KOMPUTEROWE PRZETWARZANIE WIEDZY

Kolekcja prac 2014/2015
pod redakcją Tomasza Kubika

KOMPUTEROWE PRZETWARZANIE WIEDZY

Kolekcja prac 2014/2015
pod redakcją Tomasza Kubika

Skład komputerowy, projekt okładki
Tomasz Kubik

ISBN 978-83-930823-7-7

Wydawca
Tomasz Kubik

Druk i oprawa
I-BiS sc., ul. Sztabowa 32, 50-984 Wrocław

Skład niniejszej książki wykonano w systemie LaTeX

SPIS TREŚCI

SŁOWO WSTĘPNE

Niniejszy tom jest kolejną pozycją w cyklu prezentacji prac z dziedziny komputerowego przetwarzania wiedzy, zredagowaną przez dra inż. Tomasza Kubika. Autorami prac są studenci Wydziału Elektroniki Politechniki Wrocławskiej, studiujący w semestrze letnim roku akademickiego 2014/2015 na studiach II stopnia, na specjalności Robotyka kierunku Automatyka i robotyka. Książka obejmuje 107 stron druku i składa się z dziewięciu rozdziałów, których celem jest przedstawienie wybranych metod i narzędzi do przetwarzania wiedzy oraz ich zastosowań do rozwiązania konkretnych zadań praktycznych. Autorzy i tytuły rozdziałów są następujące:

1. Kamil Mówiński: *Semantyka rozmyta*
2. Zuzanna Różycka, Agata Leś: *Obsługa strumieni pracy*
3. Marcin Ciopcia, Daniel Gut: *Wersjonowanie i agregacja w systemach rozproszonych*
4. Hanna Sienkiewicz: *Generowanie ofert biznesowych w systemie rekomendacyjnym*
5. Kacper Nowosad, Mateusz Stachowski: *Interfejsy wyszukiwania*
6. Alicja Jurasik, Błażej Kowalczyk, Paweł Urbaniak: *Problemy sterowania w zamkniętej pętli z systemem wizyjnym*
7. Jacek Drewniak: *Planowanie ruchu przy ograniczonych zasobach*
8. Filip Grzeszczak, Marek Ulita: *Rozpoznawanie obiektów 3-wymiarowych w chmurze punktów*
9. Michał Drwięga: *Określanie lokalizacji źródła dźwięku z wielopunktowego nasłuchu*

W celu przybliżenia Czytelnikowi treści tomu scharakteryzujemy pokrótce rozdziały nr 3, 6 i 9.

- *Wersjonowanie i agregacja w systemach rozproszonych*: W rozdziale przedstawiono koncepcję systemu wersjonowania i archiwizacji stron www, pozwalającego na ich monitorowanie, rejestrację zmian i aktualizację. Szczególną uwagę poświęcono architekturze i bezpieczeństwu systemu. W celu zapewnienia stabilności, elastyczności i bezpieczeństwa systemu zastosowano gotowe narzędzia dostępne w systemie operacyjnym Linux. Zaprojektowano system złożony

z modułu archiwizacji stron www, generacji funkcji skrótu, zarządcy bazy danych, modułu wyzwalacza, modułu synchronizacji i modułu interakcji z użytkownikiem. Przedstawiono opis działania systemu z uwzględnieniem jego instalacji, obsługi i konfiguracji.

- *Problemy sterowania w zamkniętej pętli z systemem wizyjnym*: Rozdział przedstawia koncepcję i projekt systemu wizyjnego przeznaczonego do określania położenia na scenie drużyny robotów biorących udział w konkurencji zwanej *PuckCollect* zawodów *RobotChallenge*, oraz zbieranych krążków. Przedmiotem projektu są roboty biorące udział w zawodach, system wizyjny, system rozpoznawania sceny, protokół transmisji danych, a także algorytm sterowania robotów uwzględniający model ich kinematyki i pozwalający na unikanie przeszkód. Zaprojektowany system przeszedł pomyślnie testy.
- *Określanie lokalizacji źródła dźwięku z wielopunktowego nasłuchu*: Rozdział został poświęcony zagadnieniu lokalizacji źródła dźwięku dla potrzeb sterowania robotem. Po omówieniu metod lokalizacji przedstawiono w nim opis konstrukcji i architektury układu sterowania czterokołowego robota mobilnego. Robot został wyposażony w sensor *Kinect* umożliwiający lokalizację źródła dźwięku i nawigację w kierunku źródła. Układy detekcji dźwięku i lokalizacji jego źródła zaimplementowano w postaci komponentów systemu ROS. Przeprowadzono testy działania robota.

Oddawany do druku zbiór prac jest adresowany do Czytelników zainteresowanych praktycznymi aspektami komputerowego przetwarzania wiedzy.

Prof. Krzysztof Tchoń,
opiekun specjalności Robotyka,
Wrocław, luty 2016

ROZDZIAŁ 1

SEMANTYKA ROZMYTA

K. Mówiński

W niniejszym rozdziale omówiono zagadnienia związane z budową wyszukiwarki opartej na indekserze Solr oraz bazie wiedzy DBpedia (`http://dbpedia.org/sparql`). Istota zastosowanego w nim algorytmu wyszukiwania polega na generowaniu wyników z wykorzystaniem nie tylko słów kluczowych podanych przez użytkownika, ale również niewymienionych jawnie słów dodatkowych o podobnym znaczeniu. Wyszukiwarka ma być odporna na proste błędy popełniane przez użytkowników wynikające z nieznajomości zasad ortografii lub z pomyłek typograficznych.

1.1. Wprowadzenie

W ostatnich latach z powodu udostępniania coraz to nowych i obszerniejszych źródeł informacji koncepcja budowy analitycznych aplikacji uległa zmianie. Dane są publikowane w formie strukturalnej bądź niestrukturalnej, przyjmują postać obrazów, plików wideo czy dźwięków. Ze względu na tak wielką różnorodność coraz trudniej jest ująć wszystkie udostępniane zasoby w strukturalne ramy baz czy hurtowni danych. Coraz trudniej też jest rozpoznawać występujące między nimi asocjacje.

Semantyczne modelowanie wiedzy opiera się na wykorzystaniu danych kontekstowych i ich relacji z analizowanym zagadnieniem. Polega na opisywaniu co dane znaczą i do czego pasują. Pozwala poznać fakty dotyczące sprawców, obiektów, motywów, umiejscowienia w czasie i przestrzeni, parametrów ilościowych. Zastosowanie tego modelu pozwala łączyć całkowicie różne pod względem typu dane i informacje oraz umożliwia generowanie odpowiedzi na podstawowe pytania: Kto? Co? Dlaczego? Kiedy? Gdzie? Jak? Ile? (ang. *Who*? *What*? *Why*? *When*? *Where*? *How*? *How Much*?).

Jednym z bardziej znanych narzędzi stosowanych przy analizach angielskich tekstów jest *wordnet* [1, 2, 1]. Jego polskim odpowiednikiem jest *Słowosieć*. Narzędzie to pozwala na wyszukiwanie *synsetów* – zbioru wyrazów o tym samym znaczeniu, np. butelka, butla i flaszka. Termin *synset* pochodzi od angielskiego *synonim set* (zbiór synomimów).

Budowa słowosieci opiera się na założeniu, że język można zamodelować jako zbiór pojęć powiązanych ze sobą relacjami lingwistycznymi. W powstałej sieci każdy wyraz zdefiniowany jest poprzez relację do innego słowa. Dla przykładu przy wyszukaniu *synsetów* dla wyrażenia „*cat*" otrzymuje się następujący wynik:

- *cat.n.01*, zwierzę domowe,
- *guy.n.01*, potoczna nazwa na młodego, niedoświadczonego mężczyznę,
- *cat.n.03*, plotkująca, złośliwa kobieta,
- *big_cat.n.01*, duże koty, żyjące na dzikich terenach (lwy, tygrysy itp.).

Jak widać *wordnet* zwraca oprócz zbiorów synonimów także ich opisy. Kolejność wyrażeń wskazuje, jak mocno dane pojęcie jest związane z podanym, zaś liczby pojawiające się na końcach rekordów pozwalają odróżnić kolejne ich znaczenia.

Zwiększanie zakresu wyszukiwania poza słowa kluczowe zdefiniowane przez użytkownika nie jest pomysłem nowym [3]. W praktyce do semantycznego wyszukiwania często używa się tezaurusów – zarządzanych słowników synonimów, zorganizowanych w układzie pojęciowym. Formalnie tezaurus definiowany jest jako „słownik odzwierciedlający strukturę pola semantycznego danego języka informacyjno-wyszukiwawczego, obejmujący deskryptory, ich relacje oraz reguły stosowania". Najczęściej tezaurusy dotyczą wąskiej grupy dziedzinowej, jak np. EuroVoc (`http://eurovoc.europa.eu/`) czy też tematycznej, jak np. „tesaurus Harry'ego Pottera" Andrzeja Polkowskiego i Joanny Lipińskiej.

1.1.1. Wordnet

Wordnet to wielka baza leksykalna, zawierająca czasowniki, rzeczowniki, przymiotniki oraz przysłówki angielskie, pogrupowane w *synsety*. Zbiory te zawierają także relacje lingwistyczne do innych *synsetów*. Wordnet pozornie wygląda więc jak *thesaurus*, jednak różnica tkwi w sposobie wyszukiwania, który jest oparty na znaczeniach danego wyrażenia. *Wordnet* składa się z około 117 000 połączonych ze sobą synsetów. Dodatkowo każdy rekord posiada własny opis oraz przykład użycia danego wyrażenia.

Najczęściej wykorzystywaną relacją są tzw. hypernonimy, które łączą obiekty o szerszym znaczeniu z bardziej specyficznymi (jak w przykładzie łóżko – łóżko piętrowe). Relacja ta jest przechodnia, tak więc jeśli fotel jest rodzajem krzesła, a krzesło jest meblem, to fotel także jest meblem. Kolejna używaną relacją są meronimy, tzn. wyrażenia, które są składnikiem szukanego. Najprostszym przykładem meronimu jest: krzesło – noga od krzesła, auto – silnik samochodowy. Relacja ta nie „idzie w górę", ponieważ może być charakterystyczne tylko dla danej grupy, tak więc każde krzesło ma nogi, ale nie każdy mebel musi je mieć.

1.1.2. Wykrywanie błędów

Podczas wyszukiwania informacji użytkownicy często popełniają błędy redagując własne zapytania. Są to najczęściej błędy wynikające z pominięcia, przestawienia lub dodania nadmiarowych znaków. Do rozpoznawania i korekty pomyłek można stosować różne metody [4]. Często proponuje się do tego zastosować sztuczne sieci neuronowe. Podejście to wymaga jednak bardzo dużej ilości pró-

bek dla wielu różnych wyrazów w danym języku. Sieci takie można wytrenować posługując się zarejestrowanymi zestawami zapytań oraz wygenerowanymi odpowiedziami.

Innym sposobem na wykrywanie błędów proponowanym w literaturze jest liczenie odległości edycyjnej wyrazów. Odległość edycyjna mówi, jak bardzo dane słowo różni się od szukanego, przy czym miarą tej różnicy może być odległość Levenshteina.

W poruszanych zagadnieniach bardzo ważnym czynnikiem jest wykorzystywany przez użytkowników układ klawiatury. Pisząc bowiem na komputerze o wiele częściej popełnia się błędy wynikające z wciśnięcia przypadkowego przycisku obok pożądanej litery niż błędy wynikające z nieznajomości ortografii. W niniejszym rozdziale skupiono się na podejściu zaproponowanym w pracy [4], które jest rozwinięciem metody obliczania odległości Levenshteina.

1.1.3. Przechowywanie wyników wyszukiwania

W opisywanym rozwiązaniu do przechowywania wyników wyszukiwania użytkowników użyto *mongodb*. Jest to otwarty, nierelacyjny system bazodanowy o dużej skalowalności. Pozwala na przechowywanie danych w postaci dokumentów serializowanych do postaci json, z możliwością ich indeksowania i tworzenia hierarchii.

Najważniejszą kolekcją, jaką utworzono w trakcie prac, jest `KPW_statistic`. Przechowuje ona wyniki wyszukiwań użytkowników oraz link, który został przez nich wybrany po wyświetleniu wyników. Zabieg ten zastosowano celowo, aby w przyszłości wyszukiwanie odbywało się sprawniej (aby wyniki były lepiej posortowane). Przykład zawartości tej kolekcji przedstawiono na listingu 1.1.

Listing 1.1: Przykłady wpisów w kolekcji `KPW_statistic`

```
> db.KPW_statistic.find();
{"_id" : ObjectId("5526deb4e3bdea3e977758f3"),
 "count" : 1,
 "user_id" : ObjectId("5526cef9e3bdea3811e70977"),
 "url" : "http://dictionary.reference.com/browse/trial",
 "searched" : "Test"}
{"_id" : ObjectId("5526ded6e3bdea3f1ac242b3"),
 "count" : 3,
 "user_id" : ObjectId("5526cef9e3bdea3811e70977"),
 "url" : "http://en.wikipedia.org/wiki/Trial",
 "searched" : "Trial"}
```

1.1.4. DBpedia

DBpedia to baza wiedzy zbudowana w oparciu o dziedzinowe ontologie, zasilana danymi z Wikipedii, pozwalająca na zadawanie zapytań w sposób ustrukturyzowany (na temat budowy ontologii i realizacji wyszukiwaniu informacji z jej wykorzystaniem opublikowano wiele prac, np. [5]). Jej zawartości można przeglądać w formie trójek RDF: podmiot, właściwość i obiekt, wykorzystując przy tym język zapytań SPARQL (ang. *SPARQL Protocol And RDF Query Language*)

Przy formułowaniu zapytań można wykorzystać możliwości języka *SPARQL* [6] i potraktować niektóre wyrazy jako słowa kluczowe. Zostają one pominięte przy wyszukiwaniu, jednak precyzują postać wyników wyszukiwania. Dla *DBpedi* ustanowiono 5 słów kluczowych o następującym znaczeniu:

- `sameas` – najczęściej zwraca URL pojęcia o tym samym znaczeniu jak dane pojęcie (np. wynikiem `sameas` dla wyrazu „cat" może być „Mamma", „Animal", jak również „Kot"),
- `abstract` – wskazuje na opis danego wyrażenia,
- `subject` – wybiera kategorie hasła,
- `label` – zwraca różne tłumaczenia podanego słowa,
- `type` – podobnie jak *subject*, wyszukuje kategorie,
- `external` – zwraca linki do zewnętrznych serwisów, które często występują w przypisach do hasła.

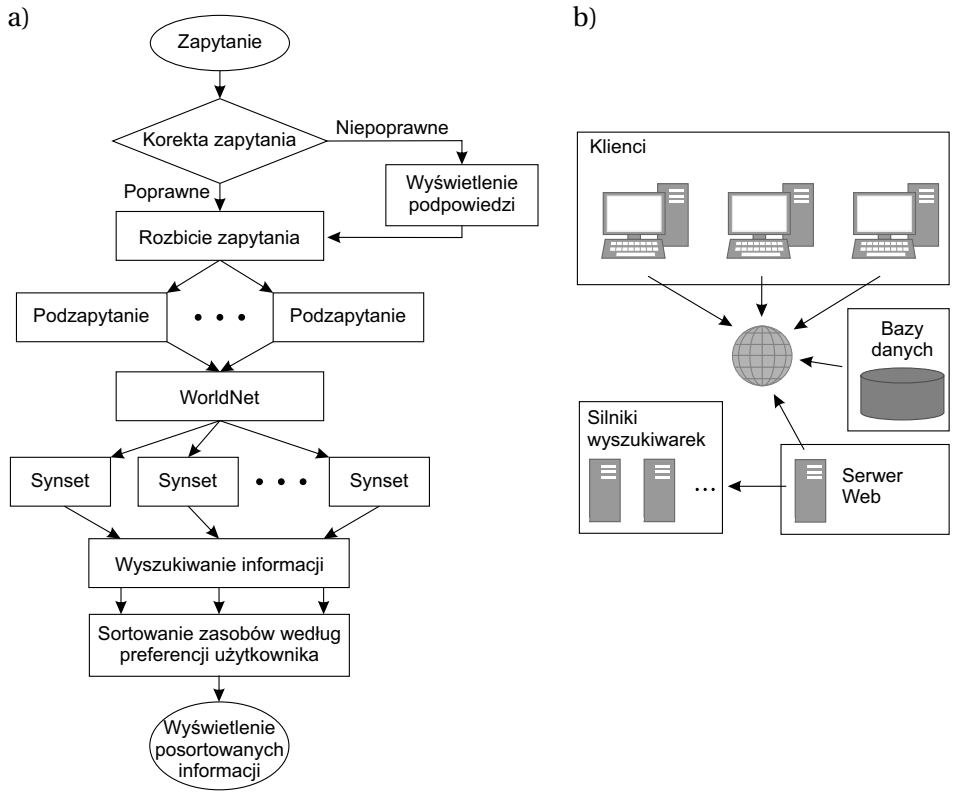
1.2. Zaimplementowane rozwiązanie

Architektura zaproponowanego rozwiązania opiera się na kaskadowym przetwarzaniu zapytania. Pierwszym etapem jest sprawdzenie poprawności frazy i przy ewentualnej pomyłce wyświetlenie proponowanego zapytania. Kolejnym etapem jest podział szukanej frazy na sensowne części, np. przez wykrycie znaków interpunkcyjnych, spójników lub przez rozdział po białych znakach. Uzyskane w ten sposób „podzapytania" są kierowane do silnika obsługującego *słowosieć*. Silnik ten po przetworzeniu zapytania zwraca wyniki w postaci listy *synset'ów*. Następnie dla każdego synonimu generowane jest niejawnie dodatkowe zapytanie. Ostatnim etapem jest posortowanie wyników używając wcześniejszych preferencji i historii wyszukań użytkownika. Przebieg algorytmu pokazano na rysunku 1.1a, zaś architekturę sprzętową zbudowanego rozwiązania na rysunku 1.1b.

1.2.1. Punkty wyszukiwania

Stworzony system wyszukiwania pozwala również na wydobywanie informacji ze innych źródeł. W pliku konfiguracyjnym określa się klasę, która będzie odpowiedzialna za „backend" wyszukiwarki. Klasa ta musi udostępniać metodę `query`, która przyjmuje jako parametr listę słów do wyszukiwania a zwracająca zbiór obiektów klasy `SearchResult`. Jej kod został zaprezentowany na listingu 1.2. Do stworzenia instancji tej klasy konieczne jest podanie następujących parametrów:

- `uri` - adres do zasobu,
- `title` - nazwa pod jaką identyfikuje się dany zasób,
- `content` - wycinek lub całość treści reprezentowanej przez zasób,
- `key_words` - lista słów kluczowych, przez które został zidentyfikowany zasób,
- `probability` - trafność danego zasobu.



Rys. 1.1: Architektura rozwiązania: a) przebieg algorytmu, b) silnik wyszukiwania

Listing 1.2: Klasa zawierająca wyniki wyszukiwania

```
class SearchResult:
    def __init__(self, uri, title,
                 content, key_words, probability=1.0):
        self.uri = uri
        self.title = title
        self.content = content
        self.key_words = key_words
        self.probability = probability

    def __eq__(self, other):
        return self.title == other.title

    def __hash__(self):
        return hash(self.title)
```

1.3. Korekcja błędów

Do poprawnego działania algorytmy wymagana jest lista słów dostępnych w danym języku. W stworzonej aplikacji wykorzystano słownik Scrable do-

stępny pod adresem `https://raw.githubusercontent.com/kamilmowinski/kpw/master/words.txt`. Działanie algorytmu można opisać w następujących krokach:

- przygotuj listę słów jako kolekcję,
- jeśli podane słowo występuje w kolekcji to je zwróć,
- jeśli podane słowo nie występuje, stwórz zbiór najczęściej popełnianych pomyłek (listing 1.3),
- spróbuj dopasować słowo do wcześniej wytworzonego zbioru i zwróć najczęściej powtarzany wynik.

Listing 1.3: Metoda zwracająca zbiór możliwych pomyłek

```
class WordCorrection:
    alphabet = 'abcdefghijklmnopqrstuvwxyz'

    def edits(self, word):
        splits = [(word[:i], word[i:])
                  for i in range(len(word) + 1)]
        deletes = [a + b[1:] for a, b in splits if b]
        transposes = [a + b[1] + b[0] + b[2:]
                      for a, b in splits if len(b) > 1]
        replaces = [a + c + b[1:] for a, b in splits
                    for c in self.alphabet if b]
        inserts = [a + c + b for a, b in splits
                   for c in self.alphabet]
        return set(deletes + transposes + replaces + inserts)
```

Następnie, wykorzystując odpowiednie konstrukcje języka python, sprawdzana jest poprawność danego słowa. Cały kod tej procedury można zapisać w kilku linijkach zaprezentowanych na listingu 1.4.

Metoda `correct` służy do sprawdzania poprawności słowa i w razie błędu zwraca najbardziej podobne. Stworzona klasa jest prosta w użyciu, nie wymaga żadnych dodatkowych czynności poza stworzeniem obiektu oraz wywołanie powyższej metody z parametrem typu `string`.

Listing 1.4: Metoda rozpoznające pomyłki

```
def known_edits(self, word):
    return set(e2 for e1 in self.edits(word)
               for e2 in self.edits(e1) if e2 in self.nwords)

def known(self, words):
    return set(w for w in words if w in self.nwords)

def correct(self, word):
    candidates = (self.known([word])
                  or self.known(self.edits(word))
                  or self.known_edits(word) or [word])
    return max(candidates, key=self.nwords.get)
```

1.4. Silnik wyszukiwarki

Wyszukiwarka opiera się na preferencjach użytkownika, a sortowanie wyników jest w niej uzależnione od wcześniejszych jego wyborów. Prezentację rezultatu wyszukiwania można podzielić na następujące etapy:

1. sprawdzanie poprawności słów i ewentualna propozycja korekcji,
2. generowanie nowych wyrażeń z podanego,
3. wyszukiwanie z podanych wyrazów,
4. sortowanie wyników.

W pierwszym etapie system sprawdza poprawność wyrażenia (pod względem błędów ortograficznych) i proponuje użytkownikowi wyszukanie z poprawionym tekstem. Następnie system dzieli zapytania po białych znakach, odrzuca słowa jedno i dwuliterowe, a następnie generuje nowe wyrażenia na podstawie lingwistycznych relacji. Kod, który realizuje ten etap, zamieszczono na listingu 1.5.

Listing 1.5: Kod generujący dodatkowe wyrażenia

```
def get_another_query(self, q):
    ret = list()
    for query in q.split('␣'):
        if len(query) > 2:
            synsets = wn.synsets(query, pos=wn.NOUN)
            for synset in synsets:
                ret.append(synset.name())
    ret += q.split('␣')
    return ret
```

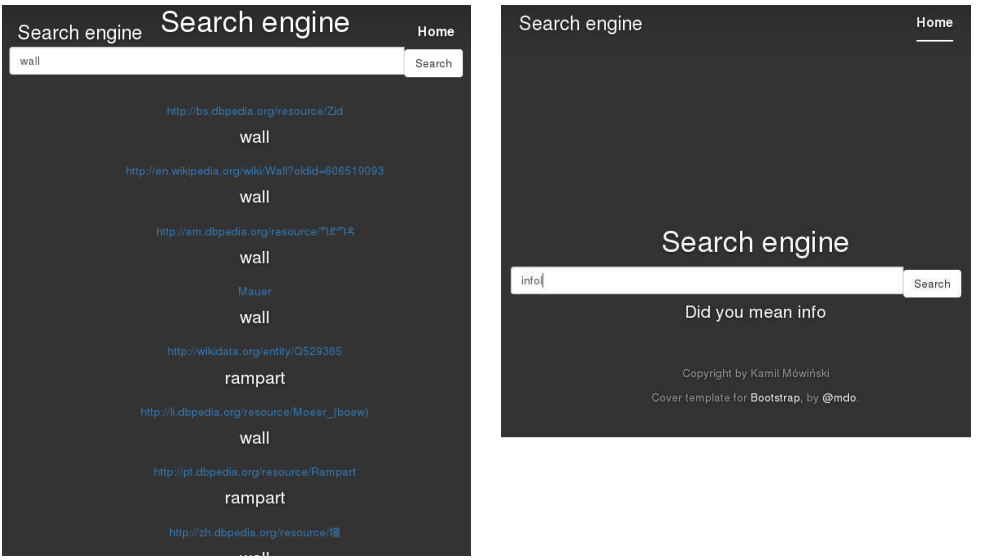
Następnie zlokalizowana zostaje klasa odpowiedzialna za wyszukiwanie informacji i odpalana jest metoda `query` z listą wyrażeń. Wynik wyszukiwania jest przechowywany w liście, która podlega sortowaniu. Jest to ostatni etap generowania rezultatów. Realizuje je metoda zaprezentowana na listingu 1.6

Listing 1.6: Sortowanie wyników wyszukiwania

```
def sort_result(self, results):
    ret = list()
    urls = map(lambda x: x.uri, results)
    stats = Statistic.objects.filter(
        url__in=urls, user=self.request.user).order_by('-count')
    urls = map(lambda x: x.uri, results)
    for stat in stats:
        ret.append(results.pop(urls.index(stat.url)))
    for result in results:
        ret.append(result)
    return ret
```

Zrzuty ekranu z wyglądu działającej aplikacji zostały przedstawione na rysunku 1.2.

a) b)



Rys. 1.2: Zrzut ekranu a) z przykładem wyszukania, b) z podpowiedzią

1.5. Podsumowanie

Celem niniejszej pracy było stworzenie systemu, który pozwalałby na semantyczne wyszukiwanie. W zaproponowanym rozwiązaniu wykorzystano narzędzie *Wordnet* wraz z nowatorskim algorytmem sortowania wyników. Zaimplementowano także algorytm wykrywania prostych pomyłek w zapytaniach i proponujący poprawiony ich tekst. Do zalet stworzonej architektury można zaliczyć wykorzystanie dobrego modelu, jaki dostarcza *Wordnet*. Pozwala to na swobodne wyszukiwanie skojarzonych różnymi relacjami wyrazów i dzięki temu zwiększa zakres słów kluczowych wykorzystywanych w zapytaniach. Wadą systemu jest konieczność komunikacji z wieloma źródłami danych. Aplikacja musi posiadać połączenie z *Wordnetem*, bazą danych, silnikiem *DBpedi* i indekserem *Solr*.

Literatura

[1] Ch. Yang, Sh.-J. Wu. A Wordnet Based Information Retrieval on the Semantic Web. *7th International Conference on Networked Computing and Advanced Information Management (NCM), 2011*, strony 324–328, 2011.

[2] A.K. Barman, J. Sarmah, S.K. Sarma. WordNet Based Information Retrieval System for Assamese. *2013 UKSim 15th International Conference on Computer Modelling and Simulation*, strony 480–484, 2013.

[3] S. Chawla, P. Bedi. Improving information retrieval precision by finding related queries with similar information need using information scent. *Proce-*

edings - 1st International Conference on Emerging Trends in Engineering and Technology, ICETET 2008, strony 486–491, 2008.

[4] A.J. Szanser. Automatic error-correction in natural languages. *Information Storage and Retrieval*, 5(4):169 – 174, 1970.

[5] K. Yoshinaga, T. Terano, N. Zhong. Multi-lingual intelligent information retriever with automated ontology generator. *1999 Third International Conference on Knowledge-Based Intelligent Information Engineering Systems. Proceedings (Cat. No.99TH8410)*, strony 62–65, 1999.

[6] B. Ducharme. *Learning SPARQL*. O'Reilly Media, wydanie 2, 2013.

ROZDZIAŁ 2

OBSŁUGA STRUMIENI PRACY

Z. Różycka, A. Leś

Zachowanie prawidłowego przebiegu procesów oraz sprawnego przepływu informacji to kluczowe zagadnienia we wszystkich niemal dziedzinach działalności człowieka. Formalne podejście do rozwiązywania występujących przy tym problemów jest kojarzone z koncepcją „strumieni pracy" (ang. *workflows*).

W niniejszym rozdziale przedstawiono kilka standardów modelowania przepływów pracy oraz przykłady narzędzi wspierających to modelowanie. Dokonano również prezentacji prostego przypadku użycia notacji BPMN.

2.1. Wprowadzenie

Pomysł na wprowadzenie ogólnych metod wspierających analizę przepływu informacji w organizacji pojawił się w 1970 roku. Właśnie wtedy Skip Ellis, pracownik Xerox PARC, wprowadził w swojej firmie model przepływu pracy bazujący na sieci Petriego. Jednak dopiero w latach dziewięćdziesiątych dwudziestego wieku idea wdrażania modeli przepływu pracy w firmach zaczęła rozwijać się na poważnie. Było to możliwe dzięki postępującej powszechnej komputeryzacji i skutkowało powstaniem standardów zapisu w modelowaniu biznesowym.

Modelując strumienie pracy opisuje się przebieg zadań wykonywanych przez różne grupy ludzi lub aplikacji (a mówiąc ogólniej: przez różnych *agentów* działających na różnych szczeblach zarządzania) w celu spełnienia postawionych założeń [1]. Opis ten powinien być spójny oraz zgodny z rzeczywistością. Dlatego przystępując do modelowania należy poprawnie zidentyfikować procesy biznesowe, a także zachodzące w nich związki.

Do projektowania i monitorowania procesów biznesowych opracowano wiele narzędzi informatycznych: aplikacji (wspierających użytkownika podczas prac projektowych) oraz notacji (potrzebnych do uściślenia języka używanego przy projektowania) [2]. Wśród notacji istotną rolę odgrywa standard BPMN (ang. *Business Process Model and Notation*). Umożliwia on opisywanie procesów biznesowych w sposób graficzny, zrozumiały dla człowieka. Przed jego powstaniem nie było żadnego, powszechnie przyjętego standardu do opisywania związków zachodzących nie tylko wewnątrz jednostki organizacyjnej, ale także pomiędzy róż-

nymi agentami. Zaletą BPMN jest uniwersalność, jednoznaczność oraz łatwość interpretacji. Opisy wykonane w tej notacji można konwertować do postaci wymaganej przez inne standardy. Nic więc dziwnego, że BPMN został uznany jako standard międzynarodowy. Aktualna wersja standardu BPMN 2.0 została opublikowana w 2011 roku [3]. Przydatność tego standardu w zastosowaniach inżynierskich zilustrowano przykładem modelowania przepływu informacji w inteligentnym budynku. Przykład ten zamieszczono na końcu niniejszego rozdziału.

2.2. Przegląd standardów

2.2.1. EPC

Łańcuch procesów sterowany zdarzeniami (ang. *Event-driven Process Chain*, EPC) jest notacją starszą od notacji BPMN. Można powiedzieć, że wszystko, co można przedstawić w notacji EPC da się również przedstawić za pomocą BPMN. Ponadto w przypadku BPMN stosunkowo łatwo można przetłumaczyć model na inne notacje. W przypadku EPC jest to możliwe, choć bywa dużo bardziej skomplikowane. EPC pozwala na znacznie szerszą analizę biznesową, istotną z punktu widzenia procesów biznesowych. EPC cechuje podejście ukierunkowane bardziej na biznes, natomiast BPMN – na technologie informacyjne.

Diagramy EPC umożliwiają przedstawienia procesu jako łańcucha przeplatających się zdarzeń oraz działań. Zależności pokazane na tak skonstruowanym drzewie mogą występować tylko pomiędzy poszczególnymi działaniami. W drzewie tym mogą występować rozgałęzienia (AND, OR, XOR). Zdarzenie nie jest synonimem działania – można je przedstawić jako punkt na osi czasu. Sam proces zaczyna się tzw. zdarzeniem inicjalizującym, a kończy zdarzeniem końcowym.

2.2.2. BPEL

BPEL (ang. *Business Process Execution Language for Web Services*) jest językiem bazującym na XML, o notacji przeznaczonej do specyfikowania procesów biznesowych bazujących na usługach sieciowych. Specyfikacja ta może przyjmować formę wykonywalną oraz formę abstrakcyjną. Forma wykonywalna to opis zachowanie się uczestników biznesowych interakcji w postaci skryptów interpretowanych i wykonywanych przez maszyny procesów biznesowych w środowisku zgodnym ze standardem BPEL. Forma abstrakcyjna odpowiada modelowi protokołów wykorzystywanych podczas biznesowych interakcji. Zawiera definicje wzorców wymiany komunikatów bez ujawniania wewnętrznego zachowania ich nadawców i odbiorców. Stąd BPEL określany jest jako język orkiestracji usług sieciowych, a nie choreografii.

2.2.3. UML

Język UML (ang. *Unified Modelling Language*) jest językiem modelowania i dokumentowania różnorodnych systemów i procesów. Pozwala przedstawić ich aspekty statyczne i dynamiczne w postaci graficznej. Dzięki swej przejrzystości,

spójności oraz przyjazności stał się on jednym z podstawowych narzędzi inżynierii oprogramowania. Jest używany między innymi do przeprowadzania analizy wymagań (gdzie służy do komunikacji pomiędzy zamawiającymi a producentami oprogramowania), opisywania architektur rozwiązań, projektowania struktur danych oraz wytwarzania kodu.

2.2.4. IDEF

IDEF (ang. *Integrated Definition*, wcześniej ang. *ICAM Definition*) to cała rodzina technik modelowania stosowanych w inżynierii systemów i inżynierii oprogramowania. Początki IDEF sięgają lat siedemdziesiątych dwudziestego wieku, gdy zaczęto poszukiwać sposobów na lepsze opisywanie i tym samym zrozumienie systemów przez ich twórców i użytkowników. Pierwsze opracowania powstały w środowisku produkcyjnym i były związane ze zintegrowanym, komputerowo wspomaganym wytwarzaniem. Opracowane metody miały służyć doskonaleniu analizy procesów i zachodzącej w nich wymiany informacji. Wkrótce zaadoptowano je szerzej, w dziedzinach związanych z rozwojem oprogramowania.

Obecnie istnieje 16 technik modelowania IDEF, oznaczonych od IDEF0 do IDEF14 (włączając IDEF1X), różniących się rodzajem wykorzystywanej informacji w procesie modelowania. Techniki te są wykorzystywane do tworzenia graficznej reprezentacji różnych systemów, analizy modeli, tworzenia modeli systemów, wspierania przejścia pomiędzy modelami i do innych celów. Najczęściej stosowane są techniki IDEF0 i IDEF3 opracowane w 1995 roku (opracowanie technik jak IDEF7, IDEF10, IDEF11, IDEF 12 oraz IDEF13 nie wyszło poza fazę początkowej definicji). Zakres tematyczny technik IDEF wygląda następująco:

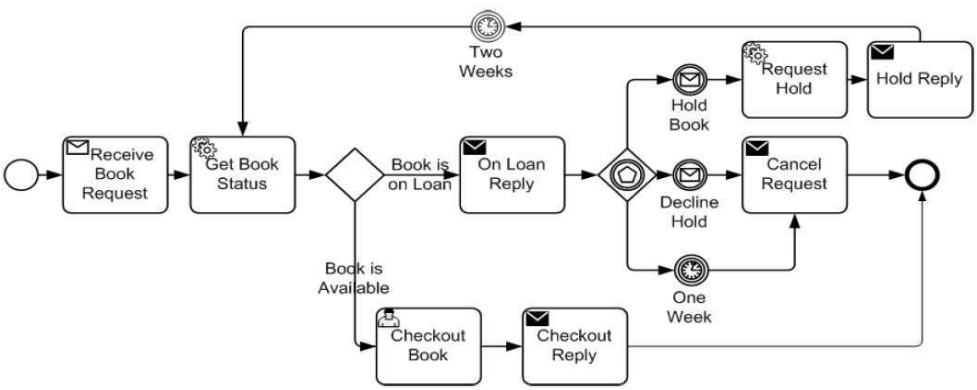
- IDEF0 – tworzenie modelu funkcjonalnego, tj. strukturalnej reprezentacji czynności i procesów (w tym współbieżnych) w modelowanym systemie.
- IDEF1 – tworzenie modelu informacyjnego, tj. struktury i charakteru zależności między wyrażeniami a obiektami, do których się odnosi w ramach modelowanego systemu lub obszaru. Daje możliwość pokazania przepływu informacji w procesie. IDEF1X rozszerza ją o zagadnienia projektowania.
- IDEF2 – tworzenie modelu dynamiki, tj. charakterystyki czasowej zachowania modelowanego systemu czy obszaru badań.
- IDEF3 – opis procesów, gdzie „proces biznesowy to uporządkowany ciąg wydarzeń angażujący osoby, surowce, energię i wyposażenie, który to ciąg zaprojektowany jest do osiągnięcia określonego rezultatu". Technika ta ma zwiększać wydajność analizy procesów biznesowych, ułatwiać opis systemu i wspierać zarządzanie projektem.
- IDEF4 – projektowanie obiektowe.
- IDEF5 – opis ontologii.
- IDEF6 – opis koncepcji projektu (racjonalne projektowanie).
- IDEF7 – audyt systemów informatycznych.
- IDEF8 – projektowanie interakcji człowiek-system.
- IDEF9 – wykrywanie więzów biznesu.
- IDEF10 – implementacja modelu architektury.

2.2.5. DFD

DFD (ang. *Data Flow Diagram* – diagram przepływu danych) pozwala na wizualizowanie procesu jako sieci procesów funkcyjnych połączonych potokami i zbiornikami danych. Diagram DFD składa się z procesu (funkcji, jak np. wyślij towar do klienta, reprezentowanej przez owalne pole), przepływu (zobrazowanego strzałką dochodzącą lub wychodzącą z procesu), magazynu (niedomkniętego prostokąta) oraz terminatora (prostokąta, odpowiadającego np. klientowi). DFD cechuje się dużą prostotą, hierarchicznym układem i odseparowaniem przepływu informacji od magazynowania informacji. Nie można jednak określić w nim dynamiki procesu, nie ma logiki działania funkcji oraz brak jest możliwości prezentacji funkcji w strukturze organizacji.

2.3. BPMN

BPMN (ang. *Business Process Model and Notation* — notacja i model procesu biznesowego) to formalizm służący do graficznego modelowania procesów biznesowych. Modelowanie procesów biznesowych to „działania związane z transformacją wiedzy o funkcjonowaniu wybranego obszaru (biznesowego) w modele odwzorowujące procesy realizowane w organizacji” [4]. Sam proces biznesowy definiuje się jako pewną uporządkowaną sekwencję czynności, które w konsekwencji doprowadzą do wytworzenia dobra. Sekwencja ta nie musi być zawsze taka sama. Także pojęcia *dobra* nie należy rozumieć w sposób materialny czy dosłowny. Dobrem może być towar, usługa, informacja lub wartość intelektualna. W opisie standardu często pojawia się słowo *notacja*. Jest to umowny, ściśle określony zbiór symboli, liter lub znaków wraz ze sposobem (zbiorem reguł, zasad) używania tych symboli. Dzięki niej można wizualizować przebieg procesu oraz opisywać wszystkie związki między poszczególnymi komponentami (z których składa się ów proces). Wszystko po to, by opis ten był przejrzysty, zrozumiały i czytelny. Przykład zastosowania notacji BPMN przedstawiono na rysunku 2.1.



Rys. 2.1: Przykładowy przebieg strumienia pracy w notacji BPMN (źródło: `http://brsilver.com/bpmn-method-and-style-an-example/`)

Cechą BPMN jest stabilna, graficzna notacja, pozwalająca opisać przebieg procesów oraz przepływ informacji między tymi procesami (przepływ komunikatów). Daje ona także możliwość ukazywania relacji, które mogą być realizowane przez różne podmioty. Mając model zjawiska można uruchamiać jego symulacje. Pozwala to zmniejszyć koszty i czas projektowania, eliminować usterki. Grafy zbudowane w BPMN mogą pomóc w ocenie wydajności czy wielkości potrzebnych zasobów.

W standardzie BPNM położono nacisk na metodykę, a nie na narzędzia. Określone w nim zasady opisu procesów pozostają niezmienne, choć same ich implementacje mogą być różne. O rosnącym znaczeniu standardu świadczy fakt, iż należy on do podstawowych i preferowanych form opisu procesów biznesowych nie tylko w Polsce, ale także w całej Unii Europejskiej.

2.3.1. Symbole

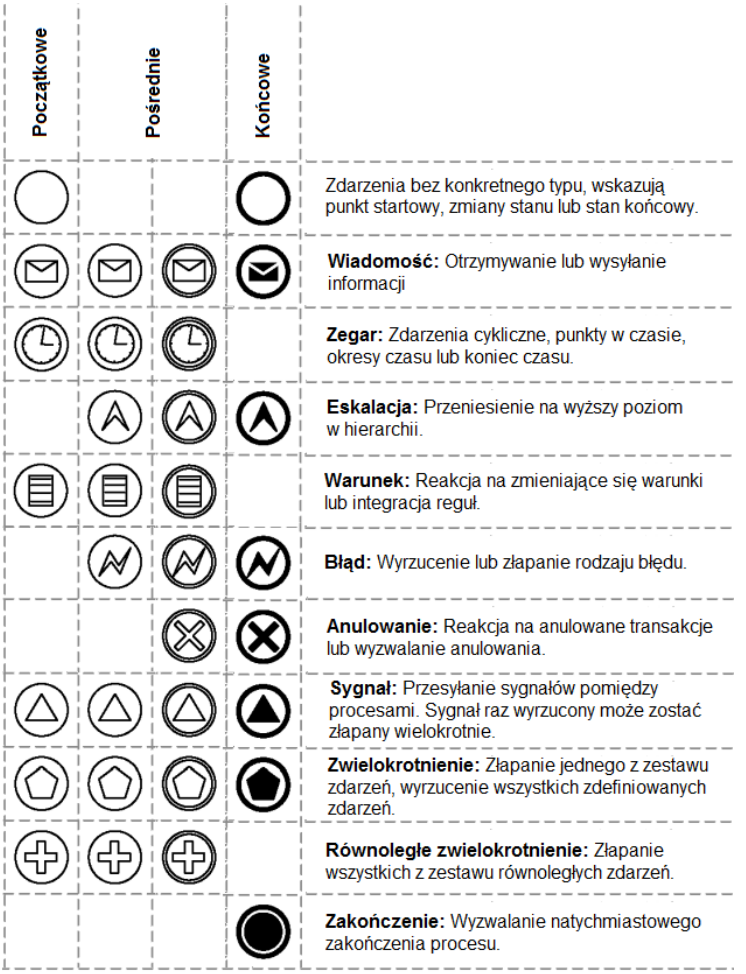
Notacja graficzna BPMN operuje na symbolach. Ich liczba, razem z możliwymi kombinacjami dwóch lub więcej różnych symboli, jest znacząca. Choć użycie tej notacji w wielu przypadkach jest intuicyjne, to jednak nie jest trywialne. Najogólniej można powiedzieć, że notacja BPMN składa się z czterech głównych klas elementów. Są to: elementy przepływu (ang. *flow objects*), połączenia (ang. *connecting objects*), miejsca realizacji (ang. *swimlanes*) oraz artefakty (ang. *artifacts*).

Elementy przepływu

Elementy te stanowią największą i najczęściej używaną grupę elementów. Przybierają postać znaków graficznych reprezentujących zdarzenia (ang. *events*) i czynności (ang. *activities*). Do tej kategorii zalicza się również bramki logiczne (ang. *gateways*).

Zdarzenia odzwierciedlają fakty, który wystąpiły bądź mogą wystąpić w modelowanym procesie. Ich pojawienie się wpływa na przebieg procesu. Jeśli dane zdarzenie wystąpiło, to znaczy, że było czymś spowodowane, a jeśli wystąpi, to spowoduje skutek. W zależności od miejsca wystąpienia na diagramie zdarzenia można podzielić na 3 typy: początkowe, pośrednie i końcowe (rys. 2.2). Symbolem zdarzenia zawsze jest koło, które może zawierać w środku dodatkowy znak graficzny, jak np. kopertę lub zegar. Puste koło o cienkich liniach oznacza zawsze początek procesu, pogrubiona linia oznacza zakończenie gałęzi procesu. Zdarzenia występujące w środku przepływu są oznaczone podwójną obwódką. Ponadto występują dwa szczególne przypadki zdarzeń: *łapanie* (ang. *catching*) oraz *rzucanie* (ang. *throwing*). Pierwszemu odpowiada symbol pusty (oznaczający, że proces odbiera zdarzenie), drugiemu – symbol wypełniony (oznaczający, że zdarzenie zostało wysłane, np. wiadomość lub dane).

Czynności reprezentują prace wykonywane przez uczestnika danego procesu. Można je podzielić na czynności proste lub złożone – w przypadku rozgałęzień i wystąpienia podprocesów. Podprocesy są oznaczane dodatkowym znakiem graficznym zwanym znacznikiem, który określa, jak zachowa się dana czynność przy

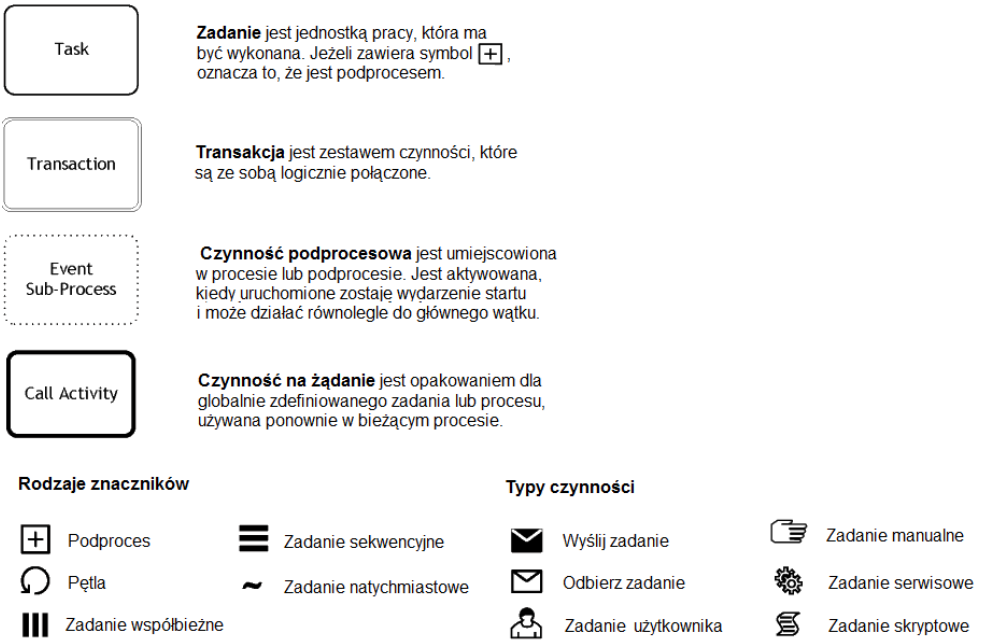


Rys. 2.2: Zestawienie symboli zdarzeń wraz z objaśnieniami

jej wykonaniu. Wszystkie czynności są reprezentowane prostokątami z zaokrąglonymi krawędziami i mogą mieć różne typy odpowiednio do kategorii zadania, które ma być wykonane. Na rysunku 2.3 przedstawiono symbole reprezentujące różne czynności, ich znaczniki oraz typy.

Bramki logiczne umożliwiają kontrolę przebiegu procesu. Można je porównać do elementów decyzyjnych, choć nie można ich z nimi utożsamiać. W notacji BPMN bramki nie mają funkcji decyzyjnych – te spoczywają na czynnościach. Zadaniem bramek jest odpowiednie kierowanie strumieni przepływu. Mają one zawsze kształt rombu z dodatkowym elementem w środku (rys. 2.4). Wyjątkiem jest bramka XOR, która najczęściej jest reprezentowana jako pusty romb. Ze względu na realizowane funkcje wyróżnia się następujące bramki:

- *Bramka równoległa* (AND) – może być używana do rozdzielenia przepływu pracy, gdy wszystkie strumienie wychodzące mają być aktywowane równocze-



Rys. 2.3: Zestawienie symboli dla czynności, ich typów oraz znaczników

Rys. 2.4: Zestawienie symboli bramek logicznych

śnie. Gdy łączy równoległe gałęzie, to czeka, aż otrzyma sygnały ze wszystkich wchodzących gałęzi, aby aktywować swój strumień wyjściowy.

- *Bramka niewykluczająca* (OR) – może rozdzielać strumień pracy, gdy aktywowana jest dowolna liczba gałęzi wyjściowych (może również nie być aktywowana żadna gałąź w wyniku niespełnienia warunków – taka sytuacja spowoduje błąd). Bramka OR może także pracować jako bramka łącząca strumienie pracy i zawsze czeka, aż otrzyma dane ze wszystkich aktywnych gałęzi.
- *Bramka złożona* (Complex) – jest specjalnym rodzajem bramki do specjalnych zastosowań. Zawiera zachowania odpowiadające za łączenie i rozdzielanie strumieni pracy, niewystępujące w innych bramkach.
- *Bramka wykluczająca sterowana zdarzeniami* (Event-based XOR) – dla każdego kolejnego wystąpienia zdarzenia zaczyna nową instancję procesu.

- *Bramka wykluczająca sterowana danymi* (Data-based XOR) – może rozdzielać strumień danych, ale przekierowuje go do dokładnie jednej gałęzi. Może również łączyć strumień danych i wtedy czeka, aż informacje pojawią się na dokładnie jednej gałęzi wejściowej zanim przekieruje potok.
- *Bramka równoległa sterowana zdarzeniami* (Event-based AND) – zaczyna nową instancję procesu, gdy będą aktywne wszystkie kolejne wystąpienia zdarzenia.

Połączenia

Połączenia symbolizują związki między łączonymi elementami na diagramie. Również tutaj można wyróżnić podkategorie:

- *przepływ sekwencji lub procesu* (ang. *sequence flow*) – szereguje (pokazuje kolejność) przepływu czynności realizowanych w danym procesie i jest reprezentowany linią ciągłą;
- *przepływ komunikatów, wiadomości* (ang. *message flow*) – symbolizuje wymianę komunikatów miedzy różnymi uczestnikami biorącymi udział w danym procesie i jest reprezentowany linią przerywaną;
- *asocjacje, powiązania* (ang. *associations*) – pozwalają dołączyć dodatkowe informacje do elementów przepływu i są reprezentowane linią kropkowaną. Strzałka na końcu asocjacji wskazuje kierunek powiązania.

Miejsca realizacji

Miejsca realizacji służą grupowaniu pewnych obiektów procesu i pozwalają na przypisanie ich do jednostki organizacyjnej (osoby lub roli). Wyróżniamy w tej klasie *pule* (ang. *pools*), które symbolizują uczestników procesu (np. firmę) oraz *tory* (ang. *lanes*), które organizują czynności i są umieszczane wewnątrz pul.

Artefakty

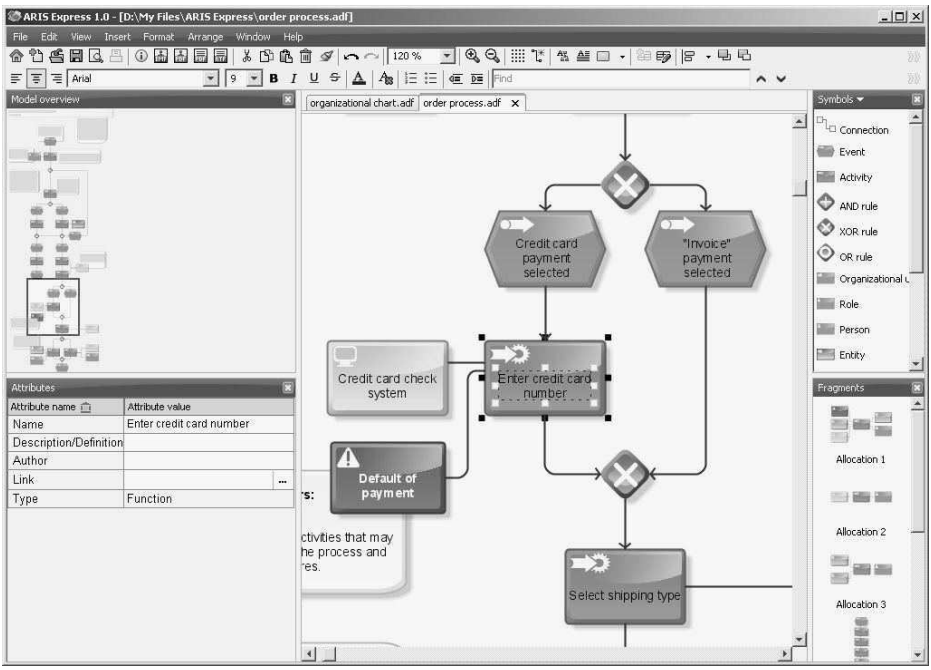
Artefakty pozwalają na dołączenie dodatkowych informacji potrzebnych podczas modelowania procesu. W klasie tej znajdują się *obiekty danych* (ang. *data objects*), które można dołączać do elementów przepływu (nie mają jednak na nie wpływu) i które mogą zawierać w sobie różnego rodzaju informacje (np. wymagania, czynności lub efekty); *grupy* (ang. *groups*), które symbolizują związki i łączą pewne elementy na diagramie; *adnotacje* (ang. *adnotations*), które zawierają dodatkowe informacje dla odbiorcy dołączane do diagramu.

2.4. Aplikacje wspomagające modelowanie

2.4.1. ARIS

ARIS (ang. *Architecture of Integrated Information Systems*) jest jednym z wiodących narzędzi do mapowania i reorganizacji procesów. Powstało ono na bazie badań prof. Augusta Wilhelma. Podstawową funkcją ARIS jest opisywanie danych

procesu i ich organizacji za pomocą modelu. Można nim także przeprowadzać analizy kosztów i czasu, opisywać przetwarzanie danych i zarządzać przepływem pracy oraz wspierać zarządzanie jakością. Dostępna dla niego jest duża baza modeli referencyjnych, na podstawie których można tworzyć nowe modele. Przykładowy widok interfejsu tego programu pokazano na rysunku 2.5.

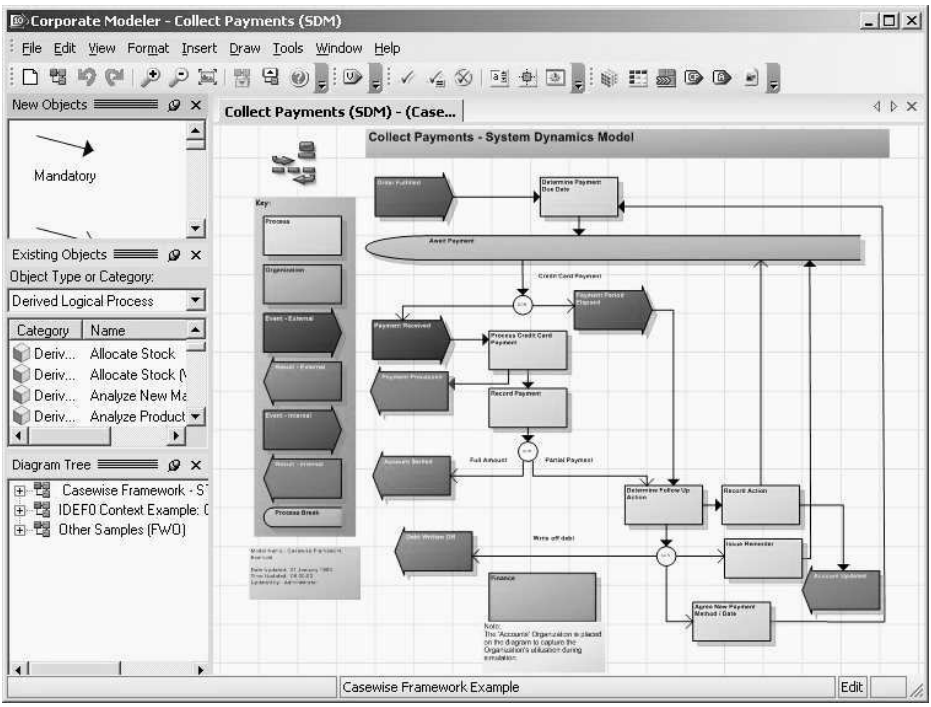


Rys. 2.5: Interfejs programu ARIS

2.4.2. Corporate Modeler

Narzędzie to powstało w 1989 roku jako produkt firmy Casewise. Zalicza się je do narzędzi typu CASE (ang. *computer aided system engineering*). Faktycznie Corporate Modeler służy do analiz organizacyjnych, nie zaś do samego projektowania systemu. Z jego pomocą można modelować środowisko, w którym działa dany system, uwzględniając procesy biznesowe i ich różne scenariusze.

Corporate Modeler jest rozwiązaniem niezależnym od konkretnej metodologi. Co za tym idzie, dostarcza wiele uniwersalnych narzędzi, pozwalając na ich zastosowanie odpowiednio do wybranej metodyki. Sama struktura programu zbudowana jest z modułów, które korzystają ze wspólnej bazy danych (zawierającej diagramy oraz obiekty wraz z atrybutami). To wszystko jest niezbędne do budowy modelu procesu. Proces można przedstawić jako pewien łańcuch czynności, który rozpoczyna się wydarzeniem oraz kończy rezultatem. Dane dotyczące procesu mogą być ilustrowane w różny sposób, np. jako diagram. Przykładowy widok interfejsu tego narzędzia pokazano na rysunku 2.6.

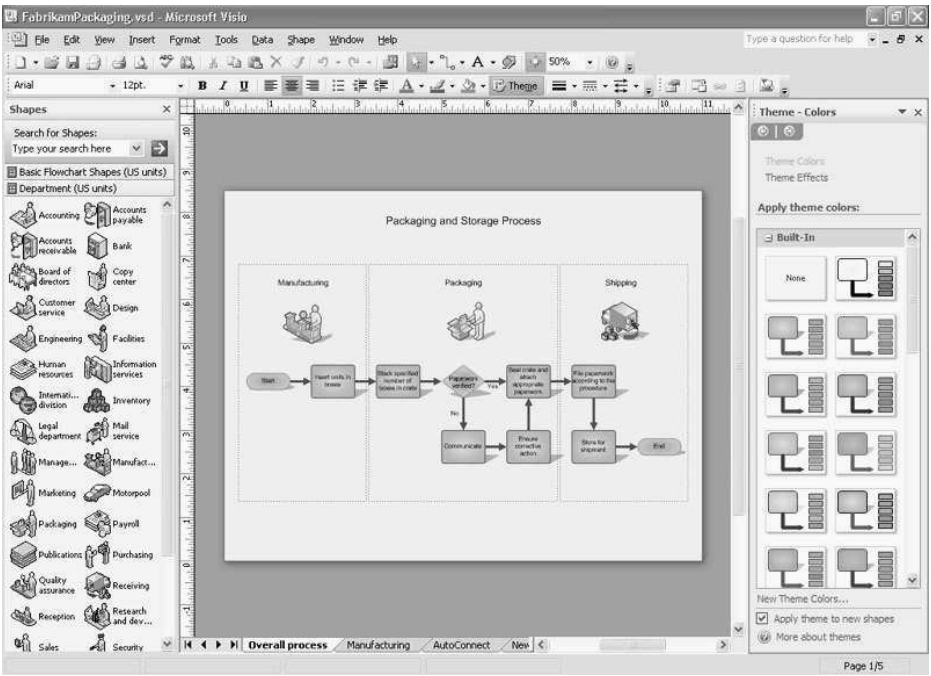


Rys. 2.6: Interfejs programu Corporate Modeler

2.4.3. Microsoft Visio

Oprogramowanie to jest częścią pakietu aplikacji Microsoft Office. Umożliwia edytowanie diagramów zgodnie ze standardem BPML oraz IDEF0. Pozwala na korzystanie z gotowych szablonów, jak i tworzenie nowych diagramów. Rozpoczynając z nim pracę należy wybrać z listy szablonów odpowiednią opcję: IDEF0 diagram model, WORK ROW diagram, Organistaion Chart, Basic FlowChart lub Cross Funcional Flowchart. Tworzone schematy mogą być integrowane z różnego rodzaju źródłami danych, np. Excel, Access i SQL.

Standard IDEF0 opisuje proces jako przejścia między czynnościami, korzystając z mechanizmów i kontroli. Diagramy Cross Functional opisują przejścia procesu między działami w firmie lub organizacji. Mogą one posłużyć jako opis procesu z różnych punktów widzenia, jako opis czynności i rejestr stanów procesu, jako narzędzie do analiz czynności lub wzajemnego oddziaływania. Diagramy Cross Functional są stosunkowo złożone, co czyni je obiektami trudnymi do tworzenia i zarządzania. Visio pozwala na sprawdzenie poprawności tego typu diagramów oraz umożliwia tworzenie diagramów przepływu. Dają one możliwość dobrego zilustrowania procesu produkcyjnego, logistycznego, przepływu informacji czy automatyzacji istniejących już procesów biznesowych. Najczęstsze podejście to tworzenie prostych i niezwykle czytelnych, ogólnych diagramów przepływu. Na rysunku 2.7 przestawiono przykładowy projekt w Microsoft Visio.



Rys. 2.7: Interfejs programu Microsoft Visio

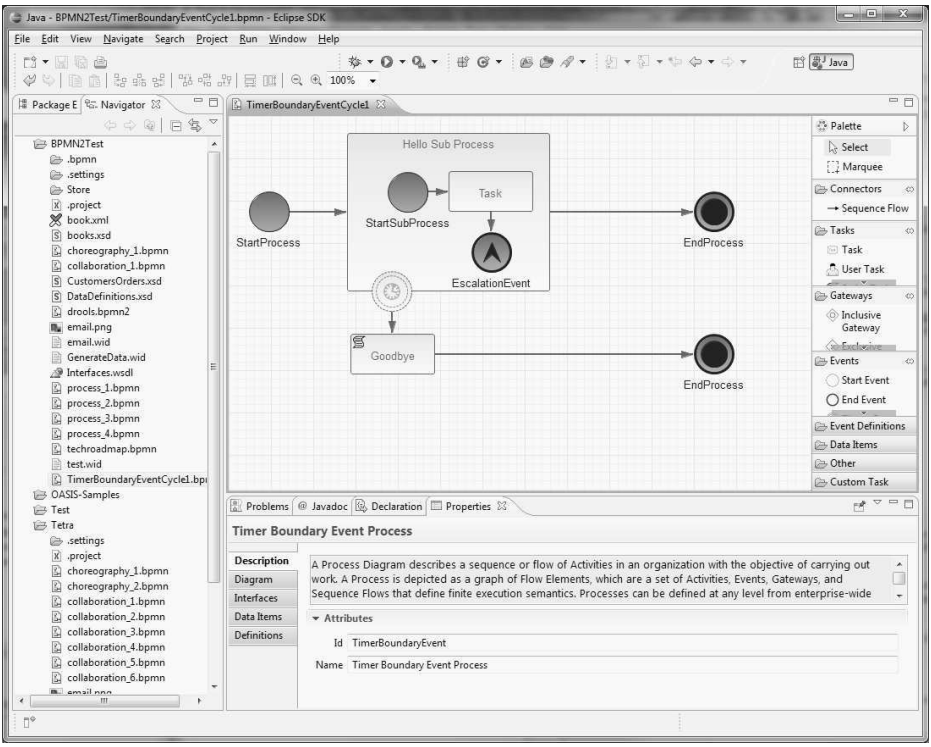
2.4.4. Eclipse BPMN2 Modeler

Platforma Eclipse została stworzona w 2001 roku jako środowisko programistyczne typu open source, a jej rozwój jest wspierany przez IBM. Platforma sama w sobie nie dostarcza narzędzi niezbędnych do tworzenia kodu czy budowania aplikacji. Rozszerzanie funkcjonalności zapewnione jest poprzez obsługę szerokiej gamy wtyczek (ang. *plugins*).

BPMN2 Modeler jest wtyczką służącą do graficznego modelowania procesów biznesowych. Narzędzie to jest oparte na Eclipse Graphiti oraz wykorzystuje BPMN 2.0 EMF meta model. Pozwala tworzyć opisy procesów biznesowych w składni XML BPMN 2.0 oraz wspiera BPMN DI (ang. *BPMN Diagram Interchange*). Istotną cechą tego narzędzia jest możliwość obsługi niemal wszystkich konstrukcji występujących w standardzie BPMN (w tym *lanes*, *pools*, *annotations* oraz wszystkich typów węzłów BPMN2). Na rysunku 2.8 przestawiono standardowy wygląd interfejsu użytkownika dostarczany przez tę wtyczkę.

2.5. Przepływ informacji w inteligentnym domu

W celu zobrazowania uniwersalności notacji BPMN zbudowano sieć zależności reprezentujących przepływ informacji w inteligentnym budynku. Przykład ten miał posłużyć udowodnieniu tezy, iż notacja BPMN nie tylko jest użyteczna do rozwiązywania złożonych kwestii przepływu informacji w biznesie, ale dzięki



Rys. 2.8: Interfejs dostarczany przez BPMN Modeller w środowisku Eclipse

prostocie i przejrzystej strukturze można ją stosować w zadaniach o bardzo praktycznym, inżynierskim charakterze. Dobór takiego przykładu wydaje się też ciekawy i niestandardowy, gdyż pozwala spojrzeć na sieć zależności w inteligentnym budynku jak na zbiór pewnych, częściowo zależnych od siebie procesów.

Model stworzono w środowisku Microsoft Visio. W modelu tym wyróżniono cztery pule:

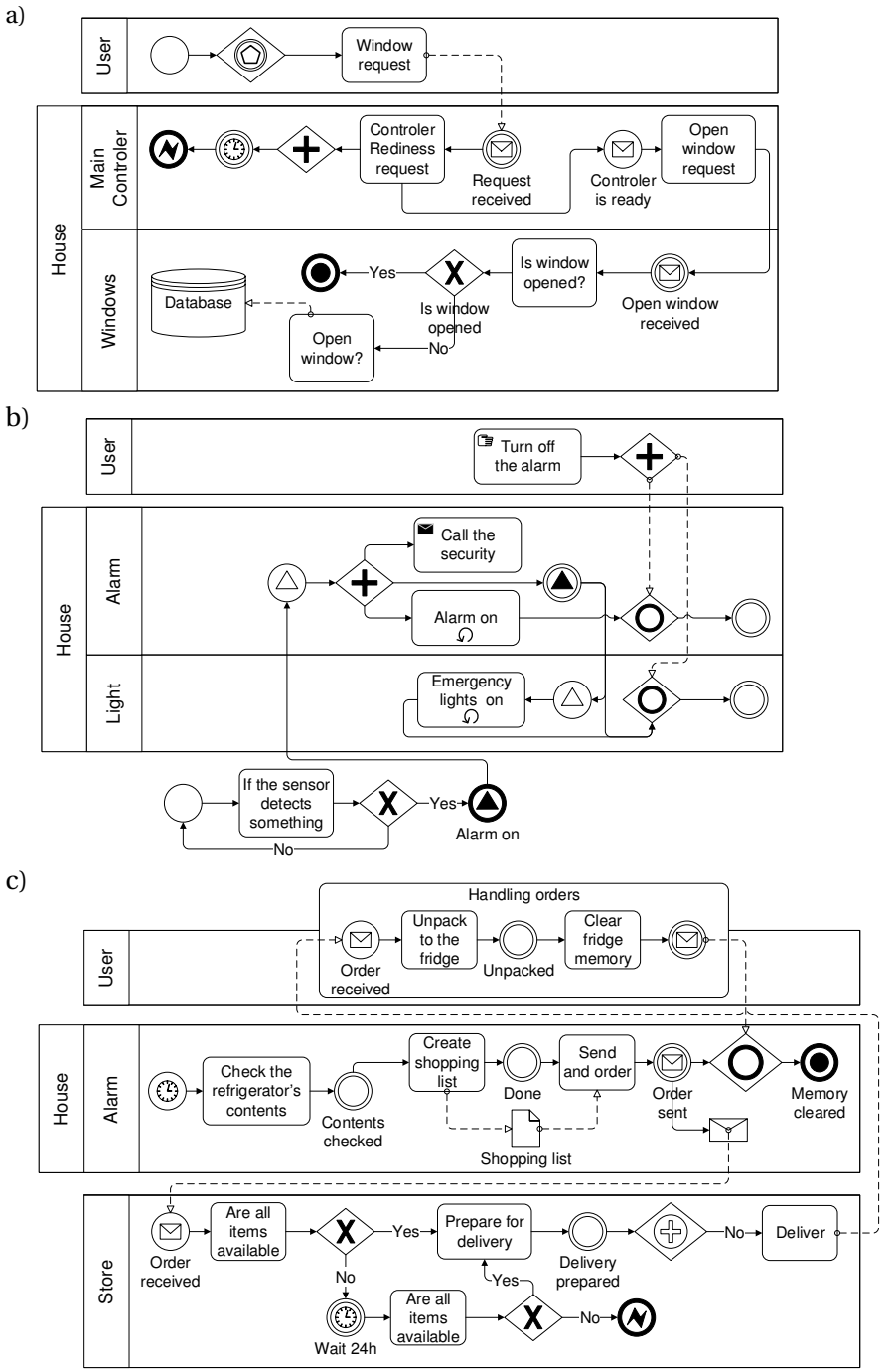
- *Human user* – reprezentuje użytkownika wysyłającego komunikaty i zadania, których wykonanie zależy od rodzaju prośby, dostępnych możliwości i dyspozycyjności agentów;
- *House* – reprezentuje zautomatyzowany dom jednorodzinny lub zautomatyzowane mieszkanie. To w niej następuje główny przepływ strumieni pracy. W ramach tej puli wydzielono dziewięć torów, reprezentujących różnych agentów. Są to: *Main Controler*, który nie jest „naturalnym" elementem domu, ale jest konieczny ze względu na prowadzenie nadzoru nad poprawnym wykonywaniem pracy pozostałych agentów, oraz *Okna*, *Oświetlenie*, *Rolety*, *Klimatyzacja*, *Telewizja*, *System audio*, *System alarmowy*, *Inteligentna lodówka*;
- *Store* – reprezentuje sklep, który jest ściśle połączony z inteligentną lodówką. Odbiera zamówienia i powiadamia użytkownika o możliwości odebrania towarów (realizuje typowe funkcje sklepu internetowego).

Przepływ informacji w zamodelowanym systemie odbywa się podobnie dla wielu agentów. W trybie normalnej pracy użytkownik inicjuje start procesu poprzez wysyłanie żądania wykonania zadania. Takim żądaniem może być polecenie: „Otwórz okno". Przedstawiony stopień szczegółowości projektu nie przewiduje etapu wyboru, które okno ma być otwarte. Uznano, że jest to element informacji przetwarzanej przez *Main Controler*, natomiast każde okno otwierane jest w takim samym procesie. Stąd w modelu występuje tylko jeden przepływ zawierający agenta zwanego *Okno*.

Następnie informacja przekazywana jest do kontrolera głównego. Musi on sprawdzić, czy agent odpowiedzialny za wykonywanie określonego zadania jest dostępny w danej chwili i dopiero wtedy przekierować do niego strumień pracy. Gdy agent nie jest dostępny (realizuje już przepływ pracy lub zepsuł się) *Main Controller* musi odczekać określony przedział czasu. Po jego upłynięciu, jeżeli brak jest możliwości przekierowania strumienia, pojawia się błąd, który w trybie natychmiastowym kończy proces. W przypadku sukcesu zadanie jest kierowane do odpowiedniego agenta. Sprawdzane są wtedy warunki możliwości wykonania zadania, na przykład, czy okno nie jest już otwarte, i, jeśli to możliwe, zadanie zostaje wykonane, a proces ukończony. Przykład graficznego zapisu BPMN omawianego procesu przedstawiono na rysunku 2.9a.

Oddzielnym przypadkiem jest działanie alarmu. Alarm należy do zjawisk niedeterministycznych, na które model musi poprawnie reagować. W notacji BPMN istnieje odpowiedni znacznik oznaczający natychmiastowe wykonanie zadania. Jeśli informacja ze środowiska zewnętrznego (z czujników) o niepokojących warunkach zostanie zrealizowana, natychmiast załączy się protokół bezpieczeństwa (zostanie wyemitowany sygnał dźwiękowy, świetlny oraz automatyczny telefon do ochrony). Zastosowano tu więc przepływ sygnałów, a nie informacji. Te pierwsze bowiem są obsługiwane natychmiast, natomiast odczytanie i wykonanie wiadomości następuje w najbliższym możliwym momencie. Zadania włączenia alarmu dźwiękowego oraz świetlnego mają znaczniki pętli, w związku z czym wykonują się non stop. Procedurę alarmu może przerwać tylko użytkownik, co również jest uwzględnione w przepływie pracy. Nieprzerwane wykonywanie się alarmu, aż do wystąpienia zadania manualnego, jest realizowane za pomocą odpowiedniej bramki. Włączenie alarmu jest typem zadania wyjątkowego, ponieważ odbywa się bez udziału głównego kontrolera. Na rysunku 2.9b przedstawiono zapis takiej procedury w notacji BPMN.

Ciekawym elementem jest inteligentna lodówka. Jej zadaniem jest kontrola zawartości i autonomiczne zamawianie produktów. Realizuje to cyklicznie, na przykład co tydzień. Przepływ zadań i danych dla tego procesu przedstawiono na rysunku 2.9c. W przepływie tym zamodelowano odpowiednie zdarzenie startowe, które jest odpalane w regularnych odstępach czasu. Lodówka zamawia zakupy poprzez przesłanie informacji (zadania) do jednostki *Store*. Tam następuje proces typowy dla przetwarzania strumieni pracy w sklepach i magazynach, a więc sprawdzenie dostępności towaru, przygotowanie dostawy lub anulowanie zamówienia, ewentualne czekanie na udostępnienie towaru i dostawę. Produkty odbiera i rozpakowuje użytkownik w podprocesie *Handling orders*. Interesujące



Rys. 2.9: Wydzielone fragmenty przepływów strumieni pracy w inteligentnym domu: a) okna, b) procedura alarmowa, c) inteligentna lodówka

może być manualne zadanie kasowania pamięci lodówki zamieszczone również w tym podprocesie. Jest to warunek konieczny, aby proces zamówienia produktów został zakończony. Zakończenie zamówienia sprawia, że czas do następnego zamówienia liczony jest od początku.

2.6. Podsumowanie

Business Process Modeling Notation to graficzna notacją służącą do opisywania procesów biznesowych. Zamiarem jej twórców było stworzenie sposobu zapisu procesów biznesowych zrozumiałego zarówno dla osób niezwiązanych z informatyką, jak i dla analityków i informatyków. Zaletą tej notacji jest jej jednoznaczność oraz to, że daje możliwość modelowania praktycznie wszystkich procesów biznesowych niezależnie od specyfiki danej branży, organizacji itp.

Zaprezentowany projekt inteligentnego domu jest przykładem, na bazie którego pokazano sposób wykorzystania notacji BPMN do modelowania przepływu informacji w złożonym obiekcie. Dzięki BPMN udało się w przejrzysty i zrozumiały sposób przedstawić pozornie skomplikowane sieci zależności i powiązania. Ta użyteczność i łatwość interpretacji notacji BPMN sprawia, iż jest ona coraz chętniej wykorzystywana w przemyśle, administracji i nauce.

Literatura

[1] D. Georgakopoulos, M. Hornick, A. Sheth. An overview of workflow management: From process modeling to workflow automation infrastructure. *Kluwer Academic Publishers*, 1995.

[2] W. van der Aalst, K. van Hee. *Workflow Management Models, Methods, and Systems*. The MIT Press Cambridge, 2002.

[3] Object Management Group. Business Process Model and Notation (BPMN) Version 2.0, Styczeń 2011. `http://www.omg.org/spec/BPMN/2.0` [dostęp dnia 20 czerwca 2015].

[4] M. Moś. BPMN Notacja modelowania procesów biznesowych. `http://procesy.ue.wroc.pl/uploads/Marcin/BPMN.pdf` [dostęp dnia 20 czerwca 2015].

ROZDZIAŁ 3

WERSJONOWANIE I AGREGACJA W SYSTEMACH ROZPROSZONYCH

M. Ciopcia, D. Gut

W niniejszym rozdziale opisano projekt autorskiego systemu wersjonowania służącego do archiwizacji wybranych stron www. Oprócz składowania danych system ten ma umożliwiać ich przeszukiwanie oraz odtwarzanie według stanu z interesującej użytkownika chwili. W jego architekturze przewidziano miejsce dla kilku niezależnych, synchronizowanych ze sobą maszyn. Ma to zapewnić wysoką dostępność i niezawodność całego rozwiązania.

3.1. Wprowadzenie

Systemy wersjonowania pełnią istotną rolę w procesach rozwoju i utrzymania systemów informatycznych. Oprócz monitorowania zmian i archiwizacji danych można za ich pomocą przeprowadzać analizy porównawcze, rozwiązywać konflikty występujące podczas integrowania rozgałęzionych archiwów, wyliczać statystyki itp. Systemy te służą do wykrywanie źródeł błędów i ich naprawy, pomagają też w utrzymaniu wyników prac programistycznych i zapewnieniu jakości tworzonych produktów jak np. systemy do zarządzania kodem źródłowym `svn` czy `git`. Ponadto są one wyposażone w mechanizmy pozwalające na jednoczesną pracę wielu użytkownikom oraz wspólne zarządzanie projektem.

Coraz szersze wykorzystanie Internetu jako medium do publikacji informacji spowodowało, iż nawet w tekstach naukowych zaczęto umieszczać referencje do materiałów źródłowych dostępnych w Internecie, z podaniem ich adresu URL oraz informacji o dacie dostępu. Niestety, zdarza się, że podczas lektury takich opracowań cytowany zasób nie jest już dostępny albo też istnieje, jednak w zmienionej postaci. Rodza się więc pytania: W jaki sposób zapewnić większą żywotność zasobów? Co zrobić, aby dało się pozyskać zasoby w pierwotnej postaci? Odpowiedziami na te pytanie są systemy wersjonowania i archiwizacji stron www. Pozwalają one monitorować wskazane zasoby i rejestrować zachodzące zmiany. Istnieje też wielu innych powodów, dla których systemy te można uznać za niezwykle interesujące [1].

W niniejszym rozdziale przedstawiono koncepcję budowy systemu archiwizacji stron www. Omówiono takie kwestie, jak: architektura systemu, topologia sieci synchronizująco-archiwizującej, skalowalność oraz podstawowa analiza bezpieczeństwa proponowanych rozwiązań. Działanie systemu przypominać ma działanie serwisów `Google Cache` (`http://webcache.googleusercontent.com/`), `Archive.org` (`http://web.archive.org/`) czy `Coral web cache` (`http://cachedview.com/`). Jednym z założeń, na których oparto wspomnianą koncepcję, było wykorzystanie gotowych narzędzi działających w systemie Linux. Założenie to miało zapewnić większą stabilność, elastyczność i bezpieczeństwo całego rozwiązania.

3.1.1. Dekompozycja problemu

Projektując system wersjonowania należy poradzić sobie z wieloma pozornie błahymi kwestiami, które rzutują na jakość końcowego produktu. Rozwiązanie musi być szybkie, łatwo skalowalne, bezpieczne i wydajne. Nie powinno narażać użytkownika na ryzyko przejęcia maszyny czy uszkodzenia bazy danych wynikłe z sytuacji, przed którą można byłoby ustrzec się na poziomie analizy funkcjonalnej oraz analizy przypadków użycia.

Podczas budowy skalowalnych systemów niezwykle praktyczne jest stosowanie reguły KISS (ang. *Keep It Simple, Stupid*), co tłumaczy się wprost jako „nie komplikuj, głuptasku". W odniesieniu do tematu rozdziału oznacza to stworzenie prostego, programowego rozwiązania, którego elastyczność i stabilność działania będzie łatwa do uzyskania. Osoba uruchamiająca taki system powinna mieć dużą swobodę w kształtowaniu jego zachowań i architektury, między innymi poprzez wprowadzanie dodatkowych, pierwotnie nieprzewidzianych zabezpieczeń (jeśli zajdzie taka konieczność).

Aby realizacja podjętego zadania była efektywna zdecydowano się zastosować metodę *dziel i rządź*. W tym celu zdekomponowano problem na mniejsze zadania, realizowane niezależnie od siebie. Przyjęto też zasadę projektowania jak najprostszych modułów, które jednocześnie zapewniałyby wysokie możliwości konfiguracji (choć może to skutkować wysokim nakładem pracy związanym z uruchomieniem systemu). Ostatecznie zaprojektowano następujące moduły:

Archiwizator stron www – moduł odpowiedzialny za pobranie zawartości strony www i zapisanie jej na dysku w celu dalszego przetwarzania (jego zadaniem jest pobranie wiernej kopii witryny www w postaci umożliwiającej jej otwarcie na komputerze lokalnym).

Generator funkcji skrótu – moduł odpowiedzialny za generowanie unikalnego identyfikatora danej kopii strony (powstały identyfikator pełni rolę detektora zmian, jego zmiana świadczy o zmodyfikowaniu zawartości danej strony).

Moduł zarządzania bazą danych – moduł zbiera informacje o znacznikach czasu i przechowywanych kopiach stron www.

Wyzwalacz – element zapewniający wyzwolenie systemu archiwizacji co określony kwant czasu (czas pomiędzy synchronizacjami powinien być tak dobrany, aby zapewnić możliwie szybką aktualizację kopii strony bez zbytniego obciążenia serwera, łącza oraz maszyny archiwizującej).

System synchronizacji – podsystem umożliwiający wymianę informacji o przechowywanych wersjach strony oraz zapewniający ich synchronizację (jeśli zostanie wykryta jakaś rozbieżność).

Moduł interakcji z użytkownikiem – zawiera interfejs programistyczny aplikacji (ang. *Application Programming Interface*, API) umożliwiający wykonywanie prac administracyjnych oraz pobieranie zarchiwizowanych stron.

3.2. Opis modułów

3.2.1. Archiwizator stron WWW

Podczas archiwizacji stron www ważnym aspektem jest nie tylko same pobieranie ich kodu źródłowego, ale także zamiana występujących w tym kodzie odnośników na lokalne odpowiedniki. Dlatego w projekcie zdecydowano się wykorzystać oprogramowania typu `wget` natywnie wspierane przez systemu Linux. Dzięki niemu archiwizowanie danej strony www można zrealizować za pomocą prostego polecenia:

```
wget -mirror -nH -P pageName url,
```

gdzie `pageName` to tag (nazwa) strony www (np. dla strony `http://www.pwr.edu.pl` będzie to `pwr.edu.pl`), zaś `url` to adres strony www do archiwizacji.

Strona www może zawierać setki lub tysiące podstron, obrazków i innych plików. Zagregowanie ich w jednym pliku powinno przyspieszyć przetwarzanie na każdym kolejnym etapie. Dobrym założeniem jest, aby takie archiwa miała jedną, ściśle określoną postać. Założenie to można spełnić poprzez zastosowanie serializacji do formatu `.tar`. Archiwa `.tar` są bardzo podatne na kompresję, można więc otrzymaną kartotekę poddać dodatkowej kompresji, zmniejszając tym samym zapotrzebowanie programu na przestrzeń dyskową (kosztem wydłużenia czasu trwania operacji archiwizacji).

3.2.2. Generator funkcji skrótu

Istnieją proste mechanizmy pozwalające generować krótkie znaczniki odpowiadające jednoznacznie zawartości dużych zbiorów danych. Mechanizmy te to tzw. funkcje skrótu. Dzięki nim rozróżnianie kolejnych instancji archiwizowanych stron nie powinno stanowić problemu. Wystarczy bowiem zapisać skrót strony www w bazie danych wraz ze znacznikiem czasu i nazwą serwisu, a otrzyma się gotowy system katalogowania zawartości stron www, zarówno w domenie nazw, jak i czasu.

Najmniejsza modyfikacja zbioru danych powoduje zmianę odpowiadającej mu funkcji skrótu. Dlatego też nadaje się ona do detekcji takich modyfikacji. Jeśli po pobraniu i wstępnym spakowaniu dana strona www posiada taką samą wartość skrótu jak odpowiadający jej wpis uprzednio skatalogowany, to znaczy, że nie wystąpiły żadne zmiany i archiwizacja w bieżącej chwili nie jest potrzebna.

W przykładowym rozwiązaniu do generowania funkcji skrótu użyto algorytmu `md5`. Nic nie stoi jednak na przeszkodzie, aby w rozwiązaniu zastosować zupełnie inną funkcję skrótu, np. `sha`.

3.2.3. Zarządca bazy danych

Kluczowe znaczenie dla skalowalności projektu ma budowa bazy danych. Należy to wziąć pod uwagę już na etapie analiz. W zaproponowanym rozwiązaniu do przechowywania zawartości bazy danych użyto pliku XML. Rozwiązanie to ma jedną, niebagatelną zaletę – generuje kompaktowe rozwiązanie, które jest przenośne i łatwo poddaje się procesowi tworzenia kopii bezpieczeństwa. Z drugiej jednak strony nakłada ograniczenia na skalowalność projektu.

W zaproponowanej wersji programu użyto parsera DOM, który jest łatwy w obsłudze oraz stosunkowo szybki. Wymaga jednak wczytania zawartości całego pliku bazy danych do pamięci RAM zanim dojdzie do przetwarzania jej zawartości. Z tego powodu rozwiązanie to nie nadaje się do użycia w wypadku projektów dużej skali (np. archiwizacji całych podsieci www), nawet wtedy, gdy pojedyncze wpisy do bazy danych będą zajmować znikomą ilość pamięci. Do parsowania większych objętościowo plików XML można użyć parsera SAX. Nie jest to rozwiązanie wydajne, szczególnie w przypadku operacji zapisu. Dla projektów dużej skali zalecane jest użycie dedykowanych baz danych SQL [2, 3].

Zaproponowana baza danych pozwala na: porównanie wartości funkcji skrótu z istniejącymi w niej wpisami, dodawanie funkcji skrótu do bazy danych wraz z informacją o znaczniku czasu (w standardzie ISO 8601) oraz porównanie dwóch baz danych w celu wykrycia rozbieżności. Strukturę pliku XML bazy danych określa następujący schemat XML Scheme (zapisany w pliku ./scripts/grammar/grammar.xsd):

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="database" type="databaseType"/>
    <xsd:complexType name="databaseType">
        <xsd:sequence>
            <xsd:element name="website" minOccurs="0"
             maxOccurs="unbounded" type="websiteType"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="websiteType">
        <xsd:sequence>
            <xsd:element name="websiteName" type="xsd:string"/>
            <xsd:element name="version" minOccurs="0"
             maxOccurs="unbounded" type="versionType"/>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="versionType">
        <xsd:sequence>
            <xsd:element name="time" type="xsd:string"/>
            <xsd:element name="md5sum" type="xsd:string"/>
            <xsd:element name="archivePath" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Ponieważ zapis danych w archiwach wiąże się z rejestracją czasu najwcześniejszego wystąpienia danej funkcji skrótu, dlatego ich prawidłowa obsługa wymaga posłużenia się pewnym źródłem czasu. Łatwo bowiem wyobrazić sobie sytuację, gdy uszkodzony RTC jednej z synchronizowanych maszyn niszczy lokalną bazę danych oraz bazy danych na innych maszynach. Dlatego w proponowanym rozwiązaniu większą niezawodność zapewnić ma wymuszenie synchronizacji serwera z serwerami usługi NTP tuż przed podjęciem jakichkolwiek czynności.

3.2.4. Wyzwalacz

Do wyzwalania programu wykorzystano narzędzie `cron` dostarczane wraz z systemem Linux i domyślnie w nich uruchomiane. Umożliwia ono wywoływanie dowolnego programu w określonej chwili lub co zadany interwał czasu z dokładnością co do minuty. Uzyskano więc rozwiązanie pewne w działaniu i zaimplementowane w sposób o wiele bardziej zoptymalizowany niż przy zastosowaniu jedynie interpretera języka Python.

Dzięki takiej decyzji oszczędzono zasoby maszyny archiwizującej, uproszczono konstrukcję programu oraz zwiększono niezawodność rozwiązania. Program stał się odporny na wszelkiego rodzaju błędy (typu: wycieki pamięci, brak dostępu do internetu, wadliwe zadziałanie), gdyż z każdą próbą synchronizacji zostaje uruchomiona nowa instancja aplikacji. Kolejną konsekwencją takiej decyzji było zastosowanie dodatkowego zabezpieczenia w postaci blokady uruchomienia modułu archiwizacji strony w wypadku, jeśli program wykryje inną aktualizację w toku. Dzięki pilnowaniu właściwej kolejności wykonywania operacji dodawania wpisu do bazy danych zapewniono bezkonfliktową pracę modułu synchronizacji i dodawania stron (jeśli synchronizacja zostanie uruchomiona z uszkodzoną bazą danych, np. bazą danych odczytaną w trakcie zapisu innego modułu, synchronizacja nie powiedzie się).

`cron` ma uruchamiać zarówno moduły archiwizujące, jak i synchronizujące, zgodnie z zapisami w pliku konfiguracyjnym (stworzona aplikacja nie jest interaktywna). Poniżej pokazano przykładową zawartość pliku `/etc/crontab` z konfiguracją pozwalającą archiwizować stronę `http://pwr.edu.pl` co 30 minut i uruchamiać synchronizację między maszynami codziennie o północy:

```
30  * * * * /opt/archiver/archiver.py -an http://pwr.edu.pl
00 00 * * * /opt/archiver/sync.sh
```

3.2.5. System synchronizacji

Synchronizacja następuje poprzez porównania dwóch baz danych i wymianę brakujących wpisów i archiwów `.tar`. Wykonuje się ją wywołując program z odpowiednimi parametrami.

W trakcie prac napotkano problemy z modułem Pythona obsługujący połączenia `SSH`. Dlatego w celu wymiany plików przez instancje archiwizatorów zastosowano narzędzia `sshfs` oraz autoryzację w oparciu o klucze (co umożliwia automatyzację procesu). Przykładowy skrypt synchronizujący dane z serwerem (`/opt/archiver/sync.sh`) zaprezentowano poniżej.

```
#!/bin/bash
mkdir /opt/archiver/sync
sshfs user@serwer.pl:/opt/archiver /opt/archiver/sync
/opt/archiver/archiver.py -s /opt/archiver/sync
fusermount -uz /opt/archiver/sync
rmdir /opt/archiver/sync
```

Dzięki jednostronnej synchronizacji i elastycznej konfiguracji poszczególnych modułów synchronizacji możliwe jest realizowanie wielu scenariuszy współpracy programów archiwizujących. Najprostszym rozwiązaniem jest rozproszenie procesu archiwizacji na wiele równoważnych węzłów, z których każdy posiada pełne archiwum, lecz zajmuje się archiwizacją jedynie wybranych stron. Wersje stron niearchiwizowanych przez dany komputer dostarczane są za pomocą mechanizmu synchronizacji z innymi węzłami.

Kolejną interesującą architekturą byłaby synchronizacja w trybie (*multi*) *master-slave*. W tym trybie archiwizacją stron www zajmują się wyspecjalizowane węzły, które nie synchronizują się z żadnym innym urządzeniem (zawierają jedynie kopie stron, które same zarchiwizowały). Maszyny typu master pełnią funkcję katalogów – synchronizują się z każdą z maszyn slave, pobierając jedynie aktualizacje archiwów. Architektura tego typu mogłaby sprawdzić się przy synchronizacji znacznej liczby stron www, gdyż operacje wymagające najwięcej zasobów (przepustowości łącza i czasu procesora) zostają rozproszone pomiędzy maszyny typu slave. Użyteczną modyfikacją powyższego rozwiązania byłoby przechowywanie na maszynach typu master jedynie katalogu wraz z odnośnikami do maszyn, na których znajduje się taka kopia. Jednakże implementacja tego typu rozwiązania jest skomplikowana i wymaga równoważenia obciążeń.

3.2.6. Moduł interakcji

W proponowanym rozwiązaniu dokonuje się wywołań programu z odpowiednimi parametrami bądź uruchomia się program w trybie interaktywnym. W wypadku obsługi programu za pomocą zewnętrznej nakładki programowej wygodnie jest korzystać bezpośrednio z bazy `XML` oraz wywoływać program w tle z wymaganymi parametrami.

3.3. Opis działania programu

3.3.1. Instalacja wymaganych modułów

Do poprawnego działania programu wymagane jest posiadanie interpretera języka Python w wersji 2.7.6 oraz biblioteki PyXB w wersji 1.2.4 (do przetwarzanie dokumentów XML z wykorzystaniem schematów). Instalacja programu polega na: i) rozpakowaniu archiwum zawierającego program; ii) przejściu do katalogu `pyxb_source/PyXB-1.2.4`; iii) wydaniu polecenia `python setup.py install`. Po instalacji wymaganych modułów (`python`, `openssh-server`, `sshfs`) oraz wykonaniu czynności opisanych powyżej program jest gotowy do użytku.

3.3.2. Obsługa programu

Plikiem wykonywalnym jest skrypt `./archiver.py`. Przyjmuje on następujące parametry:

`-h` – wyświetla pomoc,

`-a url` – pobiera stronę i zapisuje do bazy w trybie użytkownika,

`-an url` – pobiera stronę i zapisuje do bazy w trybie *non interactive*,

`-s ścieżka` – pobiera dane z repozytorium znajdującego się pod podaną ścieżką,

`-sn ścieżka` – pobiera dane z repozytorium znajdującego się pod podaną ścieżką w trybie *non interactive*,

`-p` – wyświetla listę zarchiwizowanych stron,

`-d pageName,timestamp` – rozpakowuje wybraną stronę do katalogu `./extracted_website`.

Tryb użytkownika daje możliwość zarządzania procesem dodawania poszczególnych elementów archiwum oraz podejmowania decyzji o przebiegu archiwizacji. Tryb *non interactive* jest przeznaczony dla skryptów – domyślnie nie nadpisuje starszych wersji oraz synchronizuje wszystkie elementy – nie potrzebuje więc interakcji ze strony użytkownika do poprawnego działania.

3.3.3. Konfiguracja programu do pracy w trybie automatycznym

Procedura automatyzacji procesu w zaproponowanym rozwiązaniu opiera się o konfigurację skryptu programu `crontab` oraz konfiguracji skryptów synchronizacji. Stąd poziom bezpieczeństwa danej instancji programu do archiwizacji zależy od umiejętności i wiedzy osoby konfigurującej sieć oraz istniejących zabezpieczeń. W wypadku rozproszenia procesu archiwizacji na wiele równoważnych węzłów istnieje ryzyko przejęcia dostępu do całej sieci, jeśli atakujący przejmie kontrolę nad dowolnym z komputerów. Konfiguracja master-slave zapewnia większe bezpieczeństwo, gdyż możliwości wpływania na pracę maszyny master po przejęciu dowolnej maszyny typu slave jest ograniczony.

W proponowanym rozwiązaniu używa się mechanizmów bezpieczeństwa wbudowanych w system Linux, pozwalających przypisać danemu zadaniu użytkowników z odpowiednimi uprawnieniami. Można także zabezpieczyć proces synchronizacji za pomocą takich technologii, jak: `ppp` czy `vpn`. W wypadku sieci master-slave nic nie stoi na przeszkodzie aby zastosować dodatkową warstwę ochrony, gdyż jedynie komputery typu slave wymagają dostępu do internetu. Synchronizacja maszyn master może odbywać się przez dowolny system wymiany plików, np. przez intranet czy null-modem.

3.3.4. Opis procedur wykonywanych przez program

Pobieranie strony

1. Tworzony jest katalog `./tmp`.
2. Do katalogu `./tmp` kopiowana jest strona (polecenie `wget –mirror –nH –P pageName url`).
3. Utworzone zostaje archiwum `.tar`, z głównym katalogiem zawierającym nazwę strony, w którym znajduje się zawartość archiwum.
4. Obliczona zostaje wartość skrótu `md5`.
5. Jeżeli w bazie znajduje się wpis o takim samym `md5` dla danej strony, użytkownik zostaje zapytany o wersję, którą chciałby zachować w bazie (zapewnia unikalność haszy).
6. Następuje zapis uaktualnionych informacji do bazy danych.

Synchronizacja repozytorium pomiędzy katalogami

1. Tworzony jest katalog `./tmp`.
2. Do katalogu `./tmp` kopiowany jest plik `database/database.xml` ze wskazanego folderu.
3. Następuje iteracja po skopiowanej bazie danych:
   a) Jeżeli stwierdzono, że witryna była pobrana zdalnie i lokalnie, następuje iterujemy po wersjach:
      i. Jeżeli znaleziono wersje, dla których hasz jest taki sam, każemy użytkownikowi wybrać wersję do zachowania.
      ii. Jeżeli znaleziono wersję, która została pobrana zdalnie i nie ma jej lokalnie, pytamy się użytkownika, czy chce ją dodać.
   b) Jeżeli znaleziono witrynę, która jest lokalnie, a nie była pobrana zdalnie, pytamy się użytkownika, czy chce ją dodać:
   c) Jeżeli użytkownik chce dodać witrynę, może to zrobić na dwa sposoby:
      i. Dodać wszystkie zarchiwizowane wersje z lokalizacji zdalnej.
      ii. Odpytać użytkownika o dodanie każdej wersji z osobna.
4. Zostaje zapisana uzupełniona baza danych.

Literatura

[1] K. Fitch. Web site archiving - an approach to recording every materially different response produced by a website. *The Ninth Australian World Wide Web Conference*, strony 5–9, Lipiec 2003. `http://ausweb.scu.edu.au/aw03/papers/fitch/` [dostęp dnia 20 czerwca 2015].

[2] M. Montis, M. Giacchini, S. Fantinel, M. Bellato. New data archive system for SPES project based on EPICS RDB Archiver with PostgreSQL backend. *Proccedings of PCaPAC*, strony 188–190, Karlsruhe, Germany, 2014.

[3] M. Bickley, Ch. Slominski. A MySQL-based data archiver: preliminary results. *Proceedings of ICALEPCS07*, Październik 2007. `http://www.osti.gov/scitech/servlets/purl/922267` [dostęp dnia 20 czerwca 2015].

ROZDZIAŁ 4

GENEROWANIE OFERT BIZNESOWYCH W SYSTEMIE REKOMENDACYJNYM

H. Sienkiewicz

Coraz częściej małe i duże przedsiębiorstwa w swej działalności biznesowej wykorzystują komputerowe metody i narzędzia szeroko rozumianej analizy danych. W niniejszym rozdziale przedstawiono metody eksploracji, jakie można wykorzystać w celu zwiększenia sprzedaży czy rozszerzenia rynku.

4.1. Wprowadzenie

Pozyskanie nowych klientów nie jest jedynym środkiem prowadzącym do zwiększenia sprzedaży. Ten sam efekt można uzyskać poprzez zainicjowanie nowych transakcji wśród stałych klientów. Istnieją metody wspierające takie działania: *up–sell*, *cross–sell* czy *deep–sell* [1]. Pierwsza z nich polega na proponowaniu klientowi lepszego, droższego produktu. Zadaniem cross–sell jest generowanie ofert, które wiążą się z już nabytymi przez klienta produktami czy usługami. Deep–sell ma na celu dotarcie do większej grupy pracowników z danej firmy, czyli rozreklamowanie się w grupie potencjalnych klientów, z którymi przeprowadzano już transakcje (np. poprzez polecenie firmy sprzedażowej przez zadowolonego klienta z działu X dla innych potencjalnych klientów z działu Y).

Analiza zebranych informacji o klientach i kupionych przez nich produktach pozwala znaleźć związki między klientami i produktami, jak również między samymi klientami czy samymi produktami. Przetwarzanie pozyskanych danych statystycznych, ich eksploracja czy drążenie może prowadzić do rozpoznania pewnych zależności i wzorców oraz do wysnuwania wniosków co do nowych i wydajniejszych strategii biznesowych. Jest to często wspierane komputerowo przez narzędzia statystyczne oraz metody sztucznej inteligencji i może służyć: analizie wpływu zmian cen na ilość sprzedaży; rozpoznawaniu zaniżonych cen; ocenie rentowności klientów oraz doborowi strategii dla klientów niedochodowych; opracowaniu propozycji zmian asortymentu (usunięcie produktów nieprzynoszących zysków, zastąpienie wycofanych produktów); dostosowaniu ofert do potrzeb odbiorców.

4.2. Charakterystyka systemów rekomendacyjnych

Systemy rekomendacyjne często służą do generowania spersonalizowanej oferty sprzedaży na bazie analizy profili klientów. Jako przykład posłużyć może sieć marketów *Castorama*, która prosi swoich klientów o podanie kodu pocztowego, aby zebrać informację o ich pochodzeniu. Dzięki tej wiedzy przedsiębiorstwo może jak najlepiej dostosować zaopatrzenie sklepów, biorąc pod uwagę zapotrzebowanie kupujących.

4.2.1. Profil klienta

Profil klienta powinien zawierać jak najwięcej istotnych danych. Standardowo umieszcza się w nim atrybuty charakteryzujące kupującego: adres zamieszkania, dane sprzedażowe oraz dane dotyczące sprzedanych ofert. Na podstawie profilu można badać rentowność danego klienta, a z danych o sprzedaży wywnioskować, na jakie produkty kupującego „stać" oraz ocenić rozwój jego siły nabywczej. Ujmując to ogólniej: działanie systemu rekomendacyjnego z wykorzystaniem profilu użytkownika może polegać między innymi na [2]: filtrowaniu informacji, wyznaczeniu podobieństwa między konsumentami, zarekomendowaniu produktu.

4.2.2. Filtrowanie informacji

Filtrowanie informacji powinno dostarczyć odpowiedź na pytanie, jakie cechy klienta są najistotniejsze z biznesowego punktu widzenia, a jakie nie wnoszą żadnej informacji. Do metod filtracji zalicza się [2]:

- metodę demograficzną,
- metodę kolaboratywną lub jej modyfikację bazującą na związkach między produktami,
- metodę opartą na zawartości,
- metodę hybrydową,
- metodę bazującą na przypadkach użycia.

Filtrowanie demograficzne

Dane demograficzne mogą bezpośredniego posłużyć do znalezienia związku między produktami a ich nabywcami (jak w przykładzie sieci sklepów *Castorama*). Ten sposób filtrowania jest nieskomplikowany. Może jednak nie doprowadzić do wygenerowania spersonalizowanych ofert. Dlatego przy budowie systemów rekomentacji należy łączyć różne metody filtrowania, aby rekomendacje nie opierały się tylko na sterotypowych informacjach.

Filtrowanie kolaboratywne

Podczas filtrowania tego typu tworzone są pewne grupy klientów i rekomendacje produktów, które były kupowane przez innych odbiorców z danej grupy. Ta metoda opiera się na podobieństwach między konsumentami, które najlepiej wyznaczać w systemach z dużą liczbą użytkowników. Bowiem im większa liczba

konsumentów, tym większa szansa na poprawne dopasowanie kolejnego klienta do istniejących grup. W algorytmie takiej filtracji można wyróżnić następujące kroki [3]:

1. Obliczenie podobieństwa między użytkownikami.
2. Utworzenie grup i podział konsumentów.
3. Wybranie produktów kupowanych najczęściej przez klientów z danej grupy.
4. Ewentualne zaoferowanie produktów z kroku trzeciego klientom z danej grupy, którzy ich nie kupili.

Jest to nieskomplikowana metoda, niewymagająca specyficznej wiedzy o oferowanych produktach. W algorytmie należy zwrócić baczniejszą uwagę na krok trzeci. Wybór produktów powinien odbywać się według odpowiednio sprecyzowanej reguły. Należy wziąć pod uwagę to, że procedura ta nie wygeneruje listy produktów kupowanych cyklicznie, co na pewno może zmniejszyć zysk ze sprzedaży. Ponadto automatycznie generowane oferty dóbr mogą nie adaptować się do odrębnych oczekiwań pewnych klientów. Dlatego przy wyznaczaniu reguły można wykorzystać metody opierające się na podobieństwach między produktami. W tym algorytmie należy zbudować macierz M częstości występowania par produktów. Dla danego klienta C, który dokonał już zakupu, można wyróżnić następujące kroki [3]:

1. Wybranie z macierzy M najczęściej występujących produktów z każdym produktem kupionym przez klienta C, pomijając już zrealizowane transakcje.
2. Wybranie z produktów z kroku pierwszego elementów, które najczęściej występują z produktami kupionymi przez klienta C.

Należy wybrać te produkty, które były kupowane razem z produktami klienta C, następnie należy wyodrębnić te dobra, które były najczęściej kupowane z kupionymi już produktami. Na przykład produkt A może być prawie zawsze kupowany razem z produktem B oraz D, przy czym produkt B może być zawsze kupowany razem z produktem A, zaś produkt D przez pewnych klientów był kupowany osobno.

Filtrowanie informacji oparte na zawartości

W metodzie tej korzysta się z danych sprzedażowych, czyli z historii zakupów danego konsumenta. Generowanie oferty opiera się na wyborach dokonywanych przez pojedynczego klienta i przez to może być ukierunkowane tylko na wąską grupę produktów.

Metoda filtracji bazująca na przypadkach użycia

Metoda ta wykorzystuje historię użycia systemu rekomendacyjnego [4]. Jej działanie polega na: zapamiętywaniu ofert wygenerowanych przez system pozytywnie odebranych i sfinalizowanych oraz ofert, które nie przyniosły pozytywnego skutku; konstruowaniu i zastosowaniu pewnych heurystyki dla nowych bądź stałych klientów.

Metoda hybrydowa

Podejście hybrydowe pozwala połączyć wyżej wymienione metody w dowolnych, uzupełniających się kombinacjach. Na przykład można skorzystać z danych demograficznych i wstępnie zgrupować klientów, by później zastosować metodę bazującą na podobieństwach między produktami.

4.2.3. Związki między klientami a produktami

Metody filtracji bazujące na danych demograficznych, zawartości czy przypadkach użycia polegają na znalezieniu relacji między klientem a produktem. Rozpatrując przypadki indywidualnie względem klienta, dany produkt może zostać zakwalifikowany jako produkt polecany lub niepolecany. Do technik, które na to pozwalają, zalicza się m.in.: klasyfikatory i metody analizy skupień, drzewa decyzyjne, wnioskowanie rozmyte, sieci neuronowe.

Istnieje wiele narzędzi umożliwiających przeanalizowanie opłacalności zarekomendowania danego produktu. Omówione niżej wybrane metody pozwalają oszacować skłonności klientów do: podjęcia szerszych/dodatkowych zakupów, zdecydowania się na kupno droższych produktów. Pozwalają też zwiększyć sprzedaż oferowanych artykułów.

Drzewa decyzyjne

Metoda ta polega na podziale zbioru na homogeniczne grupy, najlepiej w jak najmniejszej liczbie kroków. Budowa takiego drzewa podziału następuje od korzenia (ang. *root*), gdzie znajdują się wszystkie zgromadzone dane. Wybór atrybutu do sklasyfikowania danych i ich podziału, czyli do budowy gałęzi drzewa decyzyjnego, można oprzeć na miarze entriopii H pozyskanej informacji:

- entropia Shannona – jest to średnia ważona ilości informacji niesionej przez pojedynczą wiadomość (gdy jednostką entropii jest bit):

$$-\sum_i P(x_k)\log_2 P(x_k), \tag{4.1}$$

  gdzie $P(x_k)$ to prawdopodobieństwo wystąpienia x_k.
- miara Gini'ego – jest to suma iloczynów liczebności danej klasy w stosunku do liczebności wszystkich klas wyrażająca się wzorem:

$$\sum_{i,j} \frac{\#k_i \#k_j}{(\#K)^2},\ i \neq j. \tag{4.2}$$

  Wzór ten można zastosować, jeżeli prawdopodobieństwo *a priori* przynależności do klasy k_i liczone jest jako stosunek liczności obserwacji klasy k_i do liczności wszystkich próbek $\frac{\#k_i}{\#K}$.
- testy statystyczne, np. test χ^2, test ilorazu wiarogodności (test G).

Ostatnim członem drzewa są liście, w których znajdują się podgrupy danych. Zakończenie budowy drzewa może nastąpić, gdy wszystkie obserwacje zostały

prawidłowo sklasyfikowane (co jest raczej nierealistyczne). Można zatem określić końcową maksymalną liczbę obserwacji mającą znaleźć się w liściu drzewa. Miarą poprawności klasyfikacji może być liczba przypadków błędnie zaklasyfikowanych w stosunku do liczby wszystkich przypadków, a dla zmiennej ciągłej może to być błąd średnio–kwadratowy predykcji.

Naiwny klasyfikator Bayesa (NBC)

Metoda ta opiera się na regule Bayesa. Mając zbiór danych można określić tzw. prawdopodobieństwo *a priori* $P(k_j)$ przynależności do danej klasy k_j. Następnie określana jest szansa przynależności obserwacji do danej klasy, czyli prawdopodobieństwo warunkowe $\prod_k P(x_k|k_j)$. Zakłada się, że prawdopodobieństwa te są niezależne. W efekcie końcowym obliczane jest prawdopodobieństwo *a posteriori* dane wzorem

$$P(k_j|X) = P(k_j)\prod_k P(x_k|k_j). \tag{4.3}$$

Nowa obserwacja zakwalifikowana jest do tej klasy, dla której wartość prawdopodobieństwa *a posteriori* jest największa.

Metoda najbliższego sąsiada

Metoda ta wykorzystuje wiedzę o produktach przechowywanych w bazie danych. Na podstawie informacji o cenie czy kategorii produktu można wyznaczyć podobieństwa między ofertami. Wybrane funkcje odległości oraz podobieństwa zostały wyczerpująco opisane w pracy [2]. Jedną z najbardziej popularnych funkcji podobieństwa jest cosinusowa funkcja podobieństwa.

4.2.4. Związki między klientami

Filtrowanie kolaboratywne wykorzystuje ideę grupy klientów, na podstawie której może wygenerować rekomendowane produkty. Wyznaczanie podobieństwa między konsumentami może odbywać się poprzez klasyfikację, grupowanie czy metodę najbliższych sąsiadów. Do oceny podobieństwa między profilami są wspomniane wcześniej funkcje odległości.

Ważnym etapem jest utworzenie sąsiedztwa. W skrajnym przypadku do sąsiedztwa mogą zaliczać się wszyscy konsumenci. Jednak najczęściej stosuje się metodę najlepszych sąsiadów lub progu korelacji. Pierwsza z nich gwarantuje stworzenie sąsiedztwa, lecz może ono być niemiarodajne. Nieodfiltrowanie dalszych sąsiadów może mieć znaczący wpływ na wygenerowane oferty. Druga z metod tworzenia sąsiedztwa bazuje na stwierdzeniu, który klient znajduje się w sąsiedztwie innego klienta. Polega ona na określeniu progu korelacji i może spowodować nieutworzenie sąsiedztwa dla jakiegoś klienta, co nie pozwoli na wygenerowanie ofert. Po utworzeniu sąsiedztwa należy ocenić, jakie rekomendacje przyniosą zamierzony efekt. Mając klientów z danego sąsiedztwa

oraz historię ich zakupów można dokonać predykcji, czy dany konsument zareaguje pozytywnie na rekomendowany produkt. Predykcja taka może być wyznaczona na podstawie stosunku sfinalizowanych ofert do ofert odrzuconych. Można również wyliczyć cenę jaka ma być zaoferowana za dany produkt, wykorzystując przy tym medianę, średnią i średnią ważoną cen.

4.3. Przykład analizy danych

Dane gromadzone w bazach danych mogą wymagać wstępnego przetworzenia z uwagi na występujące w nich braki lub sprzeczności. Należy również liczyć się z takimi aspektami przetwarzania jak przeuczenie algorytmów lub ich niedouczenie. Obserwacje z pewnymi brakującymi informacjami, podobnie jak przypadki danych sprzecznych, można po prostu nie włączać do analiz.

4.3.1. Opis danych

Dane eksperymentalne zostały wygenerowane i przeanalizowanie w środowisku *R*. Paczka danych składa się z trzech następujących części:

- dane demograficzne klientów
  - unikalne numery identyfikacyjne klientów,
  - miejscowość (zakres wartości: 1, 2, 3, 4),
  - branża (zakres wartości: 1, 2, 3, 4, 5);
- dane dotyczące produktów
  - unikalne numery identyfikacyjne produktów,
  - cykl życia produktu (zakres wartości: nowy, dostępny, wycofany),
  - kategoria produktu (zakres wartości: $1, \ldots, 100$);
- historia zakupów
  - unikalne numery identyfikacyjne klientów,
  - unikalne numery identyfikacyjne produktów,
  - liczba sztuk zakupionych produktów,
  - cena jednostkowa.

4.3.2. Przygotowanie danych do analizy

Na podstawie wygenerowanych danych i późniejszych analiz zdecydowano się na sformułowanie następujących pojęć:

- funkcja wyrażająca podobieństwo kategorii produktów

$$f_1(\mathit{kategoriaX}, \mathit{kategoriaY}) = \frac{\#C_A}{\#C_O}, \tag{4.4}$$

  gdzie $\#C_A$ to liczba klientów, którzy kupili produkty z kategorii *kategoriaX* oraz *kategoriaY*, a $\#C_A$ to liczba klientów, którzy kupili produkty z kategorii *kategoriaX* lub *kategoriaY*,

- funkcja wyrażająca żywotność produktów

$$f_2(p_{id}) = \frac{\#C}{\#P}, \tag{4.5}$$

gdzie $\#C$ to liczba unikalnych klientów, którzy kupili produkt p_{id}, a $\#P$ to łączna liczba sprzedaży produktu p_{id},
- średnia cena produktów – liczona w grupach klientów podzielonych na miejscowości oraz branże

$$f_3(p_{id}, \textit{miejscowość}, \textit{branża}) = \frac{1}{n}\sum_{i=1}^{n} cena_i, \tag{4.6}$$

gdzie n to liczba zakupów produktu p_{id} dokonanych przez klientów pochodzących z miejscowości *miejscowość* oraz zajmujących się branżą *branża*,
- średnia liczba sztuk zakupionych produktów – liczona w grupach klientów podzielonych na miejscowości oraz branże

$$f_4(p_{id}, \textit{miejscowość}, \textit{branża}) = \frac{1}{n}\sum_{i=1}^{n} q_i, \tag{4.7}$$

gdzie q_i to liczba kupionych sztuk produktu p_{id}, a n to liczba zakupów produktu p_{id} dokonanych przez klientów pochodzących z miejscowości *miejscowość* oraz zajmujących się branżą *branża*,
- funkcja wyrażająca rentowność klienta

$$f_5(c_{id}) = \frac{s_1}{s_2}, \tag{4.8}$$

gdzie s_1 to suma średnich cen produktów, które kupił klient c_{id}, a s_2 to suma średnich cen produktów, które kupił konsument c_{id} liczona w grupie wszystkich klientów, którzy kupili produkty kupione przez danego klienta c_{id}.

4.3.3. Zastąpienie wycofanych produktów

Generowanie strategii biznesowych mających na celu zastąpienie wycofanych produktów i tym samych zaoferowanie ich klientom, dotychczas kupujących wycofane oferty, można zaliczyć do metod up–sell. Wycofane produkty mogły okazać się niedochodowe lub z jakichś względów przedsiębiorstwo musiało zrezygnować z ich produkcji, np. przejmując inną firmę zdecydowało się na zaprzestanie oferowania danej usługi. Pozyskanie nowego klienta jest relatywnie droższe od utrzymania stałego konsumenta, dlatego z pewnością opłaca się zaproponować ofertę zastępczą.

Algorytm

Poniższe przedstawiono kroki algorytm wyszukiwania potencjalnych produktów, które mogą zastąpić wycofany produkt.

Krok 1 Wybranie wycofanego produktu oraz jego nabywcy.

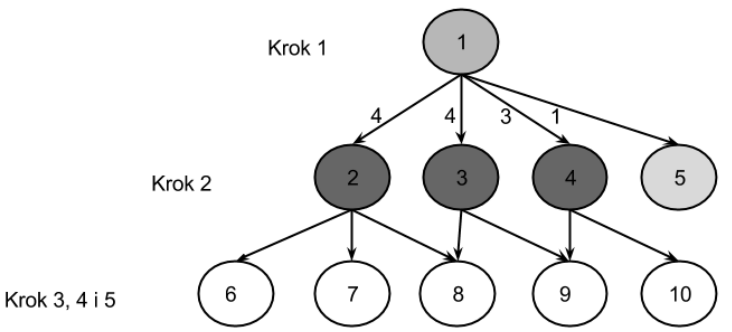
Krok 2 Wyszukanie produktów, które były najczęściej kupowane razem z wycofanym produktem przez tych samych klientów (pary produktów).

Krok 3 Wyszukanie produktów kupowanych przez klientów z tej samej miejscowości i branży co klient z kroku pierwszego, który kupił produkty z kroku drugiego.

Krok 4 Wyeliminowanie produktów z kroku trzeciego, które były kupowane razem z produktem wycofanym.

Krok 5 Wybranie produktów najbardziej podobnych do wycofanego.

Na rysunku 4.1 przedstawiono działanie algorytmu generowania zastępczych ofert. Można na nim dostrzec wybrany wycofany produkt (1), znalezione pary produktów ([1,2],[1,3],[1,4],[1,5]) oraz potencjalne zastępcze produkty (6, 7, 8, 9, 10). Liczby nad strzałkami oznaczają ile razy dana para była kupiona przez unikalnych klientów.



Rys. 4.1: Przykład działania algorytmu generowania zastępczych ofert

Ostatnim krokiem algorytmu jest wybranie potencjalnych zastępczych ofert. Zrealizowanie tego celu wymaga zastosowanie pewnej selekcji, która została dobrana drogą eksperymentalną. Poniżej przedstawiono kryterium wyboru takiego produktu:

- funkcja wyrażająca podobieństwo kategorii produktów powinna być stosunkowo mała
$$f_1(k_x, k_y) < l, \tag{4.9}$$
gdzie k_x to kategoria wycofanego produktu, k_y to kategoria potencjalnego zastępczo produktu, a l to pewna mała stała,
- różnica wartości funkcji wyrażającej średnią cenę produktów również powinna być stosunkowo mała
$$|f_3(p_{id}, \textit{miejscowość}, \textit{branża}) - f_3(P_{id}, \textit{miejscowość}, \textit{branża})| \leq g, \tag{4.10}$$

gdzie p_{id} to numer wycofanego produktu, P_{id} to numer potencjalnego zastępczego produktu, *miejscowość* oraz *branża* to miejscowość i branża klienta wybranego w kroku pierwszym algorytmu, a g to pewna stała zależna od ceny produktu wycofanego,

- różnica wartości funkcji wyrażającej żywotność produktów również powinna być stosunkowo mała

$$|f_2(p_{id}) - f_2(P_{id})| \leq h, \tag{4.11}$$

gdzie h to pewna mała stała,

- ewentualnie brana pod uwagę średnia ilość sztuk zakupionego produktu.

Główną zasadą tego algorytmu jest odnalezienie w pewnym zbiorze klientów produktów, które były kupowane jak najrzadziej przez tych samych klientów. Bowiem istnieje wtedy szansa, że produkty te są do siebie podobne, a co za tym idzie, że można używać ich zamiennie.

Przykład działania algorytmu

Do uruchomiania algorytmu należy wczytać dane z plików `.csv`. Następnie uruchomić część kodu odpowiedzialną za zastąpienie wycofanego produktu. W celu ilustracji działania algorytmu wybrano wycofany produkt o *product_id* = 10 i o następujących parametrach: liczba sztuk = 6, $f_3 = 52.64$, $f_2 = 1$. Jako, że produkt ten został kupiony tylko raz może okazać się, że algorytm wygeneruje niedopasowaną ofertę. W tabeli 4.1 pokazano częściowe wyniki algorytmu z kroków 1–4 (dla przypadku, gdy dany produkt był kupiony tylko raz, przez jednego, unikalnego klienta – stąd same jedynki w kolumnie „Wspólni klienci"). Można zauważyć, że są to dość zróżnicowane produkty pod względem ceny czy wartości funkcji f_2. Jednakże stosując dla tych danych krok 5 algorytmu daje się zauważyć podobieństwo wybranych produktów do wycofanego. W tabeli 4.2 przedstawiono listę potencjalnych produktów, które mogą zastąpić badany, wycofany produkt.

4.4. Podsumowanie

Identyfikowanie związków w analizowanych danych i opracowywanie strategii sprzedaży stało się kluczowym zagadnieniem podejmowanym w świecie biznesu. Już dziś istnieje szereg metod i narzędzi pozwalających na sprawną eksplorację danych w każdej dziedzinie gospodarki, począwszy od firm windykacyjnych aż po przedsiębiorstwa produkujące silniki. W niniejszym rozdziale scharakteryzowano krótko cechy systemów rekomendacyjnych oraz przedstawiono przykład analizy danych ukierunkowanej na wygenerowanie ofert biznesowych.

Informacja jest obecnie jednym z najistotniejszych i najdroższych dóbr świata. Jej gromadzeniem zajmują się firmy–giganty takie, jak *Facebook*, *Google* czy *Yahoo*. Odpowiednio przeanalizowana informacja posłużyć może do uzyskania znacznych zysków finansowych. Jest to chyba niezły argument, aby zainteresować się głębiej tym tematem.

Tab. 4.1: Wyniki z kroków 1–4 algorytmu

Nr produktu	Wspólni klienci	Kategoria	f_1	f_3	f_4	f_2
11264	1	104	0.09	18.9	3	1
12681	1	37	0.07	1.78	30	0.56
12701	1	37	0.07	3.18	15	0.52
13446	1	37	0.07	74.55	6	0.5
13498	1	37	0.07	7.85	3	0.6
13508	1	37	0.07	7.78	30	0.54
13511	1	37	0.07	14.27	9	1
13525	1	37	0.07	7.92	15	0.55
13672	1	37	0.07	13.57	4	0.56
13704	1	37	0.07	13.75	7	0.47
13774	1	37	0.07	20.07	3	0.57
13804	1	37	0.07	22.57	3	0.58
13815	1	37	0.07	22.44	3	0.63
13821	1	37	0.07	23.93	12	0.72
13840	1	37	0.07	24.36	3	0.75
13908	1	37	0.07	31.23	15	0.70
13935	1	37	0.07	41.43	6	0.86
14000	1	37	0.07	52.22	3	0.62
14064	1	37	0.07	63.35	36	0.81
14075	1	37	0.07	74.25	3	0.73
14251	1	37	0.07	5.27	6	0.9
14293	1	37	0.07	5.86	12	0.68
14344	1	37	0.07	4.68	67.5	0.5
14448	1	37	0.07	61.49	6	0.7
14466	1	37	0.07	107.41	3	1
201	1	5	0.16	32.43	6	0.66
202	1	5	0.16	35.40	3	0.77
23370	1	56	0.06	43.39	3	0.74
23377	1	56	0.06	31.52	18	0.83

Tab. 4.2: Wyniki z kroku 5 algorytmu

Nr produktu	Wspólni klienci	Kategoria	f_1	f_3	f_4	f_2
1293	1	37	0.07	51.74	6	0.73
1390	1	37	0.07	41.42	6	0.86
1399	1	37	0.07	52.21	3	0.62
1524	1	37	0.07	46.51	3	1.0
1732	1	1	0.03	56.28	3	0.49
2336	1	56	0.06	43.37	3	0.74

Literatura

[1] K. Tsiptsis, A. Chorianopoulos. *Data mining techniques in CRM: Inside Customer Segmentation.* John Wiley & Sons, 2009.

[2] J. Sobecki. *Rekomendacja interfejsu użytkownika w adaptacyjnych webowych systemach informacyjnych.* Oficyna Wydawnicza Politechniki Wrocławskiej, 2009.

[3] J. O'Donovan, J. Donnion. Adaptive recommendation: Putting the best foot forward. *International Symposium on Information and Communication Technologies*, 2004.

[4] A. Michalczyk. *Metoda case-based reasoning w zarządzaniu wiedzą i doświadczeniem w firmach windykacyjnych.* Wydawnictwo uczelniane Wyższej Szkoły Informatyki i Ekonomii TWP, 2010.

ROZDZIAŁ 5

INTERFEJSY WYSZUKIWANIA

K. Nowosad, M. Stachowski

W niniejszym rozdziale opisano najważniejsze cechy interfejsów użytkownika oferowanych przez usługi wyszukiwania. Dokonano w nim również przeglądu interfejsów najpopularniejszych usług oraz przedstawiono projekt rozwiązania służącego do obsługi gestów podczas wyszukiwania i uruchamiania aplikacji.

5.1. Wprowadzenie

W dobie Internetu i powszechnej komputeryzacji publikowanie danych w postaci cyfrowej stało się wyjątkowo łatwe, co szybko przełożyło się na lawinowy przyrost wolumenu udostępnianych danych. Jednak łatwość publikacji stała się źródłem pewnych problemów. Publikowane dane mogą dotyczyć zarówno spraw istotnych, jak i błahych. Z tego powodu wyszukiwanie konkretnej informacji zaczyna się mocno komplikować. Aby wspomóc ludzi w realizacji tego zadania zaczęto uskuteczniać metody porządkowania i przeszukiwania zasobów. Powstały usługi wyszukiwania, działające w najróżniejszych kontekstach i sferach.

Korzystanie z usług wyszukiwania nie zawsze jest proste i oczywiste. Z punktu widzenia użytkownika problemem może być nie tylko właściwe sformułowanie zapytania, ale również właściwa interpretacja dostarczanych odpowiedzi. Często zdarza się, że wyników wyszukiwania jest zbyt dużo, że są niewłaściwie posortowane i niekoniecznie ściśle związane z zapytaniem. Dlatego dostawcy wyszukiwarek zaczęli zwracać uwagę na projektowanie takich interfejsów, które ułatwią i przyśpieszą cały ten proces.

Projektując interfejs wyszukiwania trzeba uwzględnić wiele aspektów, jak np.: cel wyszukiwania, rodzaj wyszukiwanego zasobu, platformę klienta (np. wyszukiwarka web, aplikacja desktopowa, urządzenie mobilne) i inne. Jednak pomimo szerokiego spektrum zastosowań w projektowanych interfejsach wyszukiwania można wyróżnić trzy podstawowe pola [1]: **wyszukiwania**, **wyników**, **filtrów**.

Pole wyszukiwania jest pierwszym rzucającym się w oczy elementem interfejsu (rys. 5.1). Może ono przyjmować różne postacie, kształty, rozmiary i kolory, może być funkcją witryn, przeglądarek internetowych, aplikacji, widgetów i systemów operacyjnych.

Rys. 5.1: Pole wyszukiwania, od lewej: Google Web, Google Mobile Voice oraz Multitouch

Mówiąc o polu wyszukiwania należy zwrócić uwagę na różnorodne sposoby wprowadzania danych. Klasycznym sposobem jest wprowadzanie tekstu, słów kluczowych czy tagów za pomocą standardowych urządzeń wejścia/wyjścia (klawiatura, monitor). Inną możliwością jest sterowanie wyszukiwarką głosem (Google Mobile) i oczekiwanie, że pytanie zostanie dobrze zinterpretowane. W dziedzinie urządzeń mobilnych z ekranami typu multitouch wykorzystuje się również gesty, które są odpowiednio interpretowane.

Pole wyników jest miejscem na wyniki przeszukiwania. Ich wyświetlanie powinno być dostosowane do odbiorcy. W przypadku poszukiwania obrazu wyniki można prezentować w postaci kafli, zaś w przypadku poszukiwania zasobów tekstowych wyniki mogą przyjąć formę listy zawierającej fragmenty znalezionych rekordów, z podkreśleniem wystąpień słów kluczowych cienką linią. Zabieg ten zastosowano w wyszukiwarce google, dzięki czemu pozwala ona na prezentację dużej liczby wyników na jednej stronie. W urządzeniach wykorzystujących gesty do wyszukiwania często stosuję się animację w polu wyników. Przyciąga to użytkowników i uprzyjemnia im pracę.

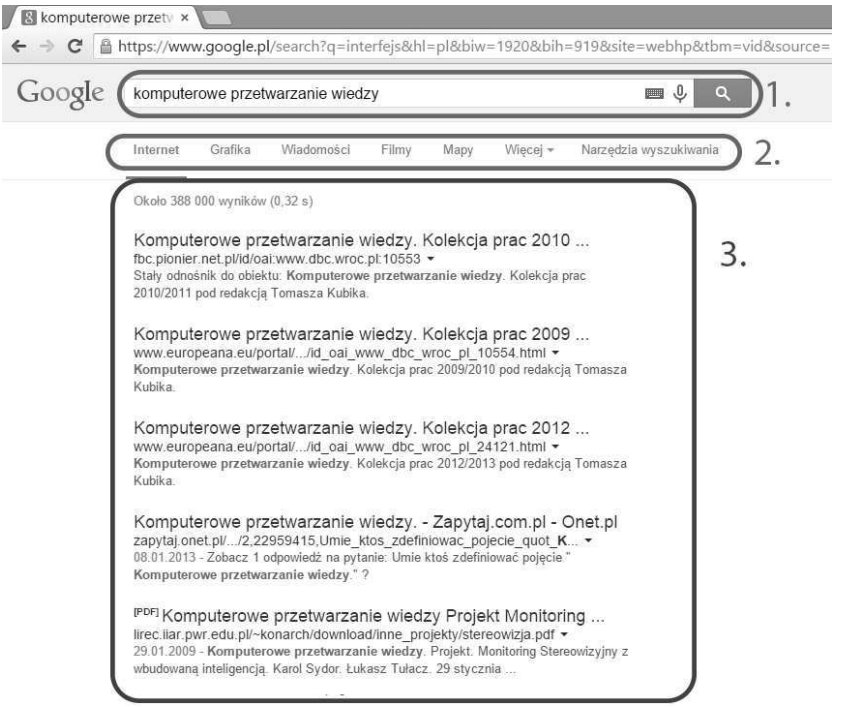
Pole filtrów jest stosowane w wyszukiwaniu zaawansowanym. Dzięki niemu można dokładniej sprecyzować warunki, jakie spełniać mają wyniki zapytania. Najczęściej używane są w sklepach internetowych, gdzie zawężanie zakresu zapytania i wyników wyszukiwania ze względu na kategorię, rodzaj produktu, cechy i cenę produktu itp. stało się swego rodzaju standardem.

5.2. Przykład interfejsów wyszukiwania w różnych dziedzinach

5.2.1. Interfejs wyszukiwarki google

Według rankingu „Gemius” wyszukiwarka google jest najpopularniejszą wyszukiwarką internetową. Jej interfejs (rys. 5.2) oraz podstawowe funkcje przedstawiają się następująco [2].

1. **Pole wyszukiwania** - w tym polu użytkownik wpisuję słowo kluczowe lub frazę. Google umożliwia też wprowadzenie frazy za pomocą głosu. Wyniki wyszukiwania pojawiają się po kliknięciu na ikonę z lupą lub naciśnięcie klawisza „Enter”.



Rys. 5.2: Interfejs wyszukiwarki google

2. Filtry - google dla wygody użytkownika udostępnia kilka filtrów tematycznych. Każdy z nich posiada dodatkowe narzędzia wyszukiwania.

Internet
- język (dowolny, język polski),
- czas dodania (kiedykolwiek, ostatnia godzina, ostatni dzień, tydzień, miesiąc, rok, zakres dat),
- dokładność (wszystkie wyniki, dokładna fraza),
- położenie.

Grafika
- rozmiar (dowolny, duże, średnie, ikony, dokładny),
- kolor (dowolny, kolorowe, czarno-białe, przeźroczyste, dominujący),
- typ (twarz, zdjęcie, obiekt clip art, grafika wektorowa, animowane),
- godzina (ostatni dzień, tydzień, zakres dat),
- prawa do użytkowania.

Wiadomości
- szukaj w Internecie (tylko język ojczysty),
- wszystkie wiadomości (blogi),
- czas dodania (kiedykolwiek, ostatnia godzina, ostatni dzień, tydzień, miesiąc, rok, zakres dat),
- sortowanie (według trafności, według daty).

Filmy – szukaj w Internecie (tylko język ojczysty),
– długość (dowolna, krótki, średni, długi),
– czas dodania (kiedykolwiek, ostatnia godzina, ostatni dzień, tydzień, miesiąc, rok, zakres dat),
– jakość (wysoka, dowolna),
– z napisami,
– źródło.

Mapy – komponent mapowy, planowanie trasy.

Książki – szukaj w Internecie (tylko język ojczysty),
– dostępność (dowolne, dostępny podgląd, elektroniczna książka google, bezpłatna elektroniczna książka google),
– rodzaj (dowolny, czasopisma, książki),
– czas dodania (XXI wiek, XX wiek, XIX wiek, zakres dat),
– sortowanie (według trafności, według daty).

Loty – wyszukiwanie lotów (kierunek, cena, data, linia, klasa).

Aplikacje – szukaj w Internecie (tylko język ojczysty),
– cena (dowolna, bezpłatne, płatne).

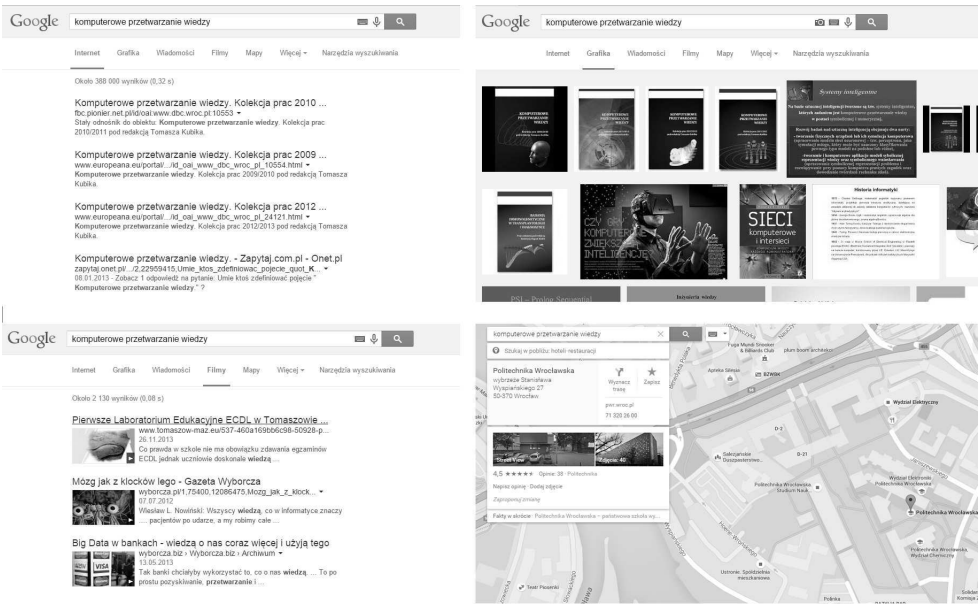
Oczywiście nie są to wszystkie filtry z interfejsu wyszukiwarki google. Stanowią one jedynie listę filtrów „podręcznych". Liczbę wykorzystanych filtrów można zwiększyć poprzez zastosowanie opcji „ukrytych", definiowanych razem z frazą do wyszukania. Przykładowo:

site:*adres domeny* – wyszukuję frazę na danej stronie,
filetyp:*typ pliku* – wyszukuję konkretny typ pliku, np. pdf, doc,
„fraza" – pozwala na dokładne wyszukiwanie frazy,
define:*fraza* – wyszuka definicję frazy.

3. Pole wyników - prezentacja tego pola różni się w zależności od zastosowanego filtru (rys. 5.3). Może to być lista, lista z miniaturą, kafelki, mapy itp.

5.2.2. Pozycjonowanie

Z działaniem stron internetowych często wiąże się realizacja jakichś celów biznesowych. Aby sprostać konkurencji firmy muszą zabiegać o użytkowników i potencjalnych klientów. Dlatego starają się, aby po wpisaniu w wyszukiwarce frazy to właśnie ich strona pojawiła się na samym początku wyników wyszukiwania lub w innym, równie dobrze widocznym miejscu. Sama wyszukiwarka google, po uiszczeniu opłat i podpisaniu umowy z jej dostawcą, oferuje możliwość wyświetlania reklam z boku lub na początku wyników (wszystko zależy od tego, czego szuka użytkownik). Aby wyniki były zaprezentowane w bardziej czytelny sposób, często towarzyszą im odpowiednie miniatury lub fragmenty tekstów. W kolejnych podrozdziałach przybliżono stosowane przy tej okazji metody.



Rys. 5.3: Pole wyników odpowiednio dla filtrów: Internet, Grafika, Film, Mapy

5.2.3. Tagi

W pierwszych wersjach HTML jedynym miejscem przewidzianym na zamieszczanie metadanych charakteryzujących daną stronę internetową była jej stopka. To w niej zamieszczano dane o autorze, prawach autorskich oraz słowa kluczowe. Należy przyznać, że przy taka skąpym zasobie informacji nawet obecnie stosowane algorytmy i systemy nie byłyby w stanie właściwie odfiltrować i pogrupować wyników wyszukiwania. Należało poszukać lepszego rozwiązania. Tym rozwiązaniem okazało się użycie tagów (znaczników) – elementów metadanych zanurzane w kodzie źródłowym portali.

Rozwój technik wykorzystujących tagi wiąże się z rozwojem technologii internetowych, a w szczególności języków znaczników (XML, XML Schema – ustandaryzowane przez W3C) i modeli danych strukturalnych (mikrodane i mikroformaty). Dzięki tym technologiom udało się zautomatyzować pozyskiwanie danych bezpośrednio ze stron portali. Tagi i dane strukturalne ułatwiają algorytmom wyszukiwarek tworzenie map przeglądanych stron, ich kategoryzowanie oraz wiązanie z innymi elementami. Dzięki nim łatwiejsze stało się tworzenie indeksów stron na cele wyszukiwania.

5.2.4. Podgląd wyników

Podczas wyprowadzania wyników wyszukiwania standardem stało się zamieszczanie krótkiego streszczenia najważniejszych informacji powiązanych z danym rekordem (w formie tekstu bądź zdjęcia) umieszczanych z prawej strony. Można to zaobserwować wpisując w pole wyszukiwania frazę: „Wrocław”. Po wy-

słaniu zapytania w odpowiedzi pojawi się zdjęcie wrocławskiego rynku, fragment mapy Polski z zaznaczonym miastem, a pod spodem takie dane, jak: powierzchnia, liczba ludności czy nawet aktualna pogoda. Również pierwsze wyniki będą odnosić do strony z Wikipedii, oficjalnej strony miasta, a także do lokalnych wiadomości wraz z miniaturami i datami (rys. 5.4). Te dodatkowe treści zostały pozyskane z danych strukturalnych (są też wynikiem inteligentnej analizy źródeł).

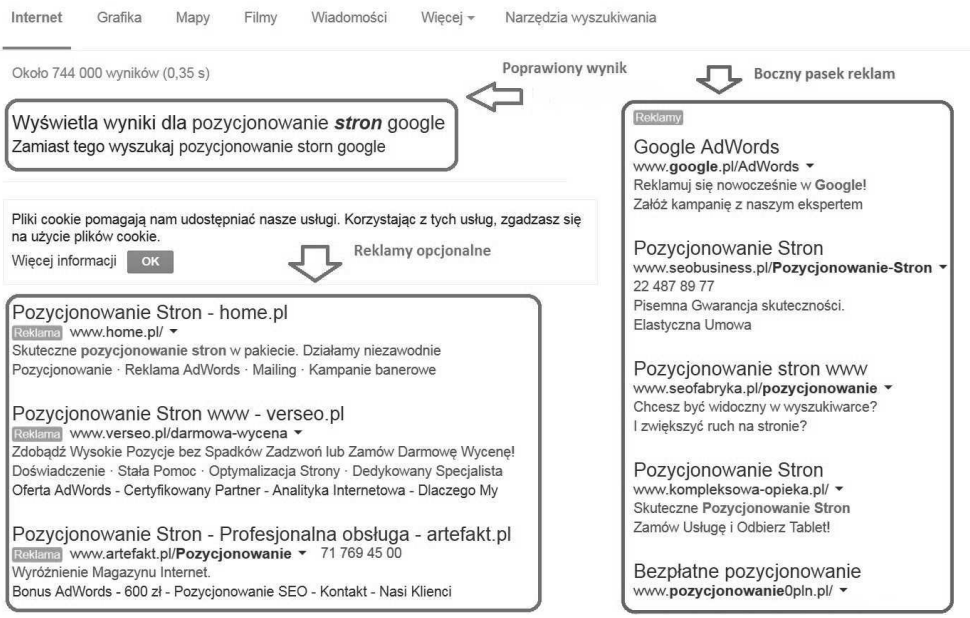
Rys. 5.4: Pasek boczny Google

5.2.5. Adwords

Pod pojęciem „Adwords” rozumiemy reklamy, które w dość sprytny i specyficzny sposób „wtapiają się” w wyniki wyszukiwania. Oczywiście również w ich przypadku dobór wyświetlanych treści, jej forma i miejsce wystąpienia nie są losowe. Są one tak dobierane przez wyszukiwarki, aby jak najbardziej trafiać do gustów klienta oraz aby zapewnić realizację zleceń reklamodawców (rys. 5.5). Obrazując to na przykładzie: wpisując w wyszukiwarkę google hasła z nazwami sprzętu elektronicznego otrzymamy w wynikach referencje do sklepów RTV oferujących ten czy inny produkt.

5.3. Urządzenia fizyczne jako interfejsy wspomagające wyszukiwanie

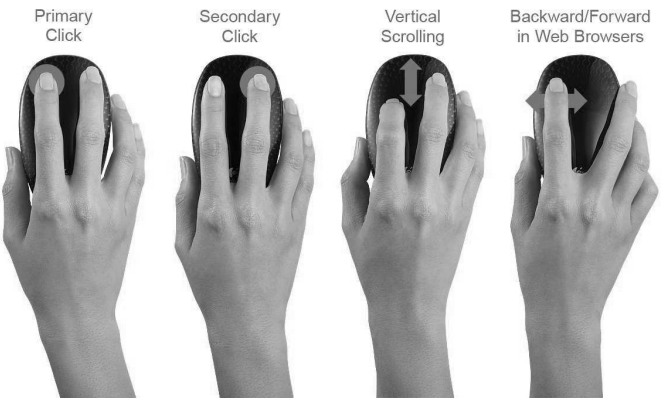
Intensywny rozwój interfejsów wyszukiwania spowodował, że rozpoczęto budować różne urządzenia fizyczne wspomagające ich obsługę. W tym podrozdziale zostanie opisanych kilka z nich.



Rys. 5.5: Sposób wyświetlania reklam w Google

5.3.1. Mysz Logitech T620

Mysz Logitech T620 jest myszą typu multitouch [3]. Jest wyposażona w laserowy sensor, który odczytuje ruchy ręki i następnie informacje o tych ruchach przesyła bezprzewodowo do komputera (rys. 5.6). Analiza tych ruchów za pomocą algorytmów po stronie komputera pozwala na wywołanie odpowiedniej reakcji. I tak dzięki rozpoznawaniu prostych i intuicyjnych gestów można bły-



Rys. 5.6: Mysz Logitech T620

skawicznie nawigować i wykonywać różne operacje w systemie. Gesty obsługiwanych przez mysz i oprogramowanie SetPoint (rys. 5.7) to, między innymi:

- podwójne stuknięcie powierzchni dotykowej jednym palcem,
- podwójne stuknięcie powierzchni dotykowej dwoma palcami,
- przesunięcie jednym palcem od prawej i lewej krawędzi,
- przesunięcie jednym palcem w dół i w górę,
- przesunięcie jednym palcem w lewo i w prawo,
- przesunięcie dwoma palcami w lewo i w prawo,
- klikniecie prawym, lewym, środkowym przyciskiem myszy.

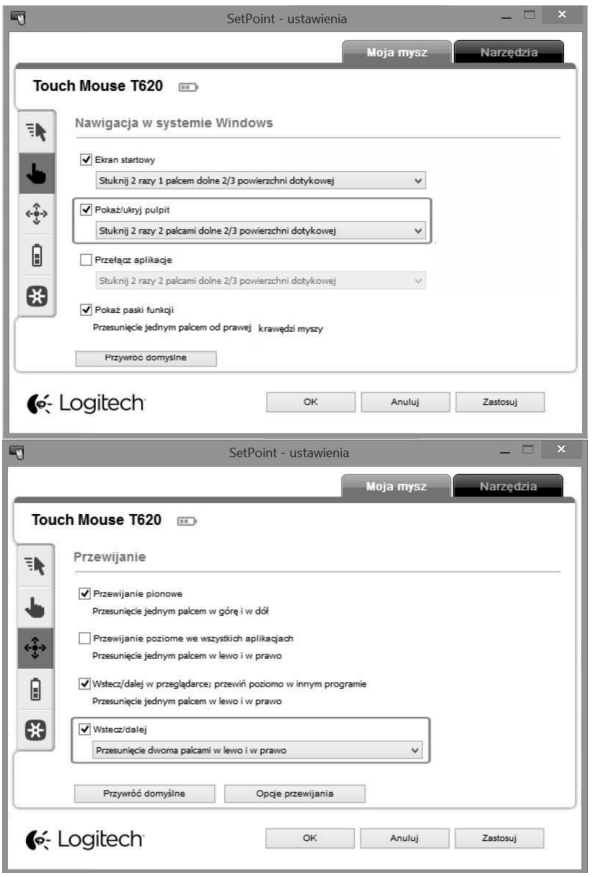
Rys. 5.7: Gesty obsługiwane przez mysz Logitech T620

Operacje odpowiadające rozpoznawanym gestom można ustawić w programie SetPoint (rys. 5.8). Mogą to być na przykład:

- przełączenie na ekran startowy (Metro),
- pokazywanie, ukrywanie pulpitu,
- przełączanie między aplikacjami,
- pokazywanie pasków funkcji,
- przewijanie pionowe,
- przewijanie poziome we wszystkich aplikacjach,
- wstecz/dalej w przeglądarce,
- wstecz/dalej w aplikacjach.

5.3.2. Ekran dotykowy

Dążenie do opracowywania coraz to wygodniejszych rozwiązań przyczyniło się do rozwoju interfejsów, w których nie wykorzystuje się już żadnych specjalistycznych kontrolerów takich jak myszy i klawiatury. Do generowania i odbierania bodźców wystarczać w nich mają efektory i sensory, jakimi człowiek dysponujemy cały czas, tj. dłonie, usta lub oczy. Można to zaobserwować na przykładzie



Rys. 5.8: Program SetPoint

urządzeń mobilnych. Wraz z wyposażeniem je w duże, wysokorozdzielcze ekrany pojawiła się potrzeba zastąpienia fizycznej klawiatury czymś innym, co pozwoli na bardziej intuicyjną, szybszą i przyjemniejszą i jednocześnie wystarczająco precyzyjną interakcję.

Pierwsze urządzenia reagujące na ruchy palców powstały w latach 60' ubiegłego wieku i wykorzystywały ramkę wokół ekranu z rozmieszczonymi w niej czujnikami optycznymi. Linie detekcji tych czujników tworzyły siatkę, dzięki czemu ich przecięcie poprzez dotknięcie palcem ekranu było łatwo rozpoznawalne i lokalizowalne.

W kolejnych wynalazkach starano się wykorzystywać fale akustyczne, jednak skomplikowana konstrukcja sensorów oraz mała ich użyteczność poskutkowała zmianą kierunku badań. W tym też czasie rozwijały się badania nad możliwością zastosowania rysików elektronicznych. Pierwotnie służyły one jedynie w eksperymentach nad rozpoznawaniem pisma, później zaczęto je wykorzystywać jako narzędzia do rysowania. W urządzeniach mobilnych i konsolach do gier zagościły one na dobre dopiero na przełomie lat 80' i 90'. Znalazły bardzo szerokie za-

stosowanie w wyspecjalizowanym sprzęcie, a o ich przydatności i popularności może świadczyć fakt, że takie urządzenie często spotka się jeszcze obecnie. Rozwiązania te opierały się na technologii oporowej, która w ostatnich latach bardzo mocno została wyparta poprzez technologię pojemnościową, która o wiele lepiej radzi sobie z dużymi rozdzielczościami i obsługą za pomocą gestów dłoni.

5.3.3. Rozpoznawanie mowy

Historia automatycznego rozpoznawania mowy sięga lat 50', kiedy to powstały pierwsze maszyny potrafiące rozpoznać poszczególne fonemy. Dziś prym w urządzeniach przenośnych i szeroko-dostępnych wiedzie firma Google. Amerykański gigant jest obecnie w posiadaniu największej liczby rozbudowanych słowników (bazy danych) oraz algorytmów potrafiących w locie rozpoznawać nawet bardzo złoże zdania.

Najlepiej metody rozpoznawanie mowy działają dla języka angielskiego. Język polski, z racji swojej charakterystyki, większej liczby głosek oraz nietypowej gramatyki, narzuca poważne ograniczenia dla istniejących algorytmów (np. algorytmów N-gram, w których przewidywane są kolejne słowa, oraz algorytmów zbudowanych w oparciu o sieci neuronowe). W związku z tym uzyskiwane wyniki nie są aż tak spektakularne.

Praktycznie każdy nowoczesny telefon posiada funkcję rozpoznawania mowy i bardzo często skrót do tej funkcji mieści się również na klawiaturze wprowadzania znaków. Świadczy to o wysokiej popularności i przydatności tej funkcji. Dobrym przykładem użycia takich rozwiązań jest kierowanie samochodem i wydawanie słownych komend do systemu nawigacji.

5.3.4. Multitouch

Pojęcie multi-touch w dosłownym tłumaczeniu oznacza wielokrotny dotyk. Zastosowane w odniesieniu do urządzeń typu tablet, smartphone, touchpad oznacza, że zostały one przystosowane do rozpoznawania dotyku i gestów wykonywanych kilkoma palcami jednocześnie. Dzięki oferowanym możliwościom implementacyjnym oraz prostocie obsługi technologia ta wyparła stosowane wcześniej rysiki. Do wykrywania wielokrotnych dotknięć najczęściej wykorzystywana jest technika pojemnościowa. Możliwe jest także wykorzystanie podczerwieni, a nawet zastosowanie kamer (do analizy położenia palców na ekranie).

Samo określenie multi-touch bardzo często dotyczy bardzo elementarnych systemów, takich jak Dual Control, Gest Touch czy Dual-Touch, które są w stanie wykryć bardzo małą liczbę jednoczesnych dotknięć i gestów. Bardziej zaawansowane urządzenia i algorytmy pozwalają na obsługę np. 56 dotknięć jednocześnie, co pozwala kilku użytkownikom jednocześnie korzystać z takiego urządzenia. Przykładem może być tutaj Microsoft PixelSense (wcześniej Microsoft Surface), czyli komputer dotykowy, zaprojektowany przez Microsoft i rozwijany pod nazwą kodową Milan. Podobne rozwiązania zagościły na dobre również na platformach Unixowych oraz urządzeniach mobilnych z najnowszymi systemami operacyjnymi typu Android, Windows Phone i iOS.

5.3.5. Kinect

W wielu domach zagościła konsola X-Box produkowana przez firmę Microsoft. Z początku służyła ona tylko do gier, głównie sportowych i tanecznych. Patrząc na jej możliwości zaczęto zadawać sobie pytanie, czy tak zaawansowanej i szeroko dostępnej technologi nie dałoby się wykorzystać poza światem gier? Czy nie można byłoby zastosować dodatkowych czujników, wiążących świat wirtualny z światem rzeczywistym? Na takim gruncie powstał czujnik ruchu Kinect.

Czujnik ruchu Kinect wyposażony został w dwie kamery, promiennik podczerwieni, macierz czterech mikrofonów kierunkowych, akcelerometr oraz napęd pozwalający na uchylanie głowicy. Oprogramowanie dostarczane wraz z kamerą Kinect jest w stanie automatycznie kalibrować czujnik w oparciu o stan gry oraz fizyczne otoczenie gracza, biorąc pod uwagę również różnego rodzaju przeszkody (np. meble). Kinect ma możliwość jednoczesnego śledzenia dowolnej liczby postaci (w praktyce 5 osób), w tym 2 aktywnych graczy, będących w stanie zmieścić się w polu widzenia kamery. Urządzenie, śledząc postacie i dokonując analizy ich ruchów, „dzieli" człowieka na 48 części, monitorowanych w czasie rzeczywistym. Pojawiła się więc możliwość wykonywania gestów palcami, rękami a nawet całym ciałem.

5.4. Projekt rozwiązania służącego do obsługi gestów

W wielu systemach istnieją skróty klawiszowe, za pomocą których można włączać programy lub uruchamiać poszczególne funkcje. Użytkownicy, którzy bardzo często korzystają z takich opcji, znają te kombinacje klawiszy na pamięć. Niestety, przeciętny użytkownik musi wyszukiwać i uruchamiać wszystko „ręcznie". O wiele prostszym rozwiązaniem i bardziej intuicyjnym są gesty wykonywane np. na touchpadzie, znane głównie z produktów Apple. Idea ta została wykorzystana w projekcie aplikacji interpretującej gesty użytkownika.

Projekt zakładał powstanie rozwiązania służącego do obsługi gestów podczas wyszukiwania i uruchamiania aplikacji na komputerze użytkownika. Funkcjonalnie miało ono przybrać formę widgetu, na którym można byłoby „rysować" różne symbole. Za źródło sygnałów (miejsce „rysowania") miały posłużyć urządzenie fizyczne typu multi-pad. Domyślnie, po każdym rozpoznaniu symbolu miała być wyświetlana odpowiednia lista zainstalowanych programów. Na przykład wykreślenie litery „P" miało spowodować wyświetlenie listy programów graficznych zainstalowanych na komputerze (przykładowo programów Paint, Corel Draw, Adobe Photoshop, Gimp). Wybór któregoś z nich i uruchomienie miałoby następować w kolejnym kroku i dla kolejnego gestu.

Początkowo planowano zaimplementować rozpoznawanie gestów i nawigowanie tematyczne wśród programów w systemie *Linux* w środowisku programistycznym *Qt* i przy użyciu języka programowania *C++*. Środowisko *Qt* zostało wybrane, ponieważ udostępnia biblioteki *QTouchEvent* i *QGestures* wspomagające obsługę i odczytywanie z wejścia urządzenia fizycznego wykonywanych gestów [4]. Ponieważ podczas prób uruchomienia i wykorzystania powyższych bibliotek oraz odpowiednich sterowników wystąpił problem z wykrywaniem urzą-

dzenia multi-tochpad oraz z obsługą przez to urządzenie wielu palców, zmieniono koncepcję programu i platformę docelową.

Platformą docelową okazał się ostatecznie wciąż popularny system *Windows 7 Professional* w wersji 64-bitowej. Jako środowisko programistyczne wybrano *Microsoft Visual Studio 2013* z językiem *C#* (język ten posiada wsparcie w postaci bibliotek idealnie nadających się do tworzenia aplikacji okienkowych, obsługi urządzeń, portów, przeszukiwania rejestru systemu itp.). Niestety z powodu ograniczeń czasowych musiano zawęzić plany obsługi wielu gestów i ich sekwencji do rozpoznawania kilku z nich oraz wywoływania domyślnych programów z poszczególnych kategorii.

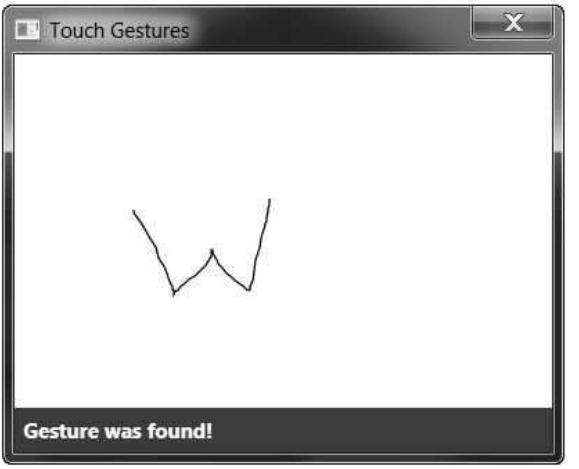
Po uruchomieniu programu przez użytkownika na pulpicie ukazuje się puste okienko z białym tłem. Na tym tle można rysować symbole za pomocą touchpada lub myszki. Rozpoznawanie rozpoczyna się po wciśnięciu lewego przycisku myszki bądź stuknięcia w gładzik. Program zaczyna klasyfikować gest w momencie zwolnienia przycisku lub po oddaleniu palca. Na dole okienka wyświetla się komunikat, czy rozpoznawanie zakończyło się sukcesem czy wręcz przeciwnie.

Zaimplementowany algorytm zachowuje listę punktów i na podstawie zmian na osi x oraz y tworzy ośmiokierunkową listę przemieszczeń. Ludzka ręka nie jest na tyle dokładna, żeby narysować idealnie prostą linię. Z tego powodu rysowaniu symboli towarzyszą tak zwane szumy. Aby odfiltrować właściwy kierunek kreślenia podzielono cały ruch na mniejsze odcinki, gdzie sprawdzano, jakie przemieszczenie dominuje. Kolejnym etapem rozpoznawania było porównanie sekwencji ruchów z sekwencjami w stworzonej wcześniej bazie gestów. Tym sposobem udało się zaimplementować działanie dla czterech gestów: P, W, N, E.

Następnie, odpowiednio do rozpoznanych gestów, następowało przeszukiwanie folderu w rejestrze systemu zawierającego listę zainstalowanych programów wraz z ich ścieżkami dostępu. Dla litery P odnalezionym programem był *Paint*, dla W - *Word*, dla N- *Notatnik* oraz dla E- *Internet Explorer*. Rozpoznawalność tych symboli pomimo braku zastosowania bardziej złożonych algorytmów okazała się być wystarczająco dobra. W celu zilustrowania wyników działania algorytmu rozpoznawania gestów utworzono widget jak na rysunku 5.9.

5.5. Podsumowanie

Nowoczesne interfejsy wyszukiwania są projektowane w taki sposób, aby jak najbardziej ułatwić i usprawnić użytkownikowi współprace z nimi. Jednym z wbudowywanych w interfejsy wyszukiwania udogodnień jest urozmaicenie możliwość zadawania przez użytkownika pytania, które nie tylko wprowadza się za pomocą teksu, ale wykorzystuje się różne urządzenia fizyczne, systemy rozpoznające mowę, wykonywane gesty itp. Ulepszeniu ulega również sposób wyświetlania wyników. Dane wynikowe są dostosowywane do odbiorcy, prezentowane w różne sposoby graficzne, są sortowane według najlepszego dopasowania lub, jeżeli interfejs wyszukiwania służy do nawigowania programów w systemie operacyjnym, wynikiem jest uruchomienie programu.



Rys. 5.9: Widget ilustrujący działanie algorytmu rozpoznawania gestów

Literatura

[1] P. Morville, J. Callender. *Wzorce wyszukiwania. Projektowanie nowoczesnych wyszukiwarek*. Wydawnictwo Helion, 2011.

[2] Google. Support Google. `www.support.google.com/?hl=pl` [dostęp dnia 20 czerwca 2015].

[3] Logitech. Touch Mouse T620. `www.logitech.com/pl-pl/product/touch-mouse-t620` [dostęp dnia 20 czerwca 2015].

[4] Qt. Qt Documentation. `www.doc.qt.io/qt-4.8/` [dostęp dnia 20 czerwca 2015].

ROZDZIAŁ 6

PROBLEMY STEROWANIA W ZAMKNIĘTEJ PĘTLI Z SYSTEMEM WIZYJNYM

A. Jurasik, B. Kowalczyk, P. Urbaniak

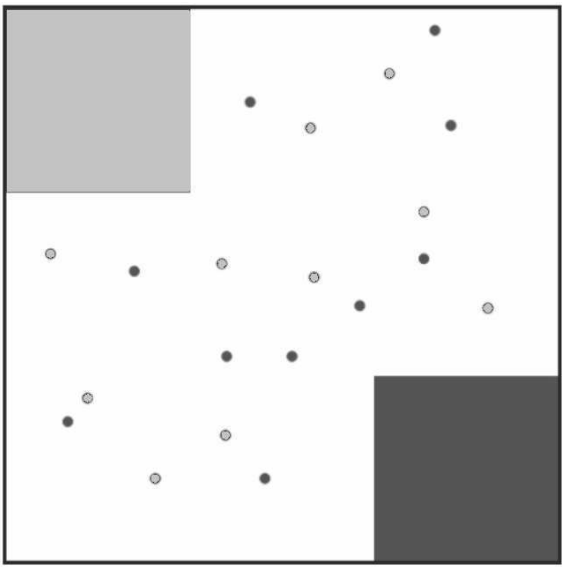
„PuckCollect" to jedna z konkurencji rozgrywanych w trakcie międzynarodowych zawodów robotów „RobotChallenge". W rozdziale opisano autorskie rozwiązanie pozwalające na realizację zadań stawianych w tej konkurencji przez rój robotów sterowanych w zamkniętej pętli z systemem wizyjnym.

6.1. Wprowadzenie

Corocznie w Wiedniu rozgrywane są międzynarodowe zawody robotów „RobotChallenge". Ich organizatorem jest INNOC (ang. *Austrian Society for innovative Computer*). Jedną z odbywających się wtedy konkurencji jest „PuckCollect". Zgodnie z regulaminem [1] celem uczestników „PuckCollect" jest zebranie jak największej liczby pucków (okrągłych krążków) rozłożonych równomiernie na obszarze o wymiarach 2,5 m × 2,5 m. Zadanie należy wykonać w jak najkrótszym czasie. W jednej potyczce biorą udział dwie drużyny, których roboty (maksymalny rozmiar 0,5 m × 0,5 m) zbierają krążki określonego koloru (10 czerwonych lub 10 niebieskich) przenosząc je do wyznaczonej części pola gry (zobacz rysunek 6.1). Punkty przyznawane są tylko za zebranie właściwych krążków, choć można zbierać też pucki przeciwników (chowając je wewnątrz robota), uniemożliwiając im zdobycie punktów. Wygrywa drużyna, która zgromadzi więcej punktów. Standardowy czas rozgrywki to ok. 5 minut.

Podczas analiz różnych wariantów realizacji celu we wspomnianej konkurencji powstała idea stworzenia roju robotów poruszających się w zamkniętym środowisku. Idea ta opiera się na pomyśle rozwiązanie problemu zbierania pucków poprzez rozczłonowanie jednego robota na kilka małych elementów. Elementy te poprzez równoległą pracę mogłyby zebrać wymaganą liczbę pucków w znacznie krótszym czasie niż jeden robot. Ponieważ gwarancją wygranej jest zebranie 11-tu dowolnych krążków, zaproponowano więc utworzenie 11 małych robotów nazwanych *SwarmBotami*. Ze względu na sporą ich liczbę starano się zminimalizować koszty związane z ich wytworzeniem. Dlatego zrezygnowano z całej odo-

metrii umieszczanej na pojedynczym robocie. Rolę sprzężenia zwrotnego przeniesiono na system wizyjny składający się z dwóch kamer umieszczonych nad polem gry. Tak powstał system wymieniony w tytule niniejszego rozdziału. W kolejnych podrozdziałach opisano dokładniej wszystkie jego elementy.



Rys. 6.1: Pole konkurencji PuckCollector z zaznaczonymi obszarami drużyn

6.2. System wizyjny

System wizyjny jest układem współpracujących ze sobą elementów: urządzeń pozyskujących informacje (kamera lub kilka kamer), urządzenia służącego do akwizycji i przetwarzania danych (typu *frame grabber*) oraz urządzenia analizującego dane (procesor lub komputer z oprogramowaniem). Jego funkcją jest automatyczna analiza wizyjna otoczenia na podobieństwo zmysłu wzroku u ludzi.

6.2.1. Kamery

Przy wyborze kamer zostały wzięte pod uwagę następujące wymagania:

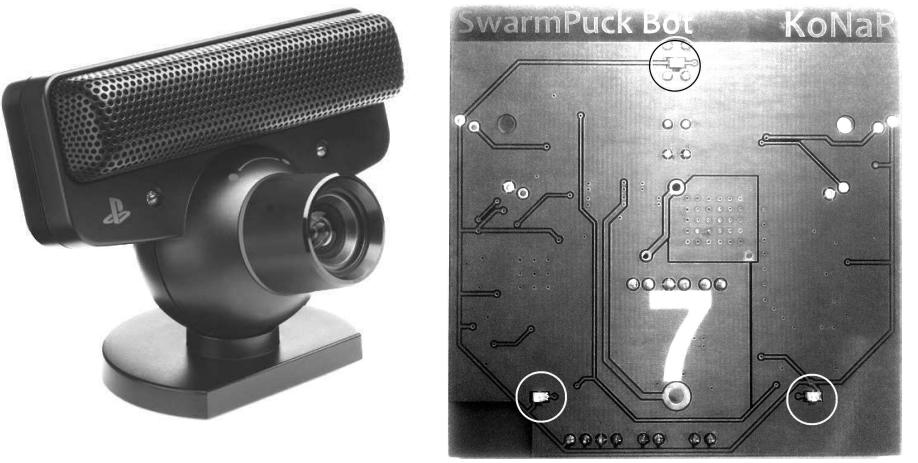
- liczba klatek rejestrowanych w ciągu sekundy (ang. *frames per second*, FPS) (ze względu na szybko zmienny obraz wartość ta powinna być wystarczająco wysoka – maksymalna zakładana prędkość *SwarmBotów* to 1 m/s);
- kąt widzenia (powinien być na tyle duży, aby obraz z kamery obejmował cały obszar rozgrywki);
- cena.

Zdecydowano się na zakup 2 kamer Playstation Eye [2]. Ich podstawowe parametry to: 120 FPS w rozdzielczości 320 × 240 lub 60 FPS w rozdzielczości 640 × 480

oraz kąt widzenia od 56 do 75 stopni (rys. 6.2a). Kamery te umieszczono na rozkładanym wysięgniku. Każda z kamer pełni inną funkcję.

- Kamera pierwsza, wyposażona w filtr światła widzialnego, jest przeznaczona do wykrywania i śledzenia poruszających się po polu gry *SwarmBotów*. Każdy robot jest identyfikowany na podstawie 3 diod emitujących światło podczerwone. Diody są ułożone w trójkąt równoramienny, co jednoznacznie pozwala określić ich pozycję (zwrot i kierunek, zobacz rysunek 6.2b).
- Kamera druga ma za zadanie zlokalizować pucki i wybrać 11 najbliższych, przypisując im rozpoznany kolor. Dodatkowo kamera ta odpowiada za wyznaczenie obszarów zabronionych — czyli przeszkód (np. robot drużyny przeciwnej).

a) b)



Rys. 6.2: Elementy systemu sterowania odpowiedzialne za pozyskiwanie i generowanie sygnałów wizyjnych *SwarmBota*: a) kamera Playstation EYE, b) płytka z rozmieszczonymi na niej diodami identyfikacyjnymi

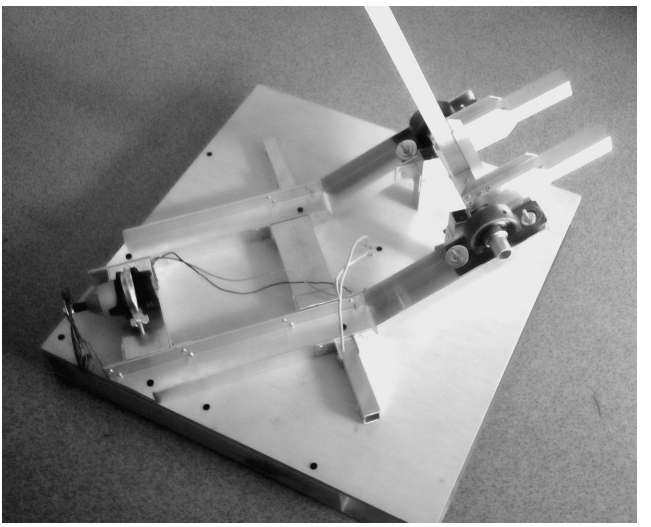
6.2.2. Stacja bazowa

Ze względu na zastosowanie nietypowego rodzaju zasilania *SwarmBotów* (omówionego dalej) konieczne było zbudowanie ładującej stacji dokującej. Statek Matka składa się z 2 płyt pokrytych warstwą miedzi, między które wjeżdżają roboty (rys. 6.3). Dodatkowo zamontowany jest na nim maszt wychylający kamery (ze względu na ograniczenia wynikające z regulaminu [1]) oraz komputer przetwarzający obraz i sterujący wszystkimi *SwarmBotami*. Ciężkie akumulatory zasilające system ładowania i składania/rozkładania masztu stanowią konieczną przeciwwagę.

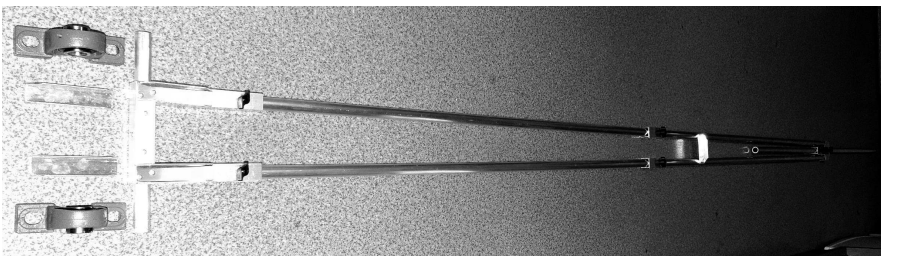
Aby kamery obejmowały cały obszar muszą znajdować się na wysokości ok. 2 m. W połączeniu z kątem widzenia kamer implikowało to konieczność wykonania masztu o długości prawie 3 m. Odpowiednią sztywność i wytrzymałość tego

elementu miały zapewnić wykorzystane aluminiowe profile i rury. Konstrukcja umożliwia regulację długości wysięgnika, co okazało się konieczne do prawidłowego skalibrowania systemu wizyjnego (rys. 6.3b).

a)



b)



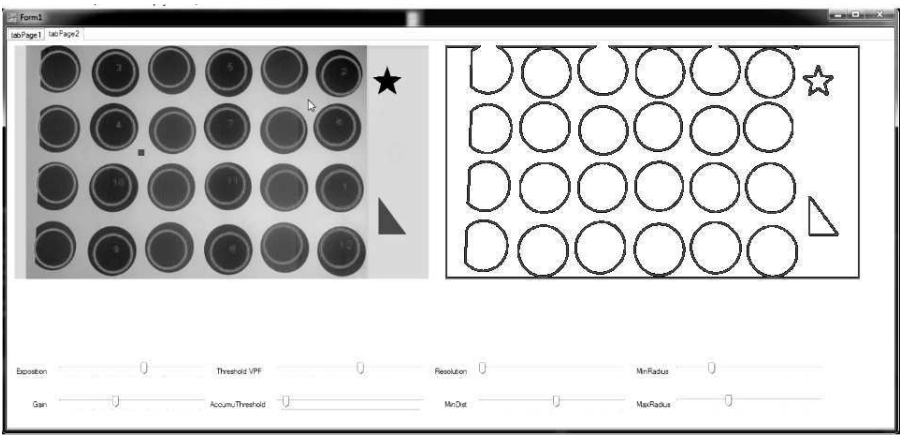
Rys. 6.3: Statek Matka: a) stacja bazowa, b) wysięgnik do mocowania kamer nad obszarem ruchu *SwarmBotów*

6.2.3. Rozpoznawanie obiektów: robot, cel, przeszkoda

Robot

Poszczególne *SwarmBoty* są rozpoznawane dzięki trzem diodom emitujących światło podczerwone (ok. 720 nm). Identyfikacja danego bota następuje, gdy algorytm znajdzie pierwszy, jasny punkt na obrazie z kamery. Następnie w jego otoczeniu poszukiwane są dwa równoodległe punkty. Jeżeli zostaną znalezione, wszystkim trzem punktom przypisywany jest nr robota, który dostał polecenie wyjazdu ze statku matki. Ponieważ algorytm ten nie jest w stanie rozpoznać wszystkich botów wyjeżdżających na raz, problem rozwiązano ustalając odpowiednią sekwencję wyjeżdżania botów. Najpierw wysyłane są roboty, które mają najdalej do celu. Dzięki temu optymalizuje się czas potrzebny na zebranie wszystkich krążków z planszy.

Cel
W celu wykrycia krążków na obrazie wykorzystana została transformacja Hough'a zaimplementowana w bibliotece OpenCV. Podejmowanie odpowiednich decyzji wymaga również śledzenie każdego krążka (dany krążek musi mieć stale to samo id, pomimo jego przesuwania). Żeby to osiągnąć wykryte krążki na bieżącej klatce są dopasowywane do krążków jakie zostały wykryte na poprzedniej klatce. Po wykryciu i dopasowaniu krążka następuję analiza koloru i podjęcie decyzji, czy krążek jest czerwony czy niebieski. Interfejs stworzonej aplikacji przedstawiono na rysunku 6.4.



Rys. 6.4: Interfejs aplikacji do rozpoznawania krążków

Przeszkoda
W celu wykrycia przeszkód obraz pochodzący z kamery poddawano progowaniu, uzyskując obraz składający się z samych zer i jedynek. W celu znalezienia konturów stosowano detektor krawędzi Canne'go. Następnie, za pomocą funkcji *findCounturs* z biblioteki OpenCV, wydzielano na obrazie zbiór punktów stanowiących krawędzie przeszkód. Pozyskane w ten sposób informacje przekazywano do części programu zajmującej się wyznaczaniem trajektorii.

6.3. Konstrukcja robotów

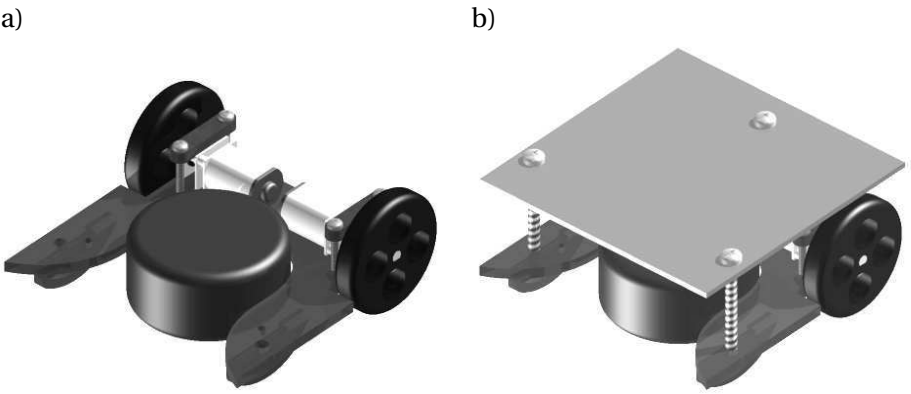
Podczas konstruowania robotów musiano uwzględniać kilka ograniczeń.

- Roboty powinny mieć niewielkie rozmiary pozwalające na zmieszczenie ich w polu startowym oraz ułatwiające bezkolizyjne poruszanie się.
- Budowa robotów powinna umożliwiać podjęcie przez nie poszukiwanego obiektu (krążka) oraz dowiezienie go do stacji bazowej.
- W konstrukcji powinno pojawić się tylko podstawowe oczujnikowanie, pozwalające na nawigację w pobliżu celu oraz rozpoznawanie koloru celu.
- Sterowanie robotami powinno być sprzęgnięte z systemem wizyjny.
- Powinien być zapewniony kanał komunikacji ze stacją bazową.

Dodatkowym założeniem był jak najniższy koszt jednostkowy robota oraz maksymalne uproszczenie montażu. Założenie to wynikało z konieczności zbudowania dużej liczby robotów. Do wypełnienia zadania wystarczyłoby 11 robotów działających jednocześnie, jednak planowano wykonać 20 egzemplarzy aby zwiększyć niezawodność system.

6.3.1. Budowa mechaniczna

Zdecydowano się na roboty klasy (2,0) ze względu na ich prosty układ mechaniczny, dużą zwrotność, proste sterowanie oraz bogatą literaturę na ich temat [3]. Roboty zostały zaprojektowane z użyciem oprogramowania *Autodesk Inventor*. Podwozie wykonano ze szkła akrylowego, ciętego laserem. Do podwozia zamocowano dwa motoreduktory lokalizując je z tyłu robota. Z punktu widzenia sterowania i unikania kolizji bardziej korzystne jest umiejscowienie napędu bliżej środka ciężkości obrysu robota. Konieczne jednak było przesunięcie napędu, aby zostawić przestrzeń na transportowany obiekt, bez zwiększania rozmiaru robota. Ponad napędem i przestrzenią transportową umieszczono płytkę PCB (ang. *Printed Circuit Board*). Ostatecznie gabaryty robota w wersji finalnej zmieszczono w obrysie 75 mm × 75 mm (rys. 6.5).



Rys. 6.5: Projekt robota: a) podwozie, układ napędowy, krążek; b) postać finalna

6.3.2. System zasilania

W robotach zastosowano nietypowy układ zasilania. Centralną rolę odegrały w nim superkondensatory. Takie źródło zasilania ma szereg zalet. Największą jest bardzo krótki czas ładowania. Dla większości typów akumulatorów maksymalny prąd ładowania jest zwykle przynajmniej 10 razy niższy od maksymalnego prądu rozładowywania i stanowi 1-2 krotność pojemności wyrażonej w Ah. Oznacza to, że pełne naładowanie trwa co najmniej pół godziny. Dla superkondensatorów wartość ta jest znacznie wyższa i pozwala na pełne naładowanie nawet w kilka sekund. W przypadku omawianych robotów pełne ładowanie trwa około 2 minut, co wynika z zastosowanego układu ładowania, a nie z ograniczeń kondensatorów.

Zaletą superkondensatorów jest również bardzo wysoki uzyskiwany prąd w stosunku do masy i pojemności. Pozwala to znacznie zmniejszyć masę robotów o krótkim docelowym czasie działania. Zastosowana bateria 4 kondensatorów, o pojemności 10 F i masie około 12 g, może dostarczyć prąd maksymalny około 48 A. Jest to znacznie więcej niż wymagają użyte silniki. Dla porównania uzyskanie takiego prądu wymagałoby użycia akumulatora litowo-polimerowego o masie 50 g.

Wadą superkondensatorów jest niskie napięcie pojedynczego ogniwa (2,5 V) oraz bardzo szybki spadek tego napięcia w miarę rozładowywania. Chcąc uzyskać stałą moc silników konieczne jest zastosowanie przetwornicy step-up, podnoszącej napięcie do poziomu 6 V wymaganego przez użyte napędy. Oznacza to spadek wydajności oraz pobieranie z kondensatorów coraz wyższego prądu przy tym samym obciążeniu silników.

Zastosowana bateria superkondensatorów pozwala na ruch robota z pełną prędkością przez około 3 minuty. Ładowanie odbywa się w stacji bazowej w sposób automatyczny. Oznacza to, że cały rój może prowadzić ciągłe działania bez konieczności jakiejkolwiek obsługi. Do ładowania użyto układu liniowego ograniczającego prąd do około 1 A. Możliwe jest jednak kilkukrotne zwiększenie prądu, co skróciłoby czas ładowania do kilku sekund i umożliwiłoby niemal nieprzerwane działanie systemu.

6.3.3. Komunikacja i rozpoznawanie

Komunikacja z systemem nadrzędnym odbywa się drogą radiową, z użyciem modułów RFM73 [4] w paśmie 2,4 GHz. Komunikacja odbywa się jednostronnie – robot jedynie odbiera polecenia od układu nadrzędnego, samemu nie transmitując żadnych danych. Każdy z robotów posiada unikalny 4-bitowy nr ID, ustalany za pomocą umieszczonego na płytce DIP-switcha. Dodatkowo możliwe jest jednoczesne programowanie wszystkich robotów drogą radiową.

U góry robota umieszczone są 3 diody podczerwone, celem łatwego wykrywania jego położenia i orientacji poprzez kamerę z odpowiednim filtrem, opisaną w podrozdziale 6.2. Diody mogą być zapalane i gaszone celem identyfikacji poszczególnych robotów na obrazie.

6.4. Sterowanie robotem klasy (2,0) w zamkniętym środowisku

Monocykl jest najprostszą, nieholonomiczną platformą mobilną klasy $(2,0)$ (rys. 6.6). Posiada dwa wejścia sterujące, a jego równania kinematyki mają następującą postać:

$$\dot{q} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{6.1}$$

gdzie:
u_1 – prędkość liniowa platformy,

u_2 – prędkość kątowa platformy,
(x, y) – położenie punktu charakterystycznego,
θ – orientacja platformy.

Oznaczając elementy wektora konfiguracji jako q_1, q_2, q_3 pola wektorowe rozważanego modelu można wyrazić następująco:

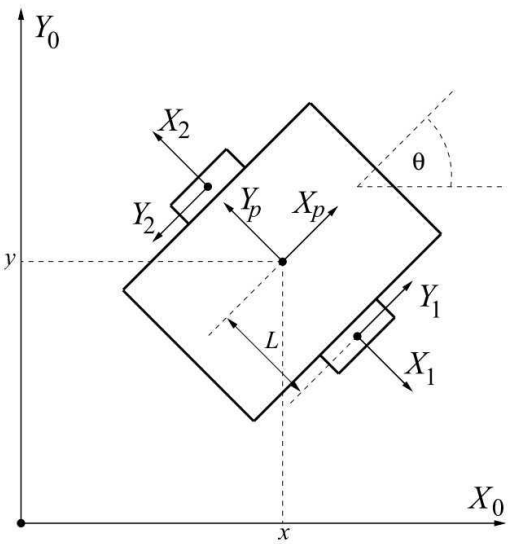
$$\boldsymbol{g}_1(\boldsymbol{q}) = \begin{bmatrix} \cos(q_3) & \sin(q_3) & 0 \end{bmatrix}^T \quad \boldsymbol{g}_2(\boldsymbol{q}) = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T.$$

Z równań tych i po zastosowaniu odpowiednich nawiasów Li'ego można wygenerować kolejne pola wektorowe (kierunki ruchu):

$$\begin{aligned} [\boldsymbol{g}_1, \boldsymbol{g}_2] &= \begin{bmatrix} \sin(q_3) & -\cos(q_3) & 0 \end{bmatrix}^T, \\ [\boldsymbol{g}_1, [\boldsymbol{g}_1, \boldsymbol{g}_2]] &= \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T, \\ [\boldsymbol{g}_2, [\boldsymbol{g}_1, \boldsymbol{g}_2]] &= \begin{bmatrix} \cos(q_3) & \sin(q_3) & 0 \end{bmatrix}^T. \end{aligned}$$

Sterowanie kinematyką zakłada, że możliwa jest bezpośrednia zmiana prędkości zapewniająca realizowanie zadanej trajektorii robota. W rzeczywistości zmianę stanu układu uzyskujemy poprzez sterowanie silnikami. Ruch robota, który jest odpowiedzią na sygnały pochodzące z napędów, to dynamika układu.

W przedstawionym rozwiązaniu sterowanie *SwarmBotem* odbywa się jedynie za pomocą kinematyki. Sprzężeniem zwrotnym informującym o stanie robota jest system wizyjny umieszczony nad obszarem, w którym porusza się robot.



Rys. 6.6: Schemat monocykla (kołowej platformy mobilnej klasy $(2,0)$)

6.4.1. Metoda pól potencjałów

Do planowania bezkolizyjnego ruchu postanowiono wykorzystać metodę pól potencjałowych (ang. *potential field method*) [5]. Jest to metoda ogólnego przeznaczenia, którą można zaadoptować do rozwiązywania różnych problemów. Ze względu m.in. na matematyczną elegancję oraz łatwość fizycznej interpretacji znajduje ona dość szerokie zastosowanie wielu aplikacjach robotów mobilnych.

Metoda pól potencjałowych należy do lokalnych metod planowania ruchu. Można ją wykorzystywać bez pełnej wiedzy o rozkładzie wszystkich przeszkód w otoczeniu robota. Ten brak może jednak prowadzić do wyliczania nieoptymalnych rozwiązań. Rekompensatą są mniejsze nakłady obliczeniowe oraz zwiększona odpornością na ewentualną zmianę warunków początkowych zadania (np. pojawienie się nowych przeszkód na scenie robota). W metodzie tej zakłada się, że ruch robota wynika z działających na niego sił:

- sił odpychających robota od przeszkód (powinny mieć niewielką wartość, gdy robot jest z dala od przeszkody i rosnąć nieskończenie, gdy robot jej dotyka);
- siły przyciągającej robota do celu (każdy pojedynczy krążek jest celem dla konkretnego *SwarmBota*).

W każdym punkcie przestrzeni wypadkowa siła wyznacza chwilowy kierunek ruchu i przemieszcza robota o małą i stałą odległość. Zachowuje się w ten sposób kontrolę nad dynamiką zmian konfiguracji robota oraz zabrania się (przy odpowiednio wysokich ładunkach potencjału) „przeskoczenia" robota przez przeszkodę. Istotną cechą tego podejścia jest niewrażliwość na kształt przeszkód.

Z zastosowaniem metody pól potencjałowych wiąże się kilku problemów. Najpoważniejszym jest występowanie minimów lokalnych. Wyjaśniając to w kilku słowach: wokół każdego minimum rozpościera się pewien obszar będący jego atraktorem. Jeśli robot wejdzie w ten obszar, a w praktyce bardzo trudno jest ocenić jego kształt i zasięg, to nieuchronnie osiągnie odpowiadające mu minimum lokalne. W takim przypadku nie da się wyznaczyć kolejnego kierunku ruchu bez odstąpienia od zasady metody.

Istnieje kilka sposobów na rozwiązanie lub uproszczenie tego problemu. Jeden z nich polega na takim zdefiniowaniu funkcji pola potencjałowego, aby nie występowały w nim minima lokalne. W pewnych warunkach, np. przy prostych przeszkodach i bardzo dobrej znajomości ich rozmieszczenia, udaje się wyznaczyć taką funkcję. Jednakże dla układów z licznymi i skomplikowanymi przeszkodami wydaje się to niemożliwe.

W sytuacji, gdy robot utknie w minimum lokalnym można zastosować różne scenariusze. Przykładowo można nakazać robotowi poruszać się wzdłuż przeszkody, aż do uzyskania pewności, że ją ominął. Można też sprawdzać, czy robot nie był już w danym położeniu wcześniej (co ma zabezpieczać przed oscylowaniem między minimami lokalnymi). Istnieją również metody, w których wykorzystuje się pewne informacje globalne. Aby uciec z obszaru minimum lokalnego można zastosować techniki błądzenia losowego lub algorytmy symulowanego odprężania, polegających z grubsza na dawaniu szansy rozwiązaniom gorszym od optymalnego.

6.4.2. Sterowanie *SwarmBotami* za pomocą metody pól potencjałowych

Praktyczne zastosowanie metody pól potencjałowych do sterowania robotem różni się nieco od tego, co podpowiadają teoretyczne rozważania. Po pierwsze, teoretyczne sterowania przyjmują postać translacji (odpowiadających ruchowi postępowemu) oraz rotacji robota. W praktyce zaś do sterowania robotem potrzebne są prędkości obrotowe napędów. Po drugie, w zbudowanych robotach występuje znaczne przesunięcie pomiędzy środkiem obrotu robota a jego środkiem geometrycznym. Dlatego należy rozważyć nieco zmienioną kinematykę, wyliczaną dla punktu przesuniętego o odległość L do przodu robota, gdzie L jest połową jego szerokości. Przyjmując L jako podstawową jednostkę odległości (tj. $L = 1$) można całe zagadnienie nieco uprościć. Współrzędne nowego puntu to:

$$\begin{aligned} x &= x_0 + \cos\theta, \\ y &= y_0 + \sin\theta, \end{aligned} \tag{6.2}$$

a jego prędkości to:

$$\begin{aligned} \dot{x} &= \dot{x}_0 - \dot{\theta}\sin\theta, \\ \dot{y} &= \dot{y}_0 + \dot{\theta}\cos\theta. \end{aligned} \tag{6.3}$$

Ponieważ $\dot{\theta} = u_2$ więc nowe równanie kinematyki przyjmie postać:

$$\dot{q} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\eta}_1 \\ \dot{\eta}_2 \end{pmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \\ 0 & 1 \\ \frac{1}{r} & \frac{1}{r} \\ \frac{1}{r} & -\frac{1}{r} \end{bmatrix} \cdot \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}.$$

Trajektoria robotów jest generowana za pomocą metody pól potencjałów. Jako dane wejściowe używany jest zbiór punktów reprezentujących krawędzie przeszkód. Robot porusza się zgodnie z gradientem ∇U pewnego pola opisanego funkcją U. W praktyce znajomość funkcji opisującej energię potencjalną tego pola jest mało istotna, gdyż łatwo wyliczyć siłę działającą na robota w punkcie, będącą właśnie gradientem tego pola.

Funkcję siły wyliczano korzystając z podejścia opisanego w [5]. Część przyciągającą (odpowiadającą za podążanie robota w kierunku celu) wyrażono wzorem:

$$F_{przy} = -\frac{x - x_d}{\|x - x_d\|}$$

gdzie: x_d jest punktem docelowym, natomiast x środkiem geometrycznym robota. Część odpychającą (odpowiadająca za odpychanie robota od przeszkód) zdefiniowano jako:

$$F_{odp} = \sum_{i=0}^{n} \frac{x - x_i}{\|x - x_i\|} \cdot \frac{1}{(\|x - x_i\| - R)^2},$$

gdzie x_i to współrzędne przeszkód w otoczeniu robota, natomiast R to założony promień bezpieczeństwa. Ponieważ w przyjętej skali robot jest kwadratem o wymiarach 2×2, a punkt x znajduje się po środku tego kwadratu, dlatego przyjęto promień bezpieczeństwa nie mniejszy niż $R = 1.5$. Należy zwrócić uwagę, że gdyby wyszczególniony punkt robota nie był środkiem geometrycznym, a np. osią obrotu, minimalna wartość R musiałaby być znacznie większa, aby obszar bezpieczeństwa obejmował całego robota.

Ze względu na postać funkcji wartość siły odpychającej gwałtownie rośnie w odległości od przeszkody bliskiej R. Natomiast dla większych odległości wartość ta staje się pomijalnie mała. Można zatem nie uwzględniać w obliczeniach przeszkód znajdujących się poza pewnym otoczeniem robota.

Przy dokładniejszej analizie problemu można zauważyć, że wielkość promienia bezpieczeństwa może być uzależniona od aktualnej prędkości robota, zaś wartość jego prędkości powinna być odwrotnie proporcjonalnie do wartości siły odpychającej. W praktyce oznacza to, iż robot może poruszać się szybko w znacznej odległości od przeszkód. Może też przejeżdżać przez wąskie przejścia, jednak przy niższej prędkości zapewniającej większą precyzję.

Reasumując założenia: wyróżniony punkt na robocie ma poruszać się zgodnie z gradientem pola, określonym kierunkiem siły

$$\nabla U = F = F_{przy} + F_{odp}.$$

Chcemy zatem, by składowe prędkości $\dot{x}$ i $\dot{y}$ były proporcjonalne do składowych f_x, f_y siły F:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} f_x \\ f_y \end{pmatrix}, \tag{6.4}$$

Siły i sterowania wiąże relacja:

$$F = G \cdot u, \tag{6.5}$$

gdzie

$$G = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

jest odwracalną macierzą obrotu o wyznaczniku równym 1. Stąd sterowania u w każdym punkcie można wyliczyć ze wzoru:

$$u = G^{-1} \cdot F \tag{6.6}$$

Rozpisując to dalej otrzymamy:

$$\begin{aligned} u_1 &= f_x \cos\theta + f_y \sin\theta \\ u_2 &= f_y \cos\theta + f_x \sin\theta \\ \eta_1 &= \frac{1}{r}(u_1 + u_2) \\ \eta_2 &= \frac{1}{r}(u_1 - u_2). \end{aligned} \tag{6.7}$$

gdzie u_1 i u_2 to składowe sterowania, zaś η_1 i η_2 to prędkości kół.

6.5. Podsumowanie

Celem niniejszego rozdziału było przedstawienie problemów i dostępnych rozwiązań związanych ze sterowaniem rojem robotów w zamkniętym środowisku z użyciem systemu wizyjnego. Przedstawiono w nim szczegóły rzeczywistej realizacji takiego systemu. Realizacja ta odbyła się w ramach projektu, na który składało się:

- zaprojektowanie i wykonanie *SwarmBotów*,
- zaprojektowanie i wykonanie Statku Matki i systemu wizyjnego,
- zaimplementowanie systemu rozpoznawania botów, pucków i przeszkód,
- stworzenie i zaimplementowanie protokołu przesyłu danych,
- zaimplementowanie metody sterowania robotami (metoda pól potencjałów).

Niektóre z elementów systemu powstały na bazie standardowych rozwiązań (jak np. algorytm wykrywania pucków). W innych zastosowano niecodzienne podejście do rozważanego problemu (np. zasilanie *SwarmBotów* z wykorzystaniem superkondensatorów).

Testy przeprowadzone na stworzonym stanowisku pozwoliły zweryfikować przyjęte założenia i oraz wybrane rozwiązania. Idea sterowania zaproponowana w podrozdziale 6.4 sprawdziła się podczas realizacji postawionych przed systemem zadań. Zastosowne rozwiązania inżynierskie w dziedzinie budowy systemu zasilania, komunikacji i przetwarzania obrazu okazały się skuteczne. System pozycjonowania robotów wykorzystujący diody IR i kamerę z filtrem zapewnił dużą dokładność i odporność na zewnętrzne czynniki. W efekcie powstał spójny i kompletny system umożliwiający prowadzenie dalszych badań.

Literatura

[1] Regulamin zawodów RobotChallenge 2014 kategorii PuckCollect. `http://www.robotchallenge.org/fileadmin/user_upload/_temp_/RobotChallenge/Reglement/RC-PuckCollect.pdf` [dostęp dnia 20 czerwca 2015].

[2] Dokumentacja techniczna kamery Playstation Eye. `http://uk.playstation.com/media/247868/7010571%20PS3%20Eye%20Web_GB.pdf` [dostęp dnia 20 czerwca 2015].

[3] A. Mazur. *Sterowanie oparte na modelu dla nieholonomicznych manipulatorów mobilnych.* Instytut Informatyki, Automatyki i Robotyki Politechniki Wrocławskiej, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław., 2009.

[4] Dokumentacja techniczna modułu RFM73. `http://www.hoperf.com/upload/rf/RFM73-Datasheet-V2.0.pdf` [dostęp dnia 20 czerwca 2015].

[5] I. Dulęba. *Metody i algorytmy planowania ruchu robotów mobilnych i manipulacyjnych.* Akademicka Oficyna Wydawnicza EXIT, 2001.

ROZDZIAŁ 7

PLANOWANIE RUCHU PRZY OGRANICZONYCH ZASOBACH

J. Drewniak

W niniejszej pracy zaprezentowano problem planowania ruchu z ograniczeniami energii dla robotów mobilnych oraz zaproponowano metody generowania optymalnej trajektorii rakiety.

7.1. Wprowadzenie

W praktycznych zastosowaniach coraz bardziej popularne stają się roboty i urządzenia nie związane na stałe z miejscem eksploatacji. Mobilność oraz elastyczność budowanych na ich podstawie systemów pozwala bowiem na zwiększenie komfortu obsługi oraz pokonywanie barier istniejących w innych rozwiązaniach. Jednak uwolnienie się od więzów okablowania wymusza konieczność zastosowania mobilnych źródeł energii, co często kończy się wprowadzeniem zasilania bateryjnego. Oznacza to, między innymi, nałożenie ograniczeń na ilość dostępnej energii do wykonania konkretnego zadania. W konsekwencji powstają limity czasu na wykonanie zadania oraz ograniczenia na moc źródła zasilania.

W przypadku urządzeń stale obsługiwanych przez ludzi dużą uwagę przykłada się do wygody ich użytkowania. Natomiast w przypadku urządzeń autonomicznych istotniejsze jest automatyczne zarządzanie zgromadzoną energią, np. poprzez planowanie własnego działania, a w szczególności – ruchu.

Zasilanie bateryjne jest stosowane w znakomitej większości robotów mobilnych (rys. 7.1). Roboty te można podzielić na kategorie według obszaru ich zastosowań: roboty czyszczące, roboty kosiarki, roboty ratownicze i inne. W szczególności dużym zainteresowaniem, zarówno w świecie nauki jaki i świecie codziennych zastosowań, cieszą się pojazdy elektryczne.

Ciekawą grupę urządzeń, których dotyczy problem zarządzania energią, stanowią obiekty latające: drony (samoloty i wielowirnikowce) oraz rakiety. Obok zasilania elektrycznego (stosowanego głównie do sterowania) wykorzystuje się w nich również różnego rodzaju paliwa (dostarczające energię potrzebną do przemieszczania się).

Rys. 7.1: Przykłady zasilanych bateryjnie, pojazdów i robotów mobilnych

7.2. Planowanie ruchu robotów mobilnych

Planowanie ruchu ze względu na ograniczone zasoby może odbywać się na wielu płaszczyznach:

- planowanie sterowania napędami z minimalizacją zużywanej energii,
- planowanie ścieżki/trajektorii z minimalizacją energii potrzebnej do jej przebycia,
- planowanie ścieżki/trajektorii z uwzględnieniem możliwości powrotu do stacji dokującej (zapewniające długoterminowe działanie),
- planowanie ścieżki/trajektorii z uwzględnieniem pożądanego sposobu osiągnięcia celu oraz wykorzystanie całej energii (w uogólnieniu związane z maksymalizacją efektów).

7.2.1. Planowanie sterowania napędami z minimalizacją zużywanej energii

Pierwszym podejściem przy planowaniu ruchu jest optymalizacja energii wytracanej na napędach i układzie jezdnym. Do tego celu niezbędne jest wyprowadzenie modelu dynamiki pojazdu pozwalającego określić związek pomiędzy zmiennymi ruchu (prędkość i przyspieszenie), a zużywaną energią. [1]. Modelem dynamiki wiążącym te zmienne jest EVPCM (ang. *electric power consumption model*):

$$P_\tau(t) = v(t)F(t) = v(t)(ma(t) + kv^2(t) + mg\mu) \tag{7.1}$$

gdzie $F(t)$ reprezentuje chwilową siłę oporu ruchu, która zawiera w sobie inercję, opór powietrza oraz opór toczenia – zakładając płaskie podłoże (w równaniach tego modelu stosuje się standardowe oznaczenia na prędkość, masę, przyspieszenie itp.).

Jakkolwiek zaproponowany model zawiera w sobie intuicyjnie najważniejsze elementy modelu, to w przypadku poruszania się z dużą prędkością nie można ignorować ciepła traconego na uzwojeniach silnika.

$$\begin{cases} F(t) = \frac{K\Phi N}{R}. \\ I(t) = KI(t)E(t) = \frac{K\Phi N}{R} \cdot v(t) = Kv(t) \end{cases} \tag{7.2}$$

$$I(t) = \frac{F(t)}{K} \tag{7.3}$$

Całkowita energia tracona przez układ równa się:

$$P[a(t), v(t)] = \frac{r}{K^2}\left(ma(t) + kv^2(t) + mg\mu\right)^2 + v(t)\left(ma(t) + kv^2(t) + mg\mu\right) \tag{7.4}$$

Jak można zauważyć równanie pozwala także uwzględnić ładowanie ogniw podczas hamowania silnikiem, gdyż $P(t) < 0$ jeśli $a(t) < 0$.

Ostatnim elementem, który można wprowadzić do rozważań, jest pochyłość trasy. Ostateczny wzór zawierający nachylenie drogi Θ to:

$$P = \frac{r}{K^2}\left(ma(t) + kv^2 + mg\mu + mg\sin\Theta\right)^2 + v(t)\left(ma(t) + kv^2(t) + mg\mu + mg\sin\Theta\right) \tag{7.5}$$

Z powyższych równań można łatwo wywnioskować, że:

- przede wszystkim im niższa prędkość tym niższy pobór energii,
- zmiany prędkości powinny być łagodne.

7.2.2. Planowanie ścieżki/trajektorii z minimalizacją energii

Kolejnym poziomem, na którym dochodzi do optymalizowania strat energii, jest planowanie trasy robota. Intuicyjnie najlepszą ścieżka jest ta najkrótsza. Problem pojawia się przy nierównościach terenu i omijaniu przeszkód. By temu zaradzić można zdyskretyzować przestrzeń ruchu i wprowadzić koszt przemieszczenia się z jednego punktu do drugiego, budując w ten sposób graf dostępnych stanów.

Tak sformułowane zadanie podobne jest do problemu komiwojażera. Jest on bardzo dobrze opisany w literaturze i istnieje wiele algorytmów oraz ich implementacji pozwalających znaleźć najkrótsze przejście pomiędzy wszystkimi punktami docelowymi. Ważnym aspektem w tego typu zadaniach jest wygenerowanie wymienionego wcześniej grafu dostępnych stanów. Istnieje kilka do tego podejść. Jednym z prostszych jest zastosowanie grafu widzialności, którego wierzchołki rozpięte są na krawędziach przeszkód. Niestety, pojawia się tam problem z płynnym przejściem pomiędzy krawędziami. Należy więc brać pod uwagę konieczność zatrzymywania się lub hamowana robota przy wierzchołkach grafu.

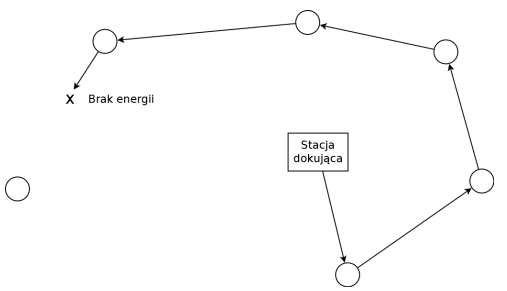
7.2.3. Planowanie ścieżki/trajektorii z uwzględnieniem możliwości powrotu do stacji dokującej

Kiedy robot powinien pracować nieustannie lub gdy jego maksymalny możliwy zapas energii jest niewystarczający do zrealizowania całego zadania jak na rysunku 7.2, należy wprowadzić kolejny poziom planowania ruchu robota. W takim przypadku robot powinien mieć możliwość powrotu do miejsca, gdzie może zregenerować baterię – może to być stacja dokująca.

Stosowanym rozwiązaniem jest statyczne określanie w każdym punkcie, w jakim znajdzie się robot, czy energia jaka mu pozostała jest wystarczająca do powrotu do miejsca ładowania, z określonym marginesem bezpieczeństwa. Jeśli robot ma zbyt mało energii, to natychmiast przerywa swoje działanie i podąża w kierunku stacji dokującej. Po naładowaniu wraca do miejsca, z którego przyjechał i kontynuuje swoje zadanie. Można zauważyć, że taka strategia wprowadza bardzo mało dodatkowego narzutu projektowego i jest stosunkowo łatwa w implementacji. Należy jedynie dodać warunek sprawdzający ilość pozostałej energii w stosunku do drogi powrotnej i oraz określić sposób realizowania drogi do stacji.

Zaproponowany wyżej algorytm nie jest optymalny w większości przypadków. Jeden z nich pokazano na rysunku 7.3. Robot próbując zrealizować przedostatni cel musiał zawrócić w połowie drogi aby zregenerować baterię. Dużo lepszym rozwiązaniem było pozorne nadłożenie drogi by doładować baterię i zrealizować resztę zadania już bez przerwy jak na rysunku 7.3.

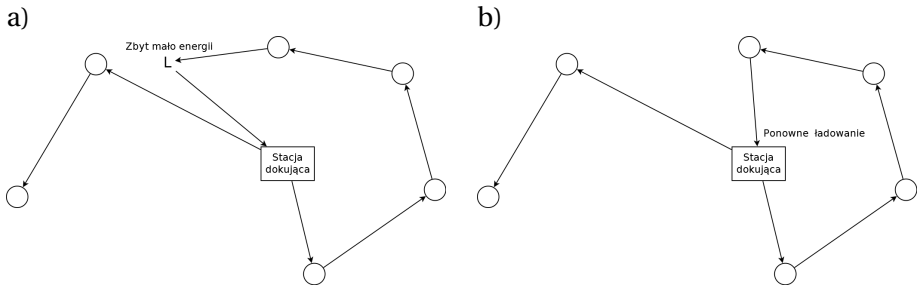
W [2] zaproponowano, by optymalnej drogi poszukiwać za pomocą algorytmów heurystycznych – w szczególności stosując przeszukiwanie tabu. By zwiększyć prędkość algorytmu jako początkowy stan zostaje podany wynik rozwiązania zadania komiwojażera.



Rys. 7.2: Ścieżka robota bez uwzględnienia ograniczeń na energię

7.2.4. Planowanie ścieżki/trajektorii z uwzględnieniem pożądanego sposobu osiągnięcia celu oraz wykorzystanie całej energii

Ostatnim, bardzo rzadko optymalizowanym aspektem ruchu, jest dążenie do zrealizowania wszystkich celów tak, by wykorzystać całą bądź prawie całą dostępną energię. Najczęściej do takiej optymalizacja dochodzi w systemach, w których robotowi nadawana jest pewna energia na początku ruchu w postaci energii



Rys. 7.3: Ścieżka robota z uwzględnieniem a) powrotu do stacji po przekroczeniu krytycznego poziomu baterii b) potrzeby ponownego naładowania robota

kinetycznej. Znaczy to tyle, że pole w sterowaniu prędkością czy przyspieszeniem jest bardzo ograniczone. Taka sytuacja ma miejsce w zrobotyzowanych pociskach rakietowych napędzanych silnikami na paliwo stałe.

7.3. Planowanie ruchu rakiety

7.3.1. Rakieta

Rakietą nazywamy każdy pojazd latający, który siłę napędową potrzebną do wykonania ruchu ruch czerpie korzystając z III prawa dynamiki Newtona. Siła ciągu powstaje w silniku rakietowym, z którego z dużą prędkością wyrzucane są produkty spalania paliwa. By układ zachował pęd całkowity, rakieta musi poruszać się w przeciwnym kierunku, proporcjonalnie do własnej masy i masy ubywającego paliwa, zgodnie z zasadą akcji-reakcji.

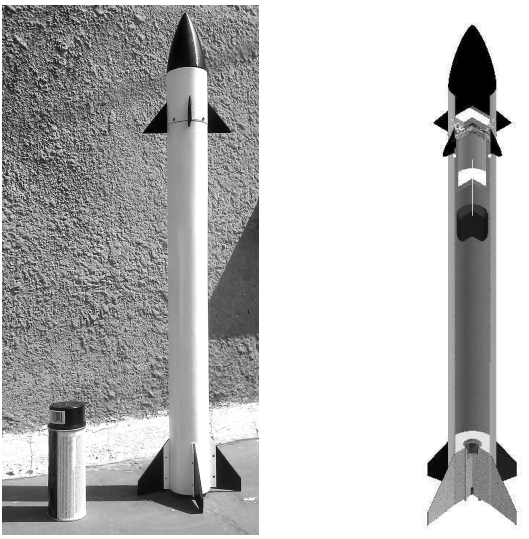
Modelowaną rakietę przedstawiono na rysunku 7.4. Jest ona napędzana silnikiem na paliwo stałe, a do sterowania lotem używa czterech symetrycznie rozłożonych wokół rakiety sterolotek. Rakieta składa się z modułu sterowania oraz korpusu. Cała elektronika oraz silniki wykonawcze zawarte są w module sterowania. W skład modułu wchodzą również takie elementy, jak: AHRS, GPS, barometr, karta SD, moduł radiowy. Taki zestaw funkcji pozwala nazwać rakietę – robotem.

7.3.2. Założenia

Aby można było planować ruch rakiety należy poczynić kilka założeń:

- Rakieta napędzana jest silnikiem na paliwo stałe, którego ciągiem nie można sterować (co określane jest z ang. *fire and forget*).
- Profil mocy silnika jest znany. Producent silnika podaje w dokumentacji siłę ciągu silnika zależną od czasu jego pracy. Testy statyczne wykonywane są na wagach tensometrycznych.
- Model matematyczny rakiety jest znany.
- Położenie rakiety i celu jest znane.
- Zapas energii elektrycznej wystarcza z zapasem na czas lotu rakiety. Pozycjonowanie sterolotek nie jest ograniczone energetycznie.

a) b)



Rys. 7.4: Rakieta a) model rzeczywisty b) model CAD

Najważniejszym z wymienionych elementów jest model rakiety [3]. W najprostszym ujęciu musi on brać pod uwagę następujące siły: siłę grawitacji, siłę ciągu silnika, siły aerodynamiczne (oporu, nośne). Każda z tych sił działa w osobnych lokalnych układach współrzędnych. Dodatkowo siły te mocno zależą od stanu, w jakim aktualnie znajduje się rakieta. Stan ten opisywany może być przykładowo:

- kątem natarcia rakiety, kątem ślizgu rakiety, kątem natarcia sterolotek,
- interwałem czasowym,
- prędkością względem opływającej cieczy,
- temperaturą i wilgotnością powietrza,
- lokalną liczbą Macha.

Uwzględnienie wymienionych sił i stanów implikują duże skomplikowanie modelu. Dlatego sterowanie na podstawie takiego modelu jest bardzo utrudnione. W [3] lotem rakiety steruję się za pomocą sterolotek. W niniejszej pracy natomiast przyjęto dodatkowe założenia:

- ruch kulisty rakiety nie będzie brany pod uwagę,
- sterowanie dotyczyć będzie jedynie pozycji początkowej rakiety.

7.3.3. Cele

Celem planowania ruchu rakiety jest:

- wyznaczenie trajektorii do celu przy ograniczonej energii,
- równoczesna maksymalizacja wysokości lotu,
- maksymalizacja prędkości końcowej.

Można zauważyć, że nie zawsze da się wyznaczyć trajektorię do zadanego celu. Maksymalizacja wysokości lotu wynika z potrzeby ominięcia większości przeszkód terenowych. Maksymalizacja prędkości końcowej implikuje maksymalizację energii kinetycznej, przy zetknięciu rakiety z celem.

Dwa ostatnie cele są ze sobą intuicyjnie sprzeczne, ponieważ biorąc pod uwagę opór aerodynamiczny, trajektoria powinna być jak najkrótsza by wytracić jak najmniej prędkości.

Współcześnie maksymalizacja prędkości końcowej pozostaje istotna, gdy rozważane jest niebezpieczeństwo wymanewrowania pocisku. Natomiast mniejszą przykłada się do niej wagę przy planowaniu wyniku misji, bowiem głowice bojowe nie potrzebują energii kinetycznej do zwiększenia zniszczeń.

Model matematyczny i planowanie ruchu

W pierwszym podejściu skorzystano z całego modelu wyprowadzonego w [3]. Uwzględnia on większość zjawisk występujących podczas lotu rakiety. Ze względu na skomplikowanie, dużą liczbę zmiennych i cztery sterowania – nie jest możliwe wyprowadzenie analitycznego wzoru wiążącego pozycję i trajektorię ze sterowaniem. Należałoby użyć algorytmów heurystycznych. Są one jednak bezużyteczne w przypadku rakiet, ponieważ komputer pokładowy nie posiada wystarczająco dużej mocy obliczeniowej, aby przetworzyć dane przed końcem misji.

W związku z wymienionymi problemami w pracy [3] złożono szczególną propozycję. Podzielono w niej planowanie ruchu rakiety na:

- myszkowanie i kołysanie rakiety,
- dotarcie do celu.

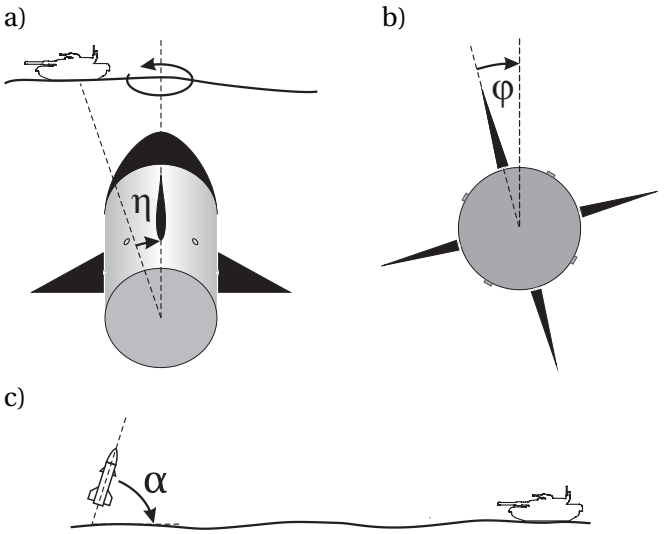
Planowanie w sferze myszkowania i kołysania rakiety (rys. 7.5a i b) sprowadza się do utrzymywania jej kursu oraz stałej orientacji lotek względem układu globalnego. W implementacji sterownika sprowadza się to do:

$$\eta \to 0, \qquad \phi \to 0 \tag{7.6}$$

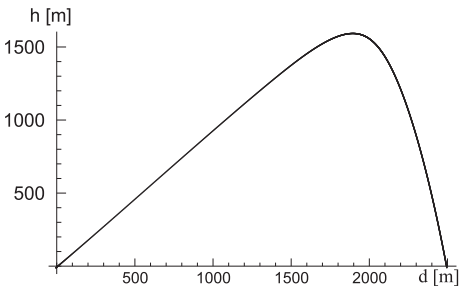
Sterownik do tego celu używa tylko lotki górnej i dolnej. Z wydajnym systemem AHRS i pozycjonowaniem GPS realizacja tego zadania staje się trywialna. Dzięki założeniu, że sterownik utrzymuje (7.6) – lewa i prawa lotka służą tylko i wyłącznie do sterowania rakietą w jednej płaszczyźnie (zobacz rysunej 7.5c).

Również ten przypadek można podzielić. Można uznać rakietę za zwykły pocisk balistyczny i, bez wprowadzania jakichkolwiek sterowań w płaszczyźnie dotarcia do celu, ustawić rakietę pod odpowiednim kątem startowym jak na rysunku 7.5a. Poleci ona wtedy trajektorią balistyczną (rys. 7.6) obliczoną na podstawie modelu [3], w którym zostały wyłączone sterowania.

Idąc krok na przód, dla jednej serii rakiet – to znaczy z tymi samymi właściwościami, a zwłaszcza z tym samym modelem silnika – można podjąć próbę wyznaczenia zależność odległości od kąta wystrzału rakiety. Model wyznaczonej krzywej powinien być dokładny, gdyż przy odległościach na jakie leci rakieta i wielkości celu pomyłka rzędu kilku metrów jest niedopuszczalna. Z tego powodu



Rys. 7.5: Planowanie w sferze a) myszkowania, b) kołysania c) odległości od celu



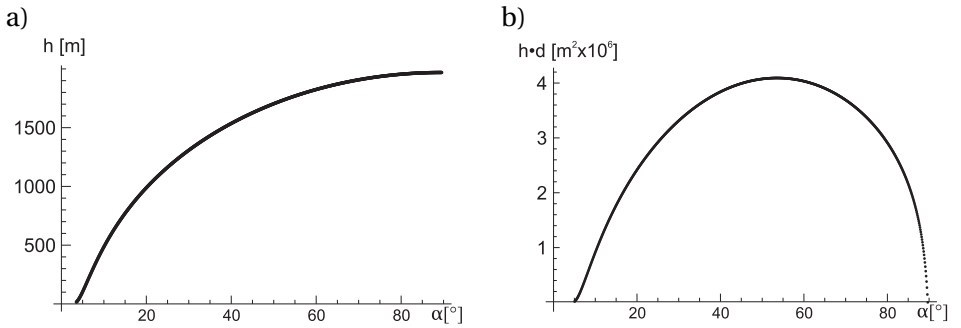
Rys. 7.6: Przykładowa trajektoria wygenerowana przez model

uzyskanej krzywej 7.6 nie da się efektywnie sparametryzować – efektywniej jest zapamiętać jej kształt w tablicy o bezpośrednim dostępie (ang. *lookup table*) i interpolować pożądane odległości i kąty. Można zauważyć, że maksymalny zasięg rakiety nie wynosi 45^o jak dla rzutu skośnego, tylko około 25^o.

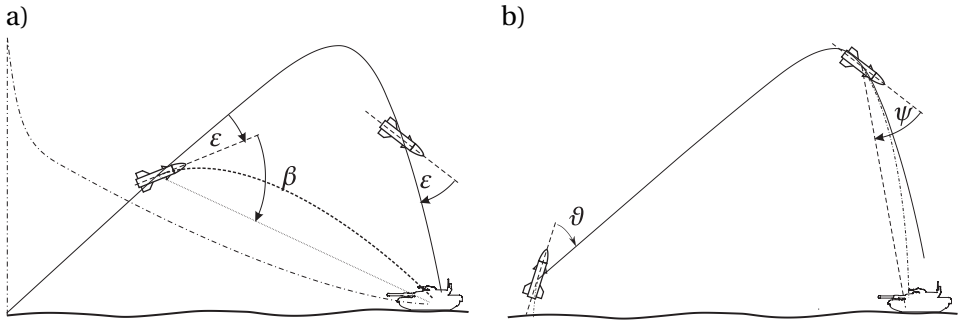
Można także wyznaczyć osiąganą przez rakietę wysokość, co pokazano na rysunku 7.7a. Jeśli zachodzi potrzeba pogodzenia ze sobą tych w pewnym stopniu sprzecznych celów, można wyznaczyć zależność ich iloczynów od kąta startu. Przykładowa zależność pokazana jest na rysunku 7.7b.

Mając na uwadze wymagania dokładnościowe można zauważyć, że platforma startowa rakiety musiałaby umożliwiać ustawianie kąta startu z bardzo dużą dokładnością. Dodatkowo, każde zaburzenie powodowałoby nietrafienie w cel. Jest to koszt otwartej pętli sterowania.

Nie należy jednak definitywnie skreślać tej metody. Można wykorzystać trajektorię i kąt startowy jako warunki początkowe, ułatwiające realizację misji. Powyższą metodę można ulepszyć. Po pierwsze, trajektorię balistyczną, zaznaczoną



Rys. 7.7: Zależność a) maksymalnego pułapu od kąta startu rakiety, b) iloczynu maksymalnego pułapu i zasięgu rakiety od kąta jej startu



Rys. 7.8: Wyznaczanie trajektorii lotu rakiety: a) trzy podejścia, b) podejście praktyczne

na rysunku 7.8 linią ciągłą, rakieta może śledzić za pomocą sterolotek – minimalizując kąt ϵ. Podejście to ma jednak wadę – można wytracić zbyt dużo energii na powrót na właściwą trajektorię. Dlatego kolejnym pomysłem jest zaprzestanie śledzenia trajektorii balistycznej po osiągnięciu pewnego pułapu. Przypadek ten zobrazowano na rysunku 7.8 linią kreskowaną. Należy wyznaczyć promień wiodący do celi i minimalizować kąt β.

Przedstawiony pomysł nie rozwiązuje problemu ustawienia rakiety pod odpowiednim kątem. Jego rozwiązaniem może być trajektoria zaznaczona na rysunku 7.8a) linią kropkowaną. Strzelec powinien ustawić rakietę pionowo lub lekko przechyloną w kierunku celu. Rakieta po osiągnięciu apogeum osiągnie zadany cel autonomicznie nurkując. Wadą tego rozwiązania jest potrzeba instalowania odpowiednio dużych sterolotek potrafiących wygenerować odpowiednią siłę nośną.

Na rysunku 7.8b) przedstawiono rozwiązanie wszystkich problemów, pod warunkiem, że rakieta posiada pierwszy stopień startujący. Strzelec może ustawić ją pionowo, przechyloną w kierunku celu. Po odpaleniu pierwszego stopnia – rakieta wytraciwszy prędkość, ustawia się pod kątem optymalnej trajektorii balistycznej. Dzięki temu będzie mogła zmieniać położenie przy optymalnej prędkości (zbyt mała prędkość spowoduje zbyt mały moment siły, zaś wychylenia lotek

przy zbyt dużych prędkościach mogą powodować niestabilny lot i duży opór aerodynamiczny). Następnie rakieta ma lecieć na właściwym silniku utrzymując kurs. Osiągnąwszy apogeum wyznacza promień wiodący do celu i minimalizuje kąt między nim, a osią rakiety. Rakieta atakując w ten sposób omija większość przeszkód terenowych, ma bardzo duży zapas energii potencjalnej oraz atakuje od góry – strony najczęściej najgorzej opancerzonej.

7.4. Wnioski

W niniejszym dokumencie przedstawiono zagadnienia związane z planowaniem ruchu przy ograniczeniach na ilość zużywanej energii. W szczególności przedstawiono metodę planowania ruchu rakiety, polegającej na przekształcenie problemu skomplikowania modelu do kilku prostszych podproblemów.

Minimalizację energii możemy obserwować na co dzień. Dzięki niej, wykonywane przez nas ruchy, wydają się nam zgrabne. Nie jest to jednak zadanie łatwe. Bez podświadomej potrzeby oszczędzania sił i podświadomego wykonywania ruchów, prawdopodobnie nie bylibyśmy w stanie efektywnie chodzić. Jeśli chcemy by dokonywał się postęp w robotyce – z założenia zmierzający ku jak najlepszemu upodobnieniu się do ludzkiej wszechstronności i adaptacyjności – powinniśmy skierować więcej wysiłków ku rozwijaniu tej dziedziny planowania.

Literatura

[1] Tao Wang, Christis G. Cassandras, Sepideh Pourazarm. Optimal motion control for energy-aware electric vehicles. *Control Engineering Practice*, 2014.

[2] H. Wei, B. Wang, Y. Wang, Z. Saho, K. Chan. Staying-alive path planning with energy optimization for mobile robots. *Expert Systems with Applications*, 2002.

[3] J. Drewniak. Modelowanie i wizualizacja rakiety modelarskiej. Praca magisterska, Politechnika Wrocławska, 2015.

ROZDZIAŁ 8

ROZPOZNAWANIE OBIEKTÓW 3-WYMIAROWYCH W CHMURZE PUNKTÓW

F. Grzeszczak, M. Ulita

W niniejszym rozdziale przedstawiono wybrane metody przetwarzania chmur punktów celem rozpoznawania obiektów trójwymiarowych. Określono w nim wymagania stawiane względem wykorzystywanych danych oraz strumieni przetwarzania wspierających efektywną klasyfikację.

8.1. Wprowadzenie

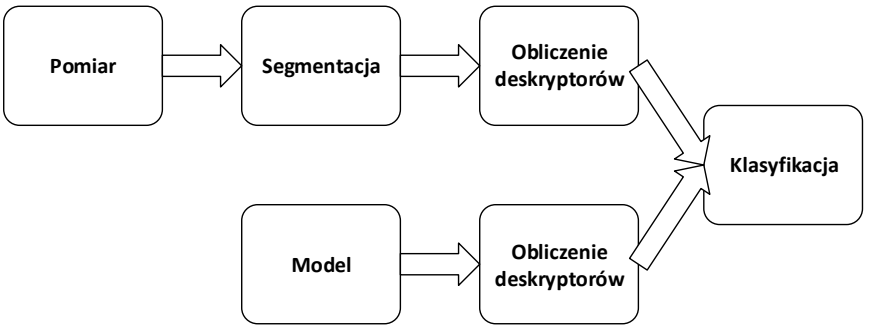
Pojęcie chmury punktów odnosi się do ustrukturyzowanego zbioru punktów, reprezentowanego w pewnym układzie współrzędnych, uzyskiwanego zazwyczaj z pomiarów wykonanych czujnikiem głębi, dalmierzem laserowym lub kamerą 3D. Chmury punktów mogą posłużyć do rekonstrukcji trójwymiarowego modelu skanowanego obiektu lub sceny. Mogą również służyć, przy zastosowaniu metod uczenia maszynowego, do rozpoznawania obiektów o znanych, trójwymiarowych modelach. Rozwijając metody analizy chmur punktów można przyczynić się do lepszej percepcji świata przez maszyny i urządzenia.

Modele stosowane podczas rozpoznawania obiektów mogą być pozyskane na dwa sposoby: przez wykonanie dokładnego skanu rzeczywistych obiektów lub przez przeprowadzenie operacji śledzenia promieni (ang. *raytracing*) na modelach obiektów stworzonych w oprogramowaniu CAD. Do poprawnej i efektywnej klasyfikacji obiektów na ich podstawie konieczne jest zastosowanie metod umożliwiających redukcję i uogólnienie danych (niezależnie od sposobu pozyskania modeli). Co więcej, metody te powinny pozwalać na możliwie precyzyjne opisywanie obiektów i być odporne na zakłócenia pomiarowe, różnice w częstotliwości próbkowania oraz w położeniu i orientacji obiektów.

8.2. Metody przetwarzania chmury punktów

Klasyfikacja chmur punktów wymaga wstępnego przetworzenia danych w celu redukcji ich ilości i generalizacji. Aby tego dokonać wykorzystuje się różne

metody segmentacji (pozwalające na wstępne uporządkowania punktów) oraz deskryptory (służące do opisu własności skupisk punktów uzyskanych na etapie segmentacji). Podstawowy potok przetwarzania chmury punktów podczas klasyfikacji przedstawiono na rysunku 8.1.



Rys. 8.1: Potok przetwarzania chmury punktów

8.2.1. Segmentacja

Segmentacja to proces podziału obrazu, w tym przypadku chmury punktów, na obszary jednorodne pod względem wybranych kryteriów. Kryteriami mogą być odległości pomiędzy punktami, ich normy lub inne własności, jak np. tekstura. Segmentacji chmury punktów służyć ma uproszczeniu zagadnienia w taki sposób, aby ułatwić dalsze przetwarzanie. Możliwa jest segmentacja punktów do wybranych brył geometrycznych, co znacząco może ułatwić klasyfikację, zwłaszcza jeśli szukany obiekt ma kształt zbliżony do wybranej bryły [1].

Segmentacja euklidesowa

Metoda ta należy do najprostszych. Punkty są grupowane w obszary według odległości euklidesowej. Jeśli odległość pomiędzy dwoma punktami jest mniejsza niż przyjęta wartość progowa, punkty są zaliczane do tego samego obszaru. W przeciwnym wypadku tworzony jest nowy obszar i punkty w pobliżu są sprawdzane w ten sam sposób.

Rozrost obszarów

W pewnym stopniu metoda rozrostu obszarów jest podobna do metody odległości euklidesowych. Jako kryterium przyjmowane jest charakter krzywych tworzonych przez punkty (ang. *smoothness constraint*). Jeśli krzywizna będzie większa niż przyjęty próg, punkty zostaną zaliczone do odrębnych obszarów.

Algorytm RANSAC

RANSAC (ang. *Random sample consensus*) to iteracyjny algorytm pozwalający na modelowanie i wyodrębnianie płaszczyzn w zbiorze danych zawierającym podłoże lub punkty nienależące do modelowanej powierzchni. W pierwszym kroku procedury RANSAC losowane są minimalne zbiory punktów, dla których wyznaczany jest model w postaci równania rozpiętej przez nie płaszczyzny. W kolejnych krokach modele te są modyfikowane przez dodawanie kolejnych punktów do utworzonych zbiorów i sprawdzanie, czy dodany punkt spełnia kryterium maksymalnej odległości od rozpatrywanej płaszczyzny. Skuteczność algorytmu rośnie wraz z liczbą iteracji. Podczas przetwarzania chmur punktów RANSAC jest wykorzystywany głównie do segmentacji i odfiltrowania punktów należących do podłoża. Redukuje to znacząco ilość danych do dalszego przetwarzania oraz poprawia skuteczność innych metod segmentacji.

8.2.2. Deskryptory

Na potrzeby klasyfikacji klastrów otrzymanych w wyniku segmentacji niezbędne jest wyznaczenie ich charakterystycznych cech. W przypadku rozpoznawania obiektów trójwymiarowych w chmurze punktów cechy te powinny być:

- niewrażliwe na transformacje obrazu,
- niewrażliwe na szum/błędy pomiarowe,
- odporne na różnice w rozdzielczości i innych parametrach, wynikające z wykorzystania różnych sensorów.

Między innymi takimi cechami odznaczają się opisane dalej deskryptory.

PFH

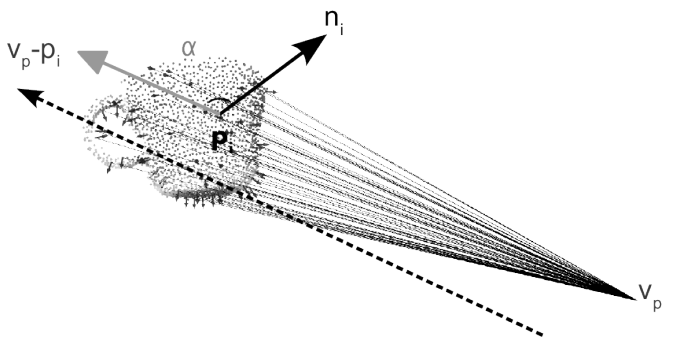
Deskryptor PFH (ang. *Point Feature Histogram*) jest wyliczany w drodze analizy geometrii klastra w otoczeniu wybranych punktów na podstawie kątów pomiędzy wektorami normalnymi sąsiednich punktów. Kąty pomiędzy wektorami normalnymi są porównywane nie tylko dla wybranego punktu startowego i jego punktów sąsiednich, ale również pomiędzy resztą punktów. Na podstawie wyznaczonych kątów oraz odległości euklidesowych pomiędzy punktami tworzone są cztery histogramy. Wynikowy wektor cech dla analizowanego klastra tworzony jest przez dodanie tych czterech histogramów.

PFH daje bardzo dokładne wyniki, w szczególności minimalizuje ilość błędnych trafień podczas klasyfikacji. Wynika to z faktu, że działając lokalnie tworzy wektory cech dobrze opisujące geometrie całego obiektu [2]. Z drugiej strony uzyskane wektory mają wymiar n^4, gdzie n to liczba przyjętych punktów startowych.

VFH

Deskryptor VFH (ang. *Viewpoint Feature Histogram*) w odróżnieniu od PFH zaliczany jest do grupy deskryptorów globalnych. Zamiast wyznaczania cech dla

zadanej liczby podobszarów, wektor cech wyznaczany jest od razu dla całego klastra. Jako punkt startowy wybierany jest punkt najbliższy centroidzie klastra. Najpierw wyznaczany jest znormalizowany wektor sensor-centroid. Następnie tworzony jest histogram kątów pomiędzy wektorami normalnymi punktów w klastrze, a wyznaczonym wcześniej wektorem. Pokazano to na rysunku 8.2. Ostatnim etapem jest wyznaczenie kątów pomiędzy wektorami normalnymi centroidy i resztą punktów, podobnie jak w deskryptorze PFH. W efekcie otrzymywane są trzy histogramy. Końcowy wektor cech tworzony jest przez dodawanie wszystkich trzech histogramów [3].



Rys. 8.2: Wektor sensor-centroid w deskryptorze VFH

SHOT

W algorytmie SHOT (ang. *Signature of Histograms of Orientations*) konstruowana jest sfera, której centroidą jest punkt, dla którego otoczenia wyznaczany jest deskryptor. Przyjmuje się, że biegun północny wyznaczonej sfery jest zgodny z kierunkiem wektora normalnego dla rozpatrywanego punktu. Następnie sfera dzielona jest na 32 części, wyznaczone przez osiem linii biegnących od bieguna południowego do północnego oraz jedną prostopadłą. Dla każdej z nich wyznaczany jest histogram kątów pomiędzy punktami znajdującymi się w danej części, a wektorem normalnym punktu startowego. Wynikowy deskryptor jest złożeniem wszystkich histogramów uzyskanych w ten sposób.

8.2.3. Klasyfikacja

Ostatnim etapem podczas rozpoznawania obiektów w chmurze punktów jest klasyfikacja z wykorzystaniem cech uzyskanych dzięki opisanym wcześniej deskryptorom. Porównywane są cechy skupisk wykrytych na przetwarzanej scenie po jej segmentacji z cechami modeli referencyjnych obiektów rozpoznawanych. Możliwe jest wykorzystanie prawie dowolnych metod klasyfikacji, w tym jednej z najprostszych: algorytmu k-najbliższych sąsiadów (kNN) [4], gdzie klasyfikacja polega na porównaniu odległości między cechami obiektu rozpoznawanego, a cechami obiektów referencyjnych (sąsiadami). Obiekt jest klasyfikowany poprzez wyznaczenie najbliższego z sąsiadów wg wybranej metryki.

8.3. Implementacja i testy wybranych algorytmów

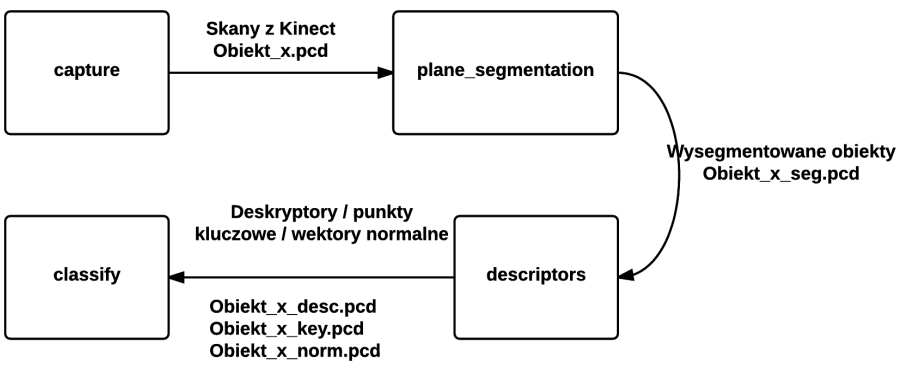
8.3.1. Środowisko testowe

Do implementacji algorytmów klasyfikacji obiektów trójwymiarowych w chmurach punktów wykorzystano bibliotekę PCL (ang. *Point Cloud Library*). Jako źródło danych pomiarowych posłużył sensor głębi Kinect.

Kinect to jeden z najpopularniejszych i łatwo dostępnych sensorów. Jego działanie polega na projekcji siatki podczerwieni, której punkty są rejestrowane przez kamerę z filtrem podczerwonym. W ten sposób pozyskiwana jest informacja o głębi obrazu i możliwe jest zrekonstruowanie informacji o położeniu poszczególnych pikseli oraz utworzenie sześciowymiarowej chmury punktów: XYZRGB [5]. PCL to open-source'owa biblioteka programowa, stworzona w laboratorium badawczym Willow Garage, dostarczająca zestawu narzędzi do przetwarzania chmur punktów w postaci API dla języków programowania C++ i Python. Wykorzystanie popularnych sensorów głębi do pozyskiwania chmur punktów jest możliwe w PCL dzięki dostarczanemu interfejsowi PCL – OpenNI.

8.3.2. Oprogramowanie

Na potrzeby testów stworzono pakiet oprogramowania składający się z podprogramów: `capture`, `plane_segmentation`, `descriptor` oraz `classify` uruchamianych w określonej kolejności (rys. 8.3).



Rys. 8.3: Oprogramowanie do rozpoznawania obiektów

Pierwszy z nich umożliwia zapisywanie do pliku chmury punktów zarejestrowanych przez sensor Kinect. Wykorzystano w nim interfejs `OpenNIGrabber` z biblioteki PCL. Program można uruchomiać w trybie zapisu ręcznego (zapis inicjowany naciśnięciem odpowiedniego przycisku) lub w trybie zapisu cyklicznego (zapis wykonywany jest cyklicznie, z zadanym interwałem czasowym).

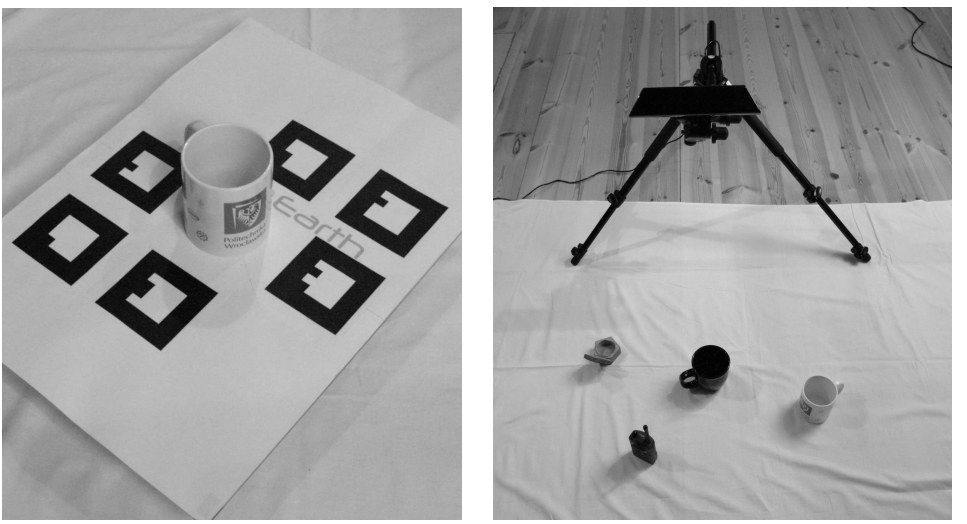
Drugi służy do wyodrębnienia obiektu z chmury punktów. Wykonuje on dwuetapową segmentację z wykorzystaniem algorytmu RANSAC (do usunięcia podłoża) oraz SAC (do wyodrębnienia obiektów o kształcie cylindrycznym).

Trzeci zapewnia realizację kolejnego etapu przetwarzania. Wyliczane są w nim deskryptory obiektów z wykorzystaniem algorytmu SHOT.

Posiadając wyodrębnione obiekty oraz wartości odpowiadających im deskryptorów możliwe jest ich klasyfikowanie i lokalizacja na scenie za pomocą programu czwartego. Program ten przyjmuje na wejście co najmniej dwa pliki: plik z chmurą punktów pochodzącą ze skanu sceny oraz przynajmniej jeden plik z modelem poszukiwanego obiektu. W pierwszym kroku ze sceny usuwane jest podłoże (z wykorzystaniem RANSAC) oraz wyliczane są wartości deskryptora dla pozostałych obiektów w chmurze punktów. Następnie porównywane są deskryptory modeli i sceny z wykorzystaniem algorytmu k-najbliższych sąsiadów. Jeśli któreś ze skupisk na scenie zostanie sklasyfikowane jako poszukiwany obiekt, skan obiektu jest dopasowywany do obiektu znajdującego się na scenie z wykorzystaniem transformacji Hougha. Wynikiem klasyfikacji jest macierz transformacji z układu lokalnego kamery do układu wykrytego obiektu oraz poziom pewności tej klasyfikacji.

8.3.3. Przygotowanie modeli rozpoznawanych obiektów

Przygotowanie modeli obiektów podlegających rozpoznawaniu wymagało wykonania skanów rzeczywistych obiektów z różnych ujęć (co miało zapewnić jak najlepsze odwzorowanie obiektu w tworzonym modelu). Stworzono więc stanowisko składające się z widocznej na rysunku 8.4 obrotnicy z naniesionymi na nią markerami, nieruchomego sensora Kinect oraz oświetlenia. Skan obiektu



Rys. 8.4: Obrotnica wykorzystana przy tworzeniu modeli (z lewej) oraz stanowisko pomiarowe użyte podczas skanów klasyfikowanych obiektów (z prawej)

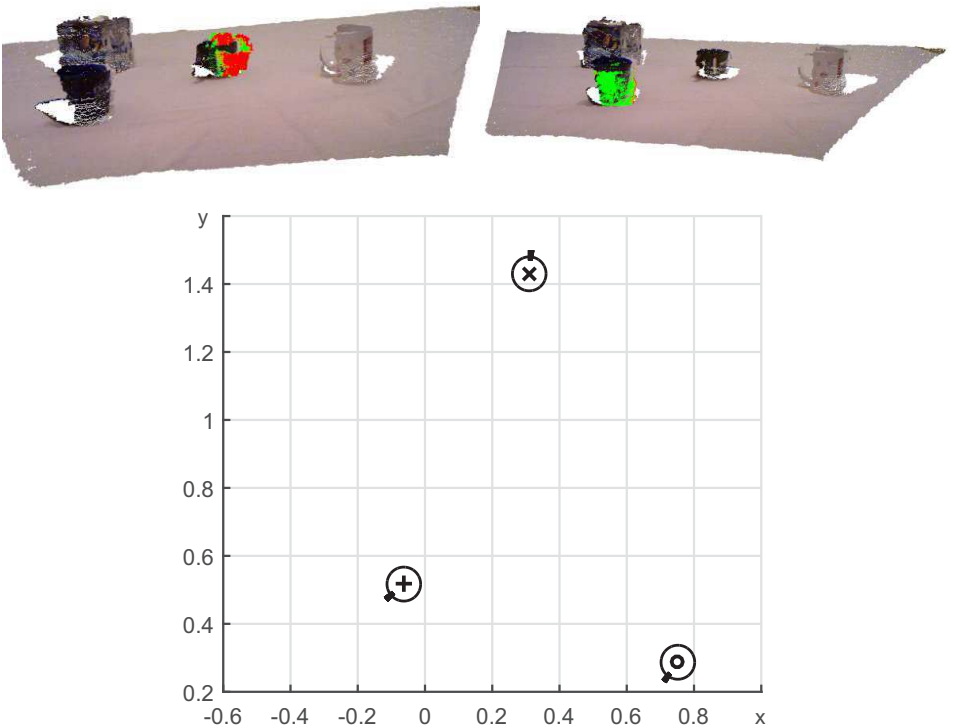
był wykonywany przez pobranie od 50 do 100 chmur punktów podczas obracania obiektem na obrotnicy. Następnie pobrane chmury punktów były poddawane segmentacji w celu usunięcia pomiarów nieodpowiadających skanowanemu obiektowi (efekt widać na rysunku 8.5).

Rys. 8.5: Przykłady wyodrębniania obiektów oraz segmentacji sceny

8.3.4. Walidacja

Badania przeprowadzono dla sceny zawierającej wszystkie uprzednio skanowane obiekty oraz dodatkowo jeden obiekt wcześniej nieznany. Wykonano pięć ujęć sceny w osi głównej oraz $\pm 45°$ i $\pm 90°$. Podczas jednego pełnego obrotu na obrotnicy dla każdego skanowanego obiektu wykonano bazę stu modeli. Dla każdej sceny i wszystkich zestawów modeli obiektu wykonano po czterdzieści wywołań programu rozpoznającego.

Na rysunku 8.6 pokazano rozkład dopasowanych obiektów na płaszczyznowe podłoża. Mimo małych odległości pomiędzy przedmiotami środek obiektu dopasowanego znajdował się w bardzo bliskiej odległości od środka obiektu na scenie. Dla zastosowanego zestawu przedmiotów we wszystkich przypadkach algorytm poprawnie klasyfikował typ obiektu, niezależnie od orientacji sceny względem kamery. Podczas dopasowania orientacji modelu do rozpoznanego obiektu na scenie pojawiały się liczne pomyłki, najprawdopodobniej wynikały z niedostatecznej dokładności skanów ze zbioru uczącego.



Rys. 8.6: Przykład wyników rozpoznawania oraz rozłożenia dopasowanych obiektów

8.4. Podsumowanie

Zaprezentowana metoda umożliwia rozpoznawanie obiektów trójwymiarowych w chmurze punktów oraz ich klasyfikację na podstawie właściwości geometrycznych. Umożliwia również lokalizację takich obiektów w przestrzeni, co może być szczególnie przydatne w systemach percepcji robotów.

Omówione zagadnienia rozpoznawania i klasyfikacji nie należą do zagadnień łatwych. Ich realizacja wymaga wieloetapowego przetwarzania danych oraz dużych nakładów obliczeniowych. Na efektywność klasyfikacji duży wpływ, oprócz zastosowanych algorytmów, ma jakość modeli wykorzystywanych do uczenia klasyfikatora. Stworzenie dobrego modelu zaś wymaga wykonania wielokrotnych pomiarów obiektu, w różnej orientacji względem sensora. Istotne przy tym jest również dokładne wyodrębnienie obiektów przez usunięcie punktów pochodzących od innych elementów znajdujących się w zasięgu pomiaru. Wykorzystanie tak stworzonego modelu oraz deskryptorów odpornych na transformacje geometryczne umożliwia efektywne rozpoznawanie i lokalizowanie obiektów w chmurze punktów. Koszt obliczeń może być redukowany przez filtrowanie pomiarów, jednak nie pozostaje to bez wpływu na jakość klasyfikacji.

Literatura

[1] G. Hetzel, B. Leibe, P. Levi, B. Schiele. 3D object recognition from range images using local feature histograms. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, wolumen 2, strony II–394. IEEE, 2001.

[2] R.B. Rusu, N. Blodow, M. Beetz. Fast point feature histograms (FPFH) for 3D registration. *IEEE International Conference on Robotics and Automation (ICRA'09)*, strony 3212–3217. IEEE, 2009.

[3] R.B. Rusu, G. Bradski, R. Thibaux, J. Hsu. Fast 3d recognition and pose using the viewpoint feature histogram. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)*, strony 2155–2162. IEEE, 2010.

[4] D.G. Lowe. Object recognition from local scale-invariant features. *Proceedings of the 7th IEEE International Conference on Computer Vision*, wolumen 2, strony 1150–1157. IEEE, 1999.

[5] A. Aldoma, M. Vincze, N. Blodow, D. Gossow, S. Gedikli, R.B. Rusu, G. Bradski. CAD-model recognition and 6DOF pose estimation using 3D cues. *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, strony 585–592, Listopad 2011.

ROZDZIAŁ 9

OKREŚLANIE LOKALIZACJI ŹRÓDŁA DŹWIĘKU Z WIELOPUNKTOWEGO NASŁUCHU

M. Drwięga

W rozdziale przedstawiono podstawowe metody lokalizacji źródeł dźwięku oraz opisano konstrukcję kołowego robota mobilnego wyposażonego w system lokalizacji źródeł dźwięku i system nawigacji.

9.1. Wprowadzenie

Systemy lokalizacji źródeł dźwięku są stosowane głównie w budowie interfejsów pozwalających człowiekowi na interakcje z maszyną. Wykorzystuje się je również w wielu innych aplikacjach. Przykładem może tu być zintegrowany system pozwalający na odtwarzanie dźwięków jedynie z wybranych obszarów otoczenia [1], składający się ze słuchawek, mikrofonów oraz układów przetwarzania sygnałów. lub systemy konferencyjne, w których kamera podąża za mówcą [2]. Prowadzone są także prace w kierunku zwiększenia skuteczności rozpoznawania mowy, bazując na dodatkowej informacji w postaci położenia użytkownika [3].

Lokalizacja dźwięków bywa wykorzystywana w rozwiązaniach militarnych i robotycznych. Służy na przykład jako narzędzie do namierzania strzelających dział samobieżnych czy chociażby snajperów. Systemy tego typu umieszczane są w hełmach żołnierzy, co pozwala im na bieżąco śledzić sytuację na polu walki [4, 5]. Tego typu lokalizacja jest obiektem zainteresowania badaczy zajmujących się robotami humanoidalnymi [6], dążących do stworzenia naturalnych kanałów komunikacji człowieka z maszyną (czyli komunikacji za pomocą mowy).

W wielu aplikacjach stosowane są całe matryce mikrofonów. Okazuje się bowiem, że w niektórych przypadkach rozwiązania naśladujące układ słuchowy człowieka są niewystarczające. Ma to miejsce np. przy występowaniu zakłóceń pochodzących z napędów czy innych źródeł dźwięku. Aby poradzić sobie z tymi problemami twórcy robota SIG [6] wykorzystali dodatkowe mikrofony umieszczone wewnątrz robota. Ich zadaniem była rejestracja dźwięków pochodzących

od wbudowanych mechanizmów. Zebrane dane dały podstawę do przeprowadzenia filtracji, dzięki czemu dźwięki rejestrowane z otoczenia mogły być uzyskane ze znacznie mniejszymi zakłóceniami.

Kolejnym zastosowaniem metod lokalizacji źródeł dźwięku jest sterowanie robotami mobilnymi w oparciu o wykryte położenie dźwięku. Jego wariantem jest podejście polegające na podążaniu robota za dźwiękiem o określonej częstotliwości, którego źródło pełni rolę znacznika. Rozwiązanie takie może być wykorzystane na przykład do naprowadzania robota na stację ładowania. Oczywiście nawigacja w kierunku zlokalizowanego źródła dźwięku musi uwzględniać omijanie pojawiających się na drodze robota przeszkód.

Innym, ciekawym ale też bardziej złożonym podejściem jest realizacja przywoływania robota za pomocą komend głosowych. Zastosowanie to jest o tyle wymagające, że oprócz lokalizowania źródła dźwięku wymagane jest tu uruchomienie systemu rozpoznawania mowy w celu wykrycia wypowiedzianych komend.

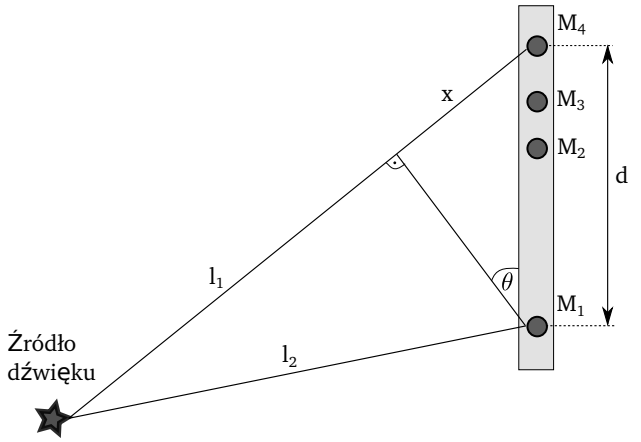
9.2. Podstawowe metody lokalizacji dźwięku

9.2.1. Międzyuszna różnica czasu – ITD

Wiele znanych metod lokalizacji dźwięku bazuje na obserwacjach systemów słuchowych występujących w przyrodzie, między innymi układu słuchowego człowieka. Do takich metod należy metoda pomiaru międzyusznej różnicy czasu ITD (ang. *Inter-aural Time Difference*), określana czasami mianem międzyusznej różnicy faz IPD (ang. *Inter-aural Phase Difference*) lub metodą różnicy czasu przybycia sygnałów TDOA (ang. *Time Difference Of Arrival*) [7, 8, 9]. Metoda opiera się na pomiarze różnicy czasu w docieraniu dźwięku do jednego i drugiego ucha. Jest to związane z pewną odległością między uszami oraz stosunkowo niewielką prędkością dźwięku w powietrzu, wynoszącą około 343 m/s przy temperaturze 20^oC. Znając prędkość dźwięku w ośrodku, odległość między odbiornikami oraz różnicę w czasach otrzymania sygnałów, można określić kierunek dźwięku na płaszczyźnie poziomej. Jeżeli jednak źródło dźwięku znajduje się dokładnie między dwoma odbiornikami, wtedy różnica czasu nie występuje.

Na rysunku 9.1 zilustrowano przypadek określania kierunku źródła dźwięku względem liniowej matrycy mikrofonów. Jedno z podejść polega na obliczeniu kierunku dla każdej z par mikrofonów, a następnie na uśrednieniu otrzymanych wyników. Postępowanie dla jednej pary mikrofonów M_1 i M_4 jest następujące. Znając czasy przelotu fali dźwiękowej oblicza się długości $l_1 + x$ oraz l_2. Zakłada się dodatkowo, że źródło dźwięku jest znacznie oddalone od matrycy i odcinki l_1 i l_2 są prawie równoległe, a $l_1 \approx l_2$ [10]. Można wtedy obliczyć kierunek źródła dźwięku za pomocą poniższego wzoru.

$$\theta = \cos\left(\frac{x}{d}\right) \tag{9.1}$$



Rys. 9.1: Określanie kierunku źródła dźwięku z liniową matrycą mikrofonów [10]

9.2.2. Międzyuszna różnica poziomów – ILD

Kolejną metodą opierającą się na biologicznych obserwacjach jest międzyuszna różnica poziomów ILD (ang. *Inter-aural Level Difference*). Metoda ta bazuje na pomiarze względnej różnicy energii sygnałów docierających do matrycy mikrofonów [7]. Do określenia kierunku źródła dźwięku wykorzystuje się fakt, że energia sygnału docierającego do mikrofonu bardziej oddalonego od źródła jest mniejsza, niż w przypadku mikrofonu znajdującego się bliżej. Metoda ta stosowana jest zwykle do mikrofonów, które znajdują się wokół pewnego obiektu tłumiącego sygnały (u człowieka sygnały tłumione są przez głowę). Technika ILD szczególnie dobrze nadaje się do sygnałów dźwiękowych o wysokich częstotliwościach, gdyż są one znacznie bardziej tłumione przez obiekty niż sygnały o niskich częstotliwościach. Gdy długości fal dźwiękowych są porównywalne lub dłuższe od obiektu, wokół którego umieszczono sensory (u człowieka głowa), wtedy zależnie od kierunku pochodzenia dźwięku i z powodu zjawiska cienia akustycznego do niektórych sensorów dochodzi znacznie stłumiona fala dźwiękowa.

9.2.3. Metoda kierunkowych mikrofonów

Metoda wykorzystuje mikrofony kierunkowe, czyli takie rozwiązanie sprzętowe, które pozwala na odbieranie dźwięku jedynie w niewielkim kącie przestrzeni. Mikrofony rozmieszcza się w taki sposób, aby każdy z nich obejmował swoim zasięgiem pewną część otoczenia [7].

9.3. Wykorzystywane narzędzia

9.3.1. Sensor Kinect

Sensor Microsoft Kinect (rys. 9.2) wyprodukowano jako urządzenie pełniące funkcję kontrolera gier, które nie wymaga bezpośredniego, fizycznego kontaktu

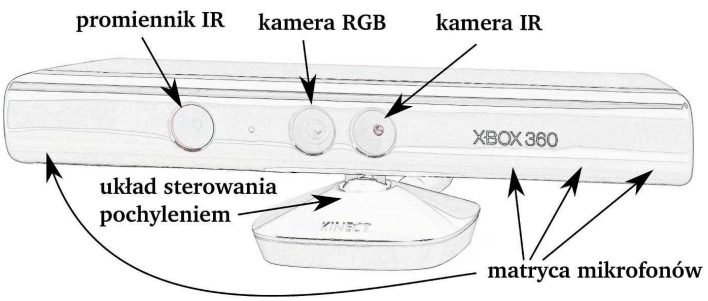
Rys. 9.2: Sensor Microsoft Kinect

z użytkownikiem. Ze względu na niewielką cenę oraz znaczne możliwości pobierania danych z otoczenia urządzenie to stało się niezwykle popularne w wielu zastosowaniach, bardzo często robotycznych. Wykorzystuje się je między innymi do nawigacji robotów mobilnych, sterowania manipulatorami itp.

Czujnik Kinect wykorzystuje metodę światła strukturalnego [11] w celu tworzenia mapy głębi, a zaimplementowane w nim algorytmy umożliwiają śledzenie poruszających się osób.

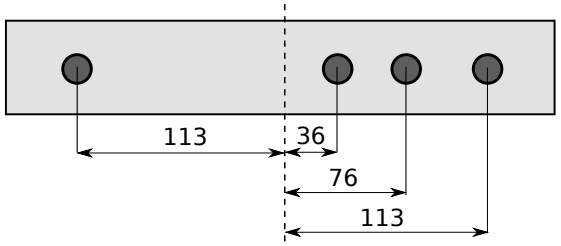
Zasięg urządzenia, w zależności od wersji, wynosi: 0,8 m – 4 m w przypadku wersji *XBOX 360* oraz 0,4 – 4 m dla *Kinect for Windows*. Kąty widzenia to około 43^o w pionie oraz 57^o w poziomie. Połączenie z komputerem odbywa się poprzez interfejs USB 2.0. W skład czujnika wchodzą następujące komponenty (rys. 9.3):

- kamera IR (podczerwieni),
- promiennik podczerwieni,
- kamera RGB,
- układ sterujący pochyleniem sensora (zawierający akcelerometr),
- matryca mikrofonów,
- jednostka obliczeniowa.



Rys. 9.3: Budowa sensora Kinect

Sensor posiada matrycę czterech mikrofonów [12]. Układ próbkujący sygnały dźwiękowe składa się z 24-bitowego przetwornika analogowo-cyfrowego (A/C). Dźwięk kodowany jest w postaci 24-bitowego sygnału PCM (ang. *Pulse Code Modulation*) o częstotliwości 16 kHz. Rozmieszczenie poszczególnych mikrofonów przedstawiono na rysunku 9.4.



Rys. 9.4: Rozmieszczenie mikrofonów w sensorze Kinect [10]

9.3.2. Biblioteka HARK

HARK (ang. ***H**onda Research Institute Japan **A**udition for **R**obots with **K**yoto University*) jest oprogramowaniem umożliwiającym tworzenie różnego rodzaju systemów programowego przetwarzania dźwięków na potrzeby robotyki [13], udostępnianym na licencji open-source.

Biblioteka HARK zawiera moduły pozwalające na lokalizację źródeł dźwięku, separację źródeł czy automatyczne rozpoznawanie mowy. Dodatkowo oferuje interfejs graficzny *FlowDesigner*, który pozwala na tworzenie komponentów w postaci schematu blokowego. Łatwość dostępu do licznych zaimplementowanych funkcji z poziomu interfejsu graficznego sprawia, że środowisko to dosyć dobrze nadaje się do tworzenia systemów słuchowych dla robotów.

9.4. Kołowy robot mobilny z systemem lokalizacji źródła dźwięku

W niniejszym podrozdziale opisano autorski projekt systemu lokalizacji dźwięku zintegrowanego z systemem nawigacji robota. Według założeń system ten powinien umożliwiać sterowanie pracą robot w trybie podążania za źródłem dźwięku i przy jednoczesnym omijaniu przeszkód.

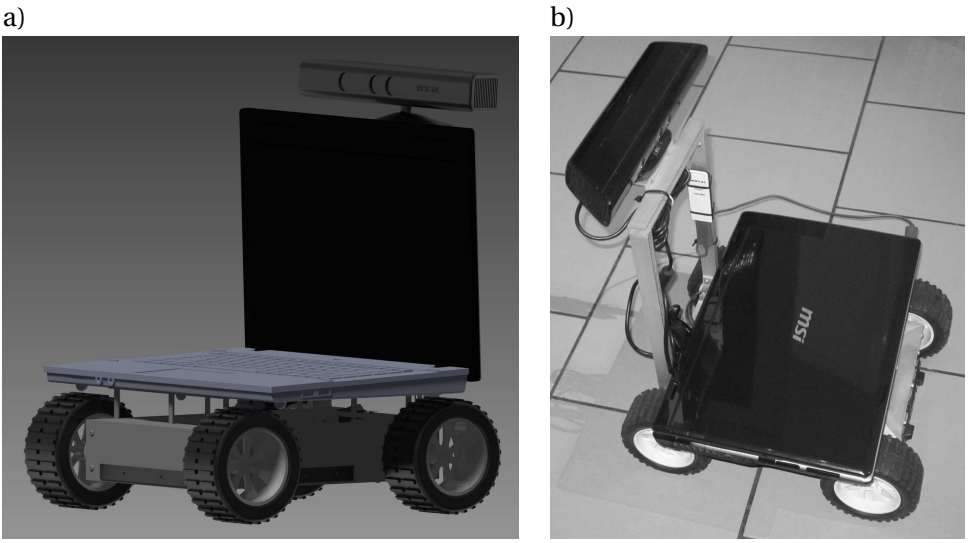
9.4.1. Opis robota

W projekcie założono, że w konstrukcji robota występować będą cztery, niezależnie napędzane koła. Na rysunku 9.5 przedstawiono projekt robota (zwizualizowany w studenckiej wersji programu Audodesk Inventor 2014) oraz zdjęcie faktycznie wykonanej konstrukcji.

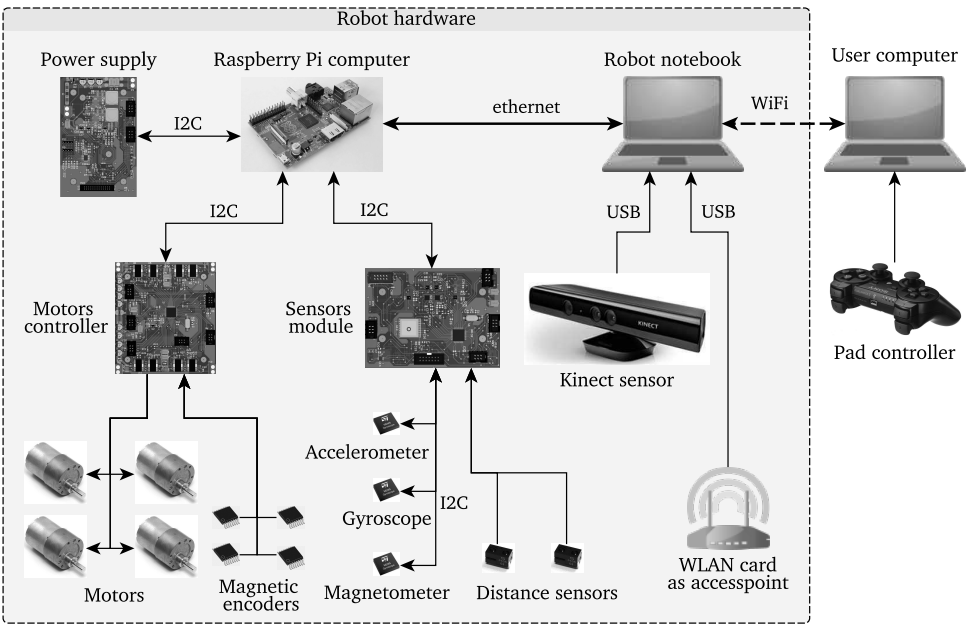
Robot wyposażony jest w bogaty zestaw sensorów: enkodery magnetyczne AS5406, akcelerometry LIS3DH, żyroskopy L3GD20, magnetometr HMC5833L, sensory odległościowe Sharp GP2Y0D340K, a także czujnik Microsoft Kinect.

Funkcje sterujące realizowane są z wykorzystaniem dwóch komputerów: Raspberry Pi w wersji B (procesor ARM 700 MHz, 512 MB RAM) oraz notebooka z procesorem Intel Core i3, posiadającego 4 GB pamięci RAM. Dodatkowo, robot wyposażony jest w dwa dedykowane moduły oparte na 32 bitowych mikrokontrolerach Freescale Kinetis L: moduł sensorów oraz sterownik silników. Na

rysunku 9.6 przedstawiono schemat kompletnej architektury sprzętowej robota, obejmujący komputer użytkownika. Szczegółowy opis podstawowej wersji platformy mobilnej można znaleźć w pracy [14].



Rys. 9.5: Wygląd czterokołowej platformy: a) model robota; b) zdjęcie konstrukcji



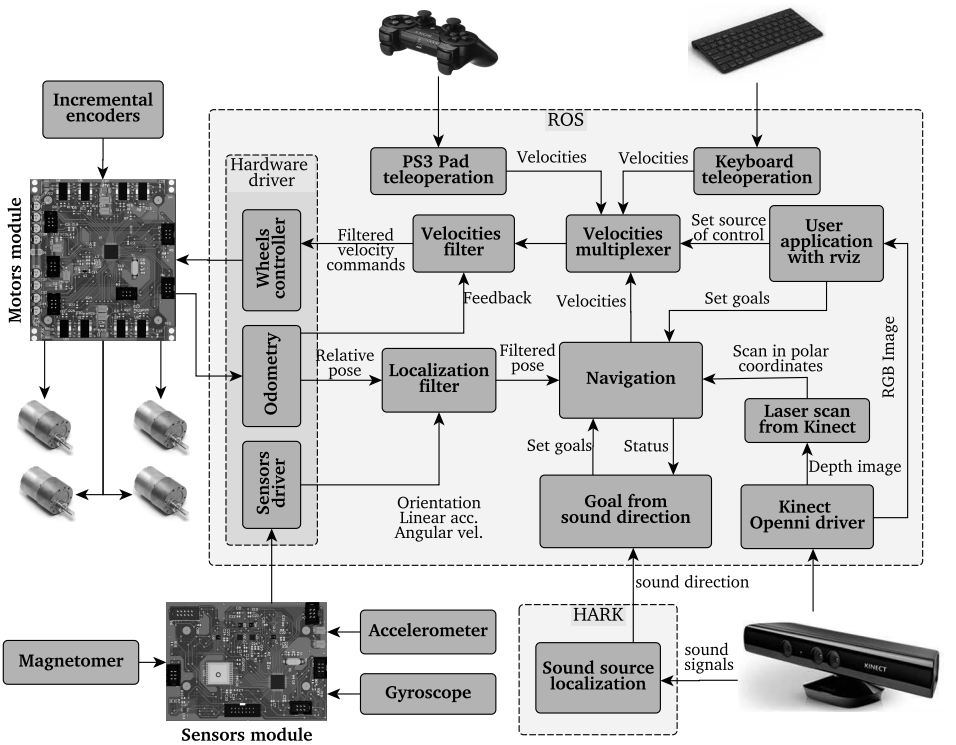
Rys. 9.6: Architektura sprzętowa czterokołowej platformy [14]

9.4.2. Architektura systemu sterowania

System sterowania robota ma strukturę wielowarstwową. Sterowanie niskiego poziomu zostało zaimplementowane na mikrokontrolerze Freescale Kinetis dedykowanego modułu, ze względu na wymagania dotyczące działania w czasie rzeczywistym RT (ang. *Real-Time*). Odpowiada ono za obsługę czterech enkoderów umieszczonych na osiach silników oraz wyznaczanie odometri na podstawie ich odczytów. Dodatkowo, moduł niskiego poziomu realizuje regulatory PID (ang. *Proportional Integral Derivative*) do sterowania silnikami robota na podstawie zadanych prędkości. Pełni również rolę systemu bezpieczeństwa układów wykonawczych robota, ponieważ został wyposażony w szereg elektronicznych zabezpieczeń.

System sterowania wysokiego poziomu zrealizowany został z wykorzystaniem środowiska ROS (ang. *Robot Operating System*) [15]. Architektura układu przedstawiono na rysunku 9.7. Wykorzystano w niej zarówno standardowe moduły środowiska ROS, jak i dedykowane moduły, stworzone w ramach projektu.

System umożliwia budowanie map, lokalizację robota oraz planowanie ruchu. Zarówno za mapowanie, jak i za planowanie ruchu odpowiada moduł *navigation*, który jest standardowym modułem systemu ROS. Na potrzeby pakietu nawigacyj-



Rys. 9.7: Architektura układu sterowania czterokołowego robota mobilnego

nego wykorzystywane są dane z sensora Kinect, w postaci mapy głębi. Mapa ta jest konwertowana do odpowiedniej postaci za pomocą komponentu *Laser scan from Kinect*. Dodatkowo, istnieje możliwość przełączania się (*Velocities multiplexer*) na sterowanie manualne z wykorzystaniem kontrolera (*PS3 Pad teleoperation*) lub klawiatury (*Keyboard teleoperation*). Bardziej szczegółowy opis sterowania wysokiego poziomu związanego z nawigacją można znaleźć w pracy [16], gdzie został przedstawiony przykład innego robota, jednak w znacznej części systemy te pokrywają się.

Do realizacji zaplanowanego zadania zaprojektowano dodatkowe moduły: *Goal from sound direction* oraz *Sound source localization*. Pierwszy z nich pozwala obliczać cele, które zadawane są do pakietu nawigacyjnego na podstawie otrzymanego kierunku źródła dźwięku. Wybierane są cele w pewnej odległości od robota i o kierunku źródła, przy czym za ewentualne omijanie przeszkód odpowiada pakiet nawigacyjny.

Poniżej umieszczono fragment kodu, w którym wyznaczany jest nowy cel robota na podstawie kierunku źródła dźwięku oraz aktualnej pozycji robota otrzymanej z algorytmu adaptacyjnej lokalizacji Monte-Carlo (AMCL).

Listing 9.1: Wyznaczanie orientacji i położenia celu

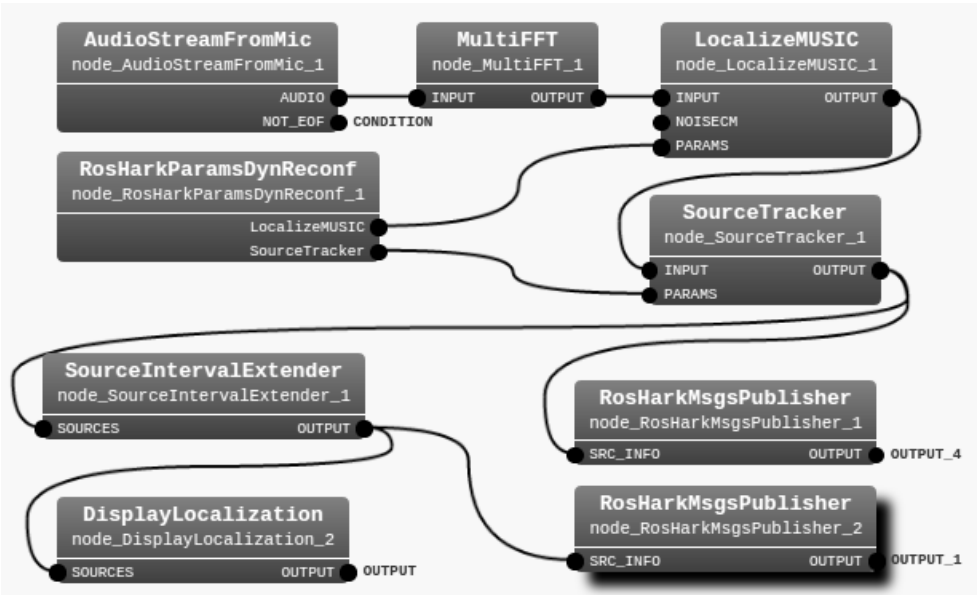
```
// Utworzenie kwaternionu z katow RPY
q1 = tf::createQuaternionFromRPY(0, 0, sourceAngle);

// Aktualna orientacja robota otrzymana z algorytmu AMCL
q2[0] = amcl_pose.pose.pose.orientation.x;
q2[1] = amcl_pose.pose.pose.orientation.y;
q2[2] = amcl_pose.pose.pose.orientation.z;
q2[3] = amcl_pose.pose.pose.orientation.w;

// Wyznaczenie docelowej orientacji robota
q3 = q2 * q1;

// Konwersja kwaternionu do RPY
tf::Matrix3x3(q3).getRPY(roll, pitch, yaw);

// Obliczenie nowego polozenia
goal_x = amcl_pose.pose.pose.position.x + distance * sin(yaw);
goal_y = amcl_pose.pose.pose.position.y + distance * cos(yaw);
```

Komponent *Goal from sound direction* otrzymuje także status działania z pakietu nawigacji na wypadek, gdyby zadany cel okazał się nieosiągalny, co wymaga podania nowego celu.

Komponent *Sound source localization* zaś odpowiada za lokalizację źródła dźwięku na podstawie sygnałów z mikrofonów umieszczonych w sensorze Kinect. Jego działanie opisano w dalszej części tego rozdziału.

9.4.3. Realizacja lokalizacji źródła dźwięku w środowisku HARK

Do lokalizacji źródła dźwięku w trybie online wykorzystano sensor Kinect oraz środowisko HARK. Zaprojektowano układ przetwarzania dźwięku według sche-



Rys. 9.8: Schemat blokowy w środowisku HARK, zrealizowanego systemu przetwarzania dźwięku w celu lokalizacji jego źródła

matu przedstawionego na rysunku 9.8. Układ ten umożliwia pobieranie danych z matrycy mikrofonów oraz lokalizowanie i śledzenie źródeł dźwięku. Układ zintegrowano ze środowiskiem ROS, co umożliwiło sterowanie platformą kołową.

Działanie układu jest następujące. Komponent *AudioStreamFromMic* odpowiada za pobieranie dźwięku z matrycy mikrofonów sensora Kinect. W systemie *Ubuntu 12.04* jest to realizowane za pośrednictwem programistycznego interfejsu ALSA (ang. *Advanced Linux Sound Architecture*). Następnie komponent *MultiFFT* oblicza dyskretną transformatę Fouriera otrzymanych sygnałów wykorzystywaną w dalszej części przetwarzania w dziedzinie częstotliwości. Komponent *LocalizeMUSIC* za pomocą wybranej metody lokalizuje wybraną liczbę źródeł dźwięku. Jego poprawne działanie wymaga odpowiedniego doboru znacznej liczby parametrów. Dzięki modułowi *RosHarkParamsDynReconf* można ustawiać parametry poprzedniego modułu z poziomu środowiska ROS, co jest stosunkowo wygodnym rozwiązaniem w przypadku testowania całego systemu.

Komponenty *SourceTracker* oraz *SourceIntervalExtender* pozwalają na śledzenie konkretnych źródeł dźwięku i przypisywanie im identyfikatorów. Pozostałe elementy układu jak: *DisplayLocalization*, *RosHarkMsgsPublisher* odpowiadają za wizualizację zlokalizowanych źródeł dźwięku oraz komunikację z komponentami systemu ROS.

9.4.4. Realizacja lokalizacji źródła dźwięku w środowisku Octave

Przeprowadzone testy dotyczyły również lokalizacji źródła dźwięku w trybie offline. Do ich przeprowadzenia wykorzystano środowisko Octave oraz zebrane

wcześniej dane z czterech mikrofonów sensora Kinect. Zaimplementowano metodę międzyusznej różnicy czasu (ITD), przedstawioną w podrozdziale 9.2.1. Metoda ta wykorzystuje sygnały z dwóch wybranych mikrofonów i wymaga znajomości odległości między nimi. Ponadto, stworzono dwie wersje algorytmu, pierwszą działającą w dziedzinie czasu, drugą działającą w dziedzinie częstotliwości. Wersję metody ITD działającej w dziedzinie czasu przedstawiono na listingu 9.2. Wykorzystuje się w niej obliczanie korelacji sygnałów dźwiękowych za pomocą funkcji *xcorr*.

Listing 9.2: Implementacja metody ITD działającej w dziedzinie czasu, w postaci funkcji środowiska Octave

```
function alpha = ITD_angle_time(left, right,d,v,fs)
% Zwraca kat do zrodla dzwieku.
% left - sygnal lewego mikrofonu,
% right - sygnal prawego mikrofonu,
% dist - odleglosc miedzy mikrofonami w metrach,
% v - predkosc dzwieku,
% fs - czestotliwosc probkowania

max_delay = floor( d / v * fs); % maksymalne opoznienie
[r, lags] = xcorr(right, left, max_delay); % korelacja
[~,n]     = max(abs(r));
delay     = lags (n) / fs; % roznica czasu odbieranych sygnalow
alpha = acosd((v * delay) / d) - 90;
return
```

9.5. Podsumowanie

W rozdziale przedstawiono podstawowe metody lokalizacji źródeł dźwięku na podstawie fizycznych własnościach fal dźwiękowych. Są one wykorzystywane w bardziej rozbudowanych algorytmach służących rozwiązywaniu problemów w konkretnych zastosowaniach.

Zawarto w nim również opis konstrukcji kołowego robota mobilnego wyposażonego w system lokalizacji źródeł dźwięku oraz system nawigacji. Zaprojektowany system lokalizacji źródeł dźwięku wykorzystuje sygnały zebrane za pomocą matrycy mikrofonów sensora Microsoft Kinect. Bezkolizyjna nawigacja realizowana jest w oparciu o mapę głębi otrzymywaną również z tego czujnika, która następnie przekształcana jest do postaci dwuwymiarowej, kompatybilnej z pakietem nawigacyjnym systemu ROS.

W praktyce realizacja systemu lokalizacji źródeł dźwięku jest zadaniem złożonym. Od strony sprzętowej wymagane są wysokiej jakości i odpowiednio rozmieszczone mikrofony oraz układy wstępnego przetwarzania dźwięku pozwalające eliminować między innymi pojawiające się echo, pogłosy, szum. Aby uzyskać zadowalające efekty wymagana jest implementacja rozbudowanych metod, dostosowanych do konfiguracji sprzętowej oraz z odpowiednio dobranymi parametrami.

W trakcie przeprowadzania testów zauważono, że przedstawiony system (zrealizowany w środowisku HARK i ROS, z użyciem sensora Kinect) umożliwia lokalizowanie kierunku źródła dźwięku jedynie w niewielkim zakresie kątów. Zakres ten określono na około 90^o, co jest w znacznej mierze związane z budową czujnika, która powoduje, że sygnały docierające pod innymi kątami są w dużym stopniu tłumione przez obudowę urządzenia. Dodatkowym ograniczeniem jest fakt, że wiele parametrów związanych z przetwarzaniem dźwięku jest w sensorze Kinect ustawiona w sposób uniemożliwiający dokonnie ich zmiany. Dotyczy to między innymi częstotliwości próbkowania. Rozwiązaniem problemów z niewielkim zakresem kątów działania może być zastosowanie innego rodzaju matrycy mikrofonów, na przykład z rozmieszczeniem ich na okręgu.

Literatura

[1] S. Basu, B.P. Clarkson, A. Pentland. Smart headphones: enhancing auditory awareness through robust speech detection and source localization. *International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2001)*, strony 3361–3364. IEEE, 2001.

[2] K. Nakamura, K. Nakadai, F. Asano, G. Ince. Intelligent sound source localization and its application to multimodal human tracking. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011)*, strony 143–148, Wrzesień 2011.

[3] I. Hara, F. Asano, H. Asoh, J. Ogata, N. Ichimura, Y. Kawai, F. Kanehiro, H. Hirukawa, K. Yamamoto. Robust speech interface based on audio and video information fusion for humanoid hrp-2. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, wolumen 3, strony 2404–2410, Wrzesień 2004.

[4] S. Hengy, S. Demezzo, P. Hamery. Sniper detection using a helmet array: first tests in urban environment. *SPIE 6562, Unattended Ground, Sea, and Air Sensor Technologies and Applications IX*, Maj 2007.

[5] P. Naz, C. Marty, S. Hengy, P. Hamery. Acoustic detection and localization of small arms, influence of urban conditions. *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, wolumen 6963 serii *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, Kwiecień 2008.

[6] K. Nakadai, H. Okuno. Real-time sound source localization and separation for robot audition. *Proceedings IEEE International Conference on Spoken Language Processing*, strony 193–196, 2002.

[7] Ch. Lenz. Localization of sound sources. Praca magisterska, ETH Zurich, 2009.

[8] Michael Shapiro Brandstein. *A Framework for Speech Source Localization Using Sensor Arrays*. Praca doktorska, Brown University, 1995.

[9] S. Paulose, E. Sebastian, P. Babu. Acoustic source localization. *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, 2013.

[10] H. Tomasson. Speaker localization and identification. Praca magisterska, Reykjavik University, 2012.

[11] *A comparative survey on invisible structured light*, wolumen 5303, 2004.

[12] N. Willson, C. Smith, D. Harris, B. Richter. Audio with Kinect. Raport instytutowy, Dept. Electrical and Computer Engineering University of Victoria, 2012.

[13] HARK. HRI-JP Audition for Robots with Kyoto University. `http://www.hark.jp/` [dostęp dnia 20 czerwca 2015].

[14] M. Drwięga. Fuzja sygnałów sensorycznych dla potrzeb lokalizacji kołowego robota mobilnego. Praca magisterska, Politechnika Wrocławska, Wrocław, 2013.

[15] ROS. Robot Operating System. `http://www.ros.org` [dostęp dnia 20 czerwca 2015].

[16] J. Jakubiak, M. Drwięga, B. Stańczyk. Verification of sensory and control system for remedi system mobile platform. *20th International Conference On Methods and Models in Automation and Robotics (MMAR 2015)*, Sierpień 2015.

Od redaktora i wydawcy

W niniejszej książce zebrano opracowania studentów II roku studiów magisterskich Wydziału Elektroniki Politechniki Wrocławskiej, kierunku Automatyka i robotyka, specjalności Robotyka, wykonane w semestrze letnim roku akademickiego 2014/2015 podczas realizacji prowadzonego przeze mnie kursu *Komputerowe przetwarzanie wiedzy*. Zawierają one opisy rożnych naukowych i inżynierskich problemów oraz metod i narzędzi stosowanych przy ich rozwiązywaniu. Opisom tym towarzyszą relacje z wykonanych projektów. Zakres poruszanych tematów można hasłowo podsumować w następujący sposób:

- wyszukiwanie wspierane technologiami sieci semantycznych web oraz systemami do indeksowania,
- metody modelowania strumieni pracy oraz przykład ich zastosowania,
- archiwizacja i wersjonowanie zasobów internetowych z wykorzystaniem narzędzi dostępnych w systemie Linux,
- analiza danych w systemach rekomendacji mająca na celu zwiększenie sprzedaży,
- budowa interfejsów wyszukiwania wykorzystujących gesty,
- sterowanie rojem robotów w układzie z systemem wizyjnym,
- planowanie ruchu przy ograniczonych źródłach zasilania na przykładzie sterowania rakietą,
- rozpoznawanie obiektów w chmurze punktów,
- lokalizacja źródeł dźwięku.

Mam nadzieję, że lektura tych opracowań okaże się interesująca dla czytelnika.

Tomasz Kubik
Wrocław, luty 2016

ISBN 978-83-930823-7-7
9 788393 082377