➢ wybrane źródła internetowe w wersji off-line

➢ tekst rozprawy w pliku `pdf`