

Politechnika Warszawska  
Wydział Elektroniki i Technik Informacyjnych  
Instytut Automatyki i Informatyki Stosowanej

Rok akademicki 2009/2010

Praca dyplomowa inżynierska

Tomasz Kuśmierczyk

**Deskryptory punktów w analizie morfologicznej  
obrazów trójwymiarowych twarzy**

Opiekun pracy:  
prof. nzw. dr hab. inż. Antoni Grzanka

Ocena .....

.....  
Podpis Przewodniczącego  
Komisji Egzaminu Dyplomowego



Specjalność: Informatyka - Systemy  
Informacyjno-Decyzyjne

Data urodzenia: 21 grudnia 1987 r.

Data rozpoczęcia studiów: 1 października 2006 r.

### Życiorys

Nazywam się Tomasz Kuśmierczyk. Urodziłem się 21.12.1987r. w Opocznie. Po ukończeniu szkoły podstawowej i gimnazjum, kontynuowałem naukę w Liceum Ogólnokształcącym im. Stefana Żeromskiego w Opocznie. W szkole średniej uczęszczałem do klasy o profilu matematyczno-informatycznym. W październiku 2006r. rozpoczęłem studia na Wydziale Elektroniki i Technik Informacyjnych Politechniki Warszawskiej na kierunku Informatyka. W okresie od lutego do czerwca 2009 w ramach programu Erasmus studiowałem na University of Southampton. Od października 2009r. jestem również studentem Wydziału Fizyki Uniwersytetu Warszawskiego na kierunku fizyka.

.....  
podpis studenta

### Egzamin dyplomowy

Złożył egzamin dyplomowy w dn. .....

Z wynikiem .....

Ogólny wynik studiów .....

Dodatkowe wnioski i uwagi Komisji .....

.....

## **Streszczenie**

*W niniejszej pracy zaprezentowano zaprojektowany i zaimplementowany system do automatycznej analizy skanów trójwymiarowych twarzy. Pozwala on na odnalezienie twarzy w wejściowej chmurze punktów oraz identyfikację regionu nosa oraz otoczeń punktów charakterystycznych na nim.*

*We wprowadzeniu teoretycznym dokonano przeglądu metod analizy obrazów 3D. Wybrano podejście oparte o deskryptory punktów. Uzasadniono wybór oraz przeanalizowano aspekty stosowania deskryptorów. Dokonano selekcji źródeł danych pomiarowych oraz przeanalizowano charakterystyki obrazów wejściowych.*

*W części praktycznej przedstawiono projekt zaproponowanego rozwiązania. Zaprezentowano kolejne fazy powstawania systemu. Opisano algorytmy zastosowane do wstępniego przetworzenia danych, estymacji wektorów normalnych do powierzchni oraz obliczania deskryptorów. Zilustrowano proces wyszukiwania twarzy w scenie. Wyjaśniono szczegółowo zaprojektowaną innowacyjną procedurę hierarchicznej lokalizacji regionów. Każdy element systemu przetestowano i omówiono oraz zinterpretowano wyniki.*

Słowa kluczowe: *biometria, analiza twarzy, deskryptory punktów, spin images, skany 3D*

## **Abstract**

Title: *Point descriptors in morphological analysis of three-dimensional face images*

*The dissertation presents a system prepared to automatically analyse three-dimensional face scans. Its design and implementation process is described. The system allows for face and nose localization in source cloud of points. Its functionality also includes identification of characteristic points on nose.*

*In theoretical introduction a review of known methods of free-form objects analysis is presented. An approach based on point descriptors is explained. Basing on its computational and algorithmic features the choice of this technique is clarified. Selection of sources is motivated. Measurement data characteristics are presented.*

*The practical part describes the design and implementation of proposed system. In the first phase source data is filtered and preprocessed. Algorithms used for normal vectors estimation and descriptors calculations are explained. Innovative procedure of hierarchical region localisation is presented. The process of face, nose and characteristic points identification, based on this procedure is revealed. Each element of the system was tested. Results are discussed and interpreted.*

Key words: *biometry, face analysis, point descriptors, spin images, 3D scans*

---

# *Spis treści*

---

<i>Spis treści</i>	<i>i</i>
<b>1 Wstęp</b>	<b>1</b>
1.1 Wprowadzenie . . . . .	1
1.2 Cel pracy . . . . .	2
1.3 Układ pracy . . . . .	3
<b>2 Tradycyjne pomiary antropometryczne</b>	<b>5</b>
2.1 Wprowadzenie i narzędzia . . . . .	5
2.2 Zarys metodyki . . . . .	6
2.3 Punkty charakterystyczne i parametry twarzy . . . . .	7
2.4 Zastosowania antropometrii . . . . .	9
2.5 Problemy antropometrii . . . . .	10
2.6 Anatomia twarzoczaszki i nosa . . . . .	10
2.7 Statystyki parametrów antropometrycznych . . . . .	11
<b>3 Charakterystyka danych</b>	<b>13</b>
3.1 Przegląd urządzeń pomiarowych . . . . .	13
3.2 Specyfika i wybór źródeł danych . . . . .	16
3.3 Baza obrazów 3D GavabDB . . . . .	17
3.4 Dane z Wydziału Mechatroniki PW . . . . .	17
3.5 Analiza porównawcza zbiorów danych . . . . .	21
<b>4 Deskryptory punktów</b>	<b>25</b>
4.1 Metody analizy obrazów 3D . . . . .	25
4.1.1 Przegląd metod . . . . .	25
4.1.2 Wybór metody . . . . .	26
4.2 Metody oparte o deskryptory punktów . . . . .	26
4.2.1 Local surface patch . . . . .	27
4.2.2 Shape context feature . . . . .	27
4.2.3 Spin image i Local shape map . . . . .	28
4.3 Aspekty deskryptorów . . . . .	30
4.3.1 Zastosowania deskryptorów . . . . .	30

4.3.2	Punkty sąsiednie . . . . .	31
4.3.3	Skala i skwantowanie . . . . .	31
4.3.4	Odległość między deskryptorami . . . . .	32
4.3.5	Dopasowywanie punktów i algorytm węgierski . . . . .	33
4.3.6	Strategie wyboru podzbioru punktów . . . . .	34
4.4	Wybór rozwiązania . . . . .	35
<b>5</b>	<b>Schemat systemu</b>	<b>37</b>
5.1	Kluczowe idee . . . . .	37
5.2	Schemat ogólny systemu . . . . .	38
<b>6</b>	<b>Wstępne przetwarzanie</b>	<b>41</b>
<b>7</b>	<b>Wektory normalne</b>	<b>47</b>
7.1	Dobór promienia otoczenia . . . . .	48
7.2	Estymacja gęstości powierzchniowej punktów . . . . .	50
7.3	Korekta zwrotów wektorów normalnych . . . . .	51
7.4	Schemat algorytmu estymacji wektorów normalnych . . . . .	52
7.5	Wygładzanie danych . . . . .	53
<b>8</b>	<b>Wybór punktów</b>	<b>57</b>
8.1	Metody wyboru punktów . . . . .	57
8.1.1	Wybór losowy . . . . .	57
8.1.2	Wybór punktów w oparciu o k-średnich . . . . .	57
8.1.3	Wybór punktów w oparciu o <i>octClustering</i> . . . . .	57
8.1.4	Wybór metody . . . . .	60
8.2	Wpływ metody wyboru punktów na jakość dopasowywania . . . . .	62
8.2.1	Auto-dopasowanie . . . . .	62
8.2.2	Dopasowanie twarzy . . . . .	63
<b>9</b>	<b>Wybór segmentu twarzy</b>	<b>67</b>
9.1	Dobór parametrów . . . . .	68
9.2	Schemat fazy wstępnej . . . . .	70
9.3	Analiza fazy wstępnej . . . . .	74
<b>10</b>	<b>Hierarchiczna lokalizacja regionów</b>	<b>77</b>
10.1	Hierarchiczna lokalizacja regionów . . . . .	77
10.1.1	Algorytm lokalizacji regionu . . . . .	77
10.1.2	Schemat algorytmu analizy . . . . .	82
10.1.3	Lokalizacja twarzy . . . . .	85
10.1.4	Analiza regionów twarzy . . . . .	86
10.1.5	Lokalizacja nosa . . . . .	87
10.1.6	Lokalizacja regionów nosa . . . . .	89
10.2	Test fazy analizy . . . . .	91
10.2.1	Dane testowe i czas obliczeń . . . . .	91
10.2.2	Test lokalizacji twarzy . . . . .	94
10.2.3	Test lokalizacji nosa . . . . .	95

10.2.4 Wpływ klasyfikacji nosa na lokalizację otoczeń punktów charakterystycznych . . . . .	96
10.2.5 Test lokalizacji otoczeń punktów charakterystycznych . . . . .	97
10.2.6 Przykładowe rezultaty . . . . .	98
<b>11 Szczegółы implementacyjne</b>	<b>101</b>
11.1 Konwertery danych . . . . .	101
11.2 Środowisko . . . . .	103
11.3 Organizacja kodu . . . . .	104
11.4 Przegląd kluczowych procedur . . . . .	105
11.4.1 findSubPatterns . . . . .	105
11.4.2 findFace, findNose, findPoint . . . . .	107
11.4.3 filterData . . . . .	108
11.4.4 findPatternCluster . . . . .	109
11.4.5 modelPreprocessing . . . . .	110
11.4.6 faceAnalysis . . . . .	112
11.4.7 noseDistances . . . . .	113
11.5 Stworzone narzędzia . . . . .	114
11.5.1 Quark . . . . .	114
11.5.2 gui3dPtSelector . . . . .	115
11.5.3 nProcessor . . . . .	119
<b>12 Podsumowanie</b>	<b>121</b>
<b>Bibliografia</b>	<b>123</b>
<b>Spis skrótów i symboli</b>	<b>125</b>
<b>Spis rysunków</b>	<b>127</b>
<b>Spis tabelic</b>	<b>130</b>



# Rozdział 1

---

## Wstęp

---

### 1.1 Wprowadzenie

Rozwój nowoczesnych technologii w ostatnich kilkudziesięciu latach spowodował początek etapu wkraczania jej w coraz to nowe obszary ludzkiej działalności. Niemniej, trzeba przyznać, że wciąż pozostają dziedziny w których dominuje metodyka opracowana na początku XX, a nawet w XIX wieku. Jedną z takich dziedzin jest antropometria – technika analizy wymiarów ludzkiego ciała.

Zakres zastosowań antropometrii jest bardzo szeroki: począwszy od wzornictwa przemysłowego przez badania antropologiczne, systemy monitoringu i autentykacji, skończywszy na medycynie. Zwłaszcza ta ostatnia dziedzina zawiera w sobie duży potencjał dla zastosowania inżynierii komputerowej. Na pograniczu medycyny i zastosowań informatyki prowadzi się badania dotyczące oceny wad postawy i jej zmienności w ciągu życia czy diagnozowania problemów zdrowotnych m. in. związanych z działaniem górnych dróg oddechowych. Prace na temat związku między rynometrią (pomiarami nosa), a kwestiami zdrowotnymi prowadzi się również na Warszawskim Uniwersytecie Medycznym przy współpracy Politechniki Warszawskiej [19]. Wiadomo, że na wyniki diagnoz dotyczących nosa wpływ ma nie tylko stan zdrowia osoby badanej ale również jej budowa antropologiczna [20], dlatego pozyskanie takiej wiedzy jest bardzo ważne w celu uwzględnienia jej wpływu na wynik badania. Potrzeba łatwego i szybkiego pozyskiwania informacji antropometrycznych jest jednym z powodów powstania niniejszej pracy.

W wersji tradycyjnej antropometrii stosuje się wciąż najprostsze narzędzia. Współcześnie jednak prowadzi się prace nad użyciem technik komputerowych w połączeniu ze znacznie bardziej rozwiniętym instrumentarium w celu automatyzacji pewnych zadań. Automatyczne pomiary antropometryczne umiejscowione są na pograniczu antropologii, mechatroniki i informatyki. Z tej pierwszej dziedziny czerpana jest wiedza o budowie ludzkiego ciała, o jego wymiarach i charakterystyce. Druga dostarcza sprzętu pozwalającego na automatyczne zbieranie powyższych informacji. Z kolei informatyka, a dokładniej biometria (ang. *biometry* - gdy mówi się o pozyskiwaniu informacji o wymiarach; *biometrics* - w przypadku zastosowań dotyczących rozpoznawania i autentykacji) zajmuje się opracowywaniem technik analizy pozyskanych danych.

Beziniwazyjne metody skaningu i fotografii pozwalają na pozyskanie obrazów (zówno dwu- jak i trójwymiarowych), które są czytelne dla ludzi, jednak wymagają skomplikowanego przetworzenia i analizy. Istniejące dotychczas opracowania dotyczą głównie algorytmów analizy fotografii. Dane dwuwymiarowe cechują się jednak znaczonymi ograniczeniami. Próby odzyskania informacji trójwymiarowej o wymiarach ciała z zapisu dwuwymiarowego - zdjęcia, napotkają na niemałe trudności. Jest to bardzo skomplikowane i właściwie nie istnieje uniwersalna technika. Problemy jakie mogą się pojawić to m.in. zakrycie i niedostępność pewnych regionów, zmienność w oświetleniu i kolorach, czy złudzenia optyczne.

Istnienie skanerów powierzchni 3D usuwa dużą część z powyższych problemów pozwalając na operowaniu bezpośrednio na danych trójwymiarowych. Jednak przy próbie wdrożenia tej technologii napotyka się na problemy, które dotyczą każdego nowego rozwiązania na rynku tj. istniejące systemy są bardzo drogie i niedoskonałe [17]. Badaniami jej dotyczącymi zajmuje się wciąż stosunkowo niewielka grupa firm i ośrodków naukowych. Pula metod i algorytmów analizy obrazów 3D jest niewielka. Dziedzina ta jest jeszcze stosunkowo słabo rozpoznana, ale już pojawiają się jej pierwsze zastosowania m.in. w medycynie. W pierwszej fazie pobierane są dane dotyczące pacjenta (jest on skanowany). Następnie są one automatycznie, bądź półautomatycznie, analizowane i przetwarzane. Przykładem może być system obrazowania trójwymiarowego żuchwy i szczęki [12] zaprojektowany na potrzeby implantologii, ortodoncji i chirurgii twarzowej. W dołączonym do niego oprogramowaniu zaimplementowano zestaw programów do badań cefalometrycznych i panoramicznych.

Podejściem bardzo obiecującym i coraz szerzej stosowanym na polu analizy obrazów trójwymiarowych jest użycie deskryptorów punktów. Są to struktury, które charakteryzuja pojedyncze punkty analizowanych obrazów. Powstało ich kilka typów w wersjach zarówno dwu- jak i trójwymiarowych. Skutecznie stosowano je do rozpoznawania i analizy obiektów dowolnego kształtu (ang. *free-form objects*) takich jak twarz, ale również pismo. Projekt systemu opisanego w niniejszej pracy oparto również w głównej mierze o deskryptory punktów. Zaproponowano i przetestowano nowe podejście polegające na hierarchicznej lokalizacji regionów.

## 1.2 Cel pracy

Zadaniem postawionym autorowi niniejszej pracy było zaprojektowanie i implementacja rozwiązania algorytmicznego, które umożliwiłoby wykorzystanie informacji zbieranej przez skanery trójwymiarowe, do analiz dotyczących twarzy, a w szczególności nosa. Podjęto próbę stworzenia narzędzia, które wyręczyłoby antropologa w ręcznym oznaczaniu punktów. Danymi przetwarzanymi przez tak zaprojektowany system, byłyby trójwymiarowe obrazy powierzchni, pochodzące bezpośrednio z urządzeń skanujących. W wyniku oczekuje się informacji o charakterystykach i parametrach twarzy, ze szczególnym uwzględnieniem nosa.

W przyszłości skanery trójwymiarowe będą stosowane powszechnie i staną się częścią urządzeń medycznych i diagnostycznych. Jednym z możliwych sposobów ich użycia jest zwymiarowanie twarzy i jej narządów. Ta ważna procedura znajduje wiele zastosowań (m.in. cefalometria). Stworzony zestaw algorytmów mógłby zostać użyty jako moduł

wstępny w systemach zbierających i analizujących takie dane. Możliwe są też inne, pozamedyczne, wdrożenia np. w systemach autentykacji, czy przy interakcji człowiek-komputer.

Zadaniem prowadzącym do celu było wyszukanie w literaturze, a następnie opracowanie różnych algorytmów przetwarzania. Dokonano ich wyboru w oparciu o ocenę efektywności. Zaprojektowano, zaimplementowano i przetestowano na rzeczywistych danych wynikowy zestaw procedur.

### 1.3 Układ pracy

Struktura rozdziałów niniejszej pracy odpowiada kolejnym etapom analizy, projektowania i testowania systemu do automatycznej analizy biometrycznej. Układ taki jest logiczną konsekwencją przebiegu procesu powstawania systemu i pozwala prześledzić kolejne fazy działań prowadzących do powstania ostatecznego rozwiązania.

Pierwszą część poświęcono tradycyjnym pomiarom antropometrycznym. Służy ona jako wstęp medyczny. Pozwala zapoznać się z tą dziedziną, ze stosowaną w niej metodą oraz narzędziami. Przybliżono w niej wiedzę antropologiczną dotyczącą budowy twarzy, a w szczególności nosa. Zaprezentowane tam informacje pozwalały zrozumieć istotę postawionego problemu, ale również dają wiedzę *a priori* o oczekiwanych rezultatach. W rozdziale 2 pokazano zarys obowiązujących w antropometrii standardów. Przedstawione w zarysie nazewnictwo i oznaczenia wypracowane zostały przez komitety ekspertów i pozwalały na wymianę informacji o anatomii człowieka.

Rozdział 3 poświęcono części technicznej opracowanego zagadnienia. Opisano pokrótko budowę istniejących systemów skaningu 3D, przybliżając ich wady i zalety. Na tej podstawie wyjaśniono wybór źródeł danych (bibliotek obrazów 3D). Dalej scharakteryzowano i dokonano wstępnej analizy obrazów trójwymiarowych, które stosowane były przy projektowaniu i analizie rozwiązania algorytmicznego.

W kolejnym rozdziale pokrótko przybliżono wiedzę dotyczącą analizy obiektów dowolnego kształtu (ang. *free-form objects*) ze szczególnym uwzględnieniem deskryptorów punktów. Zaprezentowano ich charakterystykę, przedstawiono dotychczasowe zastosowania i przybliżono aspekty ich użycia. Uzasadniono ich wybór jako narzędzia użytego do rozwiązania postawionego problemu. W części końcowej rozdziału opisano zaproponowane modyfikacje i udoskonalenia.

Rozdział 5 poświęcono projektowi i ogólnej strukturze rozwiązania algorytmicznego. Wyjaśniono w nim rozumowanie, które doprowadziło do powstania ogólnego schematu systemu. Dalej przybliżono i scharakteryzowano jego kolejne moduły, ze szczególnym uwzględnieniem ich funkcjonalności i oczekiwanych rezultatów, ale bez zagłębiania się w szczegóły algorytmiczne.

W kolejnym rozdziale 6. przedstawiono algorytm zastosowany do wstępniego przetworzenia i odszumienia danych. Wyjaśniony został sposób jego działania oraz wprowadzone modyfikacje. Zilustrowano przykładami wynik działania. Opisano test przeprowadzony na danych testowych.

Rozdział 7 poświęcono kwestiom pomocniczym, które należy rozważyć przed przeprowadzeniem obliczeń dotyczących deskryptorów punktów. Przedstawiono rozumowanie, które doprowadziło do stworzenia algorytmu estymującego kierunki wektorów nor-

malnych do powierzchni. Zaprezentowano wyniki przeprowadzonych testów. Zaproponowano algorytm służący do oceny gęstości powierzchniowej obrazów. Pokazano przykładowe rezultaty pełnej procedury służącej obliczaniu wektorów normalnych.

W pierwszej części rozdziału 8 zajęto się kwestią wyboru podzbioru punktów dla których wygenerowane zostaną deskryptory. Przedstawiono i wyjaśniono 3 podejścia. Pokazano przykładowe wyniki. Drugą część poświęcono testowi wpływu metody wyboru na jakość dopasowywania obrazów do wzorców. Omówiono i zinterpretowano uzyskane rezultaty.

Praktycznemu zastosowaniu deskryptorów punktów poświęcono rozdziały 9 i 10. W pierwszym z nich zaprezentowano algorytm wyboru segmentu danych zawierającego twarz. Opisano proces doboru parametrów i uzyskane rezultaty. Podsumowano też fazę wstępna zaproponowanego systemu. W rozdziale 10 skupiono się na fazie drugiej - analizie twarzy. Szczegółowo wyjaśniono zaproponowany algorytm lokalizacji regionów. Przedstawiono schemat procedury hierarchicznej lokalizacji oparty o ten algorytm. Przeanalizowano charakterystyki deskryptorów w różnych częściach twarzy. Opisano i zinterpretowano wyniki testów końcowych zaproponowanego rozwiązania.

Stronę implementacyjną rozwiązania omówiono w rozdziale 11. Sekcja ta służyć może jako pomoc dla osób chcących dokonać wdrożenia lub rozbudowy powstałego rozwiązania. Opisano środowisko w którym wykonany został system. Dokonano przeglądu kluczowych funkcji wraz z opisem sposobu ich użycia. Zaprezentowane i opisane zostały również narzędzia powstałe w toku prac projektowych.

Rozdział ostatni podsumowuje przeprowadzone działania. Przedstawiono w nim krótkie zestawienie rezultatów pracy. Zinterpretowano wyniki i wyciągnięto wnioski. Pokazane zostały też możliwe kierunki dalszego rozwoju stworzonego systemu oraz perspektywy jego stosowania.

## Rozdział 2

---

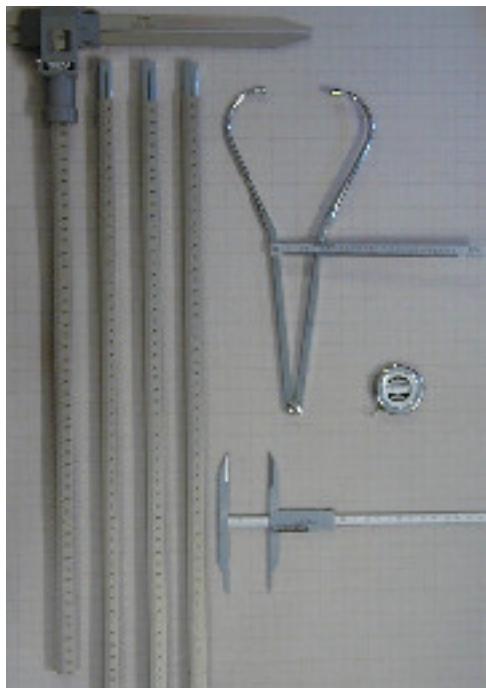
# Tradycyjne pomiary antropometryczne

---

### 2.1 Wprowadzenie i narzędzia

Antropometria to technika stosowana w antropologii. Polega ona na liczbowym opisie wymiarów ciała ludzkiego. Jej nazwa wywodzi się z greckich słów: *anthropos* – człowiek i *metrikos* – pomiar. Jest ona rozwijana od ponad 200 lat. W wersji klasycznej dokonuje się pomiarów statycznych za pomocą elementarnego zestawu narzędzi ukazanego na rysunku 2.1. Są to [17],[15] m.in.:

- waga – pomiar ciężaru
- aparat – utrwalanie wyglądu części ciała
- taśma pomiarowa – długość krzywizn i obwody części ciała
- antropometr (ang. *anthropometer*) - pomiar odległości między punktami na ciele
- suwak (cyrkiel liniowy, ang. *spreading caliper*) – pomiar średnic
- goniometr – pomiary kątów
- cyrkle kabłkowe
- fałdomierz – pomiar fałd na skórze
- narzędzie do pomiarów średnic małych otworów (oczy, nos) (ang. *sliding compass*)
- narzędzie do pomiaru wysokości głowy (ang. *head spanner*)



Rysunek 2.1: Podstawowy zestaw narzędzi do pomiarów antropometrycznych

## 2.2 Zarys metodyki

Podstawowymi pojęciami z zakresu metodologii antropometrycznej są rzuty oraz płaszczyzny i linie ciała. Najważniejszą płaszczyzną dotyczącą głowy jest płaszczyzna frankfurcka. Odpowiada ona ułożeniu głowy w którym najwyższy punkt kostnej krawędzi oczodołu i najwyższy punkt na kostnej krawędzi kanału słuchowego znajdują się w jednej płaszczyźnie (poziomej). Drugim pojęciem potrzebnym do definiowania punktów pomiarowych antropometrii są linie – ślady, jakie pozostawiają płaszczyzny na powierzchni ciała [15]. Istotną jest linia pośrodkowa przednia, która przechodzi przez środek czoła, środek wcięcia szyjnego rękojeści mostka, aż do środkowego punktu dolnego brzegu spojenia łonowego. Do precyzyjnego lokalizowania poszczególnych punktów charakterystycznych stosuje się rzuty. Rzut definiuje się jako spojrzenie na ciało lub jego część z kierunku prostopadłego do danej płaszczyzny [15], np. rzut górny (pionowy) to spojrzenie prostopadle do płaszczyzny frankfurckiej.

W celu zunifikowania wyników pomiarów antropometrycznych należy zminimalizować rozbieżności pomiędzy środowiskiem i metodą w jakim dokonuje się pomiarów. Pomiaru należy dokonywać w ułożeniu maksymalnie zbliżonym do pozycji anatomicznej: stojąc wyprostowanym, pięty złączone, pośladki, tył głowy i łopatki dotykają ściany za plecami osoby na której dokonywany jest pomiar [10]. Ramiona i palce powinny być wyprostowane. Głowa powinna być nieskręcona i ułożona w płaszczyźnie frankfurckiej.

Głównymi źródłami błędów i niepowtarzalności pomiarów dokonywanych za pomocą tej metody są:

- subiektywność ustalenia punktów charakterystycznych ciała – różne osoby różnie oceniają spełnienie kryteriów określających ich położenie

- niedokładność ułożenia ciała przy pomiarze
- precyza narzędzi

Poprawy wyników można dokonać za pomocą zwiększenia liczby osób dokonujących subiektywnej oceny pomiaru i uśredniania wyników. Wiarygodność pojedynczego wyniku można sprawdzić porównując go do wartości standardowych lub metodami statystycznymi.

## 2.3 Punkty charakterystyczne i parametry twarzy

Podstawowym zagadnieniem antropometrii jest wybór punktów lub obszarów charakterystycznych (ang. *landmarks*) na ciele pomiędzy którymi dokonywane są pomiary. Pomiary za pomocą nich dokonywane powinny spełniać następujące kryteria:

- być jednoznacznie określone i łatwe do identyfikacji
- znajdować się w obszarach łatwo dostępnych
- dobrze różnicować i charakteryzować osoby

Z punktu widzenia wrażliwości na błędy dobrze jest również gdy małe przesunięcie punktu charakterystycznego w nieistotnym stopniu wpływa na wartość zmierzonego parametru. Ważną cechą doboru punktu względem przyszłych zastosowań jest stałość jego położenia, niezależnie od przyjmowanej pozy i miny (przykładem może być identyfikacja tożsamości: osoba, która w czasie pomiaru mówiła lub śmiała się powinna zostać zaakceptowana przez system).

Medycyna definiuje wiele punktów charakterystycznych (cefalometrycznych tj. rozmieszczonych na czaszce ale rzutowanych na powierzchnię miękką) dla twarzy, jednak nie wszystkie one są użyteczne dla pomiarów antropometrycznych w każdym zastosowaniu. Dzieli się je na parzyste – te dla których istnieją dwa punkty na ciele spełniające kryterium - i nieparzyste - pojedyncze. Są to [20],[17],[15] m. in.:

- glabella (g) - punkt na kości czołowej, najbardziej wysunięty ku przodowi, znajdujący się na przecięciu linii pośrodkowej ciała z linią przeprowadzoną przez wyniosłości nadoczodołowe. W przypadku braku wyniosłości charakteryzującej głaziznę punkt należy lokalizować w miejscu przecięcia linii łączącej łuki brwiowe przez płaszczyznę środkowo – strzałkową.
- opistocranion (op) - jest to punkt na kości potylicznej najbardziej odległy od punktu glabella w płaszczyźnie pośrodkowej
- euryon (eu) - punkt na czaszce najbardziej bocznio wysunięty, leżący na kości ciemieniowej, a w części przypadków na skroniowej. Wyznaczając ten punkt należy wcześniej wykluczyć inne punkty spełniające powyższe kryterium.
- vertex (v) - punkt najwyższej położony na szczycie głowy ustalonej w płaszczyźnie frankfurckiej

- tragion (t) - znajduje się on na górnym brzegu skrawka mażowiny usznej, na wysokości górnego brzegu przewodu słuchowego zewnętrznego
- zygion (zy) - najdalej bocznie wysunięty punkt łuku jarzmowego
- entokanthion (en) – punkty położony w kącie powiekowym wewnętrznym (pomiar szerokości szpary ocznej).

Wykorzystując powyższe punkty można wyznaczyć wiele odległości. W medycynie przyjęło się stosować te najbardziej intuicyjne i które zdają się najlepiej charakteryzować człowieka:

- największa długość głowy (glabella – opistocranion, g - op)
- największa szerokość głowy (euryon – euryon, eu – eu)
- wysokość głowy uszna (vertex – tragion, v - t)
- szerokość jarzmowa twarzy (zygion – zygion, zy - zy)
- wysokość morfologiczna górną - twarzową (nasion – stomion, n – st)

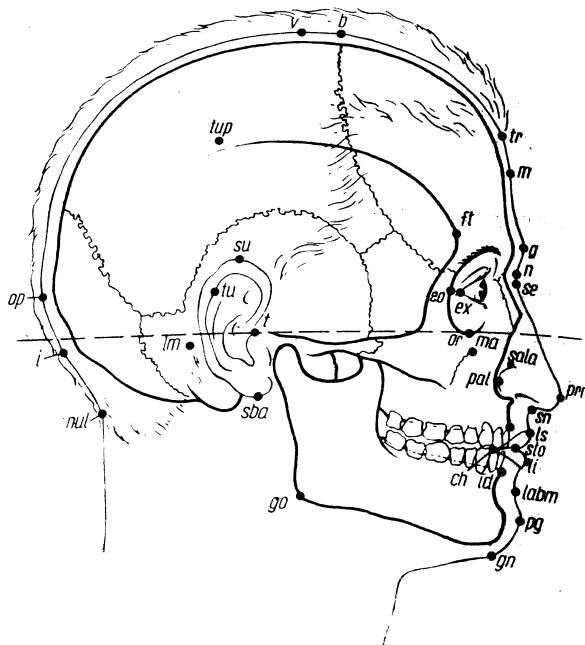
Punkty charakterystyczne dla nosa to:

- pronasale (prn) - punkt w linii pośrodkowej, w najbardziej wysuniętym ku przodowi punkcie końca nosa, przy ustawieniu głowy w płaszczyźnie uszno – ocznej.
- subnasale (sn) - punkt leżący w linii pośrodkowej ciała, na połączeniu słupka nosa i rynienki podnosowej wargi górnej. Punkt położony jest na szczycie kąta, jaki tworzy dolna krawędź przegrody nosowej i rynienka wargowa. Służy pomiarowi długości (wysokości) nosa.
- nasion (n) - punkt leżący w linii pośrodkowej, na wysokości nasady nosa
- alare (al) - najbardziej bocznie wysunięty punkt skrzydełka nosowego, służy do pomiaru szerokości nosa (w bezdechu)
- supraalare (sala) – punkt położony jest w miejscu najwyższym na bruździe skrzydła nosa.
- nariale (na) – punkt położony najniżej na krawędzi dolnej otworu gruszkowatego. Służy do pomiaru otworu nosowego.
- postalare (pal) – punkt położony z tyłu skrzydełek nosa w miejscu, gdzie skóra nosa przechodzi w skórę twarzy, najbardziej bocznie. Służy do pomiaru szerokości nasady nosa.
- stomion (st) - punkt leżący na przecięciu linii pośrodkowej ciała ze szparą ust.

Analogicznie, parametrami charakteryzującymi obszar nosa są:

- Wysokość (długość) nosa (nasion – subnasale, n – sn)

- Długość grzbietu nosa (nasion – pronasale, n – prn)
- Szerokość nosa (alare – alare, al – al) – pomiar przy bezdechu i braku napięcia skrzydełek nosa.
- Wysokość słupka nosa (długość podstawy nosa) (subnasale – pronasale, sn – prn)
- Szerokość nasady nosa (pal-pal) – pomiar ze skrzydełkami nosa u jego nasady.
- Kąt profilu nosa, nachylenie linii nasion – subnasale (n-sn) do pionu.



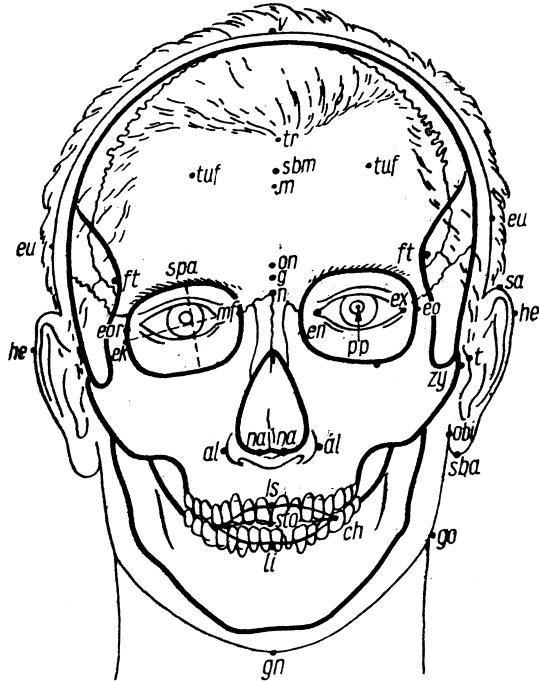
Rysunek 2.2: Punkty antropologiczne twarzy (widok z boku) [15].

## 2.4 Zastosowania antropometrii

Mając do dyspozycji zbiór pomiarów antropometrycznych można estymować rozkład wartości danych cech dla ogółu populacji. Taka informacja używana jest w czysto praktycznych zastosowaniach związanych m. in. z produkcją samochodów, szyciem ubrań czy narzędzi codziennego użytku – we wzornictwie przemysłowym [15].

Jednym z możliwych obszarów zastosowań szybkich pomiarów antropometrycznych jest przemysł odzieżowy. Problemem dla wielu osób jest dobór ubioru idealnie pasującego i wygodnego, co może okazać się trudne przy obecnym systemie "rozmiarów krawieckich" (wprowadzonym w 1941 roku wg ówczesnych standardów). Wyobrazić można sobie system, w którym klienta przychodzącego do sklepu skanuje się skanerem 3D. Uzyskany obraz poddawany jest analizie i korzystając z bazy danych dokonywana jest selekcja pasujących ubrań.

Informacje antropometryczne używane są również w medycynie np. do oceny stopnia zdrowia danego osobnika. Przykładowo, znając stopień korelacji pomiędzy zmierzonymi



Rysunek 2.3: Punkty antropologiczne twarzy (widok z przodu) [15].

wymiarami, a trudnymi do zbadania cechami charakteryzującymi zdrowie osoby, ocenić można prawdopodobieństwo problemów zdrowotnych. Znana jest np. zależność między wymiarami zewnętrznymi nosa i, trudnymi do zmierzenia, wymiarami jam nosa [19, 20].

Kolejnymi obszarami zastosowań omawianej metody są m.in. chirurgia plastyczna czy też klasyfikacja antropologiczna. Bardziej współcześnie stosować ją można w weryfikacji tożsamości, czy też przy produkcji modeli 3D do gier lub animacji komputerowych. Dziedzina zastosowań pomiarów antropometrycznych jest bardzo szeroka i stale się powiększa.

## 2.5 Problemy antropometrii

Podstawowymi problemami utrudniającymi praktyczne zastosowania antropometrii są:

- zmienność ciała ludzkiego w ciągu życia – widoczne jest to szczególnie w okresie od urodzenia do osiągnięcia dojrzałości [20]
- plastyczność twarzy i ciała jako całości
- trudność związane z dostępem do obszarów pomiarowych (np. ze względu na ubiór)

## 2.6 Anatomia twarzoczaszki i nosa

Znajomość specyfiki budowy twarzoczaszki i nosa stanowią wiedzę niezwykle istotną dla dokładnej analizy tych obszarów, jak i dla późniejszego opracowywania algorytmów

automatyzujących ich pomiary.

Twarz można podzielić w pionie na trzy obszary: od linii włosów do glabelli (część czołowa) od linii brwi do podstawy nosa (środek) od podstawy nosa do brody wyznaczoną przez gnathion (część dolna). Niezależnie od tego podziału wyróżnia się następujące regiony twarzy: czoło, brwi, oczy, nos, policzki, usta, podbródek. Podział ten jest naturalny i dlatego łatwy do użycia.

## 2.7 Statystyki parametrów antropometrycznych

Parametry charakteryzujące twarzoczaszkę jak i nos przedstawiono w poprzednich rozdziałach, jednak dla opracowania systemów pomiarowych pomocna jest wiedza na temat oczekiwanych wyników. Zestawienie wartości średnich wraz z odchyleniami standardowymi wartości pomiarów przedstawiają tabele 2.1 i 2.2.

Tablica 2.1: Zestawienie wartości parametrów twarzoczaszki dla osób powyżej 17 roku życia w okolicach Warszawy [20].

Mierzony wymiar	Średnia dla kobiet [mm]	Odchylenie standarde [mm]	Średnia dla mężczyzn [mm]	Odchylenie standarde [mm]
Największa szerokość głowy	152,43	5,09	159,64	5,40
Największa długość głowy	179,09	5,40	192,05	5,91
Wysokość uszna głowy	148,91	5,59	157,21	5,52
Szerokość jarzmowa twarzy	135,51	4,92	144,46	4,77
Wysokość twarzy górnej	74,45	12,78	78,51	4,81

Tablica 2.2: Zestawienie wartości parametrów nosa dla osób powyżej 17 roku życia w okolicach Warszawy [20].

Mierzony wymiar	Średnia dla kobiet [mm]	Odchylenie standarde [mm]	Średnia dla mężczyzn [mm]	Odchylenie standarde [mm]
Wysokość nosa	51,00	3,37	55,08	4,62
Długość grzbietu nosa	47,51	3,44	50,92	4,95
Szerokość nosa	32,17	2,80	35,82	2,85
Wysokość słupka nosa	24,61	2,32	26,77	2,53



## Rozdział 3

---

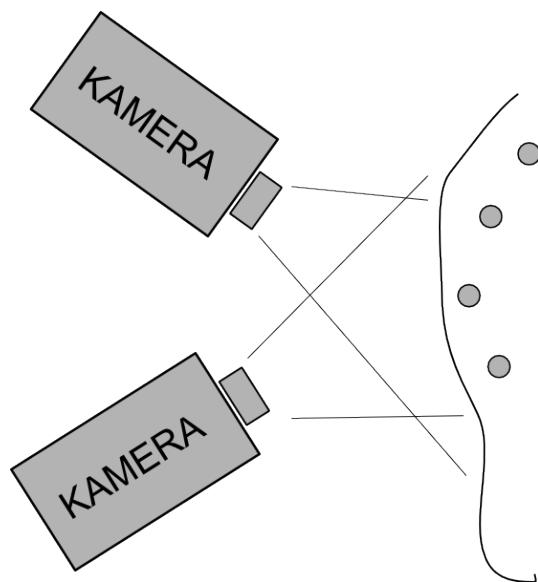
# Charakterystyka danych

---

### 3.1 Przegląd urządzeń pomiarowych

Współcześnie stosuje się kilka typów bezdotykowych skanerów powierzchni. Pozwalają one uzyskać informację o koordynatach trójwymiarowych powierzchni bez mechanicznej interakcji ze skanowanym obiektem. Dzieli się je na pasywne i aktywne [18].

Pasywne (ang. *Passive Vision*) różni od aktywnych to, że nie emitują światła operując jedynie na obrazie naturalnym. Podstawową stosowaną techniką jest fotogrammetria (ang. *photogrammetry*). Używa ona obrazów z dwóch kamer (rysunek 3.1), na których poszukuje się odpowiadających sobie punktów. Na tej podstawie następnie oblicza się informację o głębi. Wadami takiego rozwiązania są: długi czas przetwarzania danych, skomplikowana kalibracja urządzenia oraz trudność z uzyskaniem dokładnych danych (w wersji bez specjalnych znaczników punktów na powierzchni) i ograniczona liczba punktów które daje się analizować (w wersji ze znacznikami).

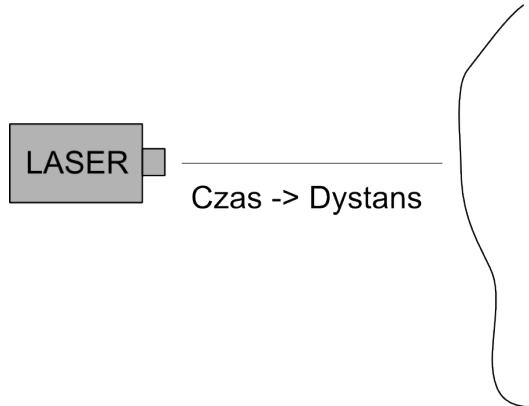


Rysunek 3.1: Schemat skanera zbudowanego w oparciu o fotogrametrię.

Znacznie lepszymi i szerzej stosowanymi rozwiązaniami są skanery aktywne. Ich działanie opiera się na rzutowaniu dodatkowego obrazu lub światła na obiekt skanowany. Wśród takich urządzeń wyróżnia się skanery punktowe (ang. *PAV - Point Active Vision*) i powierzchniowe (ang. *FAV - Full-Field Active Vision*).

Skanery punktowe składają się zwykle z lasera i kamery lub interferometru. Wyróżnia się rozwiązania oparte o:

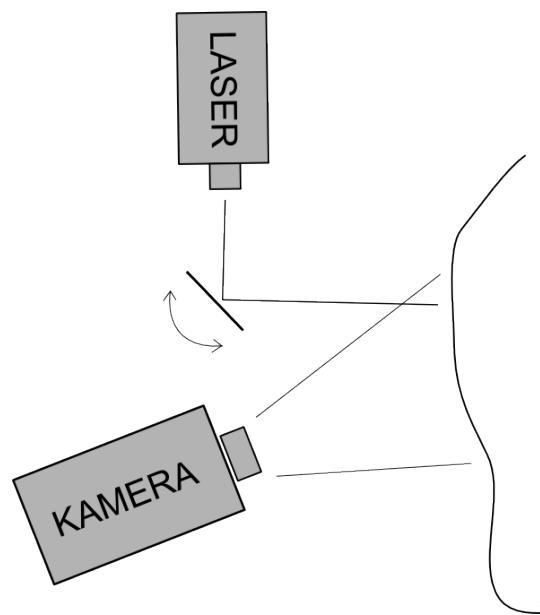
- czas lotu światła (ang. *Time-of-Flight*, rysunek 3.2) - mierzy się czas przelotu od lasera do obiektu i z powrotem. Zaletą rozwiązania jest szeroki zakres rozmiarów mierzonych obiektów. Wadą wymóg dostępności bardzo precyzyjnych urządzeń optycznych i elektronicznych. Rozwiążanie jest bardzo kosztowne.



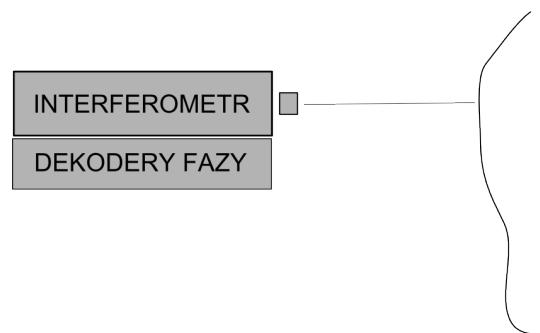
Rysunek 3.2: Schemat skanera zbudowanego w oparciu o czas lotu światła.

- skaning laserem (ang. *Laser Scanning*, rysunek 3.3) - rzutuje się światło laserowe na powierzchnię obiektu i analizuje jego położenie kamerą. Plusami rozwiązania są prostota i szybkość przetwarzania oraz wysoka dokładność. Minusem są ograniczenia geometryczne co do możliwych położen punktów na powierzchni.
- śledzenie lasera (ang. *Laser Tracking System*, 3.4) - analizuje się interferometrem wynik odbicia światła laserowego od powierzchni skanowanego obiektu. Pozycjami rozwiązań są szybkość i jakość danych. Minusem natomiast wysoki koszt.

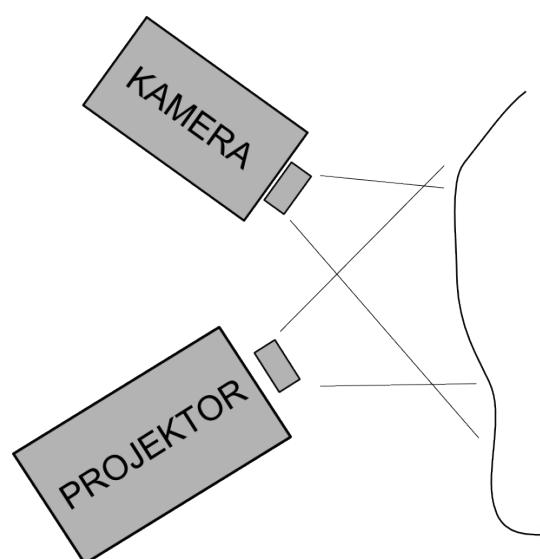
Skanery powierzchniowe składają się z projektora i kamery (rysunek 3.5). Projektor rzutuje specjalnie ustrukturyzowany obraz lub obrazy na powierzchnię skanowanych obiektów. Następnie obraz uzyskany z kamery poddawany jest analizie. Stosowanych jest kilka rozwiązań. Dwa główne to prążki Moire'a (ang. *moire fringes*) oraz światło strukturalne (ang. *structured light*). Korzyściami płynącymi z takich rozwiązań jest prosta faza analizy danych, szybkość pozyskiwania danych i stosunkowo duża dokładność danych. Minusem jest wysoki koszt projektora.



Rysunek 3.3: Schemat skanera zbudowanego w oparciu o metodę skaningu laserem.



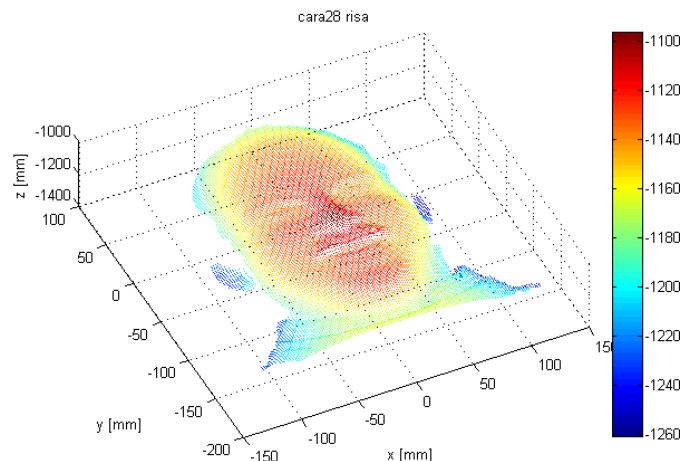
Rysunek 3.4: Schemat skanera opartego o interferometr



Rysunek 3.5: Schemat skanera opartego o światło strukturalne.

### 3.2 Specyfika i wybór źródeł danych

Każdy z powyższych skanerów generuje na swoim wyjściu zbiór/chmurę punktów (ang. *cloud of points*, rysunek 3.6) w trójwymiarowym, kartezjańskim układzie współrzędnych ( $x, y, z$ ). Każde z urządzeń korzysta z własnych, specyficznie położonych koordynat. Pojedynczy przebieg skanowania daje w efekcie pojedynczą chmurę kierunkową (ang. *directional data*) tj. widok powierzchni z jednego kierunku lub ze zbioru kierunków (np. gdy urządzenie skanujące przesuwa się wokół obiektu skanowanego). Chmurę taką charakteryzuje to, że obszary zakryte podczas skanowania w danych wyjściowych pojawią się jako "dziury". Możliwe są też nieciągłości powierzchni wynikające z innych błędów w skanowaniu. Dane mogą zawierać też fragmenty odzieży, elementy otoczenia, włosy. Dodatkowo należy pamiętać, że dane wyjściowe ze skanerów mogą mieścić w sobie przekłamania tj. przypadkowe punkty nie mające odpowiedników w rzeczywistości. Nie należy też zapominać, że niedokładność urządzeń pomiarowych wprowadza pewien rozrzułt współrzędnych punktów wokół wartości rzeczywistych.



Rysunek 3.6: Przykładowa chmura punktów powstała w wyniku skanowania powierzchni twarzy.

Opierając się na charakterystykach urządzeń pomiarowych oraz uwzględniając specyfikę danych, w projekcie i testach stworzonego systemu zdecydowano się stosować dane pochodzące z dwóch źródeł stosujących dwa najpopularniejsze i najbardziej reprezentatywne podejścia do zbierania danych trójwymiarowych: skaner laserowy (wersja z kamerą) oraz światło strukturalne. Są to rozwiązania, których koszt jest niższy od pozostałych. Wymagania co do precyzji które stawia się komponentom tak zbudowanych systemów są stosunkowo niskie, a mimo to charakteryzuje je dość wysoka jakość zbieranych danych.

Pierwszym z źródeł danych jest publicznie dostępna baza obrazów 3D trójwymiarowych twarzy - GavabDB. Drugim są dane pozyskane z Wydziału Mechatroniki Politechniki Warszawskiej dzięki uprzejmości dr inż. Roberta Sitnika. W celu uzyskania ogólności rozwiązania, w zaprojektowanym rozwiązaniu nie przyjęto żadnych założeń co do położenia (ani obrotu) danych w przestrzeni. Nic również nie zakłada się na temat numeracji czy też posortowania punktów. Informacja o kolorze często nie jest dostępna,

toteż zakłada się jej brak. Przyjmuje się natomiast, że na wejściu podawane są pojedyncze chmury kierunkowe punktów powstałe w efekcie skanowania twarzy. Obrazy powinny zawierać widok twarzoczaszki z przodu. Skala danych to 1:1, a współrzędne punktów w danych podane powinny zostać w milimetrach. **W niniejszej pracy zakłada się, że wszystkie odległości podawane są w milimetrach.**

### 3.3 Baza obrazów 3D GavabDB

Baza danych Gavab powstała na Uniwersytecie Rey Juan Carlos w Madrycie. Jest ona dostępna w internecie pod adresem [http://www.gavab.es/recursos\\_en.html](http://www.gavab.es/recursos_en.html) na licencji pozwalającej na stosowanie w zastosowaniach naukowych. Zbudowana została ona w celach badawczych dotyczących automatycznego rozpoznawania i analizy twarzy.

Bazę wykonano z wykorzystaniem skanera laserowego Minolta V1-700 [23, 13]. Każdy z obrazów 3D pozyskano w pojedynczym skanie. Czas pobierania danych nie przekraczał 1 sekundy. Przy pobieraniu danych nie czyniono żadnych prób kontroli oświetlenia. Twarze umiejscowione były w odległości około 1.5 - 2 metrów od skanera. Różnica w położeniu może powodować delikatne rozbieżności w rozdzielcości obrazów. Dane wstępnie przetworzono z wykorzystaniem oprogramowania VIVID. Zmniejszono rozdzielcość danych poprzez zastąpienie kilku punktów ich średnią, aż do uzyskania pożąданej wartości. Dodatkowo wypełniono małe nieciągłości powierzchni.

Baza zawiera 549 trójwymiarowe skany powierzchni twarzy w popularnym formacie VRML pobrane od 61 osób należących do rasy białej: 45 z nich było mężczyznami, a 16 kobietami. Ich wiek wahał się między 18, a 40 lat. W bazie znajdują się osoby z brodą, wąsami itp. Dla każdej osoby baza zawiera 2 widoki z przodu i 4 z boku w pozycji neutralnej (tabela 3.1, rysunek 3.7). Dodatkowo dołączono 3 obrazy twarzy widziane z przodu wyrażające różne emocje: śmiech (ang. *laugh*), uśmiech (ang. *smile*) i jedną losową (nie dopuszcza się jednak zakrywania twarzy np. dłonią, czy językiem). Przykładowe obrazy ukazano na rysunkach 3.8 i 3.9.

Tablica 3.1: Typy danych w bazie GavabDB.

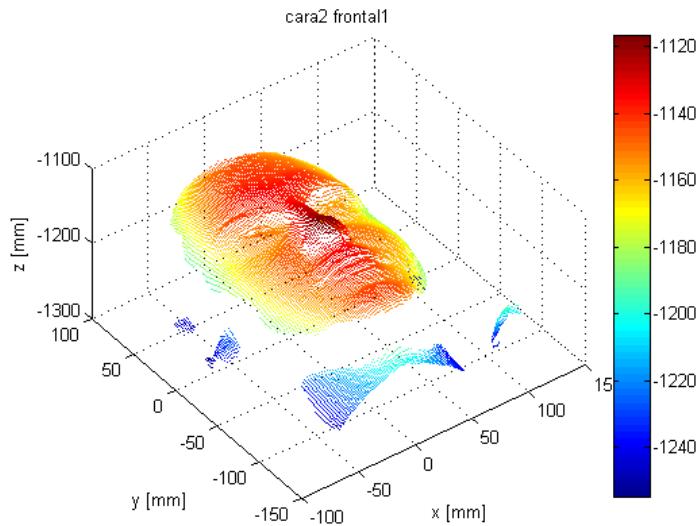
Numer widoku	Nazwa pliku	Orientacja głowy	Emocja
1	carai frontal1	Przód	Neutralna
2	carai frontal2	Przód	Neutralna
3	carai derecha	Prawy profil	Neutralna
4	carai izquierda	Lewy profil	Neutralna
5	carai arriba	Patrząc w górę (obrót o +35 stopni)	Neutralna
6	carai abajo	Patrząc w dół (obrót o -35 stopni)	Neutralna
7	carai sonrisa	Przód	Uśmiech
8	carai risa	Przód	Śmiech
9	carai gesto	Przód	Dowolna

### 3.4 Dane z Wydziału Mechatroniki PW

Dane zostały zebrane z wykorzystaniem systemu 3DMADMAC opracowanym na Wydziale Mechatroniki Politechniki Warszawskiej. System składa się z projektora DLP i



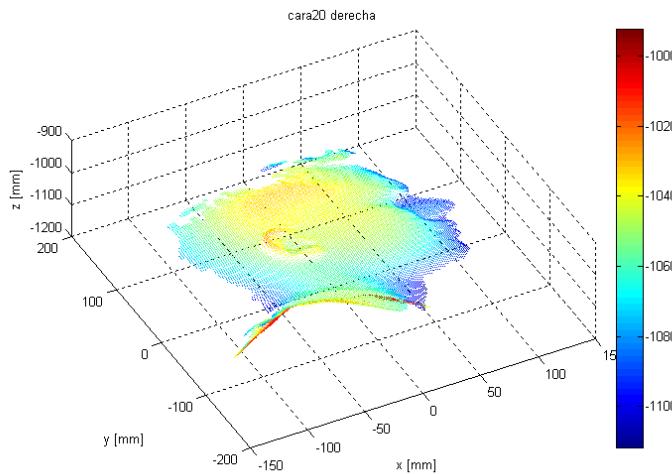
Rysunek 3.7: Ułożenie twarzy odpowiadające kolejnym skanom w bazie GavabDB [23].



Rysunek 3.8: Przykładowy skan z bazy GavabDB (widok twarzy z przodu).

kamery CCD [18]. Użyto projektora Toshiba TLP660. Rozdzielcość przestrzenna jego modulatorów to  $1024 \times 768$  pikseli. Częstotliwość odświeżania to 85Hz. Wyposażono go w obiektyw z zoomem ( $f=37\text{-}46\text{mm}$ ). Użyta kamera to Sanyo VCC3972P. Oparta jest ona o standardowy kolorowy detektor CCD  $1/3''$ . Pozwala na pracę w rozdzielcości  $720 \times 576$  pikseli z częstotliwością do 25 Hz. Wyposażono ją w obiektyw z optycznym zoomem ( $f=8\text{-}72\text{mm}$ ).

Baza zawiera 50 obrazów 3D wykonanych na grupie pracowników wydziału. Dla każdego z modeli wykonano kilkanaście różnych skanów. Każdy z obrazów zawiera nie-



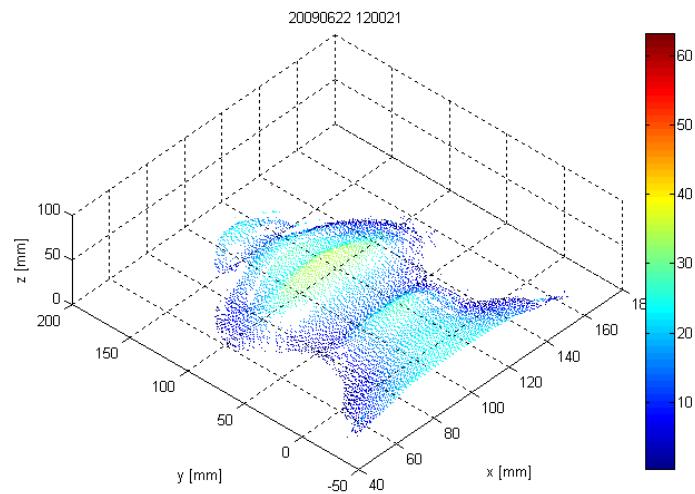
Rysunek 3.9: Przykładowy skan z bazy GavabDB (widok twarzy z prawego profilu).

obrobioną chmurę kierunkową. Mogą występować niedokładności, błędne punkty, nieciągłości powierzchni. Dane przeanalizowano organoleptycznie. Wyniki przedstawia tabela 3.2.

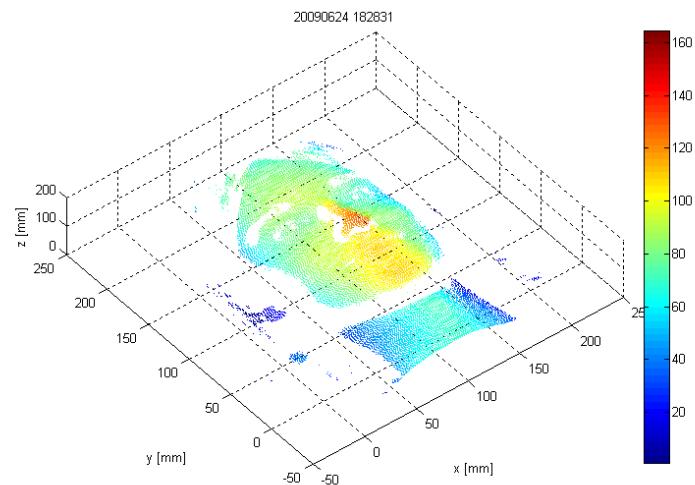
Tablica 3.2: Klasyfikacja danych z bazy Mechatroniki.

Nr grupy	Typ obrazu	Liczba obrazów 3D
1	Obraz błędny, nie zawierający twarzy	2
2	Obraz zawierający twarz	34
3	Obraz zawierający twarz, ale ze znaczącym wycięciem lub zaślonieniem części obrazu (np. czoło przysłonięte przez włosy)	8
4	Obraz zawierający twarz, ale ze znaczącymi zniekształceniami (szumy)	6

Stwierdzono, że na 6 obrazach model uśmiecha się odsłaniając zęby. Dwa obrazy zawierają ujęcie nie bezpośrednio na twarz, ale lekko z boku. Dodatkowo, pośród 48 zestawów danych zawierających twarz u 26 zaobserwowano duże uszkodzenia w okolicach obu (zarówno lewego jak i prawego) skrzydełek nosa. Uszkodzenia te uniemożliwiają analizę tych regionów. Ponadto w dużej części obrazów przypisanych do grupy 2 zaobserwowano pewne nieciągłości, które jednak wizualnie były znacznie mniejsze od tych jakie zauważono u danych z grupy 3. Przykładowe obrazy 3D widoczne są na rysunkach 3.10 i 3.11.



Rysunek 3.10: Przykładowy obraz z bazy Mechatroniki (widoczna ucięta góra część twarzy).



Rysunek 3.11: Przykładowy obraz z bazy Mechatroniki (widoczne nieciągłości w modelu i punkty nie należące do twarzy).

### 3.5 Analiza porównawcza zbiorów danych

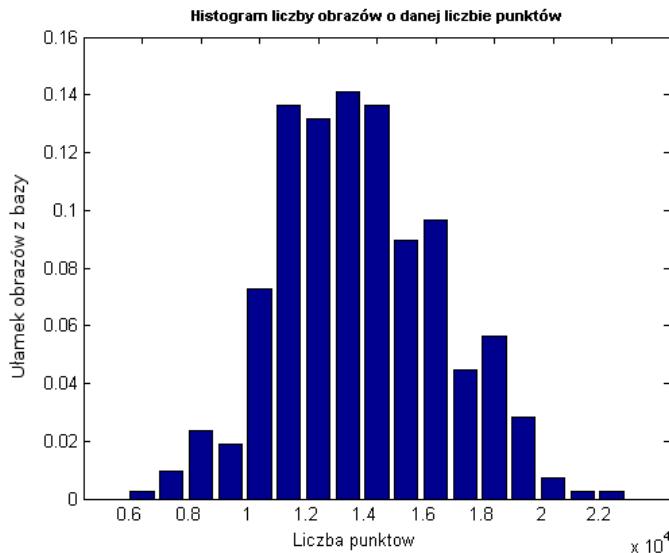
Przegląd danych z obu baz pokazał, że baza GavabDB dostarcza danych znacznie lepszych w sensie subiektywnego odbioru przez człowieka. Obrazy w bazie Mechatroniki zawierają liczne nieciągłości powierzchni oraz przypadkowe punkty. Również pod względem doboru modeli pierwszy z wymienionych zestawów danych wydaje się lepszy. Skanów dokonano na znacznie większej liczbie osób i w standardowy sposób pozwalający na przeprowadzanie analiz statystycznych np. dla wybranego ułożenia twarzy.

W celu podstawowej oceny złożoności danych wejściowych do analizy obliczono ich podstawowe statystyki. Wyniki zestawiono w tabeli poniżej. Jak widać obrazy z bazy Mechatroniki zawierają średnio około 5 razy więcej punktów (około 65 tys.) niż obrazy z bazy GavabDB (około 14 tys.). Największy w sensie liczby punktów model z bazy GavabDB jest mniejszy od najmniejszego z Mechatroniki.

Tablica 3.3: Statystyki liczby punktów w poszczególnych bazach obrazów 3D.

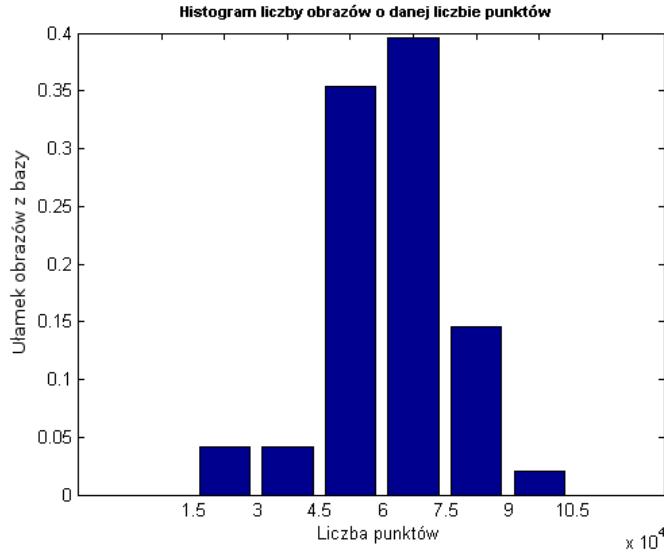
Nazwa bazy	Minimum	Maksimum	Średnia	Odchylenie standardowe
GavabDB	6 001	22 021	13 923	2 768
Mechatronika	24 804	113 295	64 933	17 837

W celu wizualizacji rozkładu liczby punktów w obu danych zaprezentowano poniższe histogramy (rysunki 3.12 i 3.13). Jak widać rozkład dla bazy GavabDB przypomina rozkład normalny o odchyleniu około 3 tys. punktów. Dla mechatroniki odchylenie standardowe wynosi około 18 tys. punktów.



Rysunek 3.12: Rozkład liczby punktów dla obrazów z bazy GavabDB.

W celu zademonstrowania jak informacje o liczbie punktów w modelu przekładają się na rozdzielcość modeli wykonano analizę liczby punktów w zależności od rozmiaru otoczenia. Wybrano losowo po 5 obrazów z każdej z baz. Dla każdego z zestawów punktów wylosowano podzbiór o rozmiarze 1% całej liczby punktów. Każdy z punktów w podzbiorze otaczano sferą o promieniu  $r_e$ . Następnie zliczano liczbę punktów wewnątrz



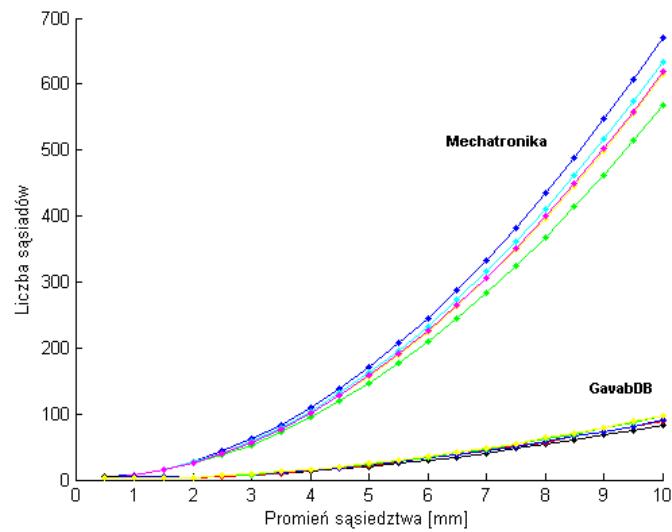
Rysunek 3.13: Rozkład liczby punktów dla obrazów z bazy Mechatroniki.

Tablica 3.4: Przykładowe dane wybrane do analizy rozdzielczości obrazów.

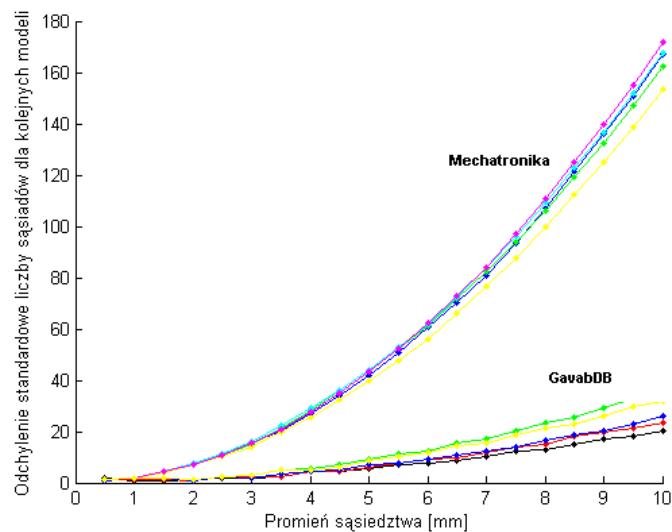
Obrazy z bazy GavabDB	Obrazy z bazy Mechatroniki
cara46 frontal1	20090622 140322
cara50 frontal1	20090625 111725
cara53 frontal2	20090623 113452
cara48 frontal1	20090625 112127
cara52 frontal2	20090625 123233

sfer. W kolejnym kroku uśredniano oraz obliczano odchylenie standardowe (w celu pokazania "stabilności" wyniku) od średniej dla każdego pojedynczego obrazu. Obliczeń dokonano dla wartości promienia  $r_e$  zmieniającej się od 0,5 mm do 10 mm, ze zmianą równą 0,5 mm. Listę testowanych danych podano w tabeli 3.4. Wyniki analizy zaprezentowano na rysunkach 3.14 i 3.15.

Jak widać dane z bazy Mechatroniki są znacznie "gęstsze". Liczbę 100 punktów w sferze otaczającej dany punkt uzyskuje się już średnio dla otoczenia około 4 mm. W celu uzyskania podobnej liczby w bazie GavabDB należy wziąć sferę o promieniu 10 mm. Tak duża różnica w rozdzielczości danych wynika z wyżej opisanego procesu wstępnego uśrednienia któremu poddane zostały dane w drugiej z wymienionych baz. Nie można więc przesądzać o użyteczności danych do analizy tylko na podstawie tej informacji. Należy pamiętać również o jakości danych z obu baz tj. o tym, że optycznie dane z bazy GavabDB wydają się znacznie lepsze.



Rysunek 3.14: Średnia liczba punktów w sąsiedztwie danego punktu w zależności od promienia sfery otaczającej dany punkt dla 10 skanów z 2 baz.



Rysunek 3.15: Odchylenie standaryzowane liczby punktów w sąsiedztwie danego punktu w zależności od promienia sfery otaczającej dany punkt dla 10 skanów z 2 baz.



## Rozdział 4

---

# Deskryptory punktów

---

### 4.1 Metody analizy obrazów 3D

#### 4.1.1 Przegląd metod

W związku z upowszechnieniem się skanerów powierzchni i obrazów przestrzennych w ostatnich latach nastąpiło znaczne natężenie prac w obszarze ich analizy. Bardzo intensywne prace prowadzi się w zakresie badania obrazów powierzchni twarzy, co spowodowane jest nadziejęmi na zastosowanie ich do weryfikacji tożsamości. Wyróżnia się podejścia:

- holistyczne – polegające na całościowym porównaniu modeli bez próby odzyskania pośredniej, użytecznej dla człowieka, informacji o charakterystyce danych.
- oparte o ekstrakcję cech – podejście w którym najpierw pozyskiwana jest informacja o cechach (wybranych punktach, charakterystykach) modelu, a dopiero odzyskana informacja o własnościach cech poddawana jest analizie.

Zgodnie z postawionym we wstępie celem tj. pozyskaniem informacji o charakterystykach wybranego regionu twarzy (nosa), interesująca jest wyłącznie druga grupa metod. W jej obszarze kierunki prowadzonych prac przypisać można do jednej z klas [21]:

- metody oparte o geometrię różniczkową (*curvature analysis-based*) – są to techniki oparte o analizę matematyczną kształtu, które biorą swój początek z prób opisu i charakteryzowania obiektów dowolnego kształtu (ang. *free-form objects*) podejmowanych od połowy lat 80-tych. Są one najszerzej znane i najbardziej rozpowszechnione.
- metody specjalizowane dla cech – grupa technik opartych o próbę odzyskiwania informacji o poszczególnych cechach. Korzystają one ze specyfiki danej cechy np. poszukiwanie czoła jako największego płaskiego obszaru twarzy.
- metody oparte o deskryptory punktów (*shape representation-based method*) – techniki w których dla każdego punktu sceny buduje się pewien zestaw danych

go charakteryzujących, a następnie próbuje dopasować punkty pomiędzy wzorcem (obrazem wzorcowym), a obrazem testowanym (chmurą punktów do przeanalizowania). Możliwa jest też analiza deskryptorów za pomocą metod sztucznej inteligencji np. próba klasyfikowania ich na podstawie punktów z zastosowaniem SVM.

#### 4.1.2 Wybór metody

Podejścia oparte o geometrię różniczkową mają najdłuższą historię. Są najbliższe intuicji, która nakazuje analizować twarz posługując się pojęciami z zakresu geometrii. Próbuje się poszukiwać obszarów o danym kształcie (np. siodło - *saddle*, dolina - *rut*, grań - *ridge* itd.), a następnie dopasowywać ich wzajemne rozłożenie [7, 9] lub też klasyfikować regiony metodami sztucznej inteligencji [25]. Wadą tych metod jest skomplikowany aparat matematyczny i duża złożoność obliczeniowa. Problemem jest też stosunkowo wysoka wrażliwość na błędy w danych wejściowych. Poza tym to, że metody te są dobrze znane i zbadane, co czyni je mało interesującymi pod kątem rozważań naukowych.

Druga grupa metod są to rozwiązania w których buduje się narzędzia opierające się o specyficzne własności poszczególnych regionów lub punktów twarzy np. wierzchołka nosa [22, 16]. Dwiema największymi wadami takiego podejścia są:

- brak ogólności - metoda zaprojektowana dla konkretnej cechy jest zwykle bezużyteczna przy próbie analizy innych cech
- bardzo wysoka wrażliwość na szумy - w większości przypadków zakłada się dane bez błędów

Deskryptory punktów należą do metod statystycznych i to czyni je odpornymi na niedoskonałości w danych. Koncepcyjnie są rozwiązaniem dość prostym, ale kolejne ich zastosowania i wariancie pokazują duży potencjał w użyciu do analizy obiektów dowolnego kształtu. Jako rozwiązanie stosunkowo młode, deskryptory są wciąż atrakcyjne z punktu widzenia badań. Ich złożoność obliczeniowa waha się w zależności od sposobu użycia co pozwala na dalsze dociekania w zakresie optymalizacji.

### 4.2 Metody oparte o deskryptory punktów

W podejściu opartym o deskryptory ze zbioru punktów analizowanego obrazu (sceny) wybierany jest ich podzbiór. Następnie dla każdego z punktów w podzbiorze obliczany jest deskryptor - struktura go charakteryzująca. Budując deskryptory zwykle korzysta się z położen względnych (np. względem punktu dla którego obliczany jest histogram) co czyni je niewrażliwymi na przesunięcia. Zwykle są też odporne na obroty. Istnieje kilka typów deskryptorów, z których najpopularniejsze to:

- *spin image* (obraz obrotu)
- *local surface patch* ("łatka" lokalnej powierzchni)
- *shape context* (kontekst kształtu)

- *local shape map* (mapa lokalnego kształtu)
- *point signature* (sygnatura punktu)

Poniżej opisano pierwsze cztery. Łączy je to, że każdy z nich do liczbowego scharakteryzowania punktu stosuje dwuwymiarowy histogram. W opisywanym rozwiązaniu ostatecznie zdecydowano się na użycie obrazów *spin image*, jednak zostały one zmodyfikowane o cechy pochodzące od pozostałych trzech typów.

### 4.2.1 Local surface patch

Ciekawą strukturą opisującą punkt wraz z jego otoczeniem jest LSP (ang. *Local Surface Patch*). LSP opisuje wycinek obiektu o promieniu  $r$  wokół punktu  $p$ . Na deskryptor punktu składają się:

- typ powierzchni określony przez indeks kształtu (ang. *shape index*)
- centroid wycinka (zauważać należy, że niekoniecznie jest on równy punktowi  $p$ )
- dwuwymiarowy histogram

Indeks kształtu jest to miara wprowadzona w geometrii różniczkowej, która mapuje lokalne kształty powierzchni w wartości  $\in [0, 1]$ . Opisuje ona kształt powierzchni niezależnie od ułożenia w przestrzeni.

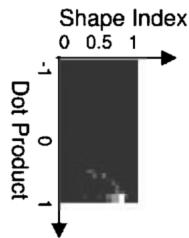
W LSP sąsiadzi (ozn.  $q$ ) punktu  $p$  zdefiniowani zostali jako punkty, których położenie zawiera się w pewnej sferze o środku w  $p$  i promieniu  $r$ :  $\|q - p\| < r$ . Dodatkowym kryterium jest, aby kierunek normalnej do powierzchni w punkcie sąsiednim  $q$  zbliżony był do kierunku normalnej w punkcie  $p$ :  $\text{acos}(\vec{n}_p, \vec{n}_q) < A$ , gdzie:

- $\vec{n}_p$  – wektor normalny do powierzchni w punkcie  $p$
- $\vec{n}_q$  – wektor normalny w punkcie  $q$  będącym potencjalnym sąsiadem
- $A$  – stała określająca próg

Kluczowym elementem LSP jest histogram. Określa on rozkład cech sąsiadów punktu  $p$ . Na jednej z osi histogramu odkładane są wartości indeksu kształtu. Druga określa wartości iloczynu skalarnego (ang. *dot product*) między kierunkami normalnymi do powierzchni w punkcie  $p$  i w punktach sąsiednich:  $\vec{n}_p \cdot \vec{n}_q$ . W celu redukcji szumu stosowana jest interpolacja dwuliniowa (ang. *bilinear interpolation*).

### 4.2.2 Shape context feature

*Shape context features* wprowadzone zostały w [2] celem rozpoznawania obiektów dwuwymiarowych, w szczególności liter pisma odrewnego. W *Shape context features* dla każdego punktu definiowany jest deskryptor: dwuwymiarowy histogram (*shape context*) liczby punktów. Z każdym punktem  $p$  dla którego budowany jest deskryptor wiąże się biegunowy układ współrzędnych. Jego początek znajduje się w tym punkcie. Oś biegunowa może mieć kierunek stały dla wszystkich deskryptorów, lub też być położona

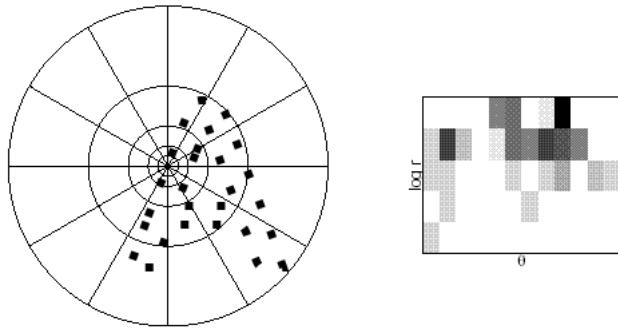


Rysunek 4.1: Przykładowy histogram wartości indeksu kształtu względem iloczynu skalarnego wektorów w LSP.

pod kątem prostopadłym do kierunku krawędzi obiektu. To drugie podejście czyni metodę odporną na obroty. W nowym układzie współrzędnych każdemu punktowi obiektu odpowiada para współrzędnych:

- $r_p$  - odległość od punktu  $p$ :  $r_p = \|p - q\|$ .
- $\Theta$  - kąt względem osi biegunowej.

Opierając się na nowych współrzędnych buduje się dwuwymiarowy histogram położenia punktów (rysunek 4.2). Na jednej osi odkładany jest  $\log(r_p)$ . Druga odpowiada wartośćom kąta  $\Theta$ .



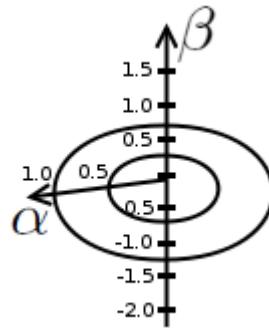
Rysunek 4.2: Przykład budowania histogramu - *Shape Context*.

### 4.2.3 Spin image i Local shape map

Reprezentacją danych 3D odporną na obroty i przesunięcia są obrazy obrotu (ang. *spin images*). Również one, opierają się o dwuwymiarowe histogramy.

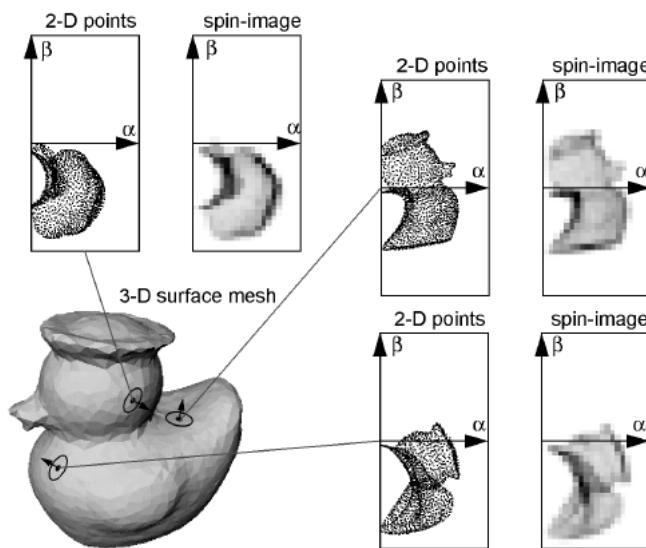
W punkcie  $p$  dla którego budowany jest deskryptor umiejscawiany jest zmodyfikowany walcowy układ współrzędnych (rysunek 4.3). Kierunek jednej z osi jest zdefiniowany przez kierunek wektora normalnego do powierzchni w tym punkcie:  $\vec{n}_p$ . Pozostałe dwie osie nie są ściśle określone. Znana za to jest płaszczyzna je zawierająca - jest ona wyznaczona jako płaszczyzna styczna (ang. *tangent*) do powierzchni obiektu w punkcie: ozn.  $Tan_p$ . Tak zbudowany układ (os i płaszczyzna) pozwala przyporządkować każdemu z punktów obiektu dwie współrzędne:

- $\alpha$  - odległość radialna. Odległość do osi określonej przez kierunek wektora normalnego w punkcie  $p$ . Symbolicznie można zapisać:  $\alpha = |q - \vec{n}_p|$ .
- $\beta$  - odległość (wraz ze znakiem determinującym położenie nad/pod) do płaszczyzny stycznej do powierzchni w punkcie:  $\beta = \text{sign}(q - \text{Tan}_p) \cdot |q - \text{Tan}_p|$ .



Rysunek 4.3: Zmodyfikowany układ współrzędnych stosowany w obrazach obrotu.

Budując histogram (*spin image*) przestrzeń dzieli się na "kubełki" odpowiadające kolejnym przedziałom wartości  $\alpha$  i  $\beta$ . Następnie zliczana jest liczba punktów obiektu trafiających do danego "kubelka". W zależności od przyjętej liczby przedziałów uzyskuje się deskryptory (histogramy) o różnej rozdzielczości. Procedurę zilustrowano na rysunku 4.4.



Rysunek 4.4: Ilustracja budowania obrazów obrotu dla różnych punktów obiektu [11].

Deskryptorem koncepcyjnie bardzo podobnym do *spin image* jest *local shape map* (LSM). Różni je od siebie sposób definiowania współrzędnej  $\alpha$ . W LSM obliczana jest ona jako odległość punktu od środka układu współrzędnych (umiejscowionego w punkcie dla którego obliczany jest deskryptor):  $\alpha = \|q - p\|$ . Dodatkowo przy budowie histo-

gramu LSM uwzględniane są jedynie punkty położone w pewnej ograniczonej odległości  $r$  od punktu dla którego budowany jest deskryptor:  $\|q - p\| < r$ .

## 4.3 Aspekty deskryptorów

### 4.3.1 Zastosowania deskryptorów

Istnieją dwa podejścia do użycia deskryptorów punktów:

1. Zastosowanie polegające na klasyfikacji pojedynczych punktów. Stosując metody sztucznej inteligencji próbuje się odnajdywać punkty spełniające dane kryterium. Takie podejście zastosowano np. w [8]. Podjęto próbę odnalezienia trzech punktów charakterystycznych twarzy: wierzchołka nosa (prn) i kącików wewnętrznych oczu (en). W tym celu klasyfikowano obrazy obrotu punktów na twarzy za pomocą SVM. Odnalezione przybliżone położenia punktów zastosowano do normalizacji ułożenia twarzy.

Wadami takiego podejścia są:

- potrzeba przygotowania odpowiednich rozmiarów i wysokiej jakości zbioru uczącego.
- bardzo wysoki koszt obliczeniowy. Wygenerowanie obrazów obrotów (a następnie klasyfikacja) dla wszystkich punktów twarzy jest niezwykle czasochłonna. Redukcja czasu wykonania następuje poprzez wstępную selekcję regionów dla których generowane będą deskryptory. Robi się to wykorzystując informację o krzywiźnie obszarów (co również jest dość kosztowne).
- wrażliwość na błędy - pojedyncze niepoprawne punkty danych mogą mieć charakterystyki bardzo podobne do tych poszukiwanych.

W [8] pokazane zostało, że **nie jest możliwe wykorzystanie obrazów obrotu do jednoznacznej identyfikacji punktów charakterystycznych twarzy**. Analiza pojedynczych deskryptorów pozwala jedynie na przybliżoną lokalizację kandydatów na punkty charakterystyczne. W celu dalszej selekcji i przybliżenia rozwiązania zastosować należy inne podejście.

2. Zastosowanie w którym dopasowuje/przyporządkowuje (ang. *match*) się deskryptory zbudowane na podzbiorze punktów analizowanego obiektu do analogicznego podzbioru wzorca. Traktując twarz jako pewną przybliżoną instancję wzorca wyszukiwanie punktów na twarzy można potraktować w kategoriach problemów znajdowania odpowiedniości (ang. *correspondence problem*) pomiędzy punktami we wzorcu i testowanym obiekcie/scenie. W [5] pokazano, że **deskryptory nie muszą mieć identycznych charakterystyk i nie muszą być przypisane do dokładnie odpowiadających sobie punktów by zostać do siebie dopasowane**. Jest możliwe np. dopasowanie wierzchołka nosa z jednego obrazu twarzy do punktu położonego lekko z boku wierzchołka w drugim obrazie. Należy jednak pamiętać, że problem ma swoją specyfikę: część punktów może nie istnieć, a mogą istnieć regiony nie uwzględnione we wzorcu.

Podejście oparte o taki sposób użycia deskryptorów znalazło swoje zastosowanie przy:

- rozpoznawaniu obiektów dowolnego kształtu (ang. *free-form objects*). W [2, 1, 24] wykorzystano (ze skutecznością 99.37% [MNIST]) *shape contexts* do rozpoznawania pojedynczych liter pisma odręcznego. Dopasowywano deskryptory z analizowanych symboli do deskryptorów wygenerowanych dla liter wzorcowych. Następnie wybierano to dopasowanie, którego odległość (sumaryczny koszt) była najmniejsza. Podobne podejście, ale do rozpoznawania twarzy z wykorzystywaniem obrazów obrotu, zastosowano w [14] oraz z wykorzystaniem sygnatur punktów w [5]. W tych przypadkach bibliotekę stanowił zbiór obrazów 3D twarzy osób.
- lokalizacji obiektów w scenie. W [11] i [16] użyto obrazów obrotu do poszukiwania obiektów w scenie 3D. Dla każdego punktu generowano deskryptor, które następnie (po redukcji wymiarowości za pomocą PCA) dopasowywano do deskryptorów obiektów z biblioteki. W [22] do podobnego celu użyto *local surface patches*.

W niniejszej pracy wykorzystano podejście nr 2 polegające na dopasowywaniu deskryptorów do deskryptorów wzorca. W algorytmie opisany w rozdziale 9 zostały one użyte w sposób częściowo zbliżony do opisanego w [11, 16]. Za ich pomocą wybierano jedną z grup punktów najbardziej pasującą do wzorca twarzy. W rozdziale 10 zaproponowano, z kolei, algorytm czerpiący z obu zastosowań. Deskryptory wykorzystane zostały tam do selekcji podzbioru punktów odpowiadającego wzorcowi.

### 4.3.2 Punkty sąsiednie

Budując histogram rozmieszczenia punktów wokół  $p$  zadecydować należy które z punktów  $q$  obrazu 3D zostaną uwzględnione w obliczeniach. W obrazach obrotu brane pod uwagę są te punkty w których normalna do powierzchni nie różni się znacznie od normalnej w punkcie  $p$ :  $\text{acos}(\vec{n}_p \cdot \vec{n}_q) < A_s$ , gdzie  $A_s$  - stała. Wadą takiego podejścia jest konieczność obliczenia wektorów normalnych we wszystkich punktach obrazu, co jest niezwykle czasochłonne. Innym podejściem jest uwzględnienie wszystkich punktów sceny - zakładając, że ostatni z przedziałów jest nieskończony. Możliwe jest też ograniczenie punktów wpływających na histogram tylko do tych które położone są w pewnej odległości mniejszej niż  $r$  (sąsiedztwie) od punktu  $p$ :  $\|q - p\| < r$ . Podejście takie zastosowano w *local shape map* i *point signature*.

### 4.3.3 Skala i skwantowanie

Istotnymi parametrami, które rozważyć należy przy budowie deskryptorów opartych o histogramy są: typ skali i jej skwantowanie.

Pierwszym krokiem obliczeń deskryptorów jest przyporządkowanie każdemu punktowi w sąsiedztwie (ozn.  $q$ ) pary liczb: jego nowych współrzędnych (do opisu położenia w przestrzeni 3D stosowane są dwie współrzędne, więc nie jest wykluczone, że te same wartości przyporządkowane zostaną do kilku, położonych w zupełnie różnych miejscach punktów np. w obrazach obrotów wszystkie punkty położone na okręgu o środku położonym na prostej wyznaczonej przez wektor normalny i równoległy do płaszczyzny

stycznej mają te same współrzędne). W kroku drugim budowany jest dwuwymiarowy histogram rozkładu punktów w nowych współrzędnych. Dla każdego przedziału współrzędnych ("kubelka") zliczone zostają punkty do niego należące (patrz rysunki 4.2 i 4.3).

Kwestią, którą należy rozważyć jest sposób podziału przestrzeni w nowych współrzędnych. Należy zadecydować na ile przedziałów zostanie podzielona każda z osi (jaka będzie rozdzielcość) oraz jak zostaną one rozmieszczone (typ skali). Im większa rozdzielcość skali tym większy wpływ na wartości histogramu mają pojedyncze punkty. Zwiększa się ilość informacji jaką niesie ze sobą deskryptor. Zwiększa się też jednak wrażliwość na błędy i wpływ efektu próbkowania powierzchni przy skanowaniu obiektów [11]. Bardzo istotne jest zbalansowanie wpływu tych czynników. W obrazach obrotu rozdzielcość uzależniona została od rozdzielcości całego modelu. W *shape contexts* skwantowanie przyjęto eksperymentalnie.

Oryginalnie dla obrazów obrotów zastosowano skalę liniową tj. kolejne przedziały mają taką samą szerokość. W *shape contexts*, z kolei, dla kątów zastosowano skalę liniową, a dla odległości logarytmiczną (patrz sekcja 4.2). Użycie skali logarytmicznej różnicuje wpływ punktów  $q$  położonych w różnych odległościach od punktu  $p$ . Obszary w mniejszej odległości reprezentowane są dokładniej, przez co ich późniejszy wpływ przy porównywaniu i dopasowywaniu deskryptorów jest większy [2].

#### 4.3.4 Odległość między deskryptorami

W celu porównania i oceny kosztów dopasowania dwóch deskryptorów wprowadza się pojęcie odległości między nimi. Odległość jest miarą podobieństwa między deskryptorami. Im jest ona mniejsza tym są one bardziej do siebie podobne. W większości opracowań stosuje się jedną z dwóch miar:

- odległość stosowana w [1, 2, 24] do dopasowywania *shape contexts*, a także w [3] przy przyporządkowywaniu *local surface patch*. Dalej umownie nazywana ona będzie *shape context measure* (SCM):

$$c_1(g, h) = \frac{1}{2} \sum_{k=1}^K \frac{[g(k) - h(k)]^2}{g(k) + h(k)} \quad (4.1)$$

W przypadku zastosowania SCM histogramy są znormalizowane (tj. suma zawartości "kubelków" wynosi 1).

- odległość oparta na korelacji liniowej Pearsona, stosowana w [16, 5, 11, 14]. Wersja o wartościach zwracanych zgodnych z *shape context measure*:

$$c_2(g, h) = \frac{1}{2} \left( 1 - \frac{\text{cov}(g, h)}{\sqrt{\text{var}(g) \cdot \text{var}(h)}} \right) \quad (4.2)$$

gdzie:

- $g, h$  - deskryptory (histogramy) między którymi liczona jest odległość. Oryginalnie są one dwuwymiarowe, ale bez wpływu na zachowanie algorytmu można je traktować jako jednowymiarowe (układając np. kolejne wiersze obok siebie).

- $k$  - numer kolejnego "kubelka" histogramu
- $K$  - liczba "kubelków" histogramu (dla obu histogramów jest ona taka sama)
- $var/cov$  - estymator wariancji/kowariancji (kolejne wartości "kubelków" traktowane są jako próby losowe)
- $c \in [0, 1]$  - koszt dopasowania między dwoma histogramami. Gdy  $c = 0$  oznacza to, że deskryptory opisują zupełnie różne regiony. W przypadku gdy  $c = 1$  to deskryptory są identyczne.

### Filtrowanie deskryptorów

W [21] zaobserwowano, że przy budowie deskryptorów wiele "kubelków" zawiera zera. Jest to spowodowane tym, że punkty obrazu w bardzo małym stopniu (lub wcale) pokrywają przedziały wartości im odpowiadające. Założono, że zera pogarszają jakość dopasowywania deskryptorów. W celu redukcji tego efektu wprowadzono proste filtrowanie: "kubelki" zerowe nie są uwzględniane przy obliczaniu odległości między deskryptorami. Możliwe są dwa podejścia:

- filtrowanie AND - dwa odpowiadające sobie "kubelki"  $g(k)$  i  $h(k)$  brane są pod uwagę przy dopasowywaniu tylko gdy oba są różne od zera:  $g(k) \neq 0$  i  $h(k) \neq 0$
- filtrowanie OR -  $g(k)$  i  $h(k)$  brane są pod uwagę przy dopasowywaniu gdy co najmniej jeden z nich jest różny od zera:  $g(k) \neq 0$  lub  $h(k) \neq 0$ .

Regułę w której wszystkie elementy histogramu brane są pod uwagę oznaczono jako NONE.

Dodatkową korzyścią z wykorzystania filtrów, o której nie wspomniano w [21], a która wykorzystana została w niniejszej pracy jest ich zdolność do usuwania części danych które nie powinny być uwzględniane przy dopasowywaniu. Możliwa jest sytuacja gdy we wzorcu w pewnym regionie nie ma punktów, a układ sceny powoduje, że w analogicznym regionie wyszukiwanego obiektu są punkty będące np. częścią innego obiektu. Używając filtra AND uniknie się wówczas wpływu tych punktów przy dopasowywaniu deskryptorów.

#### 4.3.5 Dopasowywanie punktów i algorytm węgierski

Spośród metod dopasowywania (przyporządkowywania, ang. *match*) deskryptorów (i odpowiadających im punktów) na największą uwagę zasługuje podejście zastosowane w związku z *shape contexts*. Założonym celem jest znalezienie takiego dopasowania pomiędzy parami deskryptorów (jeden z wzorca i jeden z analizowanego obrazu), które zminimalizowałoby sumaryczny koszt:

$$\min_U C = \sum_{(g,h) \in U} c(g, h) \quad (4.3)$$

gdzie przez  $U$  rozumie się zbiór dopasowań (transwersalę) między deskryptorami:  $g \in G$ ,  $h \in H$ . Liczność  $U$  jest równa mniejszej z liczności zbiorów deskryptorów wzorca i analizowanego obrazu:

$$|U| = \min(|G|, |H|)$$

gdzie  $G$  i  $H$  to odpowiednio zbiory deskryptorów z wzorca i obrazu. Oznacza to tyle, że jeśli jeden z tych zbiorów jest liczniejszy to dopasowanych zostanie tylko tyle, ile jest w drugim zbiorze.

W pozostałych podejściach stosuje się różne heurystyki, które zwykle przyspieszają czas obliczeń, jednak w wielu przypadkach powodują, że uzyskane rezultaty są dalekie od optymalnych. Kryterium 4.3 wymusza sposób dopasowania który nie jest osiągalny w innych przypadkach np. możliwa jest sytuacja kiedy przyporządkowane do siebie są deskryptory, które są od siebie bardzo odległe, ale w łącznej optymalizacji powodują najmniejszy koszt. Dodatkowo, czasochłonność procedury może zostać zbalansowana zmniejszeniem liczby deskryptorów. Na potrzeby algorytmu opisanego w rozdziale 10 inne metody mogłyby się okazać niewystarczające.

Dopasowanie punktów między wzorcem, a analizowanym obiektem (ang. *assignment problem*) w sposób spełniający kryterium 4.3 jest możliwe w czasie wielomianowym. Najpopularniejszą z metod jest algorytm węgierski z 1955 roku. Jego złożoność czasowa w wersji oryginalnej to  $O(M^4)$ , jednakże Edmonds i Karp oraz niezależnie Tomizawa pokazali, że możliwe jest jego przyspieszenie do  $O(M^3)$ , gdzie  $M$  - liczność większej z grup dopasowywanych elementów. Algorytm w najpopularniejszych implementacjach operuje na macierzach. Wiersze odpowiadają punktom wzorca, kolumny - punktom analizowanej sceny. Na przecięciu umiejscawiany jest koszt przyporządkowania.

Wynikiem działania algorytmu węgierskiego jest macierz reprezentująca dopasowanie pomiędzy punktami wzorca (wiersz), a obrazu testowanego (kolumna). Macierz zawiera maksymalnie jedną 1 w każdej kolumnie i maksymalnie jedną wartość 1 w każdym wierszu, co oznacza jednoznaczne przypisanie.

Oprócz czasu przyporządkowania dwóch zbiorów  $G$  i  $H$ , o licznościach odpowiednio:  $M_G = |G|$  i  $M_H = |H|$ , przy ocenie złożoności metody uwzględnić należy też czas przygotowania macierzy kosztów. W pierwszym kroku obliczane są deskryptory dla dwóch obrazów o liczbie punktów odpowiednio  $N_G$  i  $N_H$ . Koszty wynoszą  $O(N_G \cdot M_G)$  i  $O(N_H \cdot M_H)$ . Następnie obliczane są odległości między każdym dwoma deskryptorami:  $O(M_H \cdot M_G \cdot K)$ , gdzie  $K$  - liczba "kubelków". Dopasowywanie jest ostatnim krokiem.

#### 4.3.6 Strategie wyboru podzbioru punktów

Przy zastosowaniu deskryptorów do dopasowywania (ang. *matching*) obiektu (podzbioru punktów poszukiwanego w scenie na obrazie 3D) do wzorca kwestią podstawową jest wybór punktów dla których zostaną one obliczone. Kwestię tę rozważać można niezależnie dla obiektu i wzorca. W literaturze występuje kilka podejść do tego problemu:

1. Generacja deskryptorów dla wszystkich punktów obrazu. Metodę taką zastosowano do sygnatur punktów (ang. *point signatures*) i obrazów obrotu w [6, 11]. Wadą takiej metody jest ogromny nakład obliczeniowy zarówno przy generacji deskryptorów, jak i później przy dopasowywaniu grup deskryptorów z wzorca i z obiektu. Zaletą jest lepsza jakość dopasowywania: sumaryczny koszt dopasowania dwóch identycznych obrazów 3D jest równy 0. Omijany jest też (opisany dalej) problem względnego przesunięcia punktów.
2. Generacja deskryptorów dla punktów w których znajdują się ekstrema lokalne wartości indeksu kształtu (ang. *shape index*). Punkty takie wydają się najwyraźniej

ściej reprezentować charakterystykę danego regionu. Podejście takie pojawia się w związku z LSP w [3, 4]. Problemem jest wysoki koszt obliczenia wartości indeksu kształtu dla punktów i poszukiwanie ekstremów. Zaletą jest determinizm wyboru punktów. Ponieważ liczba wybranych punktów jest też znacznie mniejsza niż w podejściu nr 1, to koszt obliczeniowy dopasowywania jest też znacznie niższy. Dodatkowo opisywana metoda cechuje się wszystkimi pozytywnymi własnościami podejścia nr 1.

3. Generacja deskryptorów dla punktów wybranych losowo z analizowanej chmury. Preferuje się rozkład równomierny na całej powierzchni obiektu, choć nie jest to krytyczne dla stosowania [2]. Podejście to zastosowane zostało w związku z *shape contexts* w [1, 2, 24]. Opiera się ono o kluczową własność deskryptorów punktów: **punkty położone blisko siebie cechuje podobna charakterystyka** [5]. Najistotniejszą zaletą metody jest bardzo niski koszt obliczeniowy generacji. Dodatkowo możliwy jest zewnętrzny wybór (np. podanie jako parametr) liczby punktów dla których nastąpi generacja. Pozwala to wpływać m.in. na koszt dopasowywania. Im jest ich mniej tym czas dopasowywania będzie krótszy. Wadą jest brak determinizmu: za każdym razem, zbiór wygenerowanych deskryptorów będzie się nieznacznie różnił. Pogarsza się jakość dopasowywania: koszt przyporządkowania 1-1 dwóch zbiorów deskryptorów wygenerowanych na tym samym obrazie wejściowym jest niezerowy. Pojawia się problem względnego przesunięcia punktów np. można sobie wyobrazić następującą sytuację:
- a) generowane są deskryptory we wzorcu i obiekcie
  - b) następuje dopasowanie minimalizujące sumaryczny koszt
  - c) do punktu na wierzchołku nosa we wzorcu dopasowany zostaje punkt położony na grzbiecie nosa, gdyż był on najbliższy wierzchołkowi nosa z punktów wybranych do generacji deskryptorów w obiekcie

Algorytm wykonany został całkowicie poprawnie, jednak intuicyjnie uzyskane rozwiązanie nie wydaje się być do końca satysfakcyjne.

## 4.4 Wybór rozwiązania

Na potrzeby niniejszej pracy zdecydowano się zaadaptować zmodyfikowane obrazy obrotów (ang. *spin image*). Koszt obliczeniowy potrzebny do wygenerowania deskryptorów jest znacznie niższy, niż dla *local surface patch*. Są one też znacznie częściej stosowane w analizie obrazów 3D niż *local shape map*, do których są bardzo podobne.

Wprowadzając zmiany do oryginalnych deskryptorów kierowano się minimalizacją złożoności obliczeniowej i dążeniem do osiągnięcia maksymalnej prostoty przy zachowaniu wysokiej skuteczności rozwiązania.

Wprowadzone modyfikacje polegają m.in. na uproszczeniu reguły wyboru punktów sąsiednich. Uwzględnianie są wszystkie punkty  $q$  obrazu położone w odległości nie większej niż promień otoczenia  $r$  od punktu  $p$  dla którego wykonywane są obliczenia. Dzięki temu unika się konieczności estymacji wektorów normalnych dla punktów obrazu w których nie będą generowane deskryptory.

Zdecydowano się też na użycie filtrów, czego powodem jest specyfika zastosowanego podejścia (szczegóły opisano w rozdziale 5). Jak pokazano dalej wpływają one pozytywnie na sprawność rozwiązania.

W kontekście porównywania i dopasowywania sprawdzono zachowanie się obrazów obrotu dla modyfikacji zaczerpniętych z *shape contexts*:

- skali logarytmicznej
- odległości w sensie *shape context measure*

Nie wpływają one na kosztowność rozwiązania, a zwiększą efektywność stosowanych algorytmów, głównie przy dopasowywaniu. Do przyporządkowania deskryptorów między obrazami użyto **algorytmu węgierskiego**. Jego zalety opisano w rozdziale 4.3.5. W celu zbalansowania czasu jego wykonania zredukowano liczbę generowanych deskryptorów. Nie są one generowane dla całego obrazu, ale jedynie dla części punktów. W ten sposób redukuje się też czas budowania macierzy kosztów. Z podobnych побudek zdecydowano się na zastosowanie podejścia opartego o losowy wybór podzbioru punktów  $p$  dla których obliczane są deskryptory. Niewątpliwie przyspiesza to obliczenia i upraszcza algorytmy.

## Rozdział 5

---

# Schemat systemu

---

### 5.1 Kluczowe idee

Projektując opisywany system przyjęto strategię od ogólnego do szczegółowego. Wynika to z trudności w zaprojektowaniu jednolitego algorytmu, który musiałby radzić sobie ze wszystkimi problemami dotyczącymi danych jak i być dostatecznie precyzyjny w analizie.

W niniejszej pracy zaproponowano nowe zastosowanie obrazów obrotu (ang. *spin image*). Użyte zostały one do hierarchicznej lokalizacji, coraz mniejszych, regionów twarzy. W kolejnych fazach poszukuje się coraz precyzyjniejszego otoczenia punktów charakterystycznych nosa. Jak wyjaśniono we wprowadzeniu teoretycznym obrazy obrotu nie mają dostatecznej siły wyrazu by ich użyć do identyfikacji pojedynczych punktów charakterystycznych. Założono jednak, że jest możliwe ich wykorzystanie do identyfikacji podregionów twarzy.

Przy projektowaniu systemu przyjęto podejście do uczenia oparte o instancje (ang. *instance-based learning*), zwane dalej wzorcami. W oparciu o część dostępnych obrazów 3D zbudowano bibliotekę wzorców twarzy. We wzorcach następnie oznaczono konkretne regiony i podregiony. Tak zbudowaną bazę wzorców wykorzystano następnie do analizy obrazów 3D. **Założono, że obrazy obrotów budowane dla punktów należących do pewnych segmentów w obrazach 3D będą z dużą dokładnością dopasowywane do odpowiadających im segmentów w obrazach wzorcowych.** Nie zakłada się pełnej dokładności dopasowania punktów - możliwe są przyporządkowania całkowicie błędne np. lokalizując region nosa możliwe, że do części deskryptorów wzorca wygenerowanych w regionie nosa przyporządkowane zostaną deskryptory punktów z czoła lub ust. Przyjmuje się jednak, że możliwe jest odfiltrowanie punktów dopasowanych błędnie.

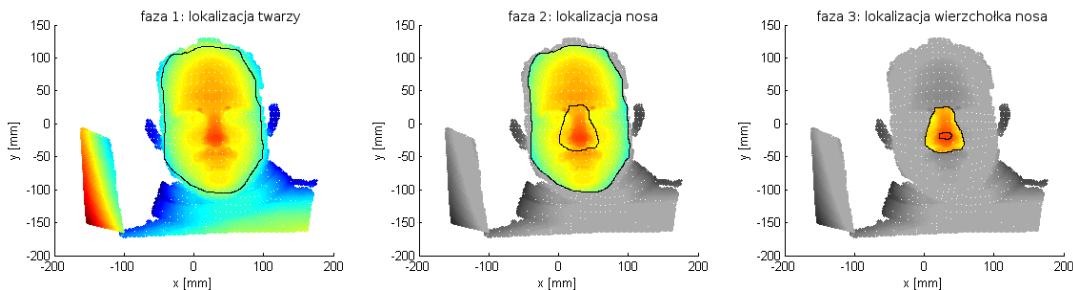
Zastosowanie hierarchicznego podejścia do lokalizacji regionów uzasadnia się dwiema przesłankami:

- kosztem obliczeniowym. Zakłada się, że w celu precyzyjnej lokalizacji regionu w scenie, odpowiednio wiele punktów powinno zostać poprawnie dopasowanych. W przypadku identyfikacji małego podzbioru (punktów), w dużej chmurze punktów, deskryptory muszą zostać wygenerowane odpowiednio gęsto. Oczekuje się, że w

poszukiwanym segmencie znajdzie się ich wystarczająco wiele prawidłowo przyporządkowanych, tak by na podstawie ich rozmieszczenia możliwe było rozróżnienie identyfikowanego regionu od tła (szczegóły wyjaśniono w rozdziałach 8 i 10).

Im większy obszar jest lokalizowany, tym można to robić z mniejszą precyzją i generując mniejszą liczbę deskryptorów. Przykładowo, jeśli do lokalizacji otoczenia wierzchołka nosa potrzeba np. 25 poprawnie przyporządkowanych punktów, to na całej twarzy rozmieścić należy ich około 500. Wprowadzając etap pośredni w postaci poszukiwania nosa koszt ten zredukować można do: 100 deskryptorów rozmieszczonych na twarzy w celu znalezienia nosa + 100 na nosie by zlokalizować poszukiwany region.

- redukcją prawdopodobieństwa błędnego dopasowania deskryptorów (punktów). Jeśli zbiór punktów (region) wybierany jest z mniejszego zbioru danych, to liczba deskryptorów wzorca do których dopasowuje się deskryptory analizowanego obiektu jest mniejsza. Mniejsza jest więc szansa błędnego dopasowania punktu z regionu do punktu spoza regionu we wzorcu.



Rysunek 5.1: Ilustracja koncepcji hierarchicznej lokalizacji regionów.

## 5.2 Schemat ogólny systemu

W zaprojektowanym systemie rozwiązanie problemu detekcji punktów charakterystycznych rozbito na trzy fazy (rysunek 5.2):

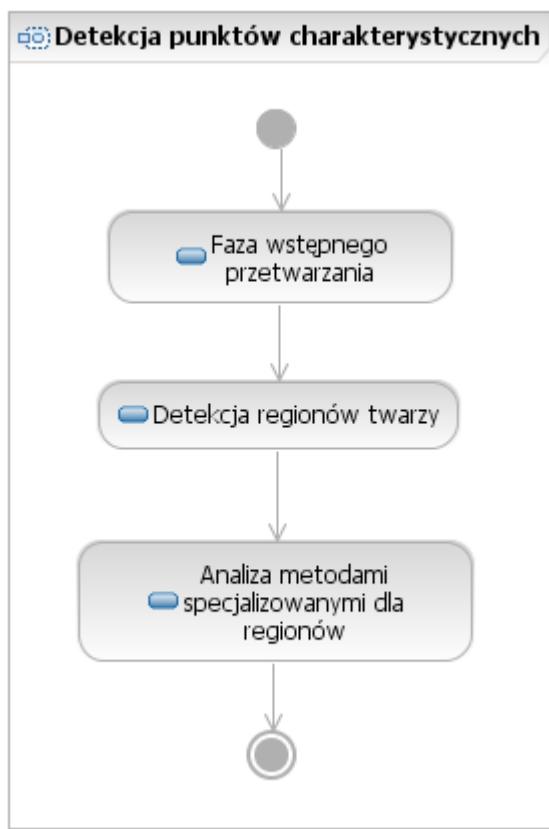
1. wstępne przetwarzanie i analiza
2. fazę lokalizacji regionów otaczających punkty charakterystyczne
3. fazę analizy regionów pod kątem identyfikacji punktów charakterystycznych

Dane wejściowe są zaszumione i zawierają błędy, więc celem fazy nr 1 jest oczyszczenie obrazu i zredukowanie wpływu niepoprawnych danych. Usuwane są błędne punkty oraz odrzucone są zbędne elementy sceny. Oczekiwany wynikiem wykonania tej procedury jest spójna chmura punktów zawierająca w sobie twarz.

Drugim krokiem jest analiza twarzy. W wyselekjonowanych wcześniej punktach wyszukiwana jest twarz. Następnie, dokonywany jest podział twarzy na coraz mniejsze regiony. Ostatecznym wynikiem działania fazy nr 2 jest identyfikacja niewielkich regionów otaczających punkty charakterystyczne na nosie.

Celem ostatniego kroku jest identyfikacja konkretnych punktów charakterystycznych na twarzy. Jak wyjaśniono we wstępnie teoretycznym nie jest to możliwe wykorzystując deskryptory punktów. W związku z tym faza ta została zredukowana do minimum. Za punkty charakterystyczne uznaje się środki ciężkości regionów zidentyfikowanych w fazie 2.

Cechą wyróżniającą zaproponowaną strukturę jest brak kroku normalizacji - pozycjonowania danych po wstępny filtrowaniu. Powodem pominięcia tego kroku jest problem z dokładnym zdefiniowaniem ułożenia, które wnosiłoby istotną informację o umiejscowieniu poszukiwanych obszarów. Innymi słowy, nie ma takiego ułożenia, które pozwalałoby w dalszych krokach ominąć dalsze dopasowywanie. Brak normalizacji zmniejsza złożoność obliczeniową, ale jednocześnie powoduje, że opisane dalej metody nie są wrażliwe na obroty i translacje.



Rysunek 5.2: Schemat struktury ogólnej zaprojektowanego systemu.



## Rozdział 6

---

# Wstępne przetwarzanie

---

Zakłada się, że dane wejściowe do algorytmu są bardzo niedoskonałe:

1. zawierają błędne punkty i fragmenty
2. pojawić się może rozrzut punktów
3. obraz zawiera twarz, ale wraz z pewnym niezdefiniowanym otoczeniem

W celu zniwelowania wpływu niedoskonałości i przygotowania obrazu do dalszej analizy zastosowano metodę opisaną poniżej. W pierwszej fazie punkty obrazy 3D zostają przefiltrowane w celu usunięcia błędnych punktów. Do ich eliminacji zaadaptowano algorytm zaproponowany w [18]. Składa on się z dwóch faz:

1. Pierwszym krokiem jest segmentacja obrazu poprzez grupy połączone punktów. Grupę połączoną definiuje się jako zbiór punktów, z których każde dwa są ze sobą pośrednio lub bezpośrednio połączone. Dwa punkty  $p$  i  $q$  są bezpośrednio połączone jeśli odległość między nimi jest nie większa niż pewna wartość progowa  $D$ :  $\|p-q\| \leq D$ . Dwa punkty są pośrednio połączone jeśli istnieje między nimi ciąg punktów bezpośrednio połączonych. Innymi słowy od każdego punktu w grupie da się przejść do każdego innego punktu w grupie wykonując skoki od punku do punktu nie większe niż  $D$ .
2. W drugiej fazie usuwane są grupy, których rozmiar (w sensie liczby punktów) jest mniejszy od wartości progowej:  $|Grupa| < S$ . W założeniu, w ten sposób, powinny zostać usunięte pojedyncze błędne punkty i grupy punktów.

Koszt obliczeniowy metody wynosi  $O(N \cdot f_n(N))$ , gdzie:

- $N$  - liczba punktów obrazu 3D
- $f_n(N)$  - funkcja określająca koszt obliczeniowy znalezienia punktu najbliższego (w sensie odległości Euklidesa) od danego punktu. W przypadku, gdy nie stosuje się żadnych dodatkowych struktur przyspieszających wyszukiwanie:  $f_n(N) = O(N)$ .

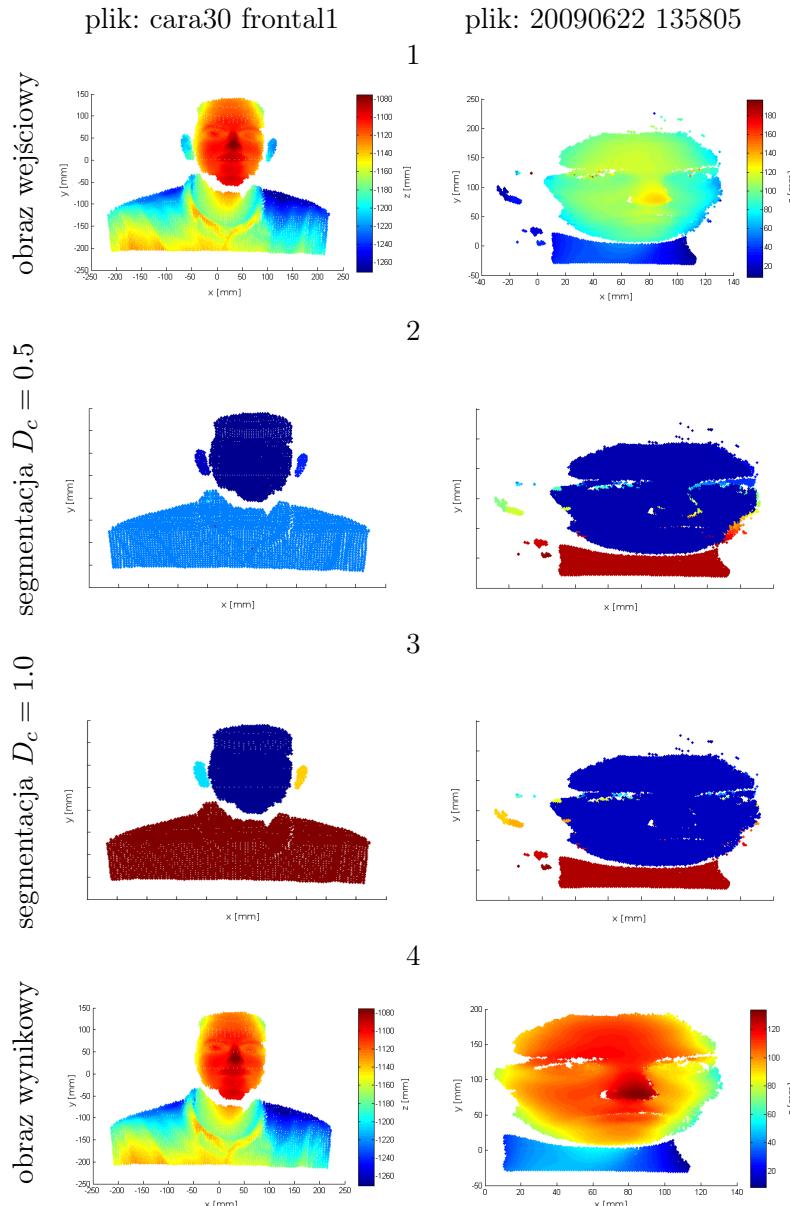
W wersji oryginalnej parametry  $D$  i  $S$  były wartościami stałymi. Takie podejście jednak uzależnia zachowanie się algorytmu od rozdzielcości analizowanego modelu. Wartości  $D$ , które poprawnie grupowały błędne piksele w osobnych segmentach dla obrazów z bazy GavabDB, okazywały się zbyt duże dla bazy Mechatroniki i powodowały "doklejanie" się błędnych segmentów do danych poprawnych. Z kolei, wartości dobrane dla drugiej z tych baz, były zbyt małe dla obrazów z pierwszej z nich i powodowały niepotrzebne "rozpadanie" się segmentów na mniejsze. W celu usunięcia powyższego problemu **zaproponowano uzależnienie parametru segmentacji od średniej odległości między sąsiednimi punktami w obrazie**.

W zaimplementowanej wersji algorytmu przed grupowaniem wykonywana jest triangulacja Delaunay'a. Wprawdzie jej złożoność obliczeniowa (dla 3 wymiarów) to  $O(3N^2)$ , jednak wykorzystano implementację bibliotecną metody, która w wykorzystywany środowisku, dla analizowanych rozmiarów obrazów, wykonuje się bardzo szybko. Co więcej może ona zostać wykorzystana wielokrotnie do wyszukiwania punktów sąsiednich danego punktu. Wykorzystując triangulację, zdefiniowano punkty sąsiednie jako punkty połączone wspólną krawędzią. Parametr segmentacji obliczany jest więc wg wzoru:  $D = D_c \cdot \text{srednia}_{e \in E} |e|$ , gdzie:

- $E$  - zbiór krawędzi między wierzchołkami (punktami) w triangulacji.
- $|e|$  - długość kolejnej krawędzi.
- $D_c$  - stała. Eksperymentalnie ustalono, że algorytm najlepiej zachowuje się dla wartości  $\in [0.5; 1]$ .

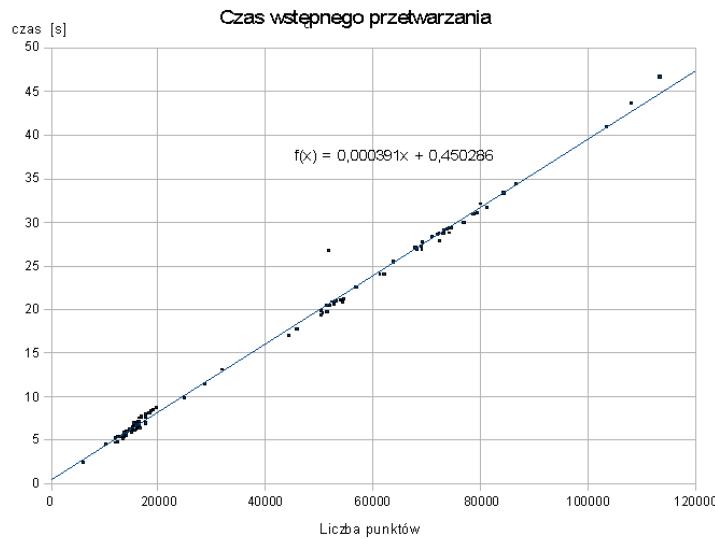
Wartość parametru  $S$  również dobierana jest w oparciu o charakterystykę danych wejściowych. Założono, że rozmiar pojedynczej grupy błędnych punktów nie stanowi więcej niż pewien ułamek rozmiaru całego obrazu tj.  $S = S_c \cdot |\text{Obraz}|$ . Arbitralnie przyjęto, że  $S_c = 10\%$ .

Zachowanie się algorytmu ilustruje rysunek 6.1. Jak widać algorytm bardzo dobrze radzi sobie z pojedynczymi błędnymi punktami i grupami punktów. Dane wyjściowe, zwłaszcza dla bazy Mechatroniki, wyglądają znacznie lepiej. Dodatkowo usuwane są zbędne części obrazów takie jak np. uszy. W celu oceny czasochłonności i efektów przetwarzania wykonano testy dla 50 modeli z bazy GavabDB i 50 obrazów z bazy Mechatroniki. Przyjęto wartość parametru  $D_c = 0.7$ . Wyniki ilustrują rysunki: 6.2, 6.3, 6.4. Jak widać czas przetwarzania rośnie bardzo wolno. Dla realistycznych rozmiarów danych zależność jest nieomal liniowa.

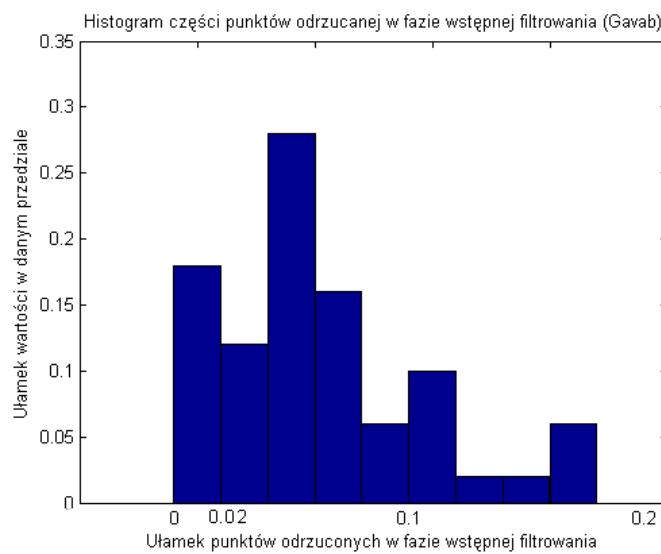


Na rysunkach 1,4 kolor oznacza trzeci wymiar. W rysunkach 2,3 różne kolory użyte zostały do rozróżnienia między segmentami.

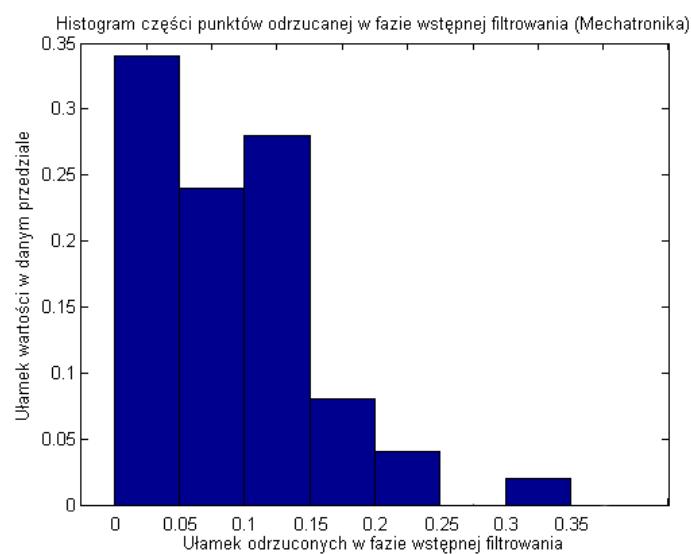
Rysunek 6.1: Zachowanie się filtrowania przez segmentację dla dwóch przykładowych plików z dwóch baz.



Rysunek 6.2: Wykres zależności czasu wstępnego filtrowania danych od liczby punktów obrazu (nawiesiona prosta regresji).



Rysunek 6.3: Histogram ukazujący jaki ułamek punktów jest odrzucany w obrazach GavabDB.



Rysunek 6.4: Histogram ukazujący jaki ułamek punktów jest odrzucany w obrazach Mechatroniki.



## Rozdział 7

---

# Wektory normalne

---

Kwestią podstawową w obliczaniu obrazów obrotu (ang. *spin images*) jest wyznaczenie kierunku wektora normalnego do powierzchni w danym punkcie  $p$ , dla którego budowany jest deskryptor. Problemem równoważnym do powyższego jest wyznaczenie płaszczyzny stycznej do powierzchni w punkcie. Mając ją wyznaczoną można w łatwy sposób wyznaczyć kierunek do niej prostopadły. Popularną metodą pozwalającą na estymację płaszczyzny stycznej jest PCA (ang. *Principal Component Analysis*).

W celu estymacji płaszczyzny stycznej (ozn.  $Tan_p$ ) do powierzchni obiektu danego jako chmura punktów, brane jest pewne otoczenie punktu  $p$ . W opisywanym rozwiązaniu przyjęto otoczenie sferyczne:  $O(p) = \{q : \|p - q\| < r_n\}$ . Im otoczenie jest mniejsze tym lepszym jego przybliżeniem jest płaszczyzna. Jej położenie wyznaczają dwa kierunki największego rozrzutu punktów, które obliczone mogą zostać właśnie z wykorzystaniem PCA. PCA jest metodą statystyczną która dla k-wymiarowych danych wyznacza nową ortogonalną bazę przestrzeni w ten sposób, że maksymalizowana jest wariancja w kierunku pierwszej, następnie drugiej i dalej, kolejnych współrzędnych.

Projektując procedurę estymującą położenie płaszczyzny stycznej odnotowano dwie kwestie ograniczające wybór wartości promienia otoczenia (sąsiedztwa)  $r_n$ :

1. zbyt mała liczba punktów w otoczeniu  $O$  powoduje bardzo niedokładne wyznaczenie kierunków największej wariancji danych. Poszerzenie otoczenia o każdy kolejny punkt powoduje znaczące ich zmiany - rozwiązanie jest niestabilne.
2. zbyt duże otoczenie powoduje, że zatrudniona jest "lokalność" tj. na kierunki rozrzutu wpływ zaczynają mieć punkty, które wpływu mieć nie powinny. Jaskrawym przykładem jest np. włączenie punktów z nosa czy uszu przy estymacji płaszczyzny stycznej do punktu położonego na policzku.

W rozwiązaniu problemu założono, że dla analizowanych danych **istnieje przedział wartości  $r_n$ , dla którego uzyskiwane przybliżenia płaszczyzny stycznej i kierunku wektora normalnego są stabilne i jednocześnie zachowują jej "lokalność"**. W tym przedziale niewielka zmiana wartości  $r_n$  nie wpływa znacząco na wyznaczony kierunek wektora normalnego  $\vec{n}_r$ .

## 7.1 Dobór promienia otoczenia

Aby dobrać wartość  $r_n$  zaprojektowano test: dla  $W$  losowo wybranych punktów obrazu 3D (przyjęto  $W = 1\% \cdot |\text{Obraz}|$ ) obliczono kierunki wektorów normalnych  $\vec{n}_r$ . Obliczeń dokonano przyjmując różne wartości promienia otoczenia:

$$r_n \in R = \{0.5, 1.0, 1.5, \dots, 9.5, 10\}$$

W ten sposób dla każdej z wartości przyjmowanych przez  $r_n$  uzyskano  $W$  wektorów. Łącznie określono ich  $|R| \cdot W$ .

Następnie, każdy z  $W$  wektorów  $\vec{n}_r$ , obliczonych dla promienia równego  $r_n$ , połączono w parę z wektorem  $\vec{n}'_r$  powstały dla tego samego punktu, ale wygenerowanym dla mniejszej wartości promienia (ozn.  $r'_n$ , gdzie  $r'_n < r_n$ ). Parę taką oznaczono przez  $(\vec{n}_r, \vec{n}'_r)$ . Dla ustalonych wartości  $r_n$  i  $r'_n$  daje to  $W$  par.

Należy ocenić czy kierunki wektorów uzyskiwane dla wartości promienia  $r_n$  i  $r'_n$  są ze sobą zgodne w sensie średnim. W tym celu zaproponowano kryterium:

$$\text{srednia}_{(\vec{n}_r, \vec{n}'_r)}(|1 - \vec{n}_r \cdot \vec{n}'_r|) < A$$

gdzie:

- $(\vec{n}_r, \vec{n}'_r)$  - para: wektor normalny obliczony dla promienia otoczenia  $r_n$  i odpowiadający mu (obliczony dla tego samego punktu) wektor policzony dla promienia  $r'_n$ . Obliczonych zostało  $W$  wektorów w obu grupach, więc analizowanych jest  $W$  par.
- $A$  - próg zgodności. Przyjęto  $A = 0.1$ .

Obliczany jest iloczyn skalarny między wektorami z każdej pary. Określa on liczbowo zgodność między kierunkami wektorów. Dla identycznych wektorów uzyskiwana jest wartość 1, dla skierowanych przeciwnie -1. Dla wektorów prostopadłych 0. Operacja  $|1 - \vec{n}_r \cdot \vec{n}'_r|$  odwraca i przesuwa zwracane wartości w przedział  $[0; 2]$ . Im wynik jest mniejszy tym kierunki są bardziej zgodne tj. dla 0 wektory są zgodne, dla 2 przeciwnie skierowane. Ostatnim krokiem jest uśrednienie, którego wynik daje informację o średnim poziomie zgodności uzyskiwanym dla całości danych. Aby stwierdzić, czy można uznać rezultaty obliczone dla promienia  $r_n$  i  $r'_n$  za zgodne wynik porównywany jest z wartością progową.

Obliczenia przeprowadzono dla każdej z wartości  $r_n \in R$  i odpowiadających im  $r'_n$ , gdzie  $r'_n \in R \wedge r'_n < r_n$ . Do każdej z testowanych wartości  $r_n$  przyporządkowano jedno  $r'_n$ . Dla wybranego  $r'_n$  spełnione jest kryterium zgodności. Dla wszystkich większych od niego wartości jest ono również spełnione. Jego zmniejszenie powoduje jednak, że przejście ono być spełnione. Wybrane  $r'_n$  mówi do jakiej wartości można zmniejszać dany promień  $r_n$  nie powodując znaczących zmian w uzyskiwanych kierunkach wektorów. Przebieg testu ilustruje poniższy pseudokod:

```
%Wybór punktów:  
W = 1% * |Obraz|
```

```

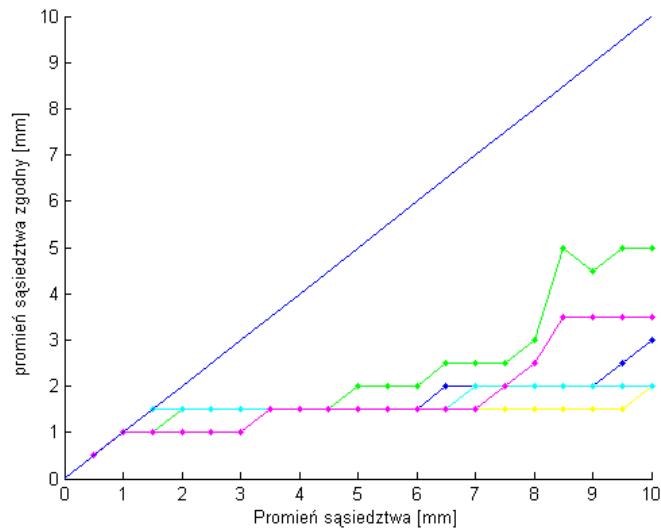
Punkty = WylosujPunkty(Obraz, W)
R = {0.5, 1.0, ..., 10.0}

%Obliczenie wektorów normalnych:
DlaKazdego Promienia r_n z R
    n(r_n) = ObliczWektorNormalne(Punkty, r_n)
    % n(r_n) - zbiór wektorów normalnych obliczonych dla W punktów
    %           przy promieniu otoczenia ustalonym na r_n

%Przyporządkowanie r_n' do r_n:
DlaKazdego Promienia r_n z R
    Zmniejszaj r_n' z R, począwszy od r_n' = r_n-0.5 do r_n' = 0.5
    Jeśli Kryterium( n(r_n), n(r_n') ) nie jest spełnione to
        przerwij zmniejszanie
    przyporządkuj do r_n wartość r_n'+0.5

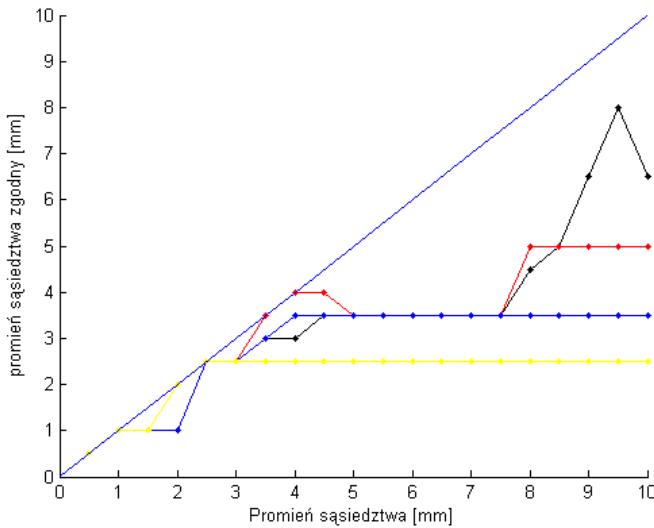
```

Test powtórzony został dla 5 obrazów z bazy GavabDB i 5 obrazów z bazy Mechatroniki. Wyniki zilustrowano na poniższych wykresach (rysunki 7.1 i 7.2). Na osi poziomej zaznaczono wartości promienia sąsiedztwa  $r_n$ . Na osi pionowej zaznaczono wybraną wartość promienia  $r'_n$ , dla którego kierunki wektorów normalnych są zgodne z tymi uzyskiwanymi dla promienia  $r_n$ . Linia 45 stopni wyznacza poziom maksymalny dla danej wartości  $r_n$  - tj. poziom który oznacza, że drobne zmniejszenie wartości promienia powoduje niezgodność uzyskiwanych wyników.



Rysunek 7.1: Ilustracja poziomów zgodności promienia sąsiedztwa dla Mechatroniki.

Wykresy potwierdzają słuszność przyjętych założeń. Zarówno dla obrazów z bazy Gavab jak i Mechatroniki dla pewnych przedziałów  $r_n$  występują obszary płaskie. Oznacza to, że do przedziału wartości  $r_n$  przyporządkowywana jest ta sama wartość  $r'_n$ . Kierunki wektorów uzyskiwanych dla przedziału promieni  $r_n$  są zgodne z kierunkami dla promienia  $r'_n$ . Na przedziałach  $r_n$  dla których wykresy są płaskie wyniki obliczeń są



Rysunek 7.2: Ilustracja poziomów zgodności promienia sąsiedztwa dla Gavab.

stabilne. Wybierając przedziały położone najbardziej na lewo, wybiera się najmniejsze wartości promienia, dzięki czemu zapewnia się "lokalność". Dla pierwszej z wymienionych baz najbardziej na lewo położone obszary płaskie występują dla  $r'_n \in [1\text{mm}, 1.5\text{mm}]$ . Dla drugiej z baz  $r'_n \in [2.5\text{mm}, 3.5\text{mm}]$ . Odnosząc się do rysunków 3.14 i 3.15 widać, że uzyskane wartości promienia w obu przypadkach odpowiadają podobnej liczbie, około  $E_p = 20$  punktów zawartych w sferze otaczającej dany punkt  $p$ .

Rozbieżność wartości promienia wynika z różnej rozdzielcości obrazów w obu bazach. W bazie Gavab dla tego samego promienia otoczenia uzyskuje się znacznie mniej punktów. Rozważania takie prowadzą do wniosku, że krokiem poprawiającym zachowanie algorytmu estymującego byłoby zdefiniowanie otoczenia punktu jako zbioru  $E_p$  najbliższych punktów - sąsiadów. Rozwiążanie takie jednak niesie ryzyko złamania ograniczenia nr 2 tj. zatraceniu "lokalności". Przykładowo dla punktu znajdującego się na brzegu obiektu jego sąsiedzi w odpowiedniej liczbie mogą być położeni w dużym oddaleniu. Tego typu problemy spowodowały odrzucenie powyższej koncepcji. W celu uzależnienia wartości promienia sąsiedztwa (otoczenia) od gęstości punktów w obrazie zaproponowano metodę estymacji gęstości powierzchniowej punktów.

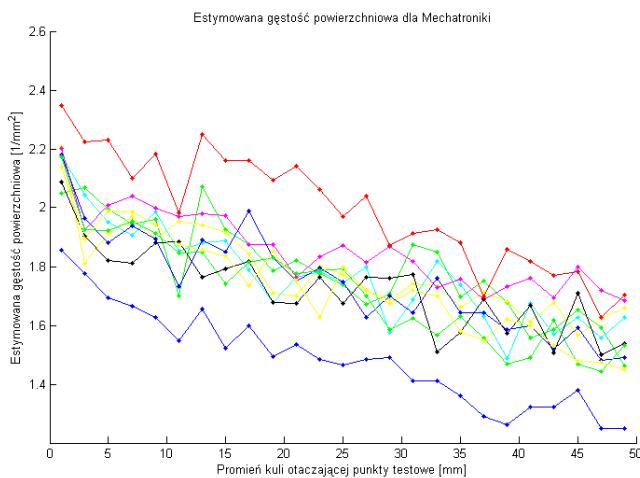
## 7.2 Estymacja gęstości powierzchniowej punktów

W celu oceny gęstości powierzchniowej punktów  $\sigma$  (liczby przypadających na jednostkę powierzchni) w obrazach zastosowano następujący algorytm:

1. losowo wybierz podzbiór  $P$  punktów obrazu ( $P = 100$ ).
2. dla każdego z wybranych punktów  $p \in P$ , policz liczbę punktów sąsiednich  $q$  w sferycznym otoczeniu  $p$ :  $k_p = |\{q : \|p - q\| < r_e\}|$ .
3. oblicz estymację:  $\sigma = \frac{srednia_p(k_p)}{\pi \cdot r_e^2}$ , ( $r_e = 2\text{mm}$ )

Tak obliczoną wartość  $\sigma$  wykorzystano do obliczania wartości promienia sąsiedztwa używanego w PCA:  $r_n = \sqrt{\frac{E_p}{\sigma \cdot \pi}}$ . Opracowany algorytm obliczania gęstości powierzchniowej przetestowano na 10 obrazach z bazy GavabDB i 10 obrazach z bazy Mechatroniki. Przeanalizowano zależność uzyskiwanego wyniku od wartości  $r_e$ . Wyniki pokazują rysunki 7.3, 7.4. Potwierdzona została uwaga o większej gęstości punktów w modelach w drugiej z baz. Dodatkowo na wykresach widoczne są dwa problemy jakie rozważyć należy dobierając parametry dla algorytmu:

- wraz ze wzrostem  $r_e$  wartość  $\sigma$  spada.
- dla zbyt małych wartości  $r_e$  uzyskiwany wynik jest niestabilny.



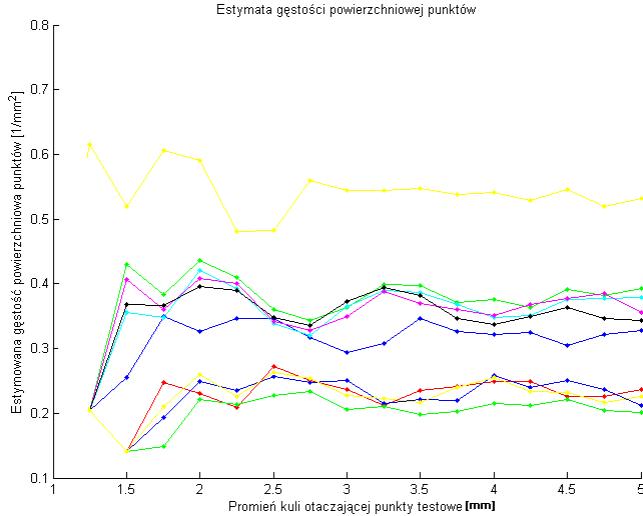
Rysunek 7.3: Zależność estymowanej gęstości powierzchniowej punktów  $\sigma$  od promienia sąsiedztwa  $r_e$  dla 10 obrazów Mechatroniki.

Wykonano również, test obliczenia wartości  $r_n$  dla 50 obrazów z każdej z baz. Rezultaty ilustrują rysunki: 7.5, 7.6. Algorytm daje wyniki bliskie oczekiwaniom. Promienie uzyskiwane dla Mechatroniki są średnio 2 razy większe niż dla GavabDB. Dodatkowo widać, że obrazy z pierwszej z baz cechują się znacznie mniejszą wariancją uzyskiwanych rezultatów.

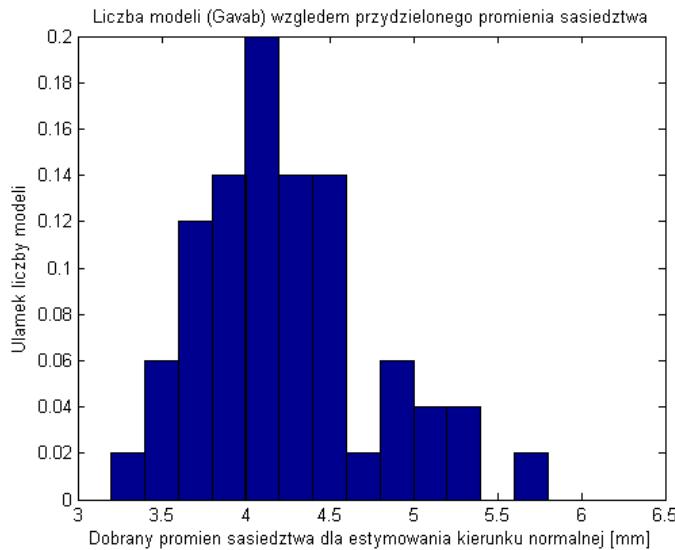
### 7.3 Korekta zwrotów wektorów normalnych

Stosując metodę PCA można wyznaczyć kierunki wektorów normalnych do powierzchni. Nie jest jednak możliwe ustalenie zwrotów: do wewnętrz/na zewnątrz. Jeden z najprostszych znanych sposobów ich unifikacji polega na tym, że zwrot wektora  $\vec{n}_p$  jest korygowany względem kierunku od punktu ciężkości chmury punktów do punktu  $p$  w którym obliczany był wektor  $\vec{n}_p$ . Algorytm jest następujący:

- oblicz wektor  $\vec{k}$  skierowany od punktu ciężkości obrazu do punktu  $p$  dla którego obliczany jest wektor normalny:  $\vec{k} = p - centroid(Obraz)$ .
- oblicz znak iloczynu skalarnego między  $\vec{k}$  i  $\vec{n}_p$ :  $znak = sign(\vec{k} \cdot \vec{n}_p)$ .



Rysunek 7.4: Zależność estymowanej gęstości powierzchniowej punktów  $\sigma$  od promienia sąsiedztwa  $r_e$  dla 10 obrazów Gavab.



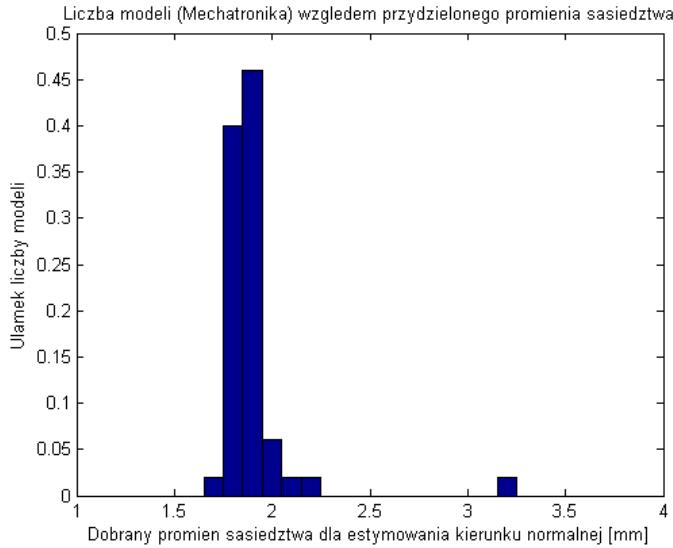
Rysunek 7.5: Histogram obliczanego promienia otoczenia  $r_n$  dla 50 obrazów z GavabDB.

- jeśli  $znak = -1$  to odwrócić wektor normalny:  $\vec{n}_p = -\vec{n}_p$

## 7.4 Schemat algorytmu estymacji wektorów normalnych

Ostateczna wersja algorytmu obliczającego wektory normalne składa się z 4 kroków (rysunek 7.7). Na wejściu podawana jest lista  $M$  punktów dla których budowane będą budowane deskryptory. W wyniku działania metody obliczanych jest odpowiadające im  $M$  wektorów. Złożoność obliczeniową poszczególnych kroków przedstawia tabela 7.1.

Zachowanie się algorytmu przedstawiono na rysunku 7.8. Dla losowego podzbioru punktów obliczono kierunki wektorów normalnych. Ich wartości wykorzystano potem do



Rysunek 7.6: Histogram obliczanego promienia otoczenia  $r_n$  dla 50 obrazów z Mechatroniki.

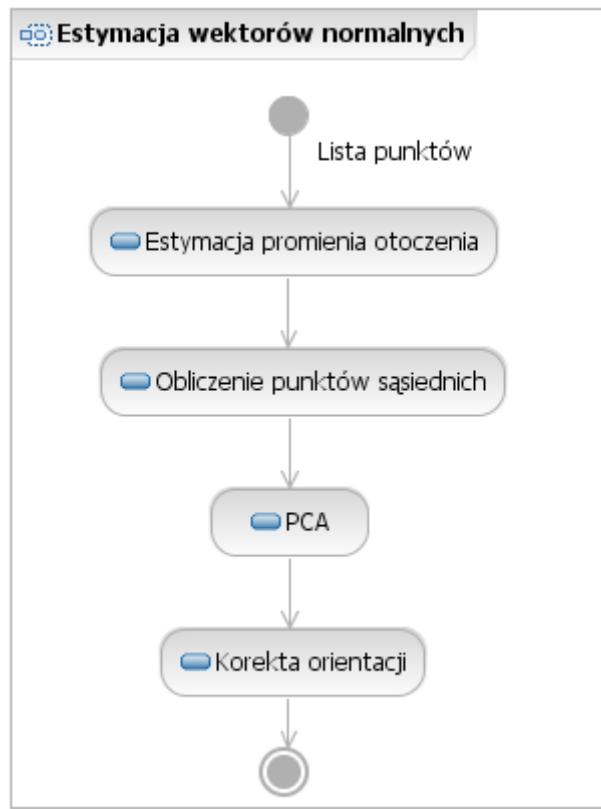
Nr	Nazwa kroku	Koszt
1	Estymacja promienia otoczenia	$O(P \cdot f_n(N))$
2	Obliczenie punktów sąsiednich	$O(M \cdot f_n(N))$
3	PCA (średnio na wektor): - odjęcie średniej - obliczenie kowariancji - wektory własne (QR+Hausholder) - posortowanie	$O(E_p)$ $O(E_p)$ $\frac{4}{3}E_p^3 + O(E_p^2)$ $O(1)$
4	Korekta orientacji	$O(M)$
*	Sumarycznie	$O(ME_p^3 + (P + M) \cdot f_n(N))$

Tablica 7.1: Złożoność obliczeniowa poszczególnych kroków obliczania wektorów normalnych.

obliczenia świecił w wizualizowanych danych. Jak widać zastosowana korekta poprawia działanie algorytmu. Większość wektorów w danych wynikowych jest skierowanych na zewnątrz obiektu - tak jak było to założone. Wektory skierowane odwrotnie pojawiają się w okolicach ust i oczu. Zakłada się, że nie wpływają one znacząco na dalsze analizy.

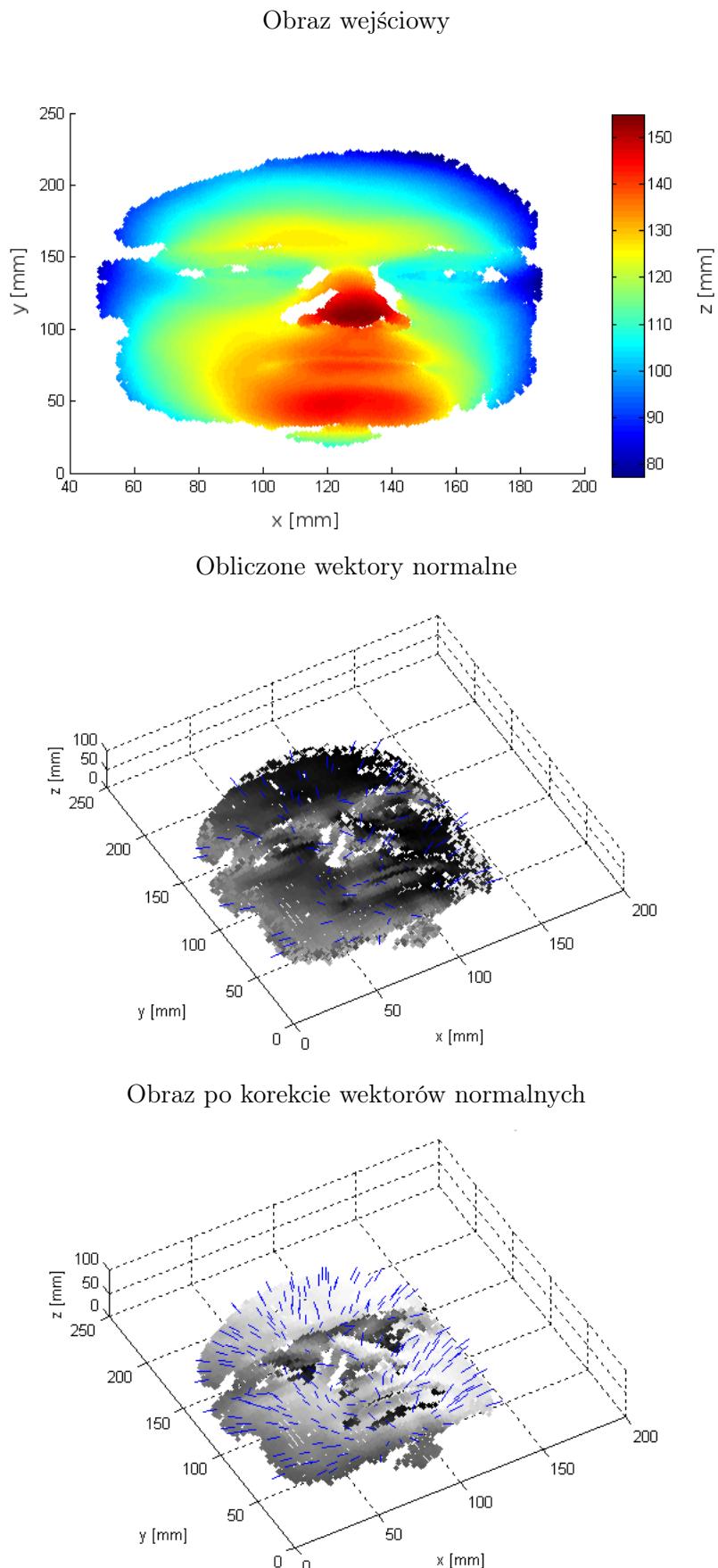
## 7.5 Wygładzanie danych

W [18] zaproponowany został algorytm wygładzania danych. W pierwszym kroku dla każdego punktu  $p$  wykonywana jest procedura estymacji płaszczyzny  $Tan_p$  stycznej do powierzchni obrazu 3D. W kolejnym kroku każdy z punktów  $p$  rzutowany jest na odpowiadającą mu płaszczyznę  $Tan_p$  - obliczony zostaje punkt  $p'$ . W ostatnim kroku  $p$  zostaje zastąpiony przez  $p'$ . Problemem opisanej metody wygładzania jest wysoka złożoność obliczeniowa równa średnio:  $O(NE_p^3)$  ( $N$  - liczba punktów obrazu,  $E_p$  - średnia liczba punktów w otoczeniu). Z tego powodu wygładzanie zostało włączone do osta-



Rysunek 7.7: Schemat algorytmu estymacji kierunków wektorów normalnych.

tecznego rozwiązania jedynie jako opcja.



Rysunek 7.8: Przykład działania algorytmu obliczającego kierunki wektorów normalnych.



## Rozdział 8

---

# Wybór punktów

---

### 8.1 Metody wyboru punktów

Przy zastosowaniu deskryptorów do dopasowywania obiektu (podzbioru punktów w obrazie 3D) do wzorca, kwestią podstawową jest wybór punktów dla których zostaną one obliczone. Na potrzeby systemu zaadaptowano podejście nr 3 z opisanych w sekcji 4.3.6 polegające na losowym wyborze punktów. Dalej zaprezentowano trzy jego wariacje.

#### 8.1.1 Wybór losowy

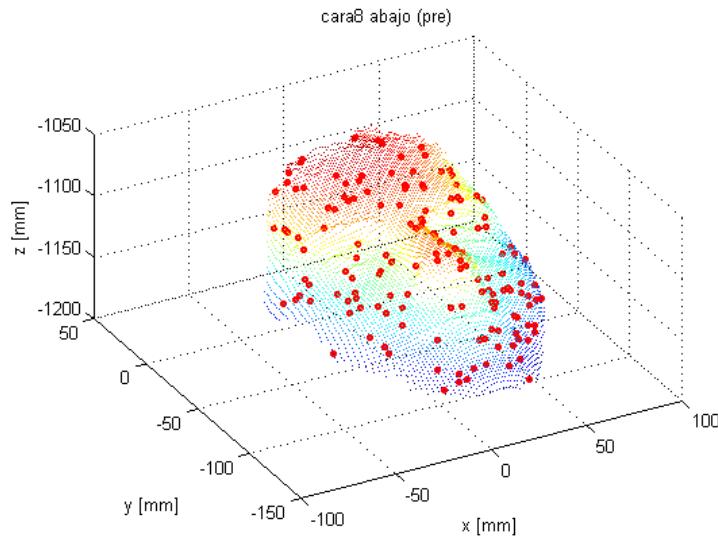
W podejściu tym generowanych jest  $M$  liczb o rozkładzie jednostajnym na przedziale  $[1, N]$ , gdzie  $N$  - liczba punktów obrazu (rozmiar chmury). Zbiór  $M$  liczb jest następnie traktowany jako indeksy punktów, które zostają wybrane do budowy deskryptorów. Złożoność metody wynosi  $O(M)$ . Przykładowy wynik jej działania pokazuje rysunek 8.1. Deskryptory zostały rozmieszczone losowo po powierzchni obrazu, jednak nie rozkładają się one równomiernie. Występują obszary o większym, jak i o mniejszym zagęszczeniu wybranych punktów. **W celu poprawy rozkładu zaproponowano zastosowanie wstępnego partycjonowania punktów.**

#### 8.1.2 Wybór punktów w oparciu o k-średnich

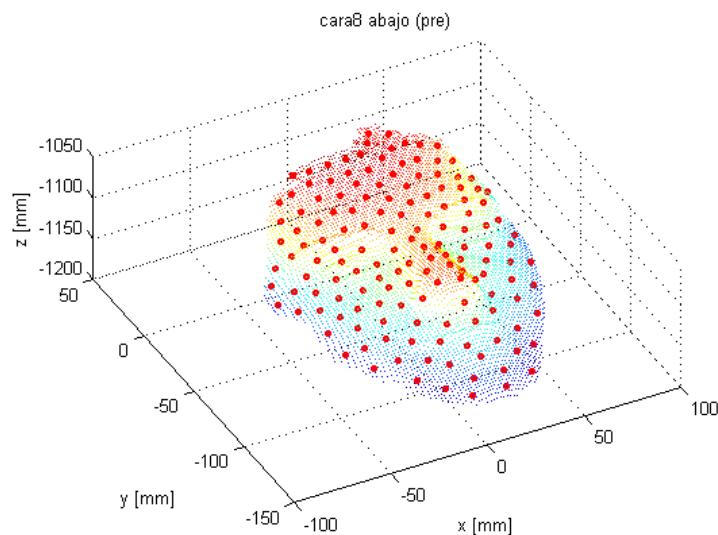
Pierwszą sprawdzoną metodą partycjonowania danych był algorytm k-średnich. Analizowany obraz 3D dzielony jest na  $M$  grup. Następnie dla każdej z grupy znajdowany jest punkt najbliższy centroidowi grupy. Przykład działania algorytmu przedstawia rysunek 8.2. Rozkład punktów jest bardzo dobry. Problemem jest jednak złożoność czasowa metody. Koszt wykonania algorytmu k-średnich dla  $M$  grup i  $N$  punktów w 3 wymiarach wynosi  $O(N^{3M})$ .

#### 8.1.3 Wybór punktów w oparciu o *octClustering*

Aby z jednej strony zredukować koszt obliczeniowy, a jednocześnie uzyskać dobry rozkład wybieranych punktów, **zaproponowano algorytm partycjonowania, oparty o koncepcję drzew ósemkowych**. Metodę nazwano *octClustering*. Schemat algorytmu



Rysunek 8.1: Przykład losowego wyboru 150 punktów do generacji deskryptorów.



Rysunek 8.2: Przykład wyboru 150 punktów metodą opartą o K-średnich.

przedstawiono na rysunku 8.3. W pierwszej fazie obszar zawierający punkty dzielony jest kolejno na coraz mniejsze, rozłączne, prostopadłościanny - węzły - o brzegach wzdłuż kierunków osi współrzędnych. Położenie każdego z nich opisane jest zestawem sześciu wartości:  $x_{min}$ ,  $x_{max}$ ,  $y_{min}$ ,  $y_{max}$ ,  $z_{min}$ ,  $z_{max}$ . Wymiary pierwszego węzła ustalane są w oparciu o położenia ekstremalne punktów  $q$  należących do obrazu:

$$x_{min} = \min_{q \in Obraz} q_x$$

$$y_{min} = \min_{q \in Obraz} q_y$$

$$z_{min} = \min_{q \in Obraz} q_z$$

$$\begin{aligned}y_{max} &= \epsilon + \max_{q \in \text{Obraz}} q_y \\x_{max} &= \epsilon + \max_{q \in \text{Obraz}} q_x \\z_{max} &= \epsilon + \max_{q \in \text{Obraz}} q_z\end{aligned}$$

gdzie:  $\epsilon$  - mała, dodatnia liczba.

W kolejnych iteracjach do podziałów wybierane są węzły obejmujące największą liczbę punktów. Punkt  $q$  obrazu uznaje się za obejmowany przez węzeł *Wezel* gdy spełnione są następujące warunki:

$$\begin{aligned}q_x &\geq Wezel.x_{min} \wedge q_x < Wezel.x_{max} \\q_y &\geq Wezel.y_{min} \wedge q_y < Wezel.y_{max} \\q_z &\geq Wezel.z_{min} \wedge q_z < Wezel.z_{max}\end{aligned}$$

W stosunku do zwykłych drzew ósemkowych różnicą jest to, że w trakcie wykonywania algorytmu pamiętana jest wyłącznie informacja o liściach. W wyniku działania pierwszej fazy otrzymywanych jest  $K$  węzłów:  $K \in [M, M + 7]$ .

Celem drugiej fazy jest zredukowanie tej liczby do  $M$ . Tak długo jak liczba węzłów jest większa od  $M$ :

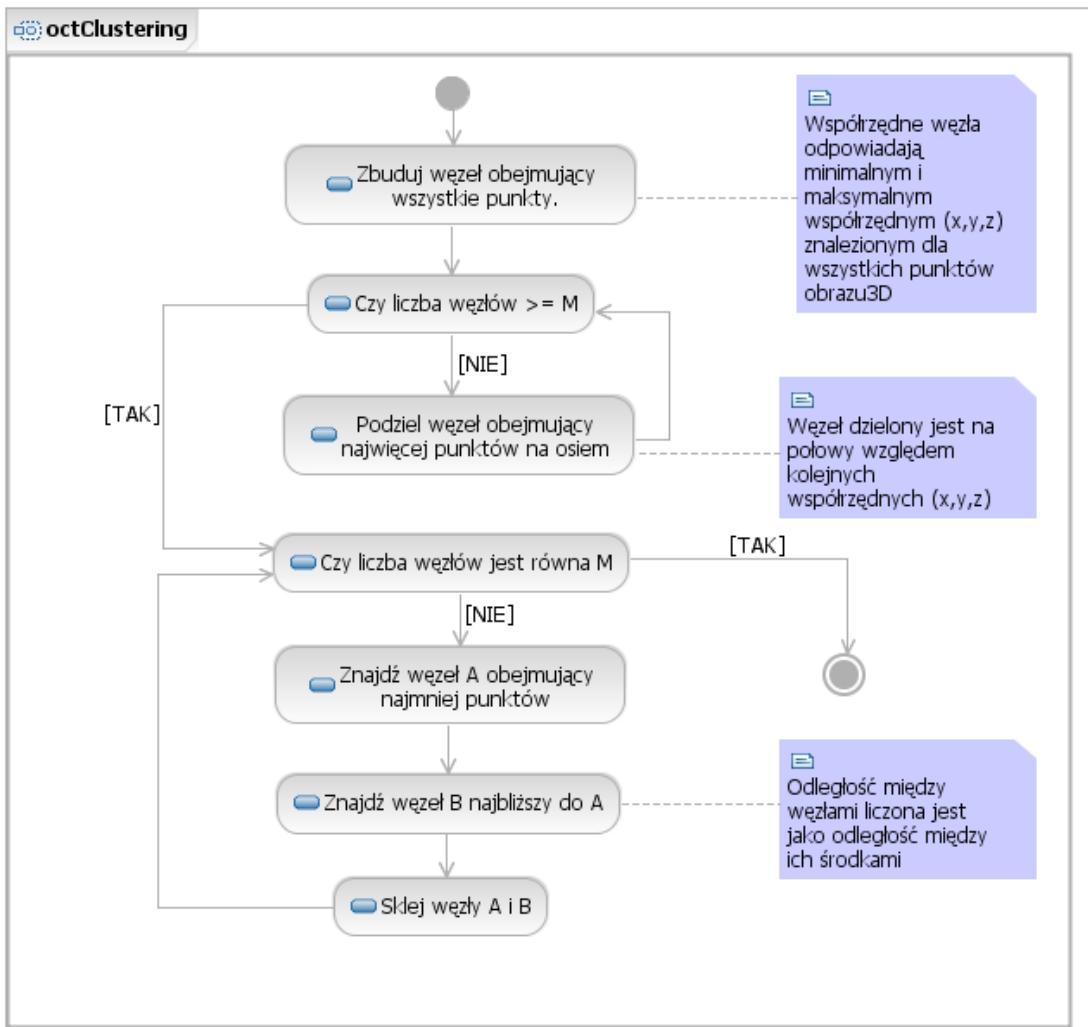
1. Znajdź węzeł  $Wezel_1$  obejmujący najmniejszą liczbę punktów.
2. Znajdź węzeł  $Wezel_2$  najbliższy do powyższego. Odległość liczona jest pomiędzy środkami prostopadłościanów.
3. Sklej dwa węzły. Nowe współrzędne określone są następująco:

$$\begin{aligned}x'_{min} &= \min\{Wezel_1.x_{min}, Wezel_2.x_{min}\} \\y'_{min} &= \min\{Wezel_1.y_{min}, Wezel_2.y_{min}\} \\z'_{min} &= \min\{Wezel_1.z_{min}, Wezel_2.z_{min}\} \\x'_{max} &= \max\{Wezel_1.x_{max}, Wezel_2.x_{max}\} \\y'_{max} &= \max\{Wezel_1.y_{max}, Wezel_2.y_{max}\} \\z'_{max} &= \max\{Wezel_1.z_{max}, Wezel_2.z_{max}\}\end{aligned}$$

W wyniku działania tej fazy dla części węzłów (maksymalnie siedmiu) złamane może zostać założenie o rozłączności. Nie wpływa to jednak w istotny sposób na dalsze zastosowanie algorytmu.

Koszt obliczeniowy pierwszej fazy wynosi  $O(M \cdot g_d(M, N))$ , gdzie  $g_d(M, N)$  - oznacza koszt kolejnych podziałów węzłów. Można ograniczyć:  $g_d(M, N) = O(N)$ . Kosz drugiej fazy również można ograniczyć przez  $O(7N)$ .

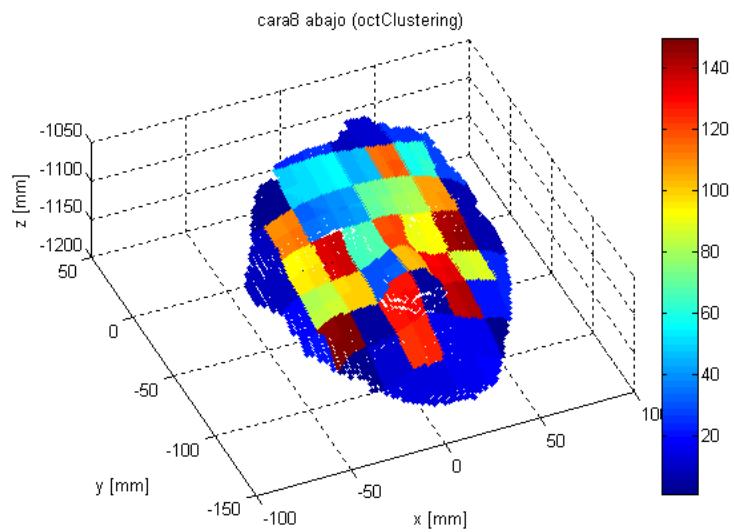
Przykład działania algorytmu partycjonowania przedstawia rysunek 8.4. Jak widać dane dzielone są na regularne obszary, które z dużą dokładnością obejmują podobną ilość punktów. Z tak wygenerowanych partycji następuje wybór  $M$  punktów. Z każdej partycji wybierany jest jeden położony najbliżej środka węzła. Przykładowy wynik działania pełnej procedury generacji punktów zilustrowano na rysunku 8.5.



Rysunek 8.3: Schemat działania algorytmu octClustering.

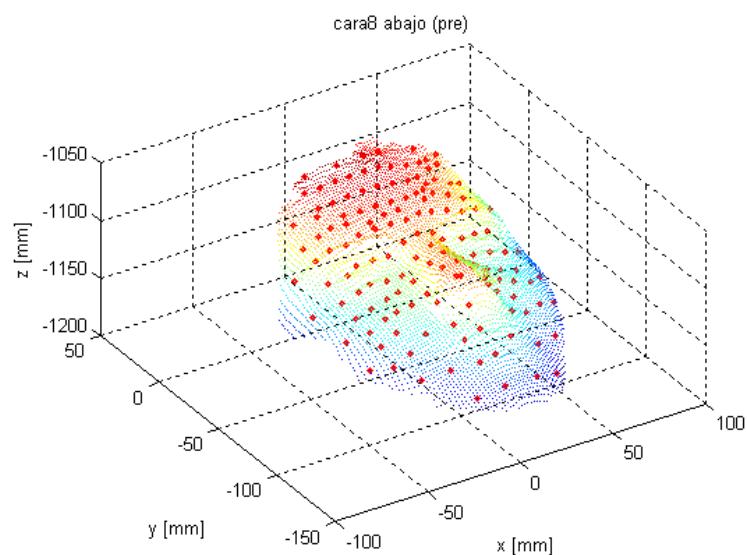
#### 8.1.4 Wybór metody

Każda z trzech metod charakteryzuje się różnymi właściwościami. Wybór w pełni losowy jest najszybszy, ale uzyskiwany rozkład jest najgorszy. Lepiej, ale też trochę wolniej zachowuje się *octClustering*. Obie te metody należy rozważyć w implementacji algorytmów analizy. Koszt obliczeniowy metody opartej o k-średnich jest bardzo wysoki. Powoduje to, że metoda ta nie nadaje się do pracy interaktywnej. Możliwe jest jednak wykorzystanie jej do przygotowania wzorców.



Kolory odpowiadają przyporządkowaniu do partycji.

Rysunek 8.4: Przykład podziału obrazu na 150 partycji algorytmem octClustering.



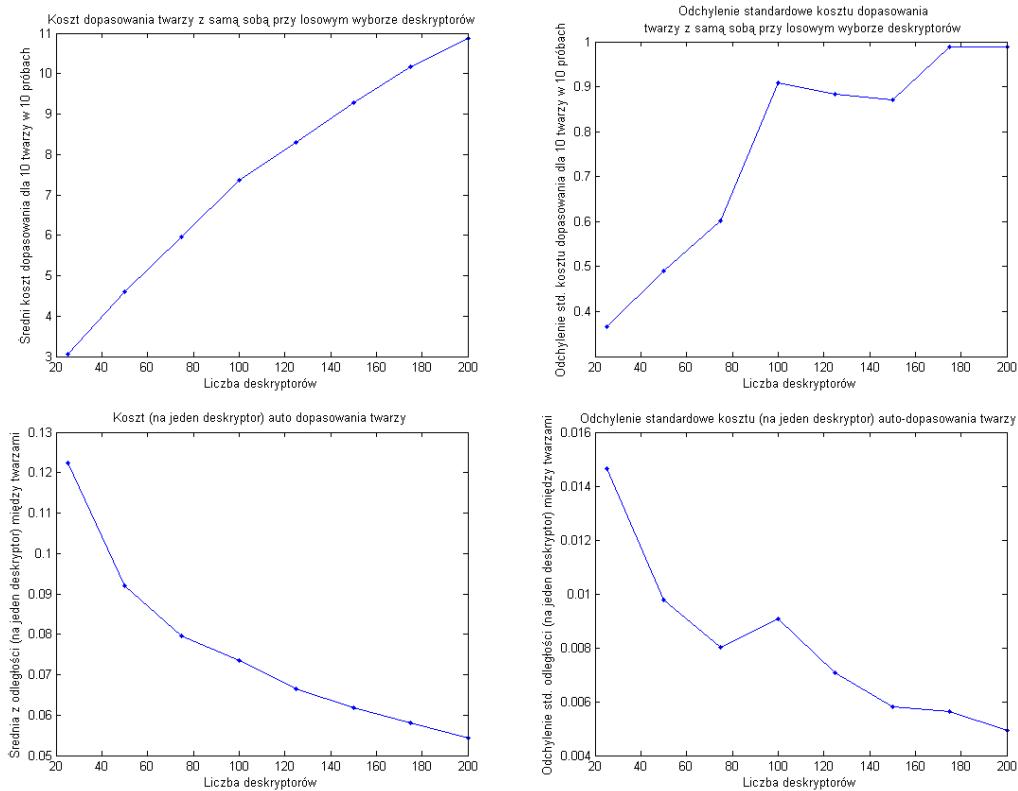
Rysunek 8.5: Przykład wyboru 150 punktów algorytmem octClustering.

## 8.2 Wpływ metody wyboru punktów na jakość dopasowywania

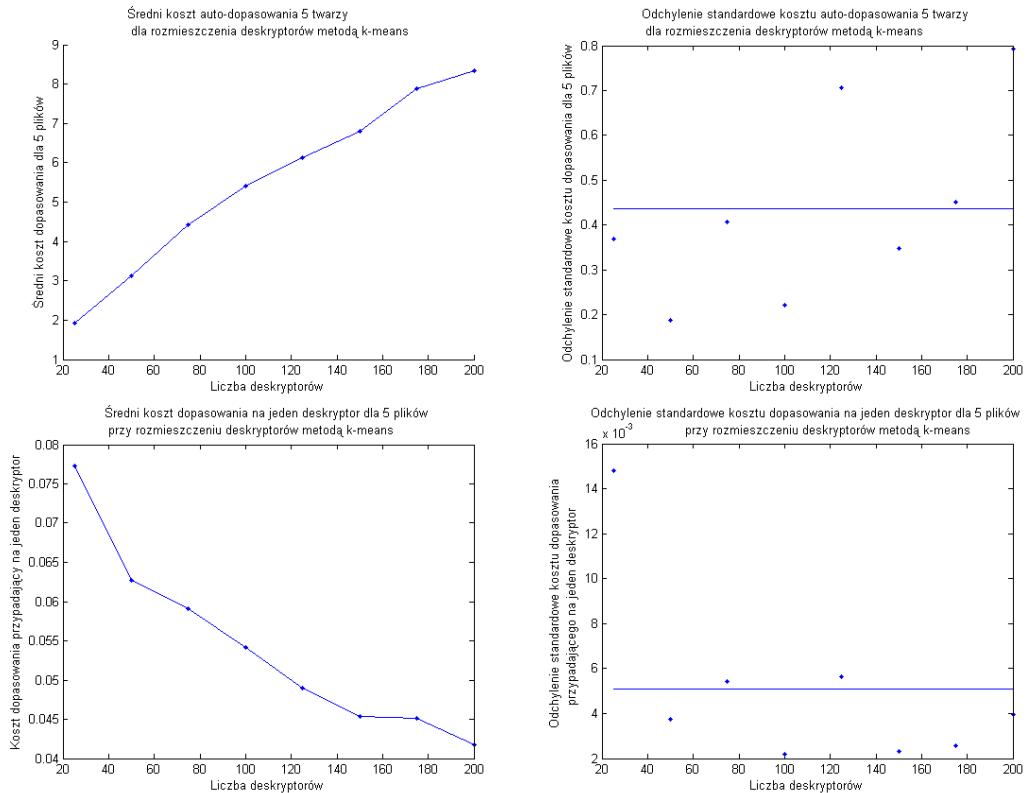
Celem sprawdzenia wpływu metod wyboru punktów, dla których budowane są deskryptory, na jakość dopasowywania między obrazami zaproponowano szereg testów. Wszystkie one przeprowadzone zostały dla deskryptorów o skalach logarytmicznych i rozdzielczości  $10 \times 10$ . Promień otoczenia ustalono na  $r = 100$ . Zastosowano filtrowanie metodą OR. Do obliczania odległości między dwoma deskryptorami użyto *shape context measure*.

### 8.2.1 Auto-dopasowanie

Dla  $L$  wcześniej przygotowanych, zawierających wyłącznie twarz, obrazów 3D,  $T$ -krotnie policzono sumaryczny koszt dopasowania obrazu do niego samego. Obliczenia powtórzono dla różnych wartości  $M$  - liczby punktów dla których budowane są deskryptory. Wyniki uśredniono po wszystkich  $L$  obrazach i  $T$  próbach. Policzono też średni koszt dopasowania przypadający na jeden deskryptor. Wyniki obliczeń dla losowego wyboru punktów ( $L = 10$ ,  $T = 10$ ) przedstawia rysunek 8.6. Analogiczne wykresy dla wyboru punktów metodą opartą o k-średnich ( $L = 5$ ,  $T = 5$ ) przedstawiono na rysunku 8.7. Podobne testy dla rozmieszczenia deskryptorów metodą *octClustering*-u są bezcelowe gdyż metoda ta jest w pełni deterministyczna. Zwarcane wybory są zawsze identyczne dla tych samych argumentów. Powoduje to, że koszt dopasowania jest równy 0.



Rysunek 8.6: Koszty dopasowania obrazów twarzy do samych siebie przy losowym wyborze deskryptorów.



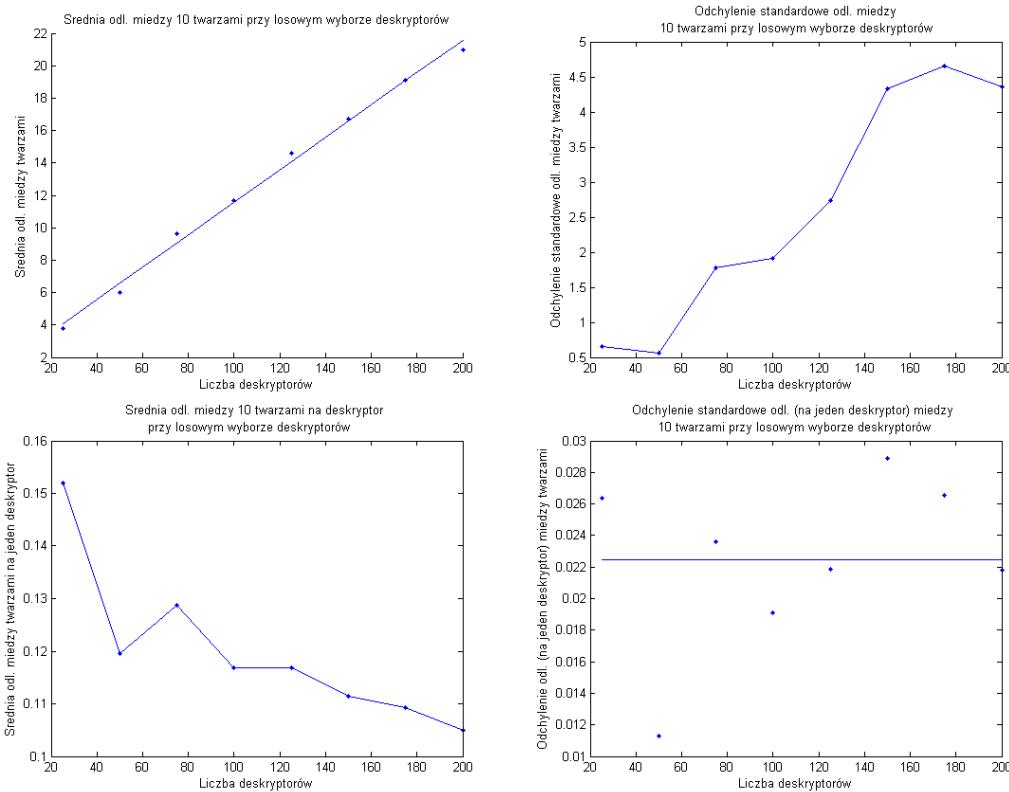
Rysunek 8.7: Koszty dopasowania obrazów twarzy do samych siebie przy wyborze deskryptorów metodą opartą o k-średnich.

Zgodnie z przewidywaniami sumaryczny koszt dopasowania zmniejsza się gdy deskryptory rozmieszczone są bardziej równomiernie (k-średnich). W obu przypadkach wraz ze wzrostem liczby deskryptorów średni koszt przypadający na pojedynczy deskryptor asymptotycznie zbiega do zera. Oznacza to, że deskryptory są coraz lepiej dopasowywane. Zmniejsza się wpływ rozrzutu wybieranych punktów. Największe nachylenie wykresu odpowiada liczbie 100. Powyżej 150 zmiany kosztu wraz ze wzrostem ich liczby są bardzo niewielkie. Co więcej różnica między wartościami uzyskiwanymi dla metody k-średnich i dla wyboru losowego maleje wraz ze wzrostem liczby deskryptorów. Pokazuje to zmniejszający się wpływ metody wyboru na jakość dopasowywania.

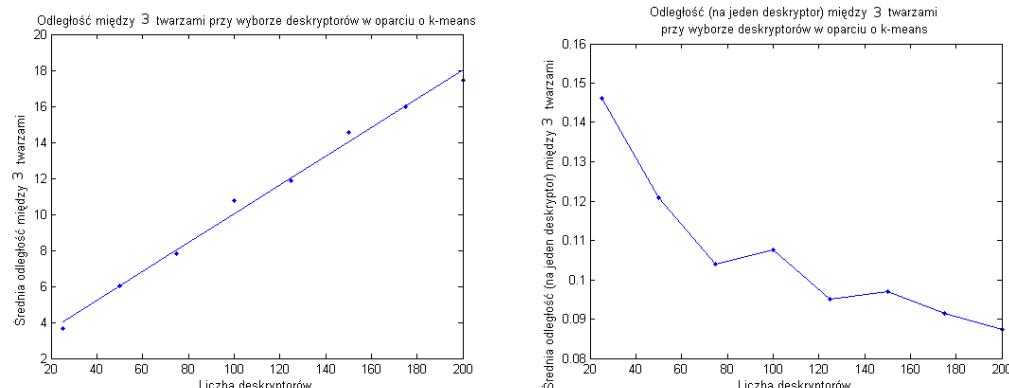
### 8.2.2 Dopasowanie twarzy

Przeprowadzono test służący ocenie sumarycznego kosztu dopasowania deskryptorów obliczonych dla różnych twarzy. Obliczono koszt dopasowania między  $L$  wcześniej przygotowanymi, zawierającymi wyłącznie twarz, obrazami 3D. Dla  $L$  zestawów deskryptorów dokonano  $T = \frac{L(L-1)}{2}$  dopasowań (każdy z każdym). Wyniki uśredniono. Policzono też średni koszt przypadający na jeden deskryptor. Wyniki obliczeń dla losowego wyboru punktów ( $L = 10$ ,  $T = 45$ ) przedstawia rysunek 8.8. Analogiczne wykresy dla wyboru punktów metodą opartą o k-średnich ( $L = 3$ ,  $T = 3$ ) przedstawiono na rysunku 8.9, a dla *octClustering-u* ( $L = 10$ ,  $T = 45$ ) na rysunku 8.10.

Wykresy, niezależnie od metody, wskazują na zależność liniową między liczbą deskryptorów a kosztem dopasowania. Różnice w nachyleniu wykresów są praktycznie



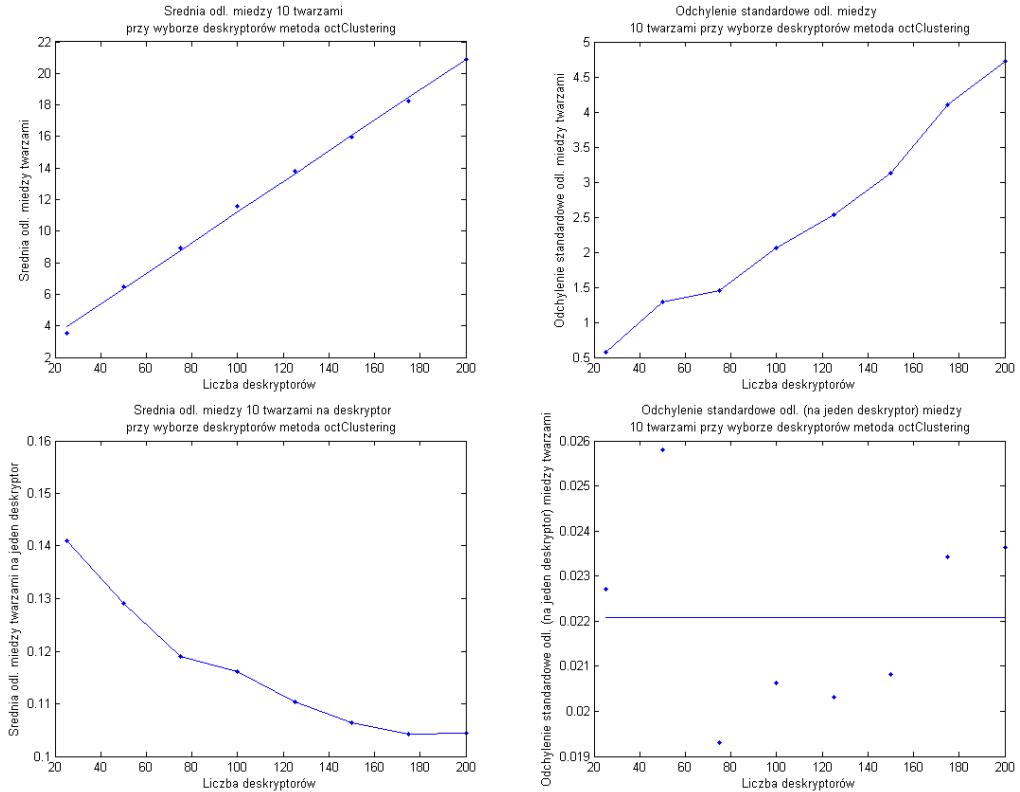
Rysunek 8.8: Koszty dopasowania obrazów twarzy przy losowym wyborze deksyryptorów.



Rysunek 8.9: Koszty dopasowania obrazów twarzy przy wyborze deskryptorów metodą opartą o k-średnich.

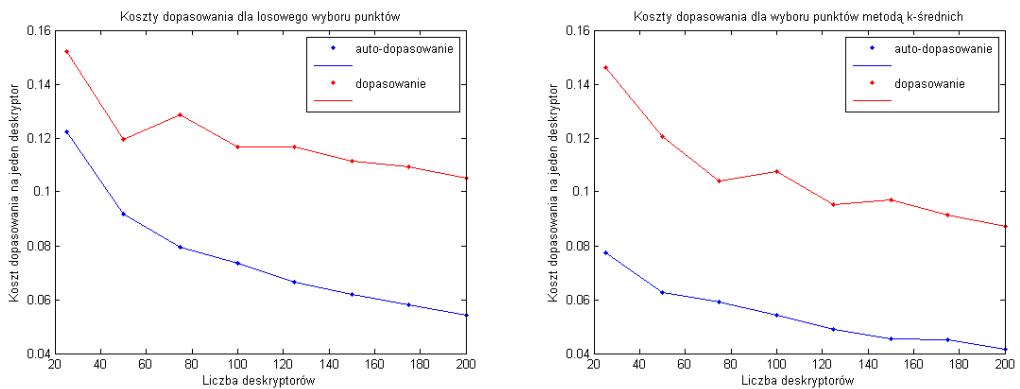
niezauważalne, co mówi o niewielkim wpływie doboru metody na jakość dopasowania. Również koszty przypadające na pojedynczy deskryptor są praktycznie identyczne. Bardzo ważną cechą wszystkich wykresów jest asymptotyczna zbieżność do zera. Wskazuje to na **wysoki stopień podobieństwa deskryptorów między różnymi twarzami, co potwierdza słuszność zastosowania podejścia opartego o wzorce (ang. *instance-based approach*)**.

Aby ułatwić ocenę wpływu zróżnicowania między różnymi twarzami na jakość dopasowywania naniesiono wykresy dotyczące samodopasowania twarzy (rysunki 8.6, 8.7) na



Rysunek 8.10: Koszty dopasowania obrazów twarzy przy wyborze deskryptorów metodą opartą o `octClustering`.

wykresy dopasowania między różnymi twarzami (rysunki 8.8, 8.9). Wynikowe wykresy zaprezentowano na rysunku: 8.11. Porównanie ich wskazuje na to, że **średni koszt (na jeden deskryptor) wnoszony przez zróżnicowanie w charakterystykach różnych twarzy jest stały** i dla powyższych parametrów wynosi około 0.06. Rezultat ten wskazuje dolne ograniczenie kosztu przy dopasowywaniu różnych twarzy. Dla losowego wyboru punktów próg ten uzyskiwany jest dla około 400-stu deskryptorów. Dla wyboru metodą opartą o k-srednich dla około 330-stu.



Rysunek 8.11: Porównanie kosztów (na jeden deskryptor) auto-dopasowania i dopasowania różnych twarzy.



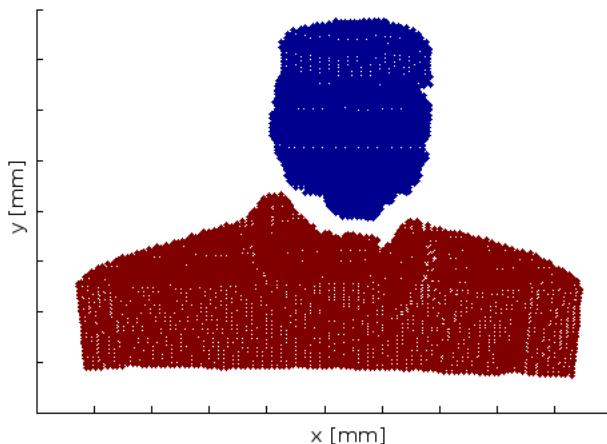
## Rozdział 9

---

# Wybór segmentu twarzy

---

Wstępne filtrowanie danych (opisane w rozdziale 6) powoduje usunięcie błędnych punktów i części zbędnych, drobnych elementów sceny. Wyjściowe dane takiego przetwarzania mogą zawierać kilka grup punktów - segmentów. Zakłada się, że częścią jednego z segmentów jest twarz. Oczywiście, może ona też zawierać inne elementy takie jak np. szyja, kolnierz. Przykładowe wyniki działania algorytmu filtrowania przedstawione są na rysunkach 9.1, 9.2. Widoczne są dwa segmenty. Zaproponowano zastosowanie deskryptorów punktów do wyboru segmentu zawierającego twarz.

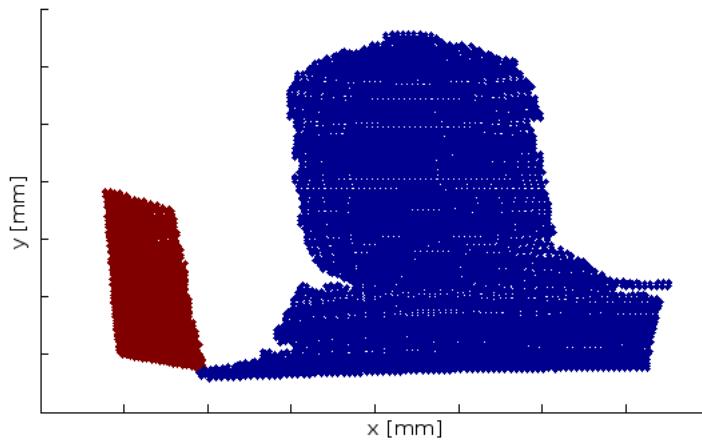


Kolory reprezentują przyporządkowanie do segmentów.

Rysunek 9.1: Przykład wyniku działania fazy wstępnego filtrowania danych (cara30 frontal1).

Algorytm wyboru segmentu składa się z następujących kroków:

1. dla każdego z segmentów:
  - a) wygeneruj zbiór deskryptorów dla wybranych punktów
  - b) dopasuj deskryptory do deskryptorów wzorca (są one wczytywane z zewnętrznego pliku) poprzez minimalizację sumarycznego kosztu (algorytm węgierski)



Kolory reprezentują przyporządkowanie do segmentów.

Rysunek 9.2: Przykład wyniku działania fazy wstępного filtrowania danych (cara28 frontal1).

- c) zachowaj wartość uzyskanego kosztu
2. wybierz segment o minimalnej wartości kosztu
3. zweryfikuj statystycznie koszt względem rozkładu uzyskiwanych wartości.

## 9.1 Dobór parametrów

Używane deskryptory (*spin images*) charakteryzowane są przez zestaw 4 parametrów (odpowiednio po dwa dla wymiaru  $\alpha$  i  $\beta$ ):

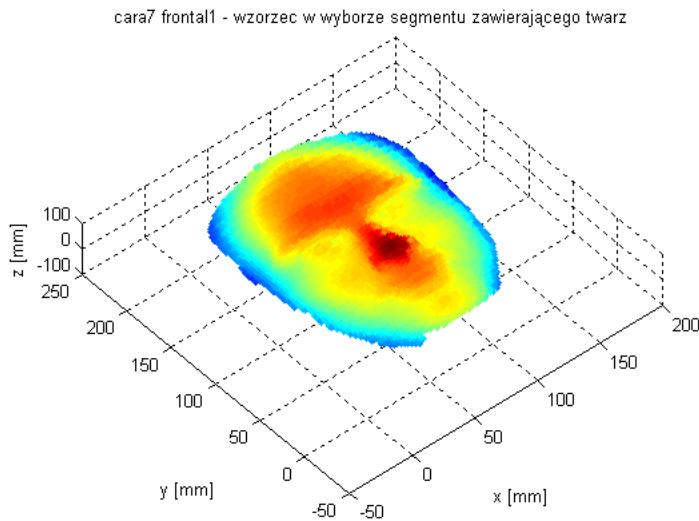
- rozdzielcość skali (liczba "kubelków").
- typ skali (logarytmiczna/liniowa).

Piątym parametrem jest promień otoczenia  $r$  używany przy ograniczeniu przestrzeni z jakiej wybierane są punkty w oparciu o które budowany jest histogram. Dopasowując zbiory deskryptorów rozważyć z kolei należy:

- typ stosowanej odległości między dwoma deskryptorami (*shape context measure* / korelacja liniowa).
- metodę filtrowania przy obliczaniu odległości (AND/OR/NONE).

Kwestią do rozważenia jest też sposób wyboru punktów dla których liczone są deskryptory. W powyższym zastosowaniu przyjęto, że wystarczający jest losowy wybór deskryptorów (opisany w sekcji 8.1.1). Za wzorzec przyjęto obszar twarzy wyselekcjonowany z obrazu cara7 frontal1. Ukazany jest on na rysunku 9.3. Ustalony został też promień sąsiedztwa  $r$ . Arbitralnie przyjęto  $r = 100$ .

Pozostałe parametry dobrane zostały eksperymentalnie. Dla dwóch przykładowych obrazów ukazanych na rysunkach 9.1, 9.2, oraz obrazu cara48 frontal1 przetestowano



Rysunek 9.3: Wzorzec stosowany w wyborze segmentu twarzy (cara7 frontal1).

kilkadziesiąt zestawów parametrów. Przesłanką stojącą za takim wyborem testowych obrazów było występowanie na nich dwóch najczęstszych problemów:

- duży segment nie zawierający twarzy (rysunek 9.1).
- duża zawartość punktów nie należących do twarzy w segmencie zawierającym twarz (rysunek 9.2).

Obrazy wstępnie przefiltrowane zostały z ustalonym parametrem filtrowania:  $D_c = 0.9$ . Następnie przetestowano zachowanie się algorytmu dla różnych konfiguracji parametrów. Wynik pierwszego z testów przedstawiono w tabeli 9.1.

Oznaczenia stosowane w tabelach 9.1, 9.2, 9.3, 9.4:

- c28/30/48f - sumaryczny koszt dopasowania dla segmentu zawierającego twarz w obrazie cara28/30/48 frontal1
- c28/30/48nf - sumaryczny koszt dopasowania dla segmentu nie zawierającego twarzy w obrazie cara28/30/48 frontal1. W przypadku większej liczby takich segmentów wybierany jest ten o minimalnym sumarycznym koszcie.
- r28/30/48 = c28/30/48nf - r28/30/48f - wartość zróżnicowania.

Wnioski z pierwszego testu:

- zastosowanie skali logarytmicznej zwiększa poziom rozróżnialności (różnicę kosztów dopasowania) między danymi dopasowanymi do wzorca i danymi odrzuconymi.
- zastosowanie odległości SCM w większości przypadków poprawia stopień rozróżnienia danych. Tendencja się zwiększa wraz ze wzrostem liczby stosowanych deksyptorów.

Tablica 9.1: Wyniki dopasowywania obrazów testowych do wzorca segmentu twarzy dla różnych parametrów (filtrowanie metodą OR).

M	K	Skala	Shape Context Measure				Korelacja liniowa			
			c28f	c28nf	c30f	c30nf	c28f	c28nf	c30f	c30nf
25	5	Lin	3.08	3.06	3.19	4.94	1.76	2.25	1.6	3.6
25	5	Log	2.18	13.17	2.52	4.33	1.16	11.78	0.39	2.99
25	10	Lin	5.36	11.77	5.44	7.25	4.79	10.85	5.18	7.7
25	10	Log	4.64	18.13	3.58	6.95	2.34	13.59	2.14	4.33
50	5	Lin	4.12	5.29	2.53	8.18	3.25	4.46	2.06	6.96
50	5	Log	2.86	22.66	2.78	8.49	1.75	20.1	0.69	6.97
50	10	Lin	6.15	22.38	7.41	11.47	7.21	21.31	6.7	10.75
50	10	Log	6.89	35.83	6.41	12.95	3.89	19.52	2.78	5.98
100	5	Lin	5.95	7.4	5.83	14.39	5.55	5.17	5.47	14.41
100	5	Log	4.59	40.92	4.99	18.39	2.56	38.97	1.53	7.74
100	10	Lin	13.26	45.52	14.05	23.21	12.35	35.88	13.41	21.13
100	10	Log	11.97	66.76	14.77	23.51	6.64	46.36	6.36	13.88

Dla obu osi histogramów zastosowano te same parametry.

- zbyt mała rozdzielcość deskryptorów lub zbyt duża prawdopodobieństwo błędów: odnotowano dwa przypadki błędnej klasyfikacji dla obrazu *cara28 frontal1*: (25 punktów, rozdzielcość 5, skala liniowa, SCM) oraz (100 punktów, rozdzielcość 5, skala liniowa, korelacja liniowa).

Na podstawie powyższych rozważań zdecydowano się zastosować 100 deskryptorów o rozdzielcościach  $10 \times 10$  i skalach logarytmicznych. Do obliczania odległości wybrano miarę SCM.

Zauważać należy, że zachowanie się algorytmu dla ustalonej liczby deskryptorów jest prawdopodobnie dość silnie skorelowane ze specyfiką danych. Można sobie wyobrazić sytuację w której segment zawierający twarz będzie tak duży, że losowe rozmieszczenie na nim deskryptorów spowoduje, że w okolicy twarzy będzie ich niewystarczająco dużo. Hipotetycznie taki układ mógłby doprowadzić do niepoprawnego zachowania się metody, jednak dla danych rzeczywistych sytuacja taka jest wysoce nieprawdopodobna.

Sprawdzono zachowanie się algorytmu z tak dobranym zestawem parametrów dla różnych metod filtrowania. Dla 3 obrazów testowych wykonano po 3 próby. Wyniki zaprezentowano w tabelach 9.2, 9.3, 9.4. Rozbieżność w rezultatach jest spowodowana losowym wyborem punktów dla których liczone są deskryptory. Porównanie zawartości tabel potwierdza, że średnio najlepszym podejściem jest filtrowanie deskryptorów metodą OR. Najgorzej wypadło filtrowanie metodą AND. Brak filtrowania daje wyniki nieznacznie gorsze od pierwszej z metod.

## 9.2 Schemat fazy wstępnej

Na fazę wstępne przetwarzania danych (pierwszy krok schematu z rysunku 5.2) składają się wyżej opisane procedury:

- filtrowania przez segmentację w grupy połączone (rozdział 6)

Tablica 9.2: Sumaryczny koszt dopasowania obrazów testowych do wzorca dla wybranych parametrów (filtrowanie metodą OR).

Lp.	cara 28			cara 30			cara 48		
	c28f	c28nf	r28	c30f	c30nf	r30	c48f	c48nf	r48
1	12.63	66.53	53.90	15.58	26.95	11.38	10.99	24.92	13.93
2	13.13	64.20	51.07	14.54	26.19	11.64	9.42	24.27	14.85
3	12.82	67.70	54.88	16.17	26.39	10.22	9.35	24.45	15.10

Tablica 9.3: Sumaryczny koszt dopasowania obrazów testowych do wzorca dla wybranych parametrów (filtrowanie metodą AND).

Lp.	cara 28			cara 30			cara 48		
	c28f	c28nf	r28	c30f	c30nf	r30	c48f	c48nf	r48
1	7.28	19.77	12.49	7.62	9.75	2.13	6.08	9.39	3.31
2	6.66	18.50	11.84	7.82	9.17	1.35	6.41	10.29	3.87
3	6.63	19.30	12.67	7.38	8.51	1.13	5.66	10.46	4.81

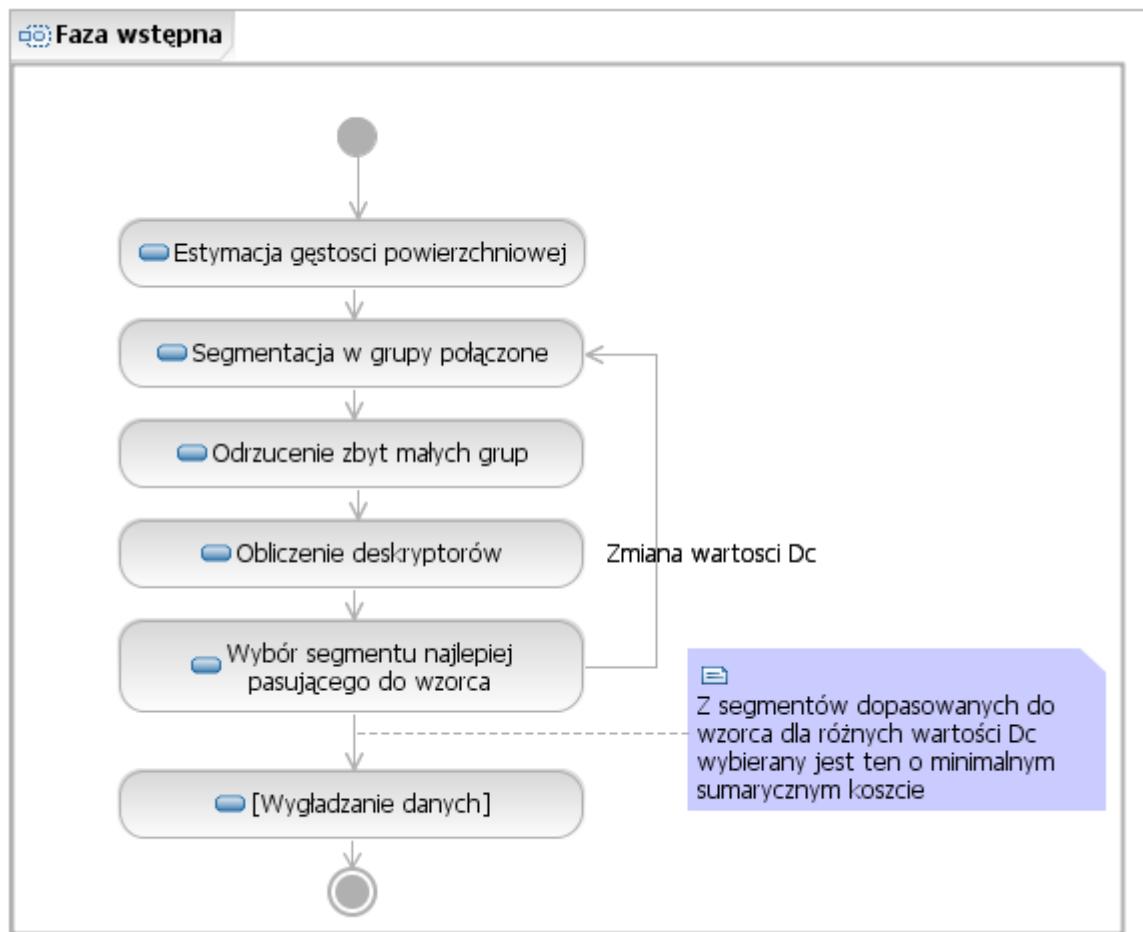
Tablica 9.4: Sumaryczny koszt dopasowania obrazów testowych do wzorca dla wybranych parametrów (brak filtrowania).

Lp.	cara 28			cara 30			cara 48		
	c28f	c28nf	r28	c30f	c30nf	r30	c48f	c48nf	r48
1	5.73	39.20	33.47	6.25	15.91	9.66	4.62	12.01	7.38
2	6.72	40.07	33.35	5.63	12.21	6.58	4.12	12.85	8.73
3	6.13	38.81	32.68	6.10	13.89	7.79	4.57	11.57	7.00

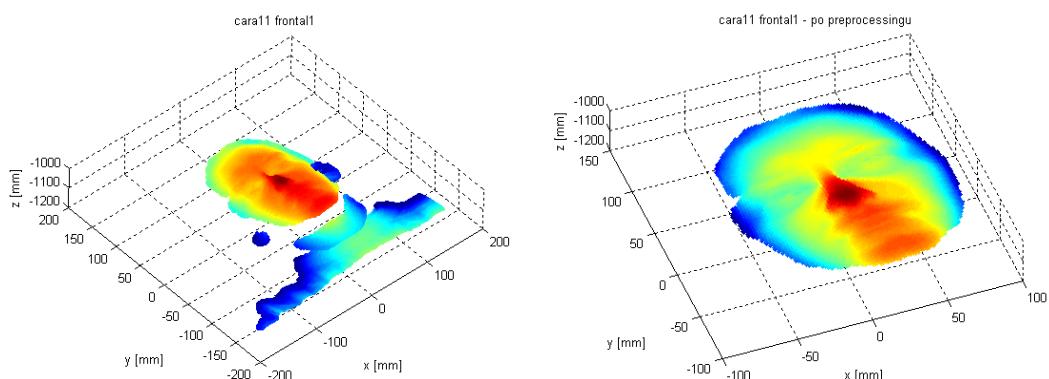
- wyboru segmentu twarzy (powyżej)
- wygładzania danych (rozdział 7.5, opcjonalnie)

Dokładny schemat działania zaprezentowano na rysunku 9.4. Pierwsze dwie z wymienionych wyżej procedur wykonywane są kilkakrotnie dla różnych wartości parametru segmentacji  $D_c$ . Każda z iteracji daje w rezultacie inną wartość sumarycznego kosztu dopasowania dla segmentu uznanego za zawierający twarz. Do dalszej analizy wybierany jest ten z segmentów, dla którego obliczony koszt jest najmniejszy.

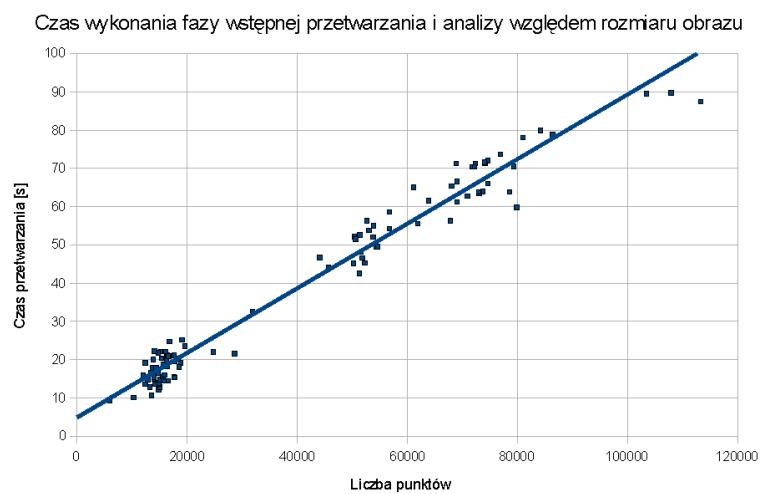
W ostatecznej wersji, w celu redukcji czasu obliczeń, sprawdzane są wyłącznie dwie wartości  $D_c$ : 0.5 i 1.0. Przykładowe wyniki działania wstępniego przetwarzani przedstawiono na rysunku 9.5. Zależność czasu wykonania od liczby punktów obrazu wejściowego ilustruje rysunek 9.6.



Rysunek 9.4: Schemat fazy wstępniego przetwarzania danych wraz z wyborem segmentu twarzy.



Rysunek 9.5: Przykład działania fazy wstępniego przetwarzania danych wraz z wyborem segmentu twarzy.



Rysunek 9.6: Zależność czasu wykonania fazy wstępnego przetwarzania względem liczby punktów obrazu.

### 9.3 Analiza fazy wstępnej

Zaimplementowaną procedurę wstępnego przetwarzania wraz z wyborem segmentu zawierającego twarz przetestowano dla kilku zestawów danych:

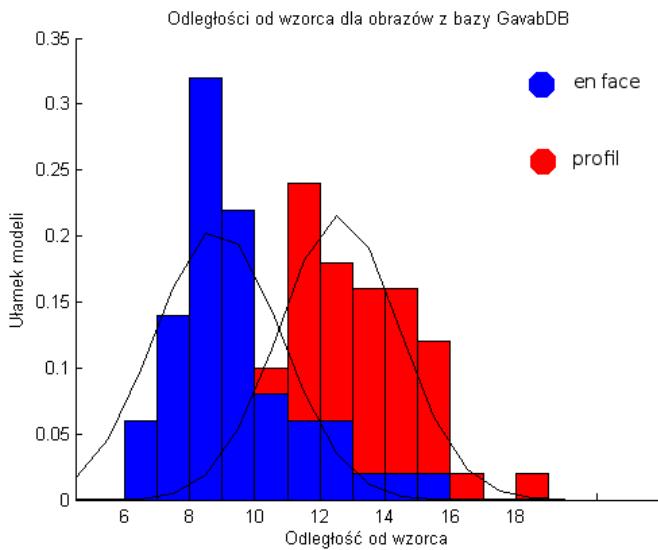
- Pierwszym zestawem danych było 100 obrazów z bazy GavabDB. Zostały one podzielone na dwie grupy: zawierające widok twarzy z przodu i z profilu. Dla wszystkich testowanych przypadków segment zawierający twarz wybrany został poprawnie. Histogram uzyskiwanych sumarycznych kosztów dopasowania obrazów do wzorca przedstawiono na rysunku 9.7.
- Na drugi zbiór testowy składało się 50 obrazów z bazy Mechatroniki. Podzielono je, zgodnie z klasyfikacją z tabeli 3.2, na trzy grupy:
  1. 34 obrazy zawierające twarz
  2. 14 obrazów zawierających fragment twarzy lub o znacznych błędach w danych
  3. 2 obrazy nie zawierające twarzy

Dla wszystkich obrazów z pierwszej grupy poprawnie wybrano segment zawierający twarz. W drugiej grupie poprawnie udało się wybrać segment w 13 przypadkach. Histogramy sumarycznych kosztów dopasowania dla grup 1 i 2 przedstawiono na rysunku 9.8. Dla grupy 3 uzyskane wartości kosztów to: 20,60 i 24,57.

- Ostatni zbiór składał się z losowo wygenerowanych 100 obrazów. Obraz o numerze  $i$ -tym wygenerowany został wg następującego schematu:
  1. Wygeneruj  $1000 \cdot i$  punktów. Każdą ze współrzędnych  $(x, y, z)$  wylosuj (stosując rozkład jednostajny) z przedziału  $[0, 200]$ .
  2. Wygeneruj  $1000 \cdot i$  punktów. Każdą ze współrzędnych  $(x, y, z)$  wylosuj (stosując rozkład jednostajny) z przedziału  $[i \cdot 300, 200 + i \cdot 300]$
  3. Połącz obie chmury punktów.

Rozkład sumarycznych kosztów dopasowania do wzorca przedstawiono na rysunku 9.9.

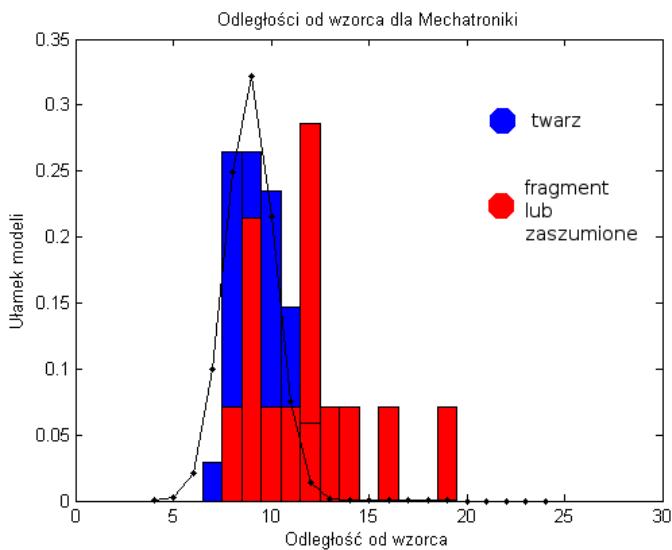
Przeprowadzone testy potwierdziły wysoką skuteczność zaprojektowanego algorytmu. Dodatkowo ukazały potencjał wykorzystania sumarycznego kosztu dopasowania (odległości) do oceny danych wejściowych. Uzyskiwane wartości z dużą skutecznością pozwalały odróżnić dane zawierające obraz twarzy od pozostałych. Opierając się na rozkładach danych ukazanych na rysunkach 9.7 i 9.8 przyjęto próg równy 16. Obrazy o koszcie dopasowania powyżej tej wartości uznane mogą zostać za niezawierające twarzy. Przyjmując założenie o rozkładzie normalnym kosztów obliczonych dla obrazów zawierających twarz (rysunki 9.7 i 9.8), obliczono, że przy tak przyjętym progu odrzuconych zostanie mniej niż 1% takich danych. Brak wystarczającej ilości skanów nie zawierających twarzy uniemożliwia podobną estymację dla klasyfikacji negatywnej. Wyniki ukazane na rysunku 9.9 pozwalają jednak oczekiwąć podobnie dobrych rezultatów.



statystyki en face: średnia = 9,35; odchylenie standardowe = 1,94

statystyki profilu: średnia = 13,09; odchylenie standardowe = 1,85

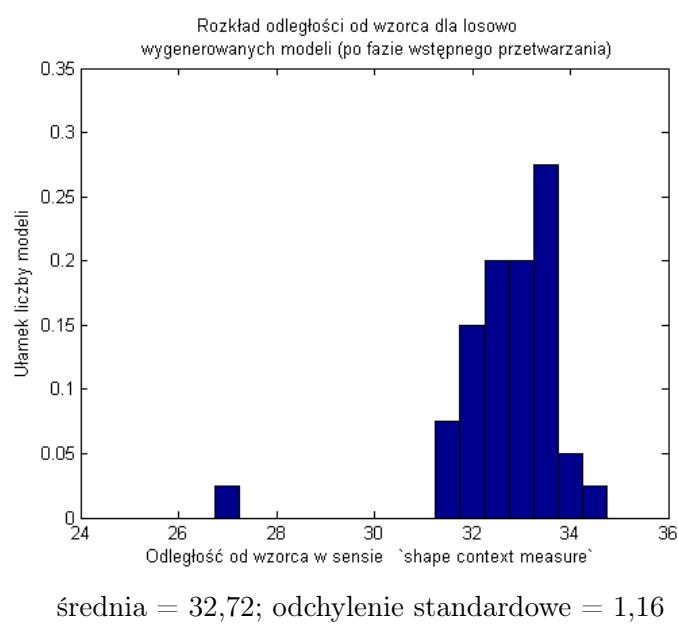
Rysunek 9.7: Rozkład odległości od wzorca dla obrazów z bazy GavabDB.



statystyki twarzy: średnia = 9,89; odchylenie standardowe = 1,23

statystyki fragmentów: średnia = 12,42; odchylenie standardowe = 2,85

Rysunek 9.8: Rozkład odległości od wzorca dla obrazów z bazy Mechatroniki.



Rysunek 9.9: Rozkład odległości od wzorca dla obrazów wygenerowanych losowo.

## Rozdział 10

---

# Hierarchiczna lokalizacja regionów

---

### 10.1 Hierarchiczna lokalizacja regionów

Wynikiem fazy wstępniego przetwarzania danych (ang. *preprocessing*) jest wybór spójnej grupy (segmentu) punktów zawierających twarz. Celem fazy II (patrz rysunek 5.2) polegającej na hierarchicznej lokalizacji regionów jest identyfikacja na twarzy pewnych charakterystycznych obszarów które mogłyby posłużyć do dalszej analizy innymi metodami.

#### 10.1.1 Algorytm lokalizacji regionu

Do lokalizacji kolejnych coraz mniejszych regionów w danych (twarz, nos, obszary nosa) zaprojektowano poniższy algorytm. Opiera się on na zastosowaniu łącznego dopasowywania deskryptorów punktów do deskryptorów wzorca. Jego zadaniem jest identyfikacja części danych (podzbioru chmury punktów) odpowiadających wzorcowi.

Wzorce zbudowane zostały w następujący sposób:

1. Z obrazu 3D wycinany jest poszukiwany obszar (zbiór punktów)  $Q$  wraz z jego pewnym otoczeniem  $Q'$  (np.  $Q$  - nos,  $Q'$  - część twarzy wraz z nosem):

$$Q \subset Q'$$

Deskryptory budowane będą na punktach wybieranych z obszaru  $Q$ , jednak w obliczeniach uwzględniane będą punkty z  $Q'$ . W związku z tym, rozmiar otoczenia przyjmuje się co najmniej taki, aby przy budowie deskryptorów, w sąsiedztwie (o promieniu  $r$ ) punktu  $p$ , dla którego budowany jest deskryptor, nie znalazły się fragmenty puste. Mogłyby one wpłynąć negatywnie na jakość dopasowywania. Ich wpływ zredukowany zostać może przez zastosowanie filtrów OR i AND, jednak w ten sposób pogarszana jest jakość dopasowywania pomiędzy poszczególnymi deskryptorami.

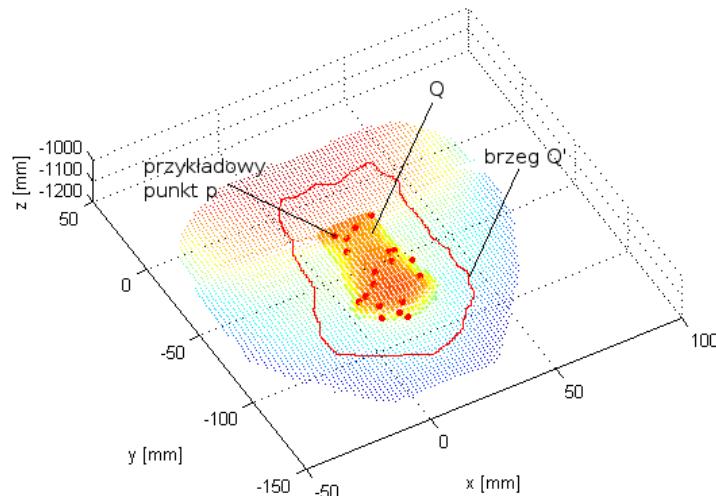
2. Z  $Q$  wybierany jest podzbiór punktów. Dla każdego z wybranych punktów  $p$ , w oparciu o punkty z  $Q'$ , budowany jest deskryptor. Zbiór deskryptorów (wraz z odpowiadającymi im punktami: rozróżnienie deskryptor/punkt wynika z kontekstu) dla obszaru  $Q$  oznaczany jest  $D_Q$ .
3. Obliczana jest powierzchnia obszaru  $Q$ :  $A_Q = \text{powierzchnia}(Q)$ . Do obliczenia powierzchni obrazu zastosowano opisany w rozdziale 7.2 algorytm estymacji gęstości powierzchniowej  $\sigma$  obrazu 3D.

$$\text{powierzchnia}(Q) = \frac{|Q|}{\sigma}$$

gdzie:

- $|Q|$  - liczba punktów  $Q$
- $\sigma$  - gęstość powierzchniowa  $Q'$

Taki sposób budowania wzorców nie dotyczy przygotowania tych służących do poszukiwania twarzy w danych. Powodem tego jest to, że otoczenie twarzy w obrazie może być zupełnie dowolne. W takim przypadku przyjęto, że  $Q = Q'$  i przy dopasowywaniu do wzorca zastosowano filtrowanie AND. Powoduje ono, że nie brane są pod uwagę obszary położone poza twarzą. Sposób budowy wzorców zilustrowano na rysunku 10.1.



Rysunek 10.1: Przykład budowy wzorca do lokalizacji regionu.

Algorytm wyszukujący w obrazie testowanym  $T'$  region  $T$  (odpowiadający regionowi  $Q$  w  $Q'$ ) przedstawiono poniżej:

0. Wygeneruj potrzebną liczbę deskryptorów na chmurze punktów  $T'$ :
- a) dla każdego kolejnego wzorca (regionu poszukiwanego)  $Q$  i odpowiadającego mu zbioru  $D_Q$  oblicz liczbę deskryptorów do wygenerowania dla obrazu  $T'$  w kontekście porównywania z  $Q$ :

$$|D_{T'}(Q)| = \lceil |D_Q| \cdot \frac{\text{powierzchnia}(T')}{\text{powierzchnia}(Q)} \rceil$$

gdzie:

- $|D_Q|$  - liczba deskryptorów wygenerowanych dla wzorca  $Q$
- $|D_{T'}(Q)|$  - liczba deskryptorów jaką należy wygenerować na chmurze  $T'$  aby uzyskać jak najlepszą jakość identyfikacji poszukiwanego, zgodnego z  $Q$ , regionu

Oczekuje się, że przy równomiernym rozmieszczeniu  $|D_{T'}(Q)|$  deskryptorów na powierzchni  $T'$  w obszarze  $T$  znajdzie się ich w przybliżeniu tyle samo, ile zostało ich wygenerowanych dla  $Q$ :  $|D_T(Q)| \approx |D_Q|$ .

- b) oblicz ile deskryptorów należy wygenerować na  $T'$  łącznie dla wszystkich wzorców:

$$|D_{T'}| = \sum_{Q \in Wzorce} |D_{T'}(Q)|$$

- c) wybierz z  $T'$  podzbiór  $|D_{T'}|$  punktów i wygeneruj dla nich zbiór deskryptorów  $D_{T'}$ . W obliczeniach uwzględnij wszystkie punkty  $T'$ . Opcjonalnie, można też uwzględnić punkty należące do otoczenia  $T'$ . Przykładowo: lokalizując region wierzchołka nosa (ozn.  $T$ ) w regionie zaklasyfikowanym wcześniej jako nos (ozn.  $T'$ ) celowym jest uwzględnienie w histogramach nie tylko punktów nosa, ale też twarzy ( $otoczenie(T')$ ). Powoduje to, że informacja jest pełniejsza.

1. Wygeneruj zbiory uczące: zbiór punktów dopasowanych do wzorców  $D_T$  i zbiór punktów odrzuconych  $D_{T''}$ :

Dla każdego kolejnego wzorca  $Q$  i odpowiadającego mu  $D_Q$ :

- a) wybierz z puli  $D_{T'}$  kolejne  $|D_{T'}(Q)|$  deskryptorów. Elementy w  $D_{T'}$  nie są posortowane w żaden sposób. Wybierając kolejne deskryptory, można się spodziewać, że punkty im odpowiadające są rozmieszczone na całym  $T'$ .

- b) dopasuj elementy  $D_{T'}(Q)$  i  $D_Q$  minimalizując sumaryczny koszt dopasowania. Dodaj do zbioru  $D_T$  elementy z  $D_{T'}(Q)$ , które zostały dopasowane do  $D_Q$ . Elementy niedopasowane dodaj do zbioru  $D_{T''}$  (oba zbiory początkowo są puste).

2. Przefiltruj elementy zbioru  $D_T$  stosując filtrowanie przez segmentację w grupy połączone (rozdział 6). Elementy odrzucone przenieś ze zbioru  $D_T$  do  $D_{T''}$ .

3. Przefiltruj elementy zbioru  $D_{T''}$  poprzez ponowną klasyfikację.

Zastosowany został zmodyfikowany klasyfikator k-nn. Modyfikacja polega na przyporządkowaniu każdej klasie wagi. Znajdowanych jest k - najbliższych sąsiadów klasyfikowanego elementu. Następnie zliczana jest reprezentacja każdej z klas. Element przyporządkowany zostaje do klasy której iloczyn liczności i wagi jej przypisanej jest największy. Jako klasy (dane uczące) podawane są zbiory  $D_{T''}$  i  $D_T$ . W wyniku działania algorytmu część elementów z  $D_{T''}$  zostaje przeniesiona do  $D_T$ .

4. Zbuduj klasyfikator traktując zbiory  $D_{T''}$ ,  $D_T$  jako dane uczące.

5. Zaklasyfikuj wszystkie punkty  $T'$ . Część nich przypisana zostanie do tej samej klasy co  $D_T$ . Są to punkty które uznane zostały za odpowiadające punktom wzorca  $Q$ . Oznacz je przez  $F$ . Pozostałe punkty zostały uznane za niepasujące do wzorca. Należy je odrzucić.
6. Przefiltruj  $F$  korzystając z filtracji przez segmentację w grupy połączone (opisanej w rozdziale 6).

Celem kroku 0 jest wygenerowanie deskryptorów, które następnie w kroku 1 będą dopasowywane do wzorców. W algorytmie dąży się do uzyskania w analizowanej chmurze punktów tej samej gęstości rozmieszczenia deskryptorów na powierzchni jaką jest we wzorcach. Przy spełnieniu takiego założenia na takiej samej powierzchni wzorca i analizowanych danych znajdzie się ich średnio taka sama liczba. Oznacza to, że na obszarze odpowiadającym wzorcowi zostanie wygenerowanych ich tyle samo co na wzorcu. W przypadku optymalnym wszystkie one zostaną następnie dopasowane do deskryptorów wzorca. Pozostałe deskryptory (te nie leżące w poszukiwanym regionie) zostaną odrzucone.

Wynikiem działania kroku nr 1 algorytmu są dwa zbiory deskryptorów (wraz z odpowiadającymi im punktami). Jeden zbiór zawiera deskryptory punktów które przyporządkowano do regionu poszukiwanego. Powinny się one znaleźć w grupie punktów chmury wejściowej należących do regionu który jest lokalizowany. Drugi zbiór składa się z elementów odrzuconych. Powinny one zostać losowo rozmieszczone po powierzchni poza tym regionem.

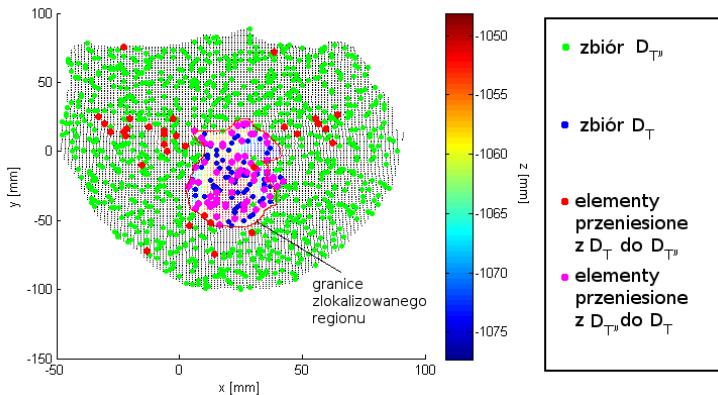
Tak rozłożone dane służą w kroku 4 do budowy klasyfikatora, który uogólniając uzyskaną informację, posłuży do klasyfikacji punktów całej chmury. Problemem jest jednak to, że w wyniku dopasowywania punktów analizowanych danych i wzorców część deskryptorów zostaje błędnie przyporządkowana.

W celu poprawy jakości tak powstały danych uczących kolejno filtryuje się oba zbiory w krokach 2 i 3. Korekta zbioru  $D_T$  opiera się na prostym założeniu o spójność poszukiwanego regionu. Pojedyncze elementy i małe grupy elementów położone z dala od większości (która w założeniu identyfikuje poszukiwany region) zostają odrzucone. W kroku 3, z kolei, usuwane są pojedyncze elementy uznane za nienależące do poszukiwanego regionu, ale znajdujące się w takim regionie, gdzie większość stanowią elementy o przeciwnym przyporządkowaniu.

Ostatni krok algorytmu koryguje wyniki klasyfikacji opierając się na założeniu podobnym do tego stosowanego w kroku 2. Przykład działania algorytmu pokazano na rysunku 10.2.

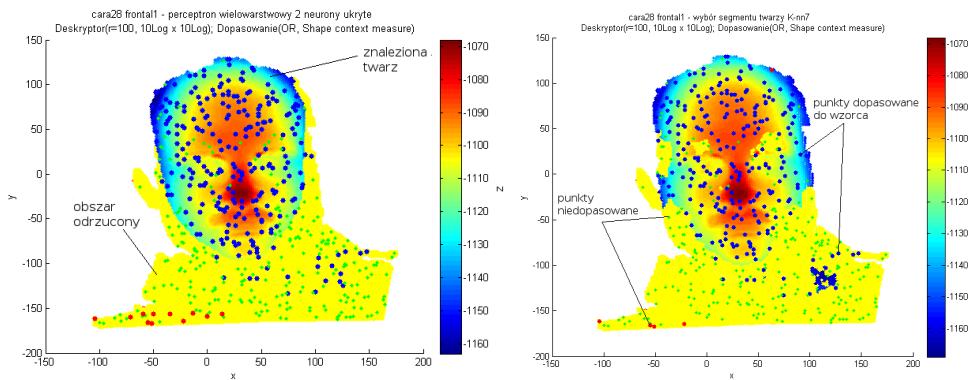
Działanie algorytmu zależy od zestawu parametrów. Sterują one jego zachowaniem w kolejnych fazach. Są to:

- liczba i wybór wzorców. Powinny one dobrze definiować klasę poszukiwanych danych. Jednocześnie nie powinno być ich zbyt dużo. Mogliby to powodować duże koszty obliczeniowe.
- parametry charakteryzujące deskryptory (promień otoczenia, rozdzielcość i typ skali) i sposób ich dopasowywania (typ filtru, stosowana odległość). Powinny one zostać dostosowane do spodziewanej zmienności danych.



Rysunek 10.2: Przykład działania algorytmu lokalizacji regionów.

- sposób wyboru punktów dla których zbudowane zostaną deskryptory (może się on różnić dla wzorców i dla analizowanych obrazów). Dla wzorców zastosowano wybór punktów metodą opartą o k-średnich (rozdział 8.1.2), co podyktowane jest dobrym rozmieszczeniem deskryptorów. Ponieważ generowane są one jednokrotnie i później wczytywane z pliku to czas ich budowania nie gra roli. Natomiast, ze względu na koszt obliczeń, dla analizowanych danych stosuje się podejście losowe (rozdział 8.1.1) lub *octClustering* (rozdział 8.1.3).
- liczba deskryptorów we wzorcach. Jest naturalnym, że zwiększając ich rozmiar, potencjalnie uzyskać można znacznie lepsze dopasowanie do wzorca i w efekcie lepsze wyszukiwanie obszarów. Jednocześnie zwiększa się, jednak, czas przetwarzania. Parametr ten powinien zostać uzależniony od tego jak dokładny wynik jest oczekiwany.
- parametry filtrowania punktów (dla których zbudowano deskryptory) poprzez grupy połączone ( $D_c$ ,  $S_c$ , krok 2).
- wagi klas przy filtrowaniu przez ponowną klasyfikację (krok 3).
- typ klasyfikatora użytego do klasyfikacji punktów wejściowego obrazu (krok 4). Rozważane były dwa typy klasyfikatora: k-nn i sieć neuronowa (perceptron wielowarstwowy z funkcją przejścia  $t(n) = \frac{1}{1+e^{-n}}$  uczony metodami wstępnej propagacji błędu i levenberga-marquardta). Wadą tego pierwszego jest ryzyko nadmiernego (lokalnego) dopasowania do danych. Powoduje to, że wynikowy zbiór punktów  $F$  jest niespójny i może zawierać nieciągłości. Ryzykiem związanym z sieciami neuronowymi jest natomiast możliwość zatrzymania procesu uczenia w minimum lokalnym, co prowadzi do kiepskiej jakości klasyfikacji. Zaletą jest dobra zdolność uogólniania, na którą można wpływać manipulując liczbą neuronów ukrytych. Przykład zachowania obu klasyfikatorów pokazano na rysunku 10.3. Test przeprowadzono dla pojedynczego wzorca *cara7 frontal1* dla którego wygenerowano 275 deskryptorów.
- parametry filtrowania punktów danych ( $D_c$ ,  $S_c$ , krok 6)



Rysunek 10.3: Przykład działania klasyfikatorów dla klasyfikacji punktów twarzy.

### 10.1.2 Schemat algorytmu analizy

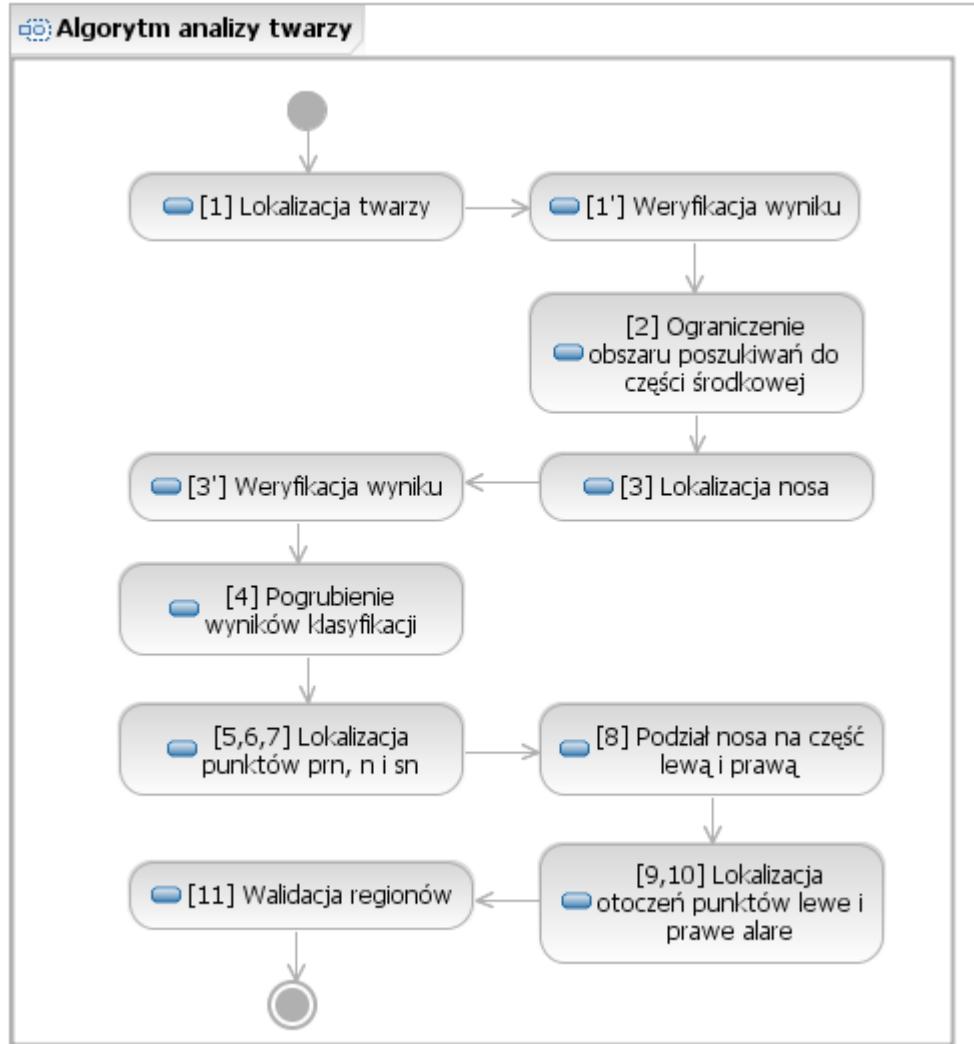
W oparciu o wcześniej opisany algorytm lokalizujący pojedynczy region zaprojektowano algorytm wyszukujący otoczenia punktów charakterystycznych nosa. Zaprezentowano go na rysunku 10.4. Składa się on z siedmiu faz, w których wykonywana jest powyższa procedura lokalizacji regionów. Jest ona wywoływaną kolejno w celu **zidentyfikowania regionów w chmurze punktów: twarzy, nosa oraz otoczenia pięciu punktów na nosie: prn (wierzchołek nosa), n, sn oraz lewego i prawego alare (skrzydełka nosa)**.

Wybór twarzy (wyznaczanej przez linię włosów i szczęki) na region identyfikowany (lokalizowany) w pierwszej kolejności wynika z założenia, że twarz jest największym obszarem, którego istnienie zakłada się w analizowanej scenie (obrazie 3D). W lokalizacji dowolnego podregionu wykorzystywana jest informacja o otoczeniu. Twarz jest największym otoczeniem (o znanej charakterystyce) dla każdego podregionu twarzy. Jej wybór więc maksymalizuje ilość informacji na podstawie której wyszukiwane mogą być regiony wewnętrzne.

O wyróżnieniu nosa w kolejnym poziomie hierarchii lokalizacji regionów zadecydowały wyniki testu opisanego w rozdziale 10.1.4. Pokazano, że ze względu na charakterystyki deskryptorów wyróżnia się on jako niezależny region. Test pokazał też, że trudne jest wyróżnienie poziomu pośredniego pomiędzy nosem, a otoczeniami punktów charakterystycznych nosa. Dodatkowo, rozmiar regionu zewnętrznego (nosa) nie jest na tyle duży w stosunku do regionów wewnętrznych (prn, n, itd.), aby opłacalne było to z punktu widzenia złożoności obliczeniowej.

Na wybór punktów charakterystycznych których otoczenia są poszukiwane wpłynęło kilka czynników. Pierwszym z nich była dostępność wzorców. Wybrane regiony są na tyle rozpoznawalne, że możliwe było przygotowanie zestawu uczącego przez osobę bez specjalnego przygotowania (co więcej w sekcji 10.1.6 pokazano, że nawet tak przygotowane dane są bardzo niedoskonałe). Drugą kwestią jest użyteczność. Wybranych pięć punktów determinuje większość wymiarów typowo obliczanych dla nosa (patrz rozdział 2.3).

W celu skrócenia czasu analizy oraz poprawy zachowania algorytmu zastosowano kilka heurystyk. Pierwsza z nich opiera się na założeniu, że obraz zawiera pełny widok twarzy oraz, że nos jest położony w części środkowej twarzy. Przy takich założeniach



Rysunek 10.4: Schemat algorytmu analizy twarzy.

zlokalizowana w kroku 1 twarz, może zostać znacznie ograniczona w kroku 2. Wyszukiwany jest środek ciężkości zaklasyfikowanego jako twarz regionu (ozn.  $T_F$ ), po czym do dalszej analizy wybierane są te punkty z  $T_F$  których odległość od tego punktu jest mniejsza niż  $r_c = 7\text{cm}$  (ozn.  $T_C$ ):

$$T_C = \{q : \|q - \text{centroid}(T_F)\| \leq r_c\}$$

W kroku 4 algorytmu zastosowano procedurę "pogrubiania" regionu nosa (ozn.  $T_N$ ). Polega ona na tym, że punkty zbioru  $T_C$  są ponownie klasyfikowane, z zastosowaniem punktów  $T_N$  jako zbioru uczącego. Do nowego zbioru  $T_M$  przyporządkowywane są punkty, których odległość od najbliższego z elementów  $T_N$  jest mniejsza niż  $r_t = 6\text{mm}$ :

$$T_M = \{q : \min_{q' \in T_N} \|q - q'\| \leq r_t\}$$

Na końcu zbiór  $T_N$  jest nadpisywany zbiorem  $T_M$ . W wyniku działania algorytmu brzegi regionu zostają 'pogrubione', a małe obszary wcześniej niezaklasyfikowane do regionu,

a przez niego otoczone, wchłonięte. Powodem zastosowania powyższej heurystyki jest dążenie do podniesienia jakości klasyfikacji. Brzegi obszaru nosa mogą zostać błędnie sklasyfikowane co mogłoby spowodować to, że punkty charakterystyczne znalazłyby się poza regionem poddawanym dalszej analizie. Zwiększenie obszaru przekazywanego do dalszej analizy redukuje to niebezpieczeństwo.

Krok 8 służy rozróżnieniu lewej części nosa od prawej. Rozróżnienie to jest konieczne, gdyż dla punktów symetrycznie położonych po obu stronach twarzy (parzystych, patrz rozdział 2.3) obliczane deskryptory są bardzo podobne. Powoduje to, że około połowy deskryptorów przy lokalizacji regionu lewego alare dopasowana zostaje w okolicy prawego alare i odwrotnie. Zaprojektowany algorytm nie uwzględnia takich sytuacji (zakłada się spójność poszukiwanego regionu) w wyniku czego konieczne jest wstępne ograniczenie obszaru poszukiwań. Obszar nosa  $T_N$  dzielony jest na dwie części: lewą  $T_L$  i prawą  $T_R$ . Odbywa się to z wykorzystaniem informacji o położeniu punktów  $prn$ ,  $n$  i  $sn$  (lub zamiast  $sn$  - środkowa ciężkość  $T_N$ ):

1. obliczane są trzy wektory  $\vec{a} = n - prn$ ,  $\vec{b} = sn - prn$  i  $\vec{i} = \vec{a} \times \vec{b}$
2. dla każdego z punktów  $q \in T_N$  obliczany jest wektor  $\vec{i}_q = q - prn$
3. jeśli spełnione jest  $\vec{i}_q \cdot \vec{i} \leq 0$  to  $q \in T_L$ , w przeciwnym przypadku  $q \in T_R$

W krokach 1, 3 i 11 algorytmu wykonywana jest weryfikacja uzyskiwanych wyników. Służy ona sprawdzeniu i ocenie wiarygodności uzyskiwanych wyników. Sprawdzane jest czy w wyniku klasyfikacji udało się dopasować do wzorców punkty danych. Dodatkowo w kroku 3 weryfikowany jest sumaryczny koszt dopasowania. W przypadku gdy jest on wyższy od ustalonego progu ustawiana zostaje flaga mówiąca o błędzie. Próg ustalono (w oparciu o wyniki z testów opisanych w sekcji 10.2.2) na wartość 15.

W kroku 11 oceniana jest wiarygodność zidentyfikowanych regionów z wykorzystaniem informacji z tabeli 2.2. Punkty charakterystyczne obliczane są jako środki ciężkości regionów ich otaczających:

- $prn = centroid(T_{prn})$
- $n = centroid(T_n)$
- $sn = centroid(T_{sn})$
- $al_{lewe} = centroid(T_{all})$
- $al_{prawe} = centroid(T_{alr})$

Obliczane są odległości typowe zdefiniowane w rozdziale 2.3:

1. Wysokość (długość) nosa:  $h = \|n - sn\|$
2. Długość grzbietu nosa:  $l = \|n - prn\|$
3. Szerokość nosa:  $w = \|al_{lewe} - al_{prawe}\|$
4. Wysokość słupka nosa:  $h_2 = \|sn - prn\|$

Dla każdej z powyższych odległości  $x = h/l/w/h_2$  obliczane są następnie po dwa wskaźniki określające wiarygodność wg wzoru:

$$p(x) = 2 \cdot (1 - F_{gauss}(|\mu - x| + \mu, \mu, \theta)); \quad (10.1)$$

gdzie:

$F_{gauss}(v, \mu, \theta)$ - wartość dystrybuanty jednowymiarowego rozkładu normalnego o środku w  $\mu$  i odchyleniu standardowym  $\theta$  obliczona w punkcie  $v$

Wskaźniki liczone są po jednym dla rozkładu (wartość średnia i odchylenie standar-dowe z tabeli 2.2) dla kobiet i jednym dla mężczyzn. Następnie obliczane są średnie wartości wskaźnika osobno dla kobiet i dla mężczyzn:  $p_{kobiety}$ ,  $p_{mężczyzni}$ . Jeśli  $p_{kobiety} > p_{mężczyzni}$  to wybierane są wskaźniki obliczone dla rozkładu dla kobiet (zakłada się, że obraz przedstawia kobietę). W przeciwnym przypadku wybierane są wskaźniki obliczone dla rozkładu dla mężczyzn. W kolejnym kroku obliczane są odległości:

$$5. \quad w_l = \|al_{lewe} - prn\|$$

$$6. \quad w_r = \|al_{prawe} - prn\|$$

Dla tak obliczonych wartości zakłada się rozkład o średniej 32mm i odchyleniu standar-dowym 4mm.

Każdej z powyższych 6 odległości przyporządkowana jest flaga określająca czy jest ona uznawana za przydatną do dalszej analizy czy nie. Jeśli dla odległości  $x$  spełniony jest warunek  $\|x - \mu_x\| > 3 \cdot \theta_x + 2mm$ , to jest ona uznawana za nieprzydatną. Powodem ustalenia dopuszczalnego odchylenia na  $3 \cdot \theta_x + 2mm$ , zamiast typowego  $3 \cdot \theta_x$ , jest próba uwzględnienia niedokładności uzyskanego położenia punktów. Przyjęte oznaczenia:

- $\mu_x$  - wartość średnia przyjętego (kobieta/mężczyzna) rozkładu dla parametru  $x$
- $\theta_x$  - odchylenie standardowe przyjętego (kobieta/mężczyzna) rozkładu dla para-metru  $x$

W ostatnim kroku, dla każdego z punktów charakterystycznych, sprawdzane jest ile z powyższych odległości, w których obliczeniu został użyty, uznawanych jest za przydatnych do analizy. Dla punktów: n, sn,  $al_{lewe}$ ,  $al_{prawe}$  możliwe są wartości 0, 1, 2. Dla punktu prn 0, 1, 2, 3, 4, 5. Im większa jest ta liczba tym obliczone położenie punktu jest uznawane za bardziej wiarygodne.

Tak dobrany wskaźnik jest bardzo prosty. Niestety brak informacji dotyczącej roz-kładu większej liczby odległości na nosie uniemożliwia stworzenie bardziej zaawansowa-nego wskaźnika. Przykładowo: dostępne są informacje o rozkładzie odległości  $\|sn - prn\|$  i  $\|sn - n\|$ . Gdyby dostępna była też informacja o oczekiwanych rezultatach  $\|sn - al_{lewe}\|$  i/lub  $\|sn - al_{prawe}\|$  możliwe byłoby znacznie precyzyjniejsze określenie oczekiwane-go położenia punktu sn względem pozostałych punktów.

### 10.1.3 Lokalizacja twarzy

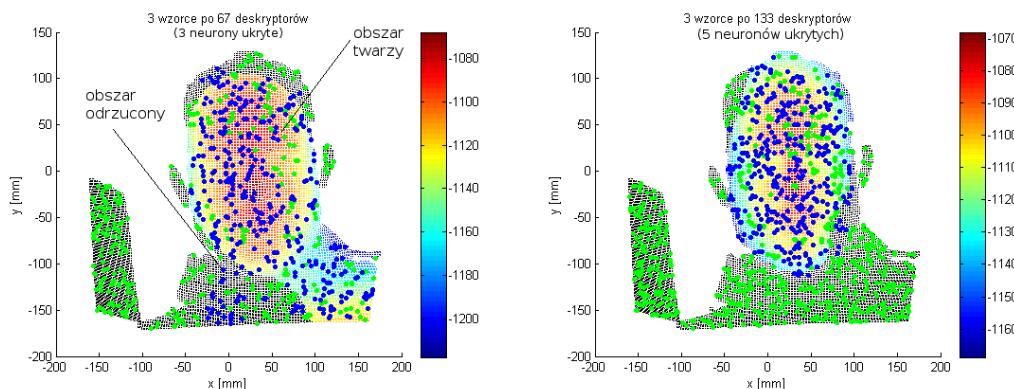
Wybór regionu twarzy, ze spójnego segmentu punktów, jest specyficzny o tyle, że nie jest możliwe zdeterminowanie we wzorcu jej otoczenia (punktów sąsiadujących z po-szukiwanym regionem). W związku z tym deskryptory budowane są wyłącznie w opar-ciu o punkty twarzy. Otoczenie pozostaje puste. W analizowanych obrazach, jednak,

nie jest ono puste. Wokół twarzy mogą wystąpić fragmenty ubrania, czy inne części ciała. W związku z tym odpowiadające sobie punkty z wzorca i obrazu mogą różnić się tym, że w tym pierwszym, część "kubelków" będzie miała wartość 0. Aby zminimalizować ich wpływ na zachowanie algorytmu dopasowującego zastosowano filtr AND, który wymusza, aby pod uwagę brane były jedynie te elementy histogramu, które w obu deskryptorach są niezerowe.

Przyjęto deskryptory o następujących parametrach (identycznych jak przy wyborze segmentu zawierającego twarz): skale logarytmiczne o rozdzielczości  $K = 10 \times 10$  i promień otoczenia  $r = 100$ . Przy dopasowywaniu użyto odległości *shape context measure*. Pozostałe parametry dobrano arbitralnie: krok 2:  $D_c = 0.5, S_c = 0.4$ , krok 6:  $D_c = 1.0, S_c = 0.4$ , krok 3:  $k = 3$ , wagi odpowiednio 5 dla twarzy i 1 dla tła. Zastosowano 3 wzorce twarzy zbudowane w oparciu o obrazy: *cara8 abajo*, *cara9 frontal2*, *cara10 risa*. Dla każdego z nich zbudowano 67 deskryptorów. Zostały one rozmieszczone metodą opartą o k-średnich.

Do klasyfikacji zastosowano sieć neuronową. W celu uniknięcia nadmiernego dopasowania (ang. *overfitting*) liczbę neuronów w warstwie ukrytej ustalono jedynie na 3. Sieć uczona jest przez 200 iteracji algorytmem wstępnej propagacji błędu, a następnie przez 50 iteracji metodą levenberga-marquadta.

Przy wyborze parametrów sieci i liczby deskryptorów czynnikiem decydującym był koszt obliczeniowy. Celem tej fazy analizy jest jedynie wyselekcjonowanie pewnego otoczenia nosa. Nie musi być ono bardzo dokładne. Przyjęto, że uzyskane rozwiązanie może być jedynie pewnym przybliżeniem twarzy. W celu poprawy zachowania algorytmu należałoby zwiększyć liczbę deskryptorów i/lub zastosować bardziej zaawansowany klasyfikator. Na rysunku 10.5 porównano zachowanie algorytmu dla 67 deskryptorów przy 3 neuronach ukrytych do 133 deskryptorów przy 5 neuronach ukrytych. Widać, że w drugim przypadku deskryptory zostały dopasowane znacznie lepiej.



Rysunek 10.5: Przykład działania algorytmu dla różnej liczby deskryptorów i różnych sieci.

#### 10.1.4 Analiza regionów twarzy

W celu oceny charakterystyk deskryptorów punktów w różnych regionach twarzy zaprojektowano następujący test:

1. Oblicz deskryptory dla wszystkich punktów obrazu 3D twarzy.
2. Dla każdego punktu  $p$  i odpowiadającego mu deskryptora:
  - a) oblicz koszt dopasowania do wszystkich pozostałych punktów  $q$  (deskryptorów):  $c_p(q)$ .
  - b) oblicz odległość punktu  $p$  do wszystkich pozostałych punktów  $q$ :  $d_p(q) = \|p - q\|$
  - c) dla każdej z odległości oblicz wagę. Obliczane są one z zastosowaniem funkcji gęstości rozkładu normalnego o środku w punkcie 0:

$$waga(d) = e^{-\frac{d^2}{2\theta^2}}$$

(gdzie:  $\theta$  - szerokość stosowanego okna), a następnie normalizowane do 1:

$$\sum_q waga(\|p - q\|) = 1$$

- d) oblicz sumę ważoną kosztów dopasowania:

$$c_p = \sum_q c_p(q) \cdot waga(\|p - q\|)$$

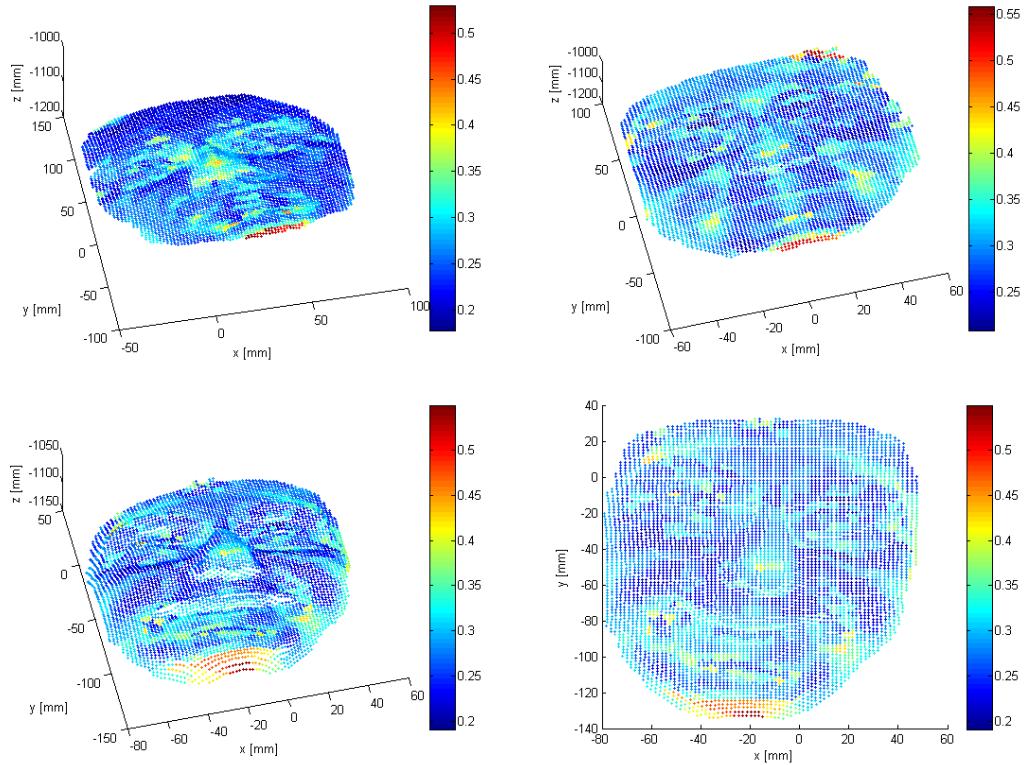
Test pokazuje jak bardzo dany punkt jest typowy dla swojego regionu. Punkty które mają bardzo podobne charakterystyki do swoich sąsiadów uzyskają niskie wartości  $c_p$ . Wysokie wartości  $c_p$  pozwalają na identyfikację regionów o nietypowej charakterystyce. Eksperyment przeprowadzono dla deskryptorów o następujących parametrach:

- promień otoczenia  $r = 100$ .
- skale logarytmiczne o rozdzielczości  $K = 10 \times 10$ .
- filtr dopasowywania OR
- odległość między deskryptorami w sensie *shape context measure*.

Wyniki dla okna o szerokości 40 dla 3 przykładowych plików przedstawiono na rysunku 10.6. Test wskazał miejsca na twarzy dla których deskryptory znacznie się wyróżniają. Obszary takie posłużyć mogą za granice regionów twarzy. Widoczne jest między innymi, że brzegi oraz okolice wierzchołka nosa są dobrze odróżnialne. W oparciu o to spostrzeżenie założono, że w kolejnych iteracjach algorytmu znajdowane będą: region nosa oraz niewielkie otoczenia punktów charakterystycznych.

### 10.1.5 Lokalizacja nosa

Do lokalizacji regionu nosa w twarzy zastosowano deskryptory o identycznych parametrach jak w sekcji 10.1.3. W oparciu o zestawienie z tabeli 2.1 zmniejszono jedynie promień otoczenia do  $r = 70$ . Tak dobrana wartość powinna powodować, że przy budowie deskryptorów na nosie uwzględniane będą punkty praktycznie z całej twarzy.



Kolor odpowiada wartości sumy ważonej kosztów dopasowania dla punktu.

Rysunek 10.6: Przykładowe wyniki analizy charakterystyk deskryptorów dla regionów.

Tablica 10.1: Parametry fazy lokalizacji nosa.

Krok algorytmu	Parametry
Filtrowanie w kroku 2	$D_c = 1.0, S_c = 0.5$
Reklasyfikacja w kroku 3	$k = 3$ , wagi: 2 / 1
Filtrowanie w kroku 6	$D_c = 0.4, S_c = 0.5$
Wybór punktów w analizowanym obrazie dla których zbudowane zostaną deskryptory	octClustering

Jednocześnie minimalizowana będzie liczba punktów spoza niej. Pozostałe parametry zawarto w tabeli 10.1.

Jako wzorce zastosowano 5 obrazów z bazy GavabDB. Na każdym z nich oznacono region twarzy  $Q_F$  i nosa  $Q_N$ . Następnie na punktach z  $Q_N$  i wykorzystując punkty z  $Q'_N = Q_F$  zbudowano w każdym z nich po 30 deskryptorów. Do dopasowywania deskryptorów zastosowano miarę SCM i filtr OR. W klasyfikacji punktów (krok 5 algorytmu opisanego w rozdziale 10.1.1) zdecydowano się na wykorzystanie zmodyfikowanego klasyfikatora k-najbliższych sąsiadów. Uznano, że w tej fazie ważniejszym od uogólniania jest precyzja wyboru. Dodatkowo krok 4 algorytmu (rysunek 10.4) - 'pogrubianie' redukuje większość negatywnych skutków stosowania klasyfikatora. Przyjęto liczbę sąsiadów  $k = 3$  oraz wagi punktów uczących odpowiednio 3 dla punktów przyporządkowanych do regionu nosa i 1 dla punktów odrzuconych.

Tablica 10.2: Parametry fazy lokalizacji regionów na nosie.

Krok/Parametr	Wartość
Rozdzielcość deskryptorów	$10 \times 10$
Typ skali	$\log \times \log$
Promień otoczenia $r$	$r = 70$
Filtr dopasowywania	OR
Koszt dopasowania	SCM
Filtrowanie w kroku 2	$D_c = 1.0, S_c = 0.3$
Reklasyfikacja w kroku 3	$k = 3$ , wagi: 5 / 1
Filtrowanie w kroku 6	$D_c = 0.7, S_c = 0.5$
Wybór punktów w analizowanym obrazie dla których zbudowane zostaną deskryptory	losowo

### 10.1.6 Lokalizacja regionów nosa

Ostatnią fazą hierarchicznej lokalizacji regionów jest wyszukiwanie obszarów wokół punktów charakterystycznych na nosie: prn, n, sn, lewe i prawe alare. Część parametrów dobranych na potrzeby algorytmu zaprezentowano w tabeli 10.2 Do klasyfikacji zastosowano zmodyfikowany klasyfikator k-najbliższych sąsiadów. Liczbę sąsiadów branych pod uwagę przy klasyfikacji ustalono na  $k = 3$ . Punktom uczącym przyporządkowany do wzorca (ang. *positive*) przypisano wagę 5. Punktom odrzuconym (ang. *negative*) przyporządkowano wagę 1.

Wzorce na potrzeby tej fazy przygotowano w sposób różniący się od faz wcześniejszych. W 5 obrazach należących do bazy GavabDB wyselekcjonowano punkty należące do twarzy ( $Q_F$ ). Następnie w tak zredukowanych danych oznaczono punkty charakterystyczne nosa: prn, n, sn, lewe i prawe alare. Z punktów obrazu wybrano następnie te które położone były w odległości mniejszej niż  $r_t = 5mm$  od powyższych punktów. Tak wybranych 5 podzbiorów danych posłużyło następnie za wzorce:  $Q_{prn}$ ,  $Q_{sn}$ ,  $Q_n$ ,  $Q_{all}$ ,  $Q_{alr}$ . Na każdym z wzorców wygenerowano po 10 deskryptorów. Punkty dla których zbudowano deskryptory wybrane zostały metodą opartą o k-średnich. W obliczeniach brany był pod uwagę obszar wcześniej oznaczony jako twarz -  $Q_F$ .

#### Ocena jakości wzorców

W celu oceny precyzji zaznaczania punktów charakterystycznych (a przez to jakości wygenerowanych wzorców) dla każdego wzorca zostały one oznaczone dwukrotnie. Policzono odległości między każdą parą tak powstałych punktów. Uśrednione pomiędzy pięcioma obrazami wyniki przedstawia tabela 10.3.

W celu oceny wpływu błędu oznaczania na zachowanie deskryptorów ( otoczenie  $r = 70mm$ , osie  $(10, \log) \times (10, \log)$  ) między każdą parą oznaczonych punktów (prn i prn', sn i sn', itd.) policzono odległość w sensie SCM. Uśrednione pomiędzy pięcioma obrazami wyniki przedstawia tabela 10.4.

Uzyskane rozbieżności wahają się między 0.01 a 0.06. W rozdziale 8.2.2 pokazano, że średni koszt dopasowania pary deskryptorów wygenerowanych na dwóch twarzach w odpowiadających sobie punktach (dla tak dobranych parametrów) wynosi około 0.06.

Tablica 10.3: Odległości między dwoma próbami oznaczenia punktu charakterystycznego nosa.

Punkt	Średnia rozbieżność [mm]	Odchylenie std. [mm]
prn	1.42	1.20
sn	2.13	1.01
n	2.94	2.34
lewe alare	2.55	1.71
prawe alare	2.73	1.83

Tablica 10.4: Koszt dopasowania (SCM) między dwoma próbami oznaczenia punktu charakterystycznego nosa.

Punkt	Koszt dopasowania	Odchylenie std.
prn	0.013	0.018
sn	0.015	0.012
n	0.050	0.087
lewe alare	0.031	0.042
prawe alare	0.037	0.037

Porównanie tych wartości pokazuje, że błąd wnoszony przez niedokładne oznaczenie punktów charakterystycznych może istotnie wpływać na jakość zbudowanych wzorców. Przy dalszym rozwoju systemu należy rozważyć inny sposób ich pozyskania.

Tablica 10.5: Czasy obliczeń kolejnych faz hierarchicznej lokalizacji regionów dla obrazów GavabDB.

Region	Średnia [s]	Odchylenie std. [s]
Twarz	25.74	6.99
Nos	46.96	8.99
Otoczenia punktów	52.86	7.66

## 10.2 Test fazy analizy

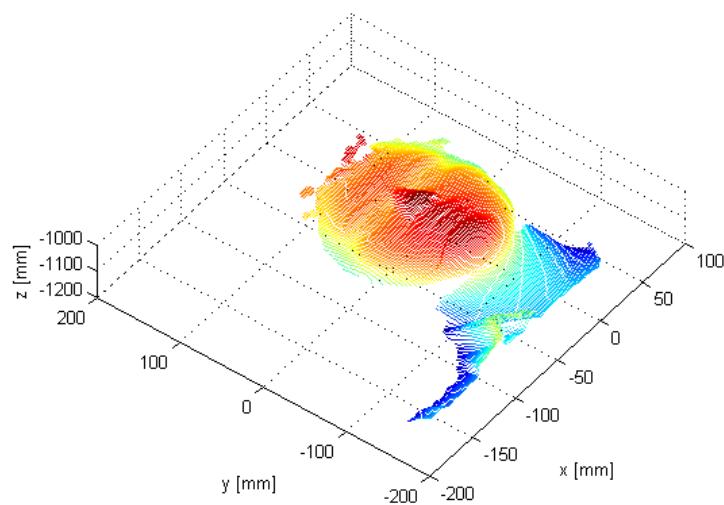
### 10.2.1 Dane testowe i czas obliczeń

W celu przetestowania zaprojektowanego systemu przygotowano testowy zbiór 25 obrazów 3 typów:

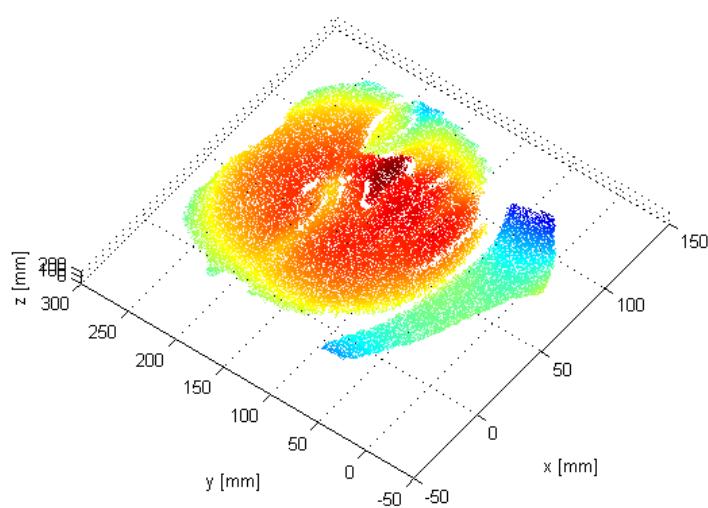
1. 15 obrazów z bazy GavabDB zawierających skany twarzy z przodu - frontal1, frontal2, arriba, abajo, risa (rysunek 10.7) - dane dobrej jakości pochodzące z tego samego zbioru w oparciu o które zbudowano bazę wzorców
2. 5 obrazów z bazy Mechatroniki (rysunek 10.8) - obrazy zawierają istotne nieciągłości w okolicach oczu i nosa
3. 5 obrazów z bazy GavabDB zawierających twarze z profilu - derecha, izquierda (rysunek 10.9)

Każdy z obrazów poddano wstępnej filtracji oraz wyborowi segmentu zawierającego twarz. Wynikowe dane poddano analizie algorytmem hierarchicznej lokalizacji regionów.

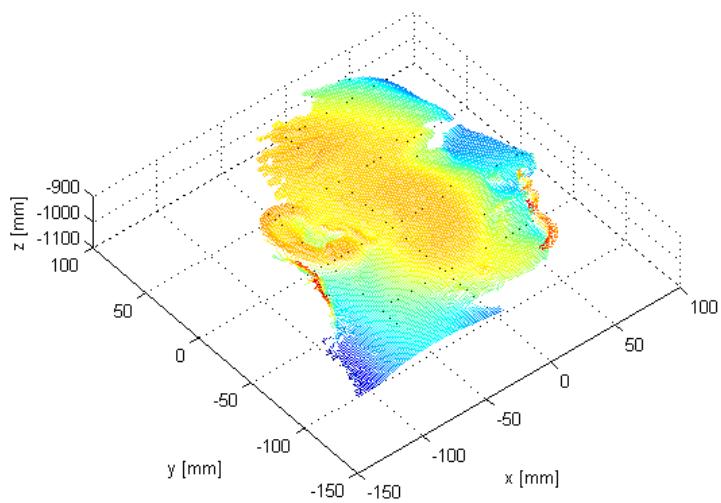
Wykres czasu analizy od liczby punktów obrazu pokazano na rysunku 10.10. Widoczna jest tendencja potwierdzająca wzrost czasu wykonania wraz ze wzrostem rozmiaru danych wejściowych. Jednak w obrębie obrazów o zbliżonej liczbie punktów przewidywanie czasu potrzebnego na obliczenia jest utrudnione. Spowodowane jest to zależnością zachowania się algorytmu od struktury danych wejściowych. Czas wykonania konkretnej fazy lokalizacji zależy od rezultatów uzyskanych w fazach wcześniejszych. Ocenę czasochłonności ułatwia tabela 10.5 w której zaprezentowano wyniki zachowania się algorytmu dla danych testowych z grup 1 i 3 (podobny rozmiar danych). Wyniki uzyskane dla lokalizacji twarzy są dwukrotnie niższe od uzyskiwanych dla nosa i otoczeń punktów. Powodem tego jest to, że punkty twarzy stanowią proporcjonalnie większe ułamek punktów sceny niż np. punkty otoczenia wierzchołka nosa punktów nosa. Generowanych jest znacznie mniej deskryptorów, więc spada koszt dopasowywania. Dzięki fazie wstępnego filtrowania i wyborze segmentu danych zredukowana została liczba deskryptorów, z których wybierane są deskryptory twarzy.



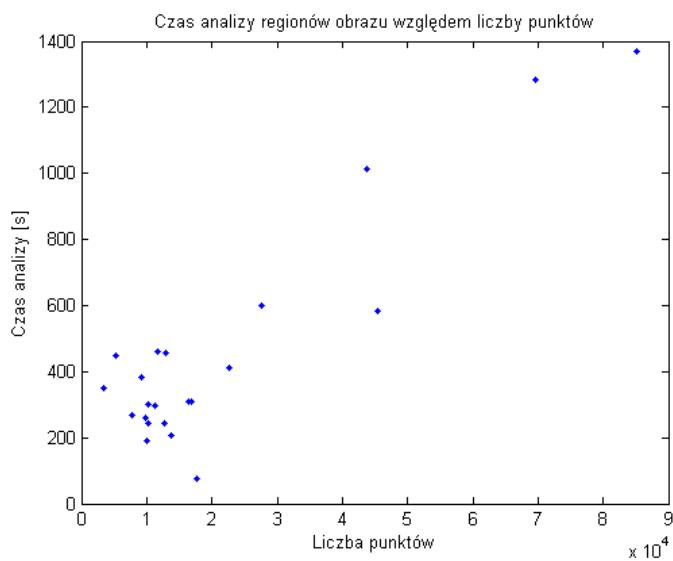
Rysunek 10.7: Przykładowy obraz z grupy 1 zbioru testowego.



Rysunek 10.8: Przykładowy obraz z grupy 2 zbioru testowego.



Rysunek 10.9: Przykładowy obraz z grupy 3 zbioru testowego.



Rysunek 10.10: Czas hierarchicznej lokalizacji regionów względem liczby punktów obrazu.

Tablica 10.6: Jakość klasyfikacji regionu twarzy.

Grupa	Dokładność (acc)		Miara $F_1$	
	Średnia	Odchylenie std.	Średnia	Odchylenie std.
1	86.23%	15.54%	90.21%	13.88%
2	79.79%	17.18%	87.30%	12.08%
3	69.88%	20.55%	78.81%	15.82%

### 10.2.2 Test lokalizacji twarzy

Aby przetestować jakość algorytmu lokalizującego twarz w obrazach testowych manualnie oznaczono punkty należące i nie należące do poszukiwanego regionu. Tak zaklasyfikowane dane porównano z wynikami automatycznej klasyfikacji. Każdy z punktów analizowanej chmury przyporządkowany został do jednej z grup:

- A (true positive) - punkty w obu przypadkach zaklasyfikowane do poszukiwanego regionu
- B (false negative) - punkty manualnie oznaczone jako należące do regionu, ale odrzucone w automatycznej klasyfikacji
- C (false positive) - punkty manualnie zaklasyfikowane jako nie należące do regionu, ale uznane za twarz w wyniku działania programu
- D (true negative) - punkty w obu przypadkach odrzucone

W oparciu o licznosći powyższych grup dla każdego z obrazów obliczone zostały miary opisujące jakość klasyfikacji:

- dokładność:  $accuracy = \frac{|A|+|D|}{|A|+|B|+|C|+|D|}$
- miara  $F_1$ :  $F_1 = 2 \frac{precision \cdot recall}{precision + recall}$ , gdzie:  $precision = \frac{|A|}{|A|+|C|}$ ,  $recall = \frac{|A|}{|A|+|B|}$

Uśrednione wyniki dla grup przedstawiono w tabeli 10.6. Biorąc pod uwagę założenia o zgrubnym dopasowaniu twarzy poczynione przy doborze parametrów oraz niską jakość danych testowych (ręczna klasyfikacja punktów siłą rzeczy cechuje się niską precyzją) uzyskane wyniki uznać należy za bardzo dobre. Zgodnie z przewidywaniami wyniki dla grupy 1 są najlepsze. Nieznacznie gorsze są rezultaty dla grupy 2, co spowodowane jest gorszą jakością danych. Rezultaty uzyskiwane dla grupy 3 są jeszcze niższe. Powodem tego jest fakt, że baza wzorców zawiera wyłącznie obrazy *en face* i dla takiego też typu danych projektowane były algorytmy.

Miarą, która dużo mówi o precyzyji lokalizacji regionu jest odległość między faktycznym środkiem ciężkości (ang. centroid) regionu, a środkiem ciężkości obszaru zaklasyfikowanego jako region. Wykorzystując punkty wcześniej oznaczone ręcznie jako twarz obliczono centroidy kolejnych obrazów testowych. Następnie policzono centroidy regionów które uznane zostały za twarz w wyniku działania algorytmu. Informację o obliczonych odległościach między tymi punktami zawarto w tabeli 10.7. Jak widać wyniki uzyskane dla grup 1 i 2 są średnio poniżej 2 cm. Biorąc pod uwagę wymiary twarzy jest to rezultat bardzo dobry. Duża wartość odchylenia standardowego spowodowana jest wpływem pojedynczych przypadków, kiedy skuteczność algorytmu dla ustalonych

Tablica 10.7: Odległości między środkami ciężkości wzorca regionu twarzy i wynikami klasyfikacji.

Grupa	Odległość średnia [mm]	Odchylenie std. [mm]
1	12.09	17.44
2	15.16	17.01
3	27.22	20.25

Tablica 10.8: Sumaryczne koszty dopasowania otrzymywane w fazie lokalizacji nosa.

Grupa	Średnia	Odchylenie std.
1	9.96	1.60
2.1	11.88	0.74
2.2	15.20	0.88
3	17.98	1.55

wcześniej parametrów okazała się niewystarczająca. **Kosztem wydłużenia obliczeń, możliwa jest poprawa jakości klasyfikacji.** Proponowanym ulepszeniem jest np. zwiększenie liczby deskryptorów. Zachowanie programu uznać można za niesatysfakcjonujące dla jednego przypadku z grupy 1 (odległość centroidów 72.92mm) i jednego z grupy 2 (44.79mm). Wyniki dla grupy 3 służą jedynie jako dane porównawcze.

W celu sprawdzenia heurystyki ograniczającej obszar przeszukiwania twarzy w celu znalezienia nosa do części środkowej na obrazach testowych manualnie oznaczono punkty należące i nie należące do nosa. Następnie sprawdzono jaki ułamek tych punktów znalazł się w obszarze centralnym  $T_C$ . Dla 14 obrazów z 1 grupy było to 100%. Dla 1 obrazu było to 24.38%. Powodem takiego wyniku było błędne zaklasyfikowanie twarzy. Dla grupy 2 ułamek ten wyniósł średnio 86% przy odchyleniu standardowym 16%. Obrazy grupy 3 są sprzeczne z poczynionym wcześniej założeniem o tym, że obraz zawiera całą twarz, w związku z tym uzyskana wartość to jedynie 50% przy odchyleniu standardowym 44%.

### 10.2.3 Test lokalizacji nosa

Do stwierdzenia z dużą skutecznością czy w analizowanych danych znajduje się region nosa wykorzystać można sumaryczny koszt dopasowania uzyskany w fazie lokalizacji nosa. Wyniki uzyskiwane dla poszczególnych grup zawiera tabela 10.8. Najniższe koszty uzyskano dla grupy 1. Nieznacznie wyższe dla grupy 2.1 - obrazy z grupy 2 w których całość nosa znalazła się w analizowanym obszarze. Koszty znacznie wyższe otrzymywano dla grupy 2.2 - mniej niż połowa nosa i 3 - profile.

Analogiczną analizę do wcześniejszej, dotyczącej klasyfikacji twarzy, przeprowadzono również dla nosa. Pod uwagę brane były wyłącznie te punkty, które wcześniej oznaczono jako należące do twarzy. Część z nich przypisano do regionu nosa. Następnie porównane one zostały z tymi które zaklasyfikowane zostały przez algorytm jako należące do nosa. Część z punktów oznaczonych przez algorytm jako należących do nosa mogło znaleźć się w miejscu które nie należy do twarzy. Odnotowano 2 takie przypadki w grupie 3. W grupach 1 i 2 problem nie wystąpił. Punkty takie nie są brane pod uwagę, gdyż takie zachowanie uznano za wynikające z błędu klasyfikacji wyższego poziomu. Dodatkowo, w wyniku klasyfikacji punktów dla 1 przypadku w grupie 1 (opisany wyżej przypadek

Tablica 10.9: Jakość klasyfikacji regionu nosa.

Grupa	Dokładność (acc)		Miara $F_1$	
	Srednia	Odchylenie std.	Srednia	Odchylenie std.
1	96.56%	0.71%	79.32%	3.88%
2	94.58%	1.15%	63.91%	10.26%
3	94.24%	2.32%	60.97%	18.21%

Tablica 10.10: Odległości między środkami ciężkości wzorca regionu nosa i wynikami klasyfikacji.

Grupa	Odległość średnia [mm]	Odchylenie std. [mm]
1	3.95	1.69
2	8.32	4.39
3	15.74	7.37

błędnej klasyfikacji twarzy) nie odnaleziono nosa (nie zaklasykowano żadnego punktu jako należącego do regionu). Podobnie stało się dla jednego przypadku w grupie 2 oraz 3 z 5 przypadków w grupie 3. Obliczone dla pozostałych danych wskaźniki przedstawiono w tabeli 10.9.

Dla 14 przypadków z grupy 1 dla których udało się odnaleźć nos obliczono odległość między środkami ciężkości: rzeczywistym (obliczonym na podstawie danych oznaczonych manualnie) i odnalezionym przez algorytm. Analogicznych obliczeń dokonano dla 4 przypadków z grupy 2 i 2 przypadków z grupy 3. Wyniki zawarto w tabeli 10.10. Na tym poziomie klasyfikacji coraz bardziej widoczny staje się wpływ jakości stosowanych danych testowych. Wyniki dla grupy 2 są średnio ponadwukrotnie gorsze od tych dla grupy 1. Spowodowane jest to dużymi uszczerbkami danych w okolicach nosa.

#### 10.2.4 Wpływ klasyfikacji nosa na lokalizację otoczeń punktów charakterystycznych

W celu przeprowadzenia analiz dotyczących regionów wokół punktów na nosie w obrazach z grup 1 i 2 oznaczono ręcznie: prn, n, sn, lewe i prawe alare. We wszystkich obrazach z grupy 2 w otoczeniach skrzydełek nosa zanotowano znaczące uszczerbki w powierzchni uniemożliwiające zidentyfikowanie części punktów. W związku z tym zamiaszt nich oznaczono punkty najbliższe ich oczekiwanej położeniu.

Tak przygotowane dane zastosowano do przeprowadzenia testu wpływu jakości klasyfikacji nosa na lokalizację regionów wokół punktów charakterystycznych. Sprawdzono czy ręcznie oznaczone punkty prn, n, sn, al w danych zawierają się w regionie zaklasyfikowanym przez algorytm jako nos -  $T_N$ :

$$prn, n, sn, alare_{lewe}, alare_{prawe} \in T_N$$

. W przypadku punktów prn, n, sn lewe alare stało się tak we wszystkich 14 przypadkach w grupie 1 dla których znaleziono nos. Prawe alare zawierało się w punktach zaklasyfikowanych jako nos w 12 z 14 przypadków. Odległości od najbliższego z punktów zaklasyfikowanych jako punkty należące do  $T_N$ : ( $|al_{prawe} - T_N|$ ) wyniosły 4.98mm i 3.16mm. Są to wyniki tego samego rzędu co odchylenia uzyskiwane przy ręcznym oznaczaniu

punktów. W 4 obrazach grupy drugiej (w których zaklasyfikowano nos) wierzchołek nosa (prn) znalazł się w  $T_N$  we wszystkich przypadkach. Podobnie jak. Punkty n i sn znalazły się poza  $T_N$  jednokrotnie (odległości 9.91mm i 5.47mm). Prawe alare znalazło się poza  $T_N$  w dwóch przypadkach. Odległości wyniosły: 9.61mm, 3.08mm.

Dla każdego ze zidentyfikowanych punktów zdefiniowano otoczenie. Składają się na nie te punkty obrazu, których odległość od wybranego punktu nosa (prn, n, sn, alare) jest mniejsza niż 5mm. Dla 14 obrazów z grupy 1 i 4 obrazów z grupy 2, w których udało się zaklasyfikować nos, obliczono jaki ułamek otoczenia punktów charakterystycznych znalazł się w punktach zaklasyfikowanych jako nos. Wyniki przedstawiono w tabeli 10.11.

Rezultaty powyższych analiz pokazały, że mimo dobrych wyników, faza klasyfikacji nosa może mieć negatywny wpływ na jakość lokalizacji punktów charakterystycznych nosa. Tabela 10.11 pokazuje, że jest ona zbyt restrykcyjna tj. region  $T_N$  w którym przeprowadzana jest dalsza lokalizacja jest często zbyt mały. Możliwym rozwiązaniem korygującym ten problem w dalszych zastosowaniach, jest zmiana parametru heurystyki 'pogrubiania' wypełniającej nieciągłości w zlokalizowanym regionie nosa. Jego zwiększenie spowodowałoby powiększenie obszaru w którym wyszukiwane są kolejne podregiony. To z kolei zredukowało prawdopodobieństwo odrzucenia istotnych regionów. Niestety, mogłoby też spowodować wydłużenie czasu obliczeń i zwiększenie prawdopodobieństwa błędnego dopasowania deskryptorów. Drugi z wymienionych negatywnych czynników można jednak zbalansować zmianą parametrów algorytmu np. zwiększając liczbę deskryptorów.

Tablica 10.11: Ułamek otoczenia punktów charakterystycznych zawarty w punktach zaklasyfikowanych jako należące do nosa.

Grupa		prn	n	sn	alare lewe	alare prawe
1	Średnio	100%	95.36%	90.08%	95.92%	94.05%
	Odchylenie std.	0%	13.05%	18.74%	15.27%	18.03%
2	Średnio	100%	61.79%	53.43%	86.84%	55.72%
	Odchylenie std.	0%	48.14%	54.07%	22.57%	50.43%

### 10.2.5 Test lokalizacji otoczeń punktów charakterystycznych

Przeprowadzono test jakości lokalizacji regionów dla otoczenia punktów charakterystycznych na nosie. Za miarę jakości przyjęto odległość między ręcznie oznaczonym punktem na nosie, a środkiem ciężkości punktów uznanych przez algorytm za należące do regionu. Dla 14 obrazów z grupy 1 obliczono odległości poszczególnych punktów. Dla 2 obrazów nie udało się zlokalizować otoczenia punktu sn (w wyniku klasyfikacji nie wybrano żadnych punktów). Rozkład uzyskiwanych wartości odległości od punktu ręcznie oznaczonego, dla pozostałych przypadków, zaprezentowano w tabeli 10.12. Zdecydowanie najlepiej odnajdowane jest otoczenie punktu prn. Większość odległości znalazła się w przedziale odpowiadającym najlepszemu dopasowaniu. Taki wynik potwierdza tezę, że punkt ten jest najbardziej charakterystyczny (co widać też w wynikach testu opisanego w rozdziale 10.1.4). Nieznacznie gorsze wyniki otrzymano dla punktu n. W jednym przypadku otrzymano rezultat zupełnie błędny. Dla punktu sn w 3 przypadkach otrzy-

Tablica 10.12: Rozkład odległości między punktami charakterystycznymi, a środkami regionów w obrazach grupy 1 (GavabDB).

Punkt	Liczba obrazów	Przedział odległości [mm]					
		0-3	3-6	6-10	10-20	20-40	40-80
prn	14	71.43%	21.43%	7.14%	0.00%	0.00%	0.00%
n	14	42.86%	42.86%	7.14%	0.00%	0.00%	7.14%
sn	12	25.00%	33.33%	16.67%	16.67%	0.00%	8.33%
<i>al_lewe</i>	14	7.14%	28.57%	35.71%	21.43%	7.14%	0.00%
<i>al_prawe</i>	14	0.00%	35.71%	28.57%	28.57%	7.14%	0.00%

W komórkach zawarto ułamek obrazów należący do przedziału.

Przedziały lewostronnie domknięte.

Tablica 10.13: Rozkład odległości między punktami charakterystycznymi, a środkami regionów w obrazach grupy 2 (Mechatronika).

Punkt	Liczba obrazów	Przedział odległości [mm]					
		0-3	3-6	6-10	10-20	20-40	40-80
prn	4	2	1	1	0	0	0
n	4	2	0	0	0	0	2
sn	2	1	0	0	0	0	1
<i>al_lewe</i>	4	0	0	0	1	3	0
<i>al_prawe</i>	4	0	1	0	1	2	0

W komórkach zawarto liczbę obrazów przypisanych do przedziału.

Przedziały lewostronnie domknięte.

mano rozbieżności powyżej 1cm. Dla punktów prawego alare i lewego alare uzyskano odpowiednio skuteczność ponad 70% i ponad 65% punktów o rozbieżności poniżej 1cm.

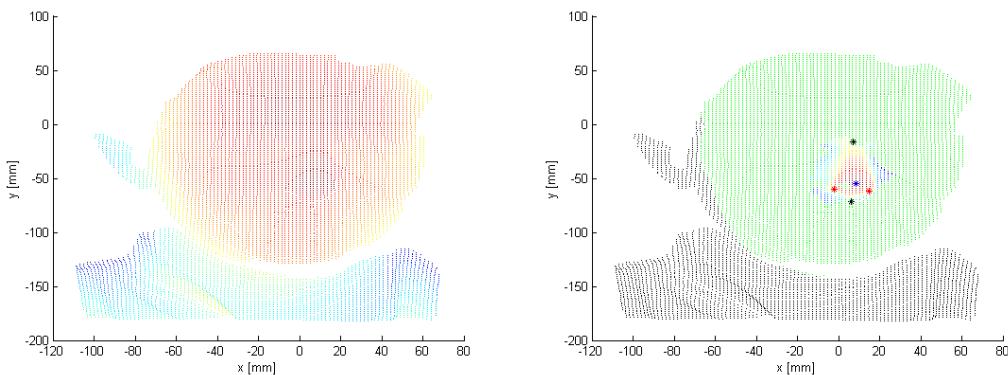
Podobne analizy przeprowadzono dla 4 obrazów z grupy 2. Wyniki przedstawiono w tabeli 10.13. Ponownie nie udało się zlokalizować 2 regionów sn. Algorytm najskuteczniej zachował się dla punktu prn. Wyniki dla n i sn są połowiczne. Najgorzej algorytm poradził sobie z lokalizacją punktów *al\_lewe* i *al\_prawe*, czego prawdopodobnym powodem były opisywane wcześniej błędy w danych.

### 10.2.6 Przykładowe rezultaty

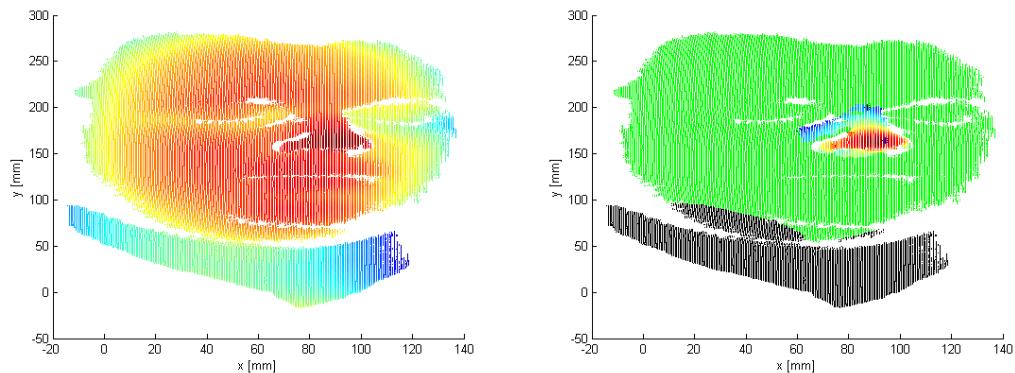
Przykładowe wyniki wykonania wszystkich faz analizy danych przedstawiono na rysunkach 10.11, 10.12, 10.13. Rysunek z lewej strony pokazuje dane wejściowe (obraz wyjściowy fazy wstępnego przetwarzania z wyborem segmentu). Po prawej stronie pokazano wynik analizy. Kolorem czarnym oznaczono regiony odrzucone. Zielony przypisany został do punktów uznanych za należące do twarzy. Kolor punktów nosa odpowiada kolorowi odpowiadającemu głębi (wymiar *z*). Na rysunkach oznaczono również środki regionów lokalizowanych na nosie. Kolorem niebieskim oznaczono wierzchołek nosa (prn). Na czerwono oznaczono punkty na skrzydełkach nosa (alare). Czarny kolor odpowiada punktom nasion i subnaasale.

Algorytm analizy najlepiej radzi sobie z obrazami przedstawiającymi widok twarzy z przodu. Dla takich danych został on zaprojektowany i tak dobierane były parametry. Na jego skuteczność istotnie wpływa jakość danych. Rysunek 10.12 pokazuje, że punkt prawego alare zidentyfikowany został błędnie. Powodem tego był nadmierny uszczerbek

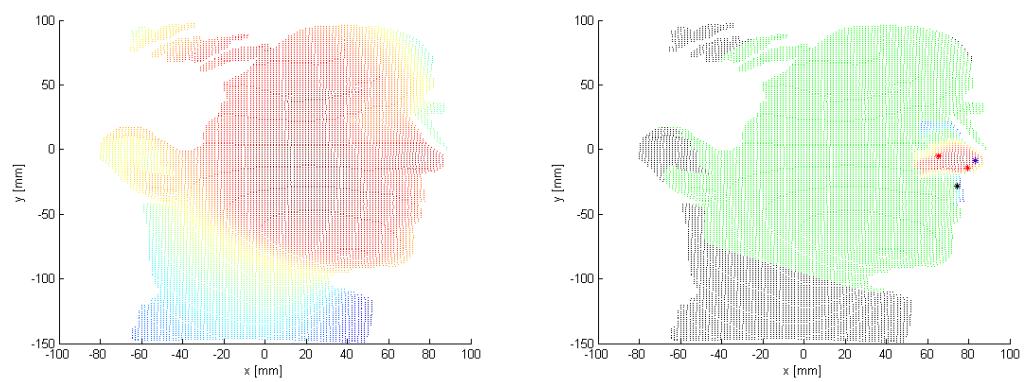
w danych, uniemożliwiający poprawne zlokalizowanie regionu. Rysunek 10.13 ukazuje zachowanie się algorytmu dla obrazów, które nie zawierają pełnej informacji o twarzy (obraz z profilu). Nawet dla części takich danych udaje się wykonać fragmentaryczną analizę. Region nosa zlokalizowany został poprawnie. Dość dobrze odnalezione zostały otoczenia punktów prn, lewego alare i subnaasale.



Rysunek 10.11: Przykład działania algorytmu analizy na obrazie en face z GavabDB.



Rysunek 10.12: Przykład działania algorytmu analizy na obrazie en face z Mechatroniki.



Rysunek 10.13: Przykład działania algorytmu analizy na obrazie profilu z GavabDB

## Rozdział 11

---

# Szczegóły implementacyjne

---

### 11.1 Konwertery danych

Obrazy 3D z bibliotek GavabDB i Mechatroniki dostarczone zostały odpowiednio w formatach VRML i chbin. W celu unifikacji i uproszczenia ładowania danych do *Matab-a*. stworzono programy konwertujące do plików tekstowych. Są to *ChBinReader* i *WlReader*. Napisano je w środowisku *Microsoft Visual Studio 2005* w języku C. Oczekiwany wynikiem wykonania programów są pliki zawierające w kolejnych wierszach współrzędne kolejnych punktów chmury punktów  $[x, y, z]$  oddzielonych tabulatorem.

#### Format chbin

Format chbin został opracowany dla systemu OGX na potrzeby przetwarzania w języku C. Plik w tym formacie zawiera zapisane jednym ciągiem następujące po sobie struktury danych:

- `sCloudsBinaryHeader_v10` - nagłówek pliku.
- `sSingleCloudBinaryHeader_v10` - nagłówek pojedynczej kierunkowej chmury punktów.
- `sCloudPoint_v10` - reprezentacja pojedynczego punktu danych.

Struktury te (jak i ich podstruktury wewnętrzne) przedstawiono poniżej:

```
typedef struct
{
    float fX_Max,fX_Min;
    float fY_Max,fY_Min;
    float fZ_Max,fZ_Min;
    SORT_STATE_v10 iSorted;
    unsigned int uiCount;
    unsigned int uiSelectedPoint;
    char pcCloudName[256];
```

```
sPointRGB_v10 colNormal;
float fNormalSize;
int iDrawCloudRGB;
sPointRGB_v10 colSelected;
float fSelectedSize;
int iDarkMode;
int iHidden;
} sSingleCloudBinaryHeader_v10;

typedef struct
{
    unsigned int uiFileFormatID_Major;
    unsigned int uiFileFormatID_Minor;
    unsigned int uiCloudCount;
    unsigned int uiAllPointsCount;
    float fX_Max,fX_Min;
    float fY_Max,fY_Min;
    float fZ_Max,fZ_Min;
    float m[16];
    sPointXYZ_v10 sRotationCenter;
    int iCloudWithSelectedPoint;
} sCloudsBinaryHeader_v10;

typedef struct
{
    float x,y,z;
    unsigned char r,g,b;
    unsigned char state;
} sCloudPoint_v10;

typedef enum
{
    SORT_NONE,
    SORT_X,
    SORT_Y,
    SORT_Z
} SORT_STATE_v10;

typedef struct
{
    float x,y,z;
} sPointXYZ_v10;

typedef struct
{
```

```
    unsigned char r,g,b;  
} sPointRGB_v10;
```

### ChBinReader

Programem konwertującym plik w formacie chbin do pliku tekstowego jest *ChBinReader*. Jest to program konsolowy - wykonywany z wiersza poleceń. Wymaga podania co najmniej jednego parametru: ścieżki do pliku wejściowego. W przypadku gdy parametr nie został podany wyświetlany jest komunikat o błędzie. Drugim (opcjonalnym argumentem) jest ścieżka do pliku wynikowego. W przypadku jej niepodania zostaje ona ustalona automatycznie na identyczną ze ścieżką pliku wejściowego. Zmieniane jest jedynie rozszerzenie pliku na .txt. Poprawnie wykonany program zwraca kod 0. W przypadku błędu zwracany jest ujemny kod błędu oraz do standardowego strumienia błędów (stderr) wysyłany jest komunikat.

### WrlReader

*WrlReader* jest konwerterem, który przekształca pliki w formacie VRML z bazy GavabDB na pliki tekstowe. Program działa w środowisku wiersza poleceń. Jego zachowanie jest analogiczne do działania programu *ChBinReader*. Obowiązkowym parametrem jest ścieżka do pliku VRML. W przypadku nie podania drugiego argumentu - ścieżki do pliku wynikowego - jest on generowany automatycznie w oparciu o pierwszy parametr. Zmieniane jest jedynie rozszerzenie na .txt. W przypadku poprawnego zachowania zwracany jest kod 0. Błędy powodują zwrócenie kodów ujemnych oraz przekazanie do strumienia błędów odpowiednich komunikatów.

*WrlReader* nie implementuje pełnej obsługi formatu VRML. Odzyskiwana jest jedynie informacja o koordynatach zapisanych punktów. Plik wejściowy przetwarzany jest wierszowo, aż do napotkania sygnatury *point*, która rozpoczyna sekwencję współrzędnych punktów. Są one kolejno kopowane do pliku wynikowego. Przerwanie przenoszenia danych powoduje sygnatura końca sekwencji ("[").

## 11.2 Środowisko

Docelowe rozwiązanie zostało zaimplementowane w środowisku *Matlab 7.6.0 (2008a)*. Za wykorzystaniem tego narzędzia stał szereg czynników:

- łatwość operowania na dużych strukturach liczbowych typu macierze i wektory (dodatekowo środowisko dostarcza wygodnego języka przetwarzania, dzięki któremu procedury przetwarzania całych chmur punktów zapisano w prosty i przejrzysty sposób).
- możliwość integracji z kodem w języku Java. Dzięki tej funkcjonalności metody filtrowania przez segmentację w grupy połączone oraz *octClustering*, które muszą zostać zaimplementowane rekurencyjnie bądź z wykorzystaniem jawniej iteracji (pętle *for*) wykonywane są efektywnie, na co nie pozwalałaby pełna implementacja tych procedur w środowisku *Matlab*.

- znaczna liczba wbudowanych pakietów narzędziowych (ang. Toolbox) zwłaszcza do sztucznej inteligencji i statystyki. Wykorzystane zostały *Statistics Toolbox*<sup>TM</sup> (metoda kmeans - segmentacja metodą k-średnich) oraz *Neural Networks Toolbox*<sup>TM</sup> (perceptron wielowarstwowy w algorytmie lokalizacji regionów).
- wsparcie dla metodologii RAD (ang. *Rapid Application Development*), która pozwoliła na błyskawiczne utworzenie i rozwój potrzebnych aplikacji narzędziowych.
- bogaty zbiór zewnętrznych bibliotek i procedur udostępnianych dla celów naukowych. W rozwiązaniu wykorzystano zewnętrzną implementację algorytmu węgierskiego. Udostępniania jest ona na licencji BSD pod adresem: <http://www.mathworks.co.kr/matlabcentral/fileexchange/20328>.
- szeroki zakres technik wizualizacji danych: wykresy 2D i 3D, histogramy.

### 11.3 Organizacja kodu

154 pliki zawierających kod *Matlab* podzielono na następujące kategorie (katalogi):

1. zAi - procedury dotyczące sztucznej inteligencji: sieci neuronowe, k-nn, *octClustering*, segmentacja w grupy połączone.
2. zArea - ocena gęstości powierzchniowej i powierzchni obiektów w obrazach 3D.
3. zDescriptors - obliczenia dotyczące deskryptorów: obliczenie obrazów obrotu, znormalizowanie histogramu, policzenie odległości między deskryptorami, filtrowania, zbudowanie macierzy kosztów oraz wybór podzbioru punktów różnymi metodami.
4. zDescInterface - zestaw nakładek (ang. *wrappers*) upraszczających użycie procedur dotyczących deskryptorów.
5. zGenerate - generacja prostych obrazów 3D (w celach testowych).
6. zGui - wizualizacja danych, wykresy oraz paski postępu przy obliczeniach.
7. zIo - obsługa plików: zapis i odczyt obrazów 3D oraz deskryptorów.
8. zMatching - procedury powiązane z dopasowywaniem punktów m.in. algorytm węgierski.
9. zNeighbors - obsługa wyszukiwania sąsiadów (punktów w otoczeniu).
10. zNormals - obliczanie kierunków wektorów normalnych do powierzchni.
11. zPre - procedury pomocnicze wstępnego przetwarzania danych.
12. zSpecialized - kod dotyczący analizy konkretnych cech twarzy m.in. podział nosa na część lewą i prawą, wybór centralnej części twarzy, ocena wiarygodności użytych odległości na nosie .
13. zTests - skrypty służące testowaniu i analizie.

14. zToolbox - procedury pomocnicze, które nie pasują do żadnej z pozostałych kategorii.
15. zTools - narzędzia wraz z interfejsem użytkownika do analizy zachowania algorytmów.
16. zUser - najbardziej zewnętrzne procedury będące implementacją konkretnych algorytmów: przetwarzania wstępnego, detekcji segmentu twarzy, lokalizacji regionów.

Dodatkowo załączono skrypt `setupEnv.m`, który powinien zostać wywołany przed użyciem pozostałych procedur. Jego zadaniem jest konfiguracja ścieżek katalogów w środowisku *Matlab*. Kod w języku Java wraz ze skompilowaną biblioteką *jar* zamieszczono w katalogu *java*. Wzorce dołączono w katalogu *patterns*.

Wewnętrznie przetwarzanie i analizę danych oparto o użycie wbudowanych w język *Matlab* macierzy i wektorów. Chmury punktów oraz informacja o kierunkach wektorów normalnych reprezentowane są w postaci macierzy  $N \times 3$ . W pojedynczym wierszu znajdują się: koordynaty jednego punktu w kolejności  $[x, y, z]$  lub współrzędne wektora normalnego  $[n_x, n_y, n_z]$ . Podobnie zbiór obrazów obrotu obliczonych dla konkretnej chmury punktów jest reprezentowany i przechowywany w postaci macierzy  $M \times K$ . W pojedynczym wierszu znajduje się zserializowany histogram przypisany do konkretnego punktu. Numery wybranych punktów przechowywane są w oddzielnym (pionowym) wektorze o rozmiarze  $M$ . W takiej samej postaci, w jakich dane są przetwarzane, są też przechowywane w plikach (do utrwalania wyników pracy użyto plików tekstowych).

## 11.4 Przegląd kluczowych procedur

### 11.4.1 findSubPatterns

Nagłówek:

```
[subPts bgPts fgSeedPts bgSeedPts, p1 p2 p3 p4, ...
matchingCost subMask] = findSubPatterns(pts, areas, descriptors, options)
```

Katalog: zUser

**Opis:** Funkcja znajduje w zbiorze punktów obrazu 3D podzbiór możliwie zgodny z wzorcem danym w postaci zestawu deskryptorów. Implementuje algorytm hierarchicznej lokalizacji regionów opisany w rozdziale 10.1.1.

Parametry:

- pts - chmura punktów  $N \times 3$  (obraz 3D) w której poszukiwany jest wzorzec ( $otoczenie(T')$ ).
- areas - wektor powierzchni wzorców ( $A_Q$ ).
- descriptors - ścieżki plików zawierających deskryptory wzorców (w postaci komórek: jedna komórka zawiera jedną ścieżkę do pliku).

- options - struktura zawierająca konfigurację zachowania się procedury. Obsługiwane pola (w przypadku braku pola używana jest wartość domyślna) to:
  - parametry generacji deskryptorów:
    - \* spinDistance - promień otoczenia ( $r$ )
    - \* alfaBins/betaBins - rozdzielczości skali
    - \* alfaAxes/betaAxes - typy skali ('lin'/'log')
  - parametry dotyczące dopasowywania deskryptorów:
    - \* matchingRule - filtr ('OR'/'AND'/'NONE')
    - \* matchingDistanceType - typ stosowanej odległości (1 - SCM, 2 - korelacja liniowa)
  - roiMask (ang. *region of interest*) - wektor binarny (maska bitowa) o długości równej liczbie punktów:  $|roiMask| = |pts|$ . Służy on identyfikacji punktów które należą do przeszukiwanego regionu i dla których podzbioru wygenerowane zostaną deskryptory (tj.  $T'$ ). Domyślnie jest on wypełniony jedynkami.
  - useNoSeedPts - sumaryczna dla wszystkich wzorców liczba deskryptorów do wygenerowania. Domyślnie 150.
  - classifier - typ stosowanego klasyfikatora (krok 4). Możliwe wartości: 'nn' – sieć neuronowa, 'knn' – k-najbliższych sąsiadów, 'wknn' – ważone k-nn. Domyślnie: 'nn'.
  - classifierOptions - struktura zawierająca parametry specyficzne dla danego klasyfikatora (zbudowanego w kroku 4 algorytmu). Obsługiwane pola struktury:
    - \* nnClassify - sieć neuronowa:
      - hiddenNeurons - liczba neuronów warstwy ukrytej. Domyślnie: 5
      - bpEpochs - liczba epok uczenia metodą wstępnej propagacji błędów. Domyślnie: 250.
      - lmEpochs - liczba epok uczenia metodą l-m. Domyślnie: 100.
    - \* knnClassify - klasyfikator k-nn:
      - k - liczba sąsiadów branych pod uwagę w głosowaniu. Domyślnie: 5.
    - \* weightedKnnClassify - ważone k-nn:
      - k - liczba sąsiadów, domyślnie: 5.
      - positiveWeight - waga klasy pozytywnej, domyślnie: 2.
      - negativeWeight - waga klasy negatywnej, domyślnie: 1.
  - parametry filtrowania wyników klasyfikacji (krok 6):
    - \* classifierOptions.connectivityCoeff = Dc, domyślnie: 0.7.
    - \* classifierOptions.clusterFraction = Sc, domyślnie: 0.4.
  - parametry wywołań procedur wewnętrznych:
    - \* connectionFiltering (krok 2 algorytmu) - filtrowanie przez połączenie punktów które przyporządkowane zostały do punktów wzorców:

- connectivityCoeff = Dc, domyślnie: 0.7.
- clusterFraction = Sc, domyślnie: 0.4.
- \* weightedKnnClassify (krok 3 algorytmu) - klasyfikator stosowany do ponownej klasyfikacji punktów które nie zostały dopasowane do wzorców:
  - k - liczba sąsiadów, domyślnie: 5.
  - positiveWeight - waga punktów dopasowanych, domyślnie: 2.
  - negativeWeight - waga punktów niedopasowanych, domyślnie: 1.

Wartości zwracane:

- subPts - punkty zaklasyfikowane jako należące do wzorca ( $F$ )
- bgPts - punkty zaklasyfikowane jako nienależące do wzorca
- fgSeedPts - punkty uczące (takie dla których wygenerowano deskryptory) dopasowane w wyniku wykonania algorytmu węgierskiego do wzorców ( $D_T$ ).
- bgSeedPts - punkty uczące, które nie zostały dopasowane do wzorców. Posłużyły one identyfikacji otoczenia poszukiwanego regionu - tła ( $D_{T''}$ ).
- p1 = bgSeedPts.
- p2 - zbiór punktów uczących zaklasyfikowany jako tło, ale odrzuconych w filtrowaniu.
- p3 = fgSeedPts.
- p4 - zbiór punktów uczących zaklasyfikowanych jako odpowiadające wzorcowi, ale odrzuconych po analizie spójności.
- matchingCost - sumaryczny koszt dopasowania deskryptorów wzorców do deskryptorów obrazu analizowanego.

#### 11.4.2 `findFace`, `findNose`, `findPoint`

Nagłówki:

```
[subPts bgPts fgSeedPts bgSeedPts, p1 p2 p3 p4, ...
matchingCost subMask] = findFace(pts)
```

```
[subPts bgPts fgSeedPts bgSeedPts, p1 p2 p3 p4, ...
matchingCost subMask] = findNose(pts, roiMask)
```

```
[subPts bgPts fgSeedPts bgSeedPts, p1 p2 p3 p4, ...
matchingCost subMask] = findPoint(pts, roiMask, ptNo)
```

Katalog: zUser

**Opis:** Metody są nakładkami na procedurę `findSubPatterns`. Opakowują konfigurację pozwalającą na wyszukanie odpowiednio: twarzy (rozdział 10.1.3), nosa (rozdział 10.1.5) i regionów wokół punktów na nosie (rozdział 10.1.6).

**Parametry:**

- pts - chmura punktów  $N \times 3$  (obraz 3D) w której poszukiwany jest region ( otoczenie( $T'$ ) ).
- roiMask (ang. *region of interest*) - wektor binarny o długości równej liczbie punktów:  $|roiMask| = |pts|$ . Służy on identyfikacji punktów, które należą do przeszukiwanego regionu i dla których podzbioru wygenerowane zostaną deskryptory.
- ptNo - numer regionu który jest lokalizowany na nosie, otoczenie punktu:
  1. prn - pronasale
  2. n - nasion
  3.  $al_{lewe}$  - lewe alare
  4.  $al_{prawe}$  - prawe alare
  5. sn - subnasale

**Wartości zwracane:**

- subPts - punkty wyselekcjonowane jako poszukiwany region ( $T$ ).
- bgPts - punkty odrzucone - uznane za tło.
- subMask - maska bitowa (o rozmiarze chmury punktów wejściowych) zawierająca 1 dla indeksów punktów wyselekcjonowanych.
- pozostałe wartości jak w procedurze `findSubPatterns`.

#### 11.4.3 `filterData`

**Nagłówek:**

```
[pts ptsClusters removedToSmall clusterSize] = ...
filterData(pts, options)
```

**Katalog:** zUser

**Opis:** Funkcja filtryuje i wstępnie przetwarza dane w celu usunięcia szumu. Implementuje filtrację przez segmentację w grupy połączone (rozdział 6).

Parametry:

- pts - dane (w postaci macierzy  $N \times 3$  punktów  $[x, y, z]$ ) do przefiltrowania.
- options - opcje filtrowania. Struktura która może zawierać następujące pola:
  - connectivityCoeff = Dc, domyślnie: 0.7.
  - clusterFraction = Sc, domyślnie: 0.4.

Wartości zwracane:

- pts - przefiltrowana chmura punktów.
- ptsClusters - wektor zawierający informację o przyporządkowaniu punktów wejściowych do segmentów.
- removedToSmall - liczba usuniętych (odfiltrowanych) punktów.
- clusterSize - wektor rozmiarów segmentów.

#### 11.4.4 findPatternCluster

Nagłówek:

```
[patternClusterNo patternMatchingCosts ...
noOfClusters clusterPts n spinImgs selIxs] = ...
findPatternCluster(pts, ptCluster, ...
patternSpinImgs, noOfSeedPoints, neighborhoodRadius, ...
spinDistance, alfaBins, betaBins, alfaAxes, betaAxes, ...
matchingRule, matchingDistanceType)
```

Katalog: zUser

Opis: Znajduje w chmurze punktów segment najbardziej przypominający wzorzec. Implementuje wyszukiwanie segmentu twarzy (rozdział 9).

Parametry:

- pts - chmura punktów  $N \times 3$  (obraz 3D) w której poszukiwany będzie segment.
- ptCluster - wektor zawierający informację o przyporządkowaniu punktów do segmentów. Jego rozmiar jest równy liczbie punktów  $N$ . Na kolejnych pozycjach zapisane są numery segmentów.
- patternSpinImgs - macierz  $M \times K$  deskryptorów. W kolejnych wierszach zawiera ona deskryptory (obrazy obrotów) wygenerowane we wzorcu.
- noOfSeedPoints =  $M$  - liczba deskryptorów które zostaną wygenerowane w analizowanym obrazie.

- neighborhoodRadius - promień otoczenia (sąsiedztwa)  $r_e$  stosowany przy obliczaniu kierunków wektorów normalnych.
- parametry deskryptorów które zostaną wygenerowane:
  - spinDistance - promień otoczenia  $r$ .
  - alfaBins - rozdzielcość osi  $\alpha$ .
  - betaBins - rozdzielcość osi  $\beta$ .
  - alfaAxes - typ osi  $\alpha$  ('lin'/'log').
  - betaAxes - typ osi  $\beta$  ('lin'/'log').
- opcje dotyczące dopasowywania (ang. *match*) deskryptorów:
  - matchingRule - metoda filtrowania deskryptorów ('OR'/'AND'/'NONE'). Domyślnie: 'OR'.
  - matchingDistanceType - metoda obliczania odległości między dwoma deskryptorami (1 - *shape context measure*, 2 - korelacja). Domyślnie: 1.

Wartości zwracane:

- patternClusterNo - numer segmentu, który został uznany za najlepiej pasujący do wzorca.
- patternMatchingCosts - koszt dopasowania segmentu.
- noOfClusters - liczba przeanalizowanych segmentów.
- dane dotyczące segmentów. W kolejnych komórkach odpowiadających kolejnym segmentom zawarto:
  - clusterPts - punkty należące do segmentu.
  - n - współrzędne kierunków wektorów normalnych.
  - selIxs - indeksy punktów należących do segmentu dla których wygenerowano deskryptory.
  - spinImgs - deskryptory wygenerowane dla punktów segmentu.

#### **11.4.5 modelPreprocessing**

Nagłówek:

```
[facePts faceCost faceMatchingCosts optimalMaxDistanceCoeff ...
removedToSmall neighborhoodRadius] = ...
modelPreprocessing(pts, options)
```

Katalog: zUser

**Opis:** Wstępne przetwarzanie chmury punktów wraz z wyborem segmentu pasującego do wzorca. Korzysta z `filterData` i `findPatternCluster`. Implementuje schemat opisany w rozdziale 9.2.

**Parametry:**

- pts - chmura punktów  $N \times 3$  (obraz 3D) do przetworzenia.
- options - struktura zawierająca konfigurację procedury. Obsługiwane pola (w przypadku braku pola używana jest wartość domyślna):
  - minFractionOfPtsInCluster =  $S_c$ , domyślnie: 0.1.
  - patternFile - plik zawierający wzorzec, domyślnie: 'patterns/faceClusterDb/cara7 frontal1 filtered raw.txt'.
  - patternDescFile - plik zawierający deskryptory wygenerowane dla wzorca, domyślnie: 'patterns/faceClusterDb/cara7 frontal1 filtered desc.txt'.
  - noOfSeedPoints =  $M$  - rozmiar podzbioru punktów dla których wygenerowane zostaną obrazy obrotu, domyślnie: 100.
  - opcje dotyczące obliczania deskryptorów:
    - \* spinDistance - promień otoczenia  $r$ , domyślnie: 100.
    - \* alfaBins - rozdzielcość osi  $\alpha$ , domyślnie: 10.
    - \* betaBins - rozdzielcość osi  $\beta$ , domyślnie: 10.
    - \* alfaAxes - typ osi  $\alpha$  ('lin'/'log'), domyślnie: 'log'.
    - \* betaAxes - typ osi  $\beta$  ('lin'/'log'), domyślnie: 'log'.
  - opcje dotyczące dopasowywania (ang. *match*) deskryptorów:
    - \* matchingRule - metoda filtrowania deskryptorów ('OR'/'AND'/'NONE'). Domyślnie: 'OR'.
    - \* matchingDistanceType - metoda obliczania odległości między dwoma deskryptorami (1 - *shape context measure*, 2 - korelacja). Domyślnie: 1.

**Wartości zwracane:**

- facePts - punkty wybranego segmentu.
- faceCost - koszt dopasowania wybranego segmentu.
- faceMatchingCosts - koszty dopasowań kolejnych segmentów do wzorca.
- optimalMaxDistanceCoeff - wybrana wartość parametru  $D_c$ .
- removedToSmall - liczba odrzuconych punktów.
- neighborhoodRadius =  $r_e$  - obliczony promień sąsiedztwa dla estymacji kierunków wektorów normalnych.

### 11.4.6 faceAnalysis

Nagłówek:

```
[face_Mask center_FaceMask nose_FaceMask noseNear_FaceMask ...
leftNose_FaceMask rightNose_FaceMask ...
prn_FaceMask n_FaceMask sn_FaceMask lal_FaceMask ral_FaceMask ...
prn n sn lal ral ...
costs t success noseFoundFlag] = faceAnalysis(pts)
```

Katalog: zUser

**Opis:** Implementacja algorytmu analizy opisanego w rozdziale 10.1.2. Procedura dokonuje analizy twarzy. Przeprowadzanych jest kolejnych 7 faz lokalizacji: lokalizacja twarzy, nosa, punktów: prn, n, sn, *al\_lewe*, *al\_prawe*. Dodatkowo po fazie lokalizacji twarzy przeprowadzana jest heurystyka ograniczająca zakres przeszukiwania do obszaru centralnego. Po fazie nosa uruchamiana jest procedura 'pogrubiająca' dla regionu nosa służąca zwiększeniu obszaru przeszukiwania i wypełnieniu nieciągłości w danych. Po zlokalizowaniu punktów (prn, n, sn) obszar nosa dzielony jest na prawą i lewą część. Łącznie wykonywanych jest 10 faz.

Parametry:

- pts - chmura punktów  $N \times 3$  (obraz 3D) do przetworzenia; segment zawierający twarz

Wartości zwracane:

- face\_Mask - maska bitowa wybierająca punkty należące do twarzy z 'pts'
- maski bitowe wybierające z twarzy (tj. z punktów pts(face\_Mask,:)):
  - center\_FaceMask - region centralny twarzy
  - nose\_FaceMask - region nosa
  - noseNear\_FaceMask - poszerzony i skorygowany region nosa
  - leftNose\_FaceMask - lewą część nosa
  - rightNose\_FaceMask - prawą część nosa
  - prn\_FaceMask - otoczenie punktu: prn
  - n\_FaceMask - otoczenie punktu: nasion
  - sn\_FaceMask - otoczenie punktu: subnasale
  - lal\_FaceMask - otoczenie punktu: lewe alare
  - ral\_FaceMask - otoczenie punktu: prawe alare
- prn n sn lal ral - wartości średkowe (ang *centroid*) regionów nosa

- costs - sumaryczne koszty dopasowań do wzorców uzyskiwane w kolejnych fazach: znajdowania twarzy, nosa i kolejnych pięciu regionów nosa. Domyślnie równe inf.
- t - czasy kolejnych 10 stopni obliczeń. W przypadku gdy któraś z faz się nie powiodła (błąd) odpowiednia wartość w wektorze t równa jest inf.
- success - maska bitowa zawierająca informację o wyniku działania kolejnej fazy lokalizacji (7 faz). Jeśli faza zakończyła się sukcesem - - region zlokalizowano, to maska zawiera wartość 1. Jeśli porażką to 0.
- noseFoundFlag - flaga mówiąca o tym czy zidentyfikowany region nosa uznano za wiarygodny.

**Uwagi:** w parametrach zastosowano maski o nazewnictwie zgodnym z formatem: A\_XMask gdzie:

- A - co konkretnie wybiera maska (np. nos)
- X - z czego maska wybiera (np. twarz)

np. lewyNosa\_TwarzMask - maska wybierająca lewą część nosa z twarzy

#### 11.4.7 noseDistances

Nagłówek:

```
[height length width heightSnPrn wl wr ...
heightP lengthP widthP heightSnPrnP wlP wrP...
sex validDist validPt] = noseDistances(prn, nasion, sn, lal, ral)
```

Katalog: zSpecialized

**Opis:** Procedura implementuje procedurę obliczenia i weryfikacji odległości charakterystycznych nosa opisaną w rozdziale 10.1.2.

**Parametry:** współrzędne kolejnych punktów na nosie (prn, nasion, sn, *allewe*, *alprawe*.

Wartości zwracane:

- odległości typowe:
  1.  $height = \|n - sn\|$
  2.  $length = \|n - prn\|$
  3.  $width = \|al_{lewe} - al_{prawe}\|$
  4.  $heightSnPrn = \|sn - prn\|$
  5.  $wl = \|al_{lewe} - prn\|$
  6.  $wr = \|al_{prawe} - prn\|$

- sex - dopasowana płeć ('M'/'F').
- wiarygodność uzyskanych rezultatów dla dopasowanej płci (zgodnie z wzorem 10.1):
  1.  $heightP = p(height)$
  2.  $lengthP = p(length)$
  3.  $widthP = p(width)$
  4.  $heightSnPrnP = p(heightSnPrn)$
  5.  $wlP = p(wl)$
  6.  $wrP = p(wr)$
- validDist - maska bitowa o długości 6, mówiąca o przydatności do analizy dla kolejnych odległości
- validPt - maska bitowa o długości zawierająca współczynniki wiarygodności przy-porządkowane kolejnym punktom (dla  $prn \in \{0, 1, 2, 3, 4, 5\}$ , dla pozostałych  $\in \{0, 1, 2\}$ )

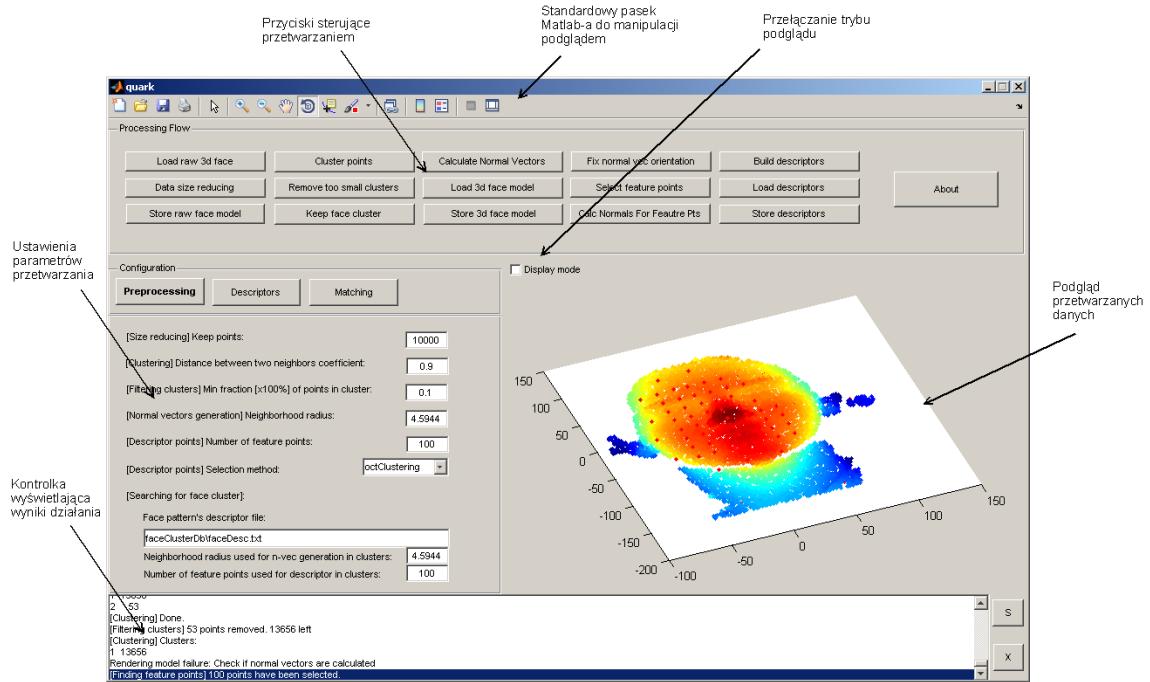
## 11.5 Stworzone narzędzia

### 11.5.1 Quark

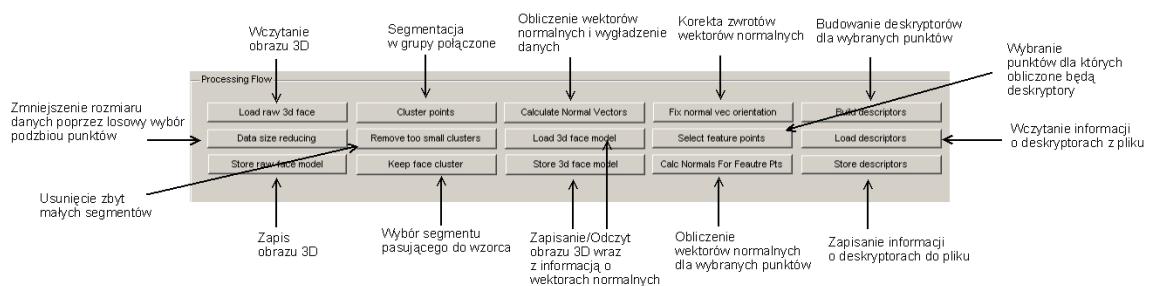
*Quark* jest narzędziem powstały w celu testowania i badania zachowania algorytmów fazy wstępne przetwarzania obrazów 3D. Wbudowano w niego też możliwość generowania deskryptorów. Dodatkowo, zaimplementowano w nim wybór segmentu najbardziej zbliżonego do wzorca. Program uruchamiany jest polecienniem *quark* (z katalogu głównego projektu). Kod źródłowy umieszczono w katalogu *zTools*.

Interfejs narzędzia ukazano na rysunku 11.1. Składa się on z 4 części:

1. górny pasek narzędziowy (rysunek 11.2). Zawiera on przyciski wywołujące po-szczególne algorytmy przetwarzania. Parametry algorytmów odczytywane są z bocznego panelu konfiguracyjnego.
2. boczny panel konfiguracyjny. Zawiera on 3 zakładki na których umiejscowiono kontrolki pozwalające na ustawianie argumentów procedur przetwarzających. Pierwsza zakładka (rysunek 11.3) odpowiada za filtrowanie danych. Pozwala również na ustawienie wzorca który użyty zostanie w algorytmie wyboru segmentu. Druga (rysunek 11.4) pozwala na dobór parametrów obliczania deskryptorów. Ostatnia (rysunek 11.5) zawiera ustawienia dopasowywania.
3. podgląd danych. Umiejscowiona z prawej strony kontrolka wizualizuje wyniki prze-twarzania i bieżący stan obrazu.
4. kontrolka informacyjna. Umiejscowiona ona została w dolnej części. Służy do wy-świetlania komunikatów o stanie programu i wyników działania poszczególnych wywołań procedur.



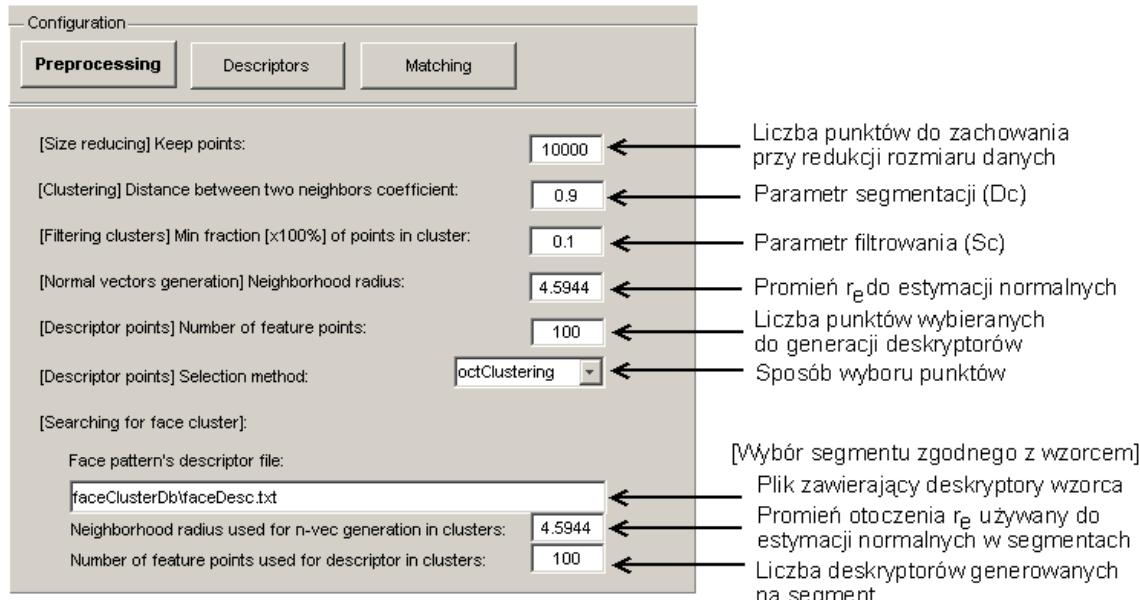
Rysunek 11.1: Interfejs programu quark.



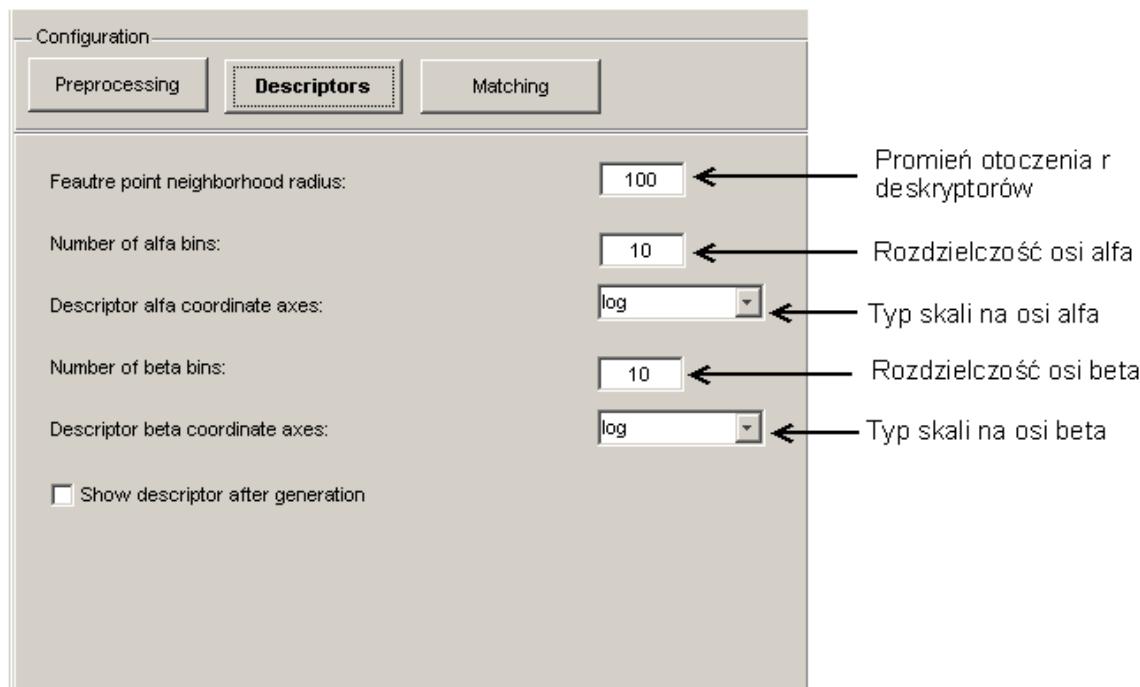
Rysunek 11.2: Górnny pasek narzędziowy programu quark.

## 11.5.2 gui3dPtSelector

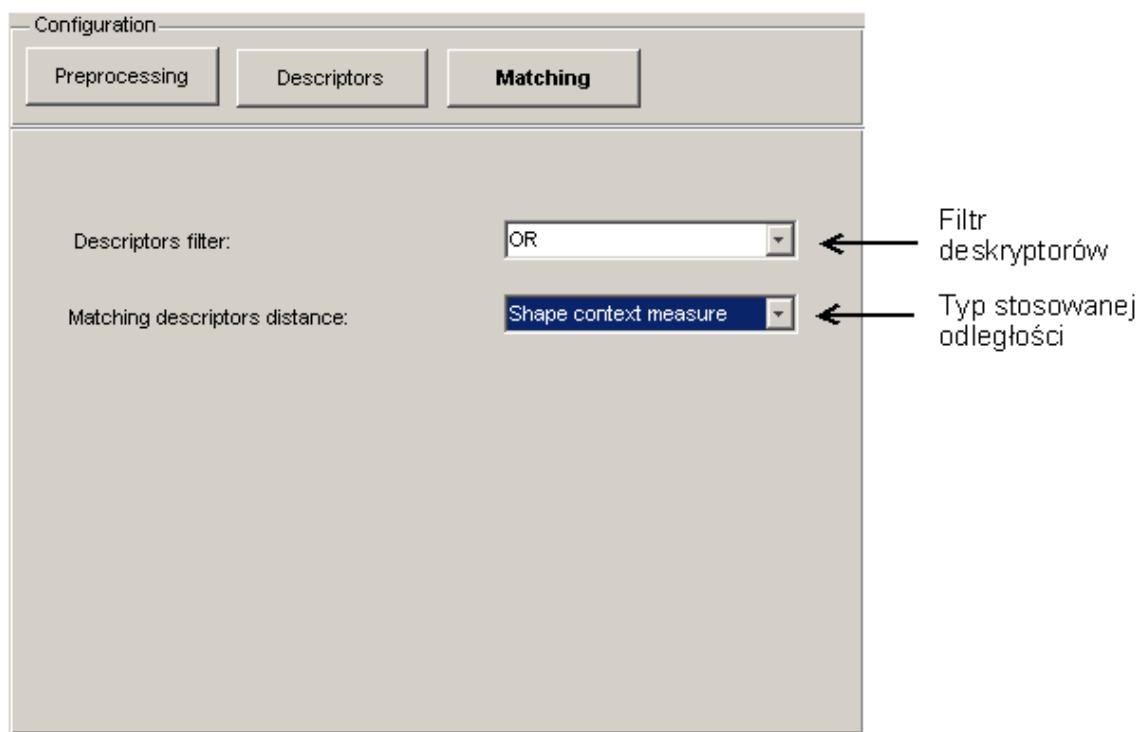
`gui3dPtSelector` jest programem służącym identyfikacji punktów w obrazie. Interfejs narzędzia ukazano na rysunku 11.6. Centralnym elementem jest kontrolka wizualizująca obraz 3D. Zawiera ona kurSOR 3D, którego aktualne położenie wyświetlane jest w górnej części okna. Przemieszczany jest on za pomocą klawiszy: `q`, `w`, `e`, `a`, `s`, `d`. Dodatkowo za pomocą przycisku `GoToPt` może on zostać przeciagnięty do najbliższego punktu obrazu. Narzędzie wykorzystane zostało do przybliżonego określenia współrzędnych punktów charakterystycznych nosa.



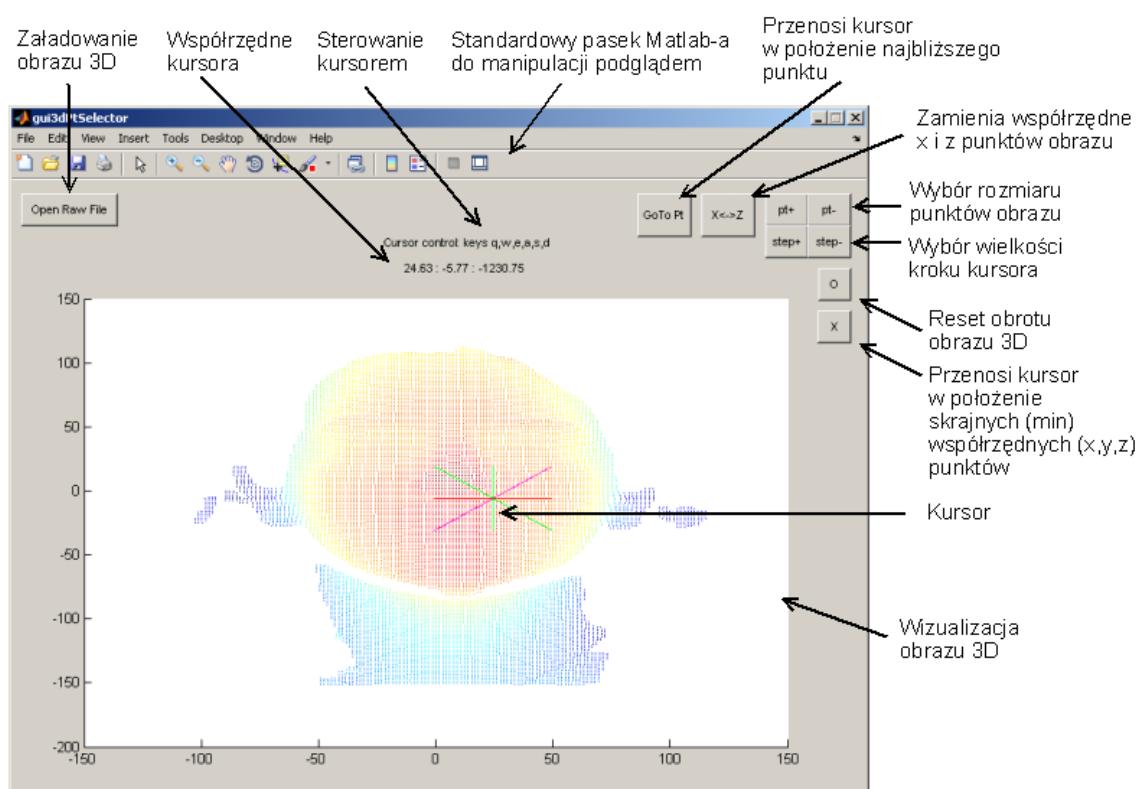
Rysunek 11.3: Boczny panel konfiguracyjny quark - wstępne przetwarzanie.



Rysunek 11.4: Boczny panel konfiguracyjny quark - obliczanie deskryptorów.



Rysunek 11.5: Boczny panel konfiguracyjny quark - ustawienia dopasowywania.

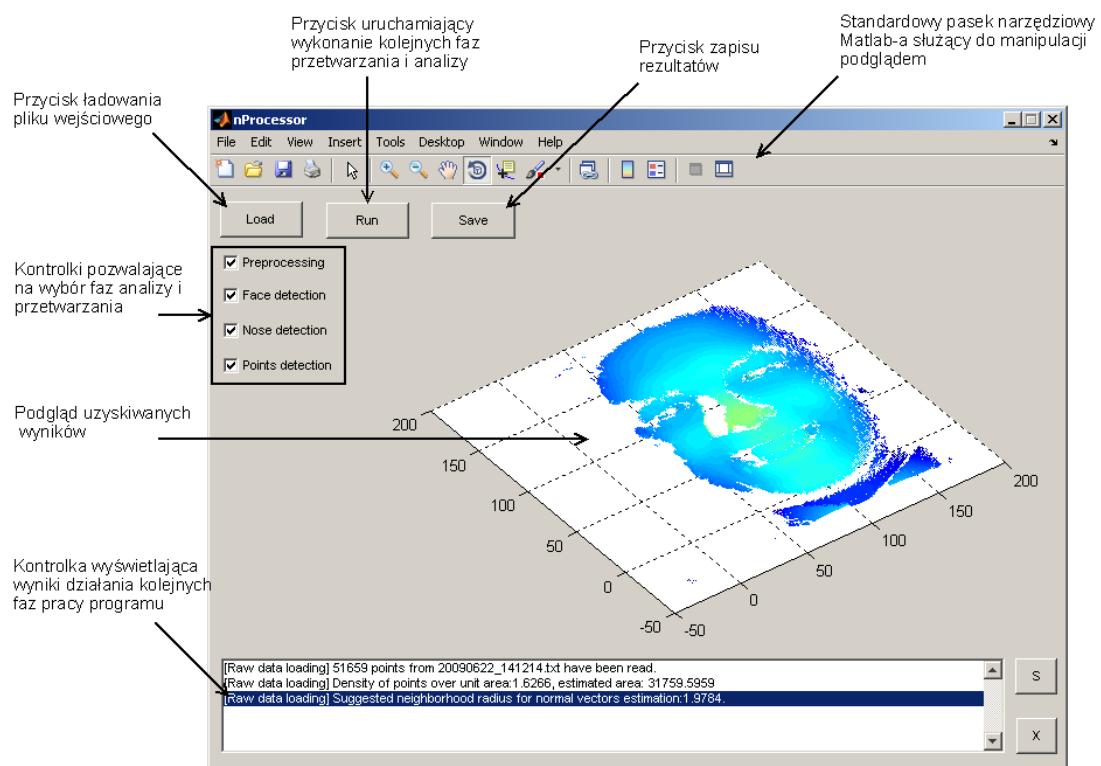


Rysunek 11.6: Interfejs programu `gui3dPtSelector`.

### 11.5.3 nProcessor

*nProcessor* jest narzędziem którego celem jest umożliwienie w sposób maksymalnie prosty użycia zaimplementowanych algorytmów. Zaimplementowano w nim użycie algorytmów wstępnej filtracji danych wraz z wyborem segmentu twarzy oraz hierarchicznej lokalizacji regionów twarzy. Interfejs programu zaprezentowano na rysunku 11.7. Składa on się z 4 części:

1. przycisków sterujących: ładowania i zapisu danych oraz przycisku uruchamiającego algorytm
2. wizualizacji danych
3. kontrolek pozwalających na wybór faz przetwarzania i analizy, które mają zostać uruchomione
4. kontrolki informacyjnej, służącej do wyświetlania wyników działania algorytmów



Rysunek 11.7: Interfejs programu nProcessor.



## Rozdział 12

---

# Podsumowanie

---

Obniżenie kosztów skanerów trójwymiarowych powoduje, że zaczynają one być stosowane w coraz szerszym zakresie. Pojawiają się ich wdrożenia w medycynie, metrologii czy systemach autentykacji. W najbliższej przyszłości ich zastosowanie pozwoli na zautomatyzowanie kolejnych procesów m.in. w antropologii. Przeszkodą, spowalniającą ich rozprzestrzenianie, jest brak algorytmów analizy danych.

W niniejszej pracy zaprezentowano stworzone rozwiązanie algorytmiczne automatyzujące proces analizy obrazów trójwymiarowych twarzy pochodzących z urządzeń skanujących. Pozwala ono na odfiltrowanie danych wejściowych i zidentyfikowanie segmentu zawierającego twarz. Umożliwia lokalizację i analizę obszaru twarzy, a w szczególności nosa i punktów charakterystycznych na nim się znajdujących.

Przeprowadzony i opisany został pełny proces analizy, projektowania i testowania systemu biometrycznego. Przedstawiono specyfikę pomiarów antropometrycznych. Dokonano przeglądu możliwych źródeł obrazów trójwymiarowych. Wybrano dwa najbardziej perspektywiczne. Zbadano specyfikę pozyskanych danych. Zaadaptowano i udoskonalono algorytm wstępnej filtracji chmur punktów.

W kolejnym kroku, zebrano i przedstawiono wybraną literaturę z dziedziny analizy obiektów trójwymiarowych dowolnego kształtu. Z możliwych podejść wybrano deskryptory punktów jako rozwiązanie proste koncepcyjnie i o dużym potencjale. Dogłębnie rozważono wszystkie aspekty ich stosowania. Porównano wszystkie możliwości. Przetestowano różne sposoby wyboru podzbioru punktów oraz ich wpływ na jakość dopasowywania. Zaprojektowano i sprawdzono procedurę estymacji wektorów normalnych do powierzchni.

W oparciu o deskryptory zaprojektowano algorytm wyboru segmentu zawierającego twarz. Dobrano parametry i przetestowano jego zachowanie dla różnych zestawów danych. Zaproponowano metodę lokalizacji podzbioru punktów zgodnego z wzorcem. W oparciu o nią stworzono procedurę wyszukującą twarz, następnie nos, a dalej otoczenia punktów nosa.

Eksperymentalnie sprawdzono prawdziwość założeń poczynionych przy opracowywaniu powyższych algorytmów. Przeprowadzone testy pokazały wysoką skuteczność lokalizacji twarzy w wejściowej chmurze punktów. Również procedura identyfikacji nosa

działa bardzo dobrze. Procedura identyfikacji punktów charakterystycznych w znaczącej części przypadków pozwala na ich zgrubną lokalizację. Skuteczność algorytmów zależy głównie od jakości danych.

Postawione we wstępnie zadanie zostało w zrealizowane. Powstały algorytmy automatycznej analizy obrazów trójwymiarowych. Zostały one wykorzystane i przetestowane na skanach twarzy, jednak **mogą one zostać zaadaptowane dla praktyczne do-**  
**wolnych obiektów.** Możliwe jest ich zastosowanie w szerokiej gamie systemów: do automatycznej analizy danych medycznych, czy przy weryfikacji użytkowników. Mogą posłużyć jako moduł lokalizujący twarz w systemach autoryzacji. **Warunkiem wdrożenia w systemach interaktywnych jest reimplementacja rozwiązania w języku kompilowanym do kodu maszynowego, co powinno zmniejszyć czas obliczeń nawet kilkudziesięciokrotnie.**

Widocznych jest szereg ścieżek rozwoju stworzonego rozwiązania. Pierwszą z nich jest opracowanie algorytmów analizujących regiony nosa i precyzyjniej lokalizujących punkty charakterystyczne. Jak wiadomo, nie jest to możliwe w oparciu o deskryptory punktów, więc należałoby wykorzystać podejście oparte o metody specjalizowane dla konkretnych cech. Innym możliwym kierunkiem prac jest analiza wpływu różnych zestawów parametrów na zachowanie się algorytmów. Wysoko prawdopodobne, że możliwe jest znaczące zwiększenie ich skuteczności przy redukcji kosztów obliczeniowych. Potencjalną ścieżką rozwoju jest też zintegrowanie rozwiązania z algorytmami analizy 2D, co często znaczająco poprawia zachowanie się procedur [4].

---

# Bibliografia

---

- [1] Serge Belongie and Jitendra Malik, editors. *Matching with Shape Contexts*. IEEE, 2000. IEEE Workshop on Contentbased Access of Image and Video Libraries (CBAIVL-2000). [cytowanie na str. 31, 32, 35]
- [2] Serge Belongie, Jitendra Malik, and Jan Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24), April 2002. [cytowanie na str. 27, 31, 32, 35]
- [3] Hui Chen and Bir Bhanu. 3d free-form object recognition in range images using local surface patches. *Pattern Recognition Letters*, 28:1252–1262, 2007. [cytowanie na str. 32, 35]
- [4] Hui Chen and Bir Bhanu. Human ear recognition in 3d. *IEEE Transactions On Pattern Analysis And Machine Intelligence*, 29(4), April 2007. [cytowanie na str. 35, 122]
- [5] Chin-Seng Chua, Feng Han, and Yeong-Khing Ho, editors. *3D Human Face Recognition Using Point Signature*. School of Electrical and Electronic Engineering, Nanyang Technological University, IEEE, 2000. Fourth IEEE International Conference on Automatic Face and Gesture Recognition (FG'00). [cytowanie na str. 30, 31, 32, 35]
- [6] Chin Seng Chua and Ray Jarvis. Point signatures: A new representation for 3d object recognition. *International Journal of Computer Vision*, 25(1):63–85, 1997. [cytowanie na str. 34]
- [7] Alessandro Colombo, Claudio Cusano, and Raimondo Schettini. 3d face detection using curvature analysis. *Pattern Recognition*, 36:444–455, 2006. [cytowanie na str. 26]
- [8] Cristina Conde, Angel Serrano, Licesio J. Rodríguez-Aragón, and Enrique Cabello, editors. *3D Facial Normalization with Spin Images and Influence of Range Data Calculation over Face Verification*. Universidad Rey Juan Carlos (URJC), Escuela Superior de Ciencias Experimentales y Tecnología (ESCET), Face Recognition and Artificial Vision Group (FRAV), IEEE, 2005. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). [cytowanie na str. 30]
- [9] G.G. Gordon, editor. *Face recognition based on depth and curvature features*. IEEE, 1992. Computer Vision and Pattern Recognition, 1992. Proceedings CVPR '92., 1992 IEEE Computer Society Conference. [cytowanie na str. 26]
- [10] Kroemer H.J. *Engineering physiology: Physiologic bases of human factors/ergonomics*. Elsevier, Amsterdam, 1986. [cytowanie na str. 6]
- [11] Andrew E. Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5), May 1999. [cytowanie na str. 29, 31, 32, 34, 127]

- [12] Kodak. *KODAK 9500 Cone Beam 3D System*. [cytowanie na str. 2]
- [13] Konica Minolta. *Non-Contact 3D Digitizer, Vivid 910/vi-910, Instruction Manual*. [cytowanie na str. 17]
- [14] Yang Li, William A.P. Smith, and Edwin R. Hancock, editors. *Face Recognition using Patch-based Spin Images*. Department of Computer Science, University of York, IEEE, 2006. 18th International Conference on Pattern Recognition (ICPR'06). [cytowanie na str. 31, 32]
- [15] Napoleon Wolański przy współautorstwie Stanisława Niemca i Miry Pyżuk. *Antropometria inżynieryjna: Kształt i wymiary ciała, a wzornictwo przemysłowe*. Książka i Wiedza, Warszawa, 1975. [cytowanie na str. 5, 6, 7, 9, 10, 127]
- [16] Salvadro Ruiz-Correa, Linda G. Shapiro, and Marina Melia, editors. *A New Signature-Based Method for Efficient 3-D Object Recognition*. Department of Electrical Engineering, Department of Statistics, University of Washington, IEEE, 2001. [cytowanie na str. 26, 31, 32]
- [17] Karla Peavy Simmons. *Body measurement techniques: a comparison of three-dimensional body scanning and physical anthropometric methods*. PhD thesis, North Carolina State University, Raleigh, North Carolina, January 2001. [cytowanie na str. 2, 5, 7]
- [18] Robert Sitnik. *A fully automatic 3D shape measurement system with data export for engineering and multimedia systems*. PhD thesis, Politechnika Warszawska, Warszawa, 2002. [cytowanie na str. 13, 18, 41, 53]
- [19] B.K. Smoliński, Grzanka A., and Gotlib T. Changes in nasal cavity dimensions in children and adults by gender and age. *Laryngoscope*, 117(8):1429–1433, August 2007. [cytowanie na str. 1, 10]
- [20] Paulina Szczęsnowicz-Dąbrowska. *Związek Pomiedzy Wynikami Rynometrii Akustycznej, a Parametrami Antropometrycznymi u Zdrowych Osobników*. PhD thesis, Akademia Medyczna w Warszawie, Warszawa, 2006. [cytowanie na str. 1, 7, 10, 11, 130]
- [21] Zhnhui Wu, Yueming Wang, and Gnng Pan, editors. *3D Face Recognition Using Local Shape Map*. Department of Computer Science and Engineering Zhejiang University, Hangzhou, IEEE, 2004. 2004 International Conference on Image Processing (ICIP). [cytowanie na str. 25, 33]
- [22] Chrnhua Xu, Yunhong Wang, and Tienii Tan, editors. *Robust Nose Detection in 3D Facial Data Using Local Characteristics*. National Lab of Pattern Recognition, Institute of Automation, CAS, Beiing, P.R.China, IEEE, 2004. 2004 International Conference on Image Processing (ICIP). [cytowanie na str. 26]
- [23] A.B. Moreno y A.Sanchez, editor. *GavabDB: A 3D Face Database*. Ed. Univ. Vigo, 2004. Proc. 2nd COST Workshop on Biometrics on the Internet: Fundamentals, Advances and Applications. [cytowanie na str. 17, 18, 127]
- [24] Marian Yusuf and Tarique Haider, editors. *Recognition of Handwritten Urdu Digits using Shape Context*. Department of Computer Science, KASBIT, Karachi, IEEE, 2004. Multi-topic Conference, 2004. Proceedings of INMIC 2004. 8th International. [cytowanie na str. 31, 32, 35]
- [25] Wenhao Zhu, Yanyun Wang, and Baogang Wei, editors. *Nose detection based feature extraction on both normal and abnormal 3D faces*. IEEE, 2008. Computer and Information Technology, 2008. CIT 2008. 8th IEEE International Conference. [cytowanie na str. 26]

---

# Spis skrótów i symboli

---

Symbol/Skrót	Opis
PCA	Principal Component Analysis (Analiza głównych składowych) - metoda redukcji wymiarowości danych
SVM	Support vector machines (Maszyna wektorów nośnych )
k-nn	klasyfikator k-najbliższych sąsiadów
LSP	Local Surface Patch ("łatka" lokalnej powierzchni) - typ deskryptora
LSM	Local Shape Map (mapa lokalnego kształtu) - typ deskryptora
SCM	Shape Context Measure - miara odległości między deskryptorami opisana wzorem 4.1
AND/OR/NONE	filtры deskryptorów opisane w rozdziale 4.3.4
prn	punkt charakterystyczny nosa - pronasale
n	punkt charakterystyczny nosa - nasion
sn	punkt charakterystyczny nosa - subnasale
al	punkt charakterystyczny nosa - alare
en	punkt charakterystyczny twarzy - entokanthion
<i>Obraz</i>	chmura punktów; obraz 3D
<i>p</i>	punkt z którym powiązany jest deskryptor
<i>r</i>	promień sfery otaczającej punkt dla którego budowany jest deskryptor; wyznacza sąsiedztwo punktu <i>p</i>
<i>r<sub>n</sub></i>	promień sfery otaczającej (wyznaczający sąsiedztwo) punktu w którym obliczany jest wektor normalny do powierzchni
<i>r<sub>e</sub></i>	promień sfery otaczającej (wyznaczający sąsiedztwo) punktu testowego przy estymacji gęstości powierzchniowej obrazu
<i>q</i>	punkt należący do sąsiedztwa punktu <i>p</i>
$\vec{n}_x$	wektor normalny do powierzchni w punkcie <i>x</i>
<i>Tan<sub>x</sub></i>	płaszczyzna styczna do powierzchni w punkcie <i>x</i>
$\alpha, \beta$	oznaczenia osi współrzędnych w dwuwymiarowych histogramach budowanych na potrzeby deskryptorów punktów
$ x - y $	symboliczne oznaczenie odległości między strukturami <i>x</i> i <i>y</i>
$ X $	moc, liczebność zbioru X
<i>sign</i>	funkcja zwracająca znak (+1/0/-1) liczby
<i>c</i>	koszt dopasowania między dwoma deskryptorami (histogramami)
<i>q/h</i>	pojedynczy deskryptor (histogram)
<i>G/H</i>	zbiór deskryptorów wygenerowany dla danego obrazu (chmury punktów)

Symbol/Skrót	Opis
$K$	liczba "kubełków" histogramu
$M$	liczba punktów dla których wygenerowano deskryptory; liczba deskryptorów obliczonych dla danego obrazu
$E_p$	liczba punktów w otoczeniu $r_n$ punktu $p$ dla którego budowany jest deskryptor
$N$	liczba punktów obrazu
$f_n(N)$	funkcja określająca koszt obliczeniowy znalezienia stałej liczby najbliższych punktów do danego punktu w chmurze $N$ punktów, $f_n(N) = O(N)$
$D_c$	współczynnik grupowania w algorytmie opisanym w rozdziale 6
$S_c$	parametr odrzucania grup w algorytmie filtrowania przez grupowanie opisanym w rozdziale 6
$\sigma$	gęstość powierzchniowa obrazów [punkt / $mm^2$ ]
<i>srednia</i>	funkcja obliczająca średnią ze zbioru liczb
$centroid(X)$	funkcja obliczająca środek ciężkości zbioru punktów $X$
$powierzchnia(X)$	funkcja estymująca powierzchnię chmury punktów $X$
$otoczenie(X)$	chmura punktów zawierająca w sobie punkty $X$ wraz z pewnym otoczeniem (punktami sąsiednimi)
$Q$	podzbiór punktów chmury punktów stanowiący wzorzec regionu
$Q'$	chmura punktów zawierająca w sobie wzorzec (otoczenie wzorca)
$D_Q$	zbiór deskryptorów (wraz z odpowiadającymi im punktami) wygenerowany na obszarze wzorca $Q$ w oparciu o punkty wzorca wraz z otoczeniem $Q'$
$A_Q$	estymowana powierzchnia wzorca $Q$
$T$	podzbiór analizowanej chmury punktów $T'$ zidentyfikowany jako odpowiadający wzorcowi
$T'$	analizowana chmura punktów w której poszukiwany jest region odpowiadający wzorcowi
$T_{T'}$	zbiór deskryptorów (wraz z odpowiadającymi im punktami) wygenerowanych na punktach analizowanej chmury punktów $T'$
$D_T$	zbiór deskryptorów (wraz z odpowiadającymi im punktami; na zbiorze analizowanym $T'$ ) przyporządkowanych do wzorca
$D_{T''}$	zbiór deskryptorów (wraz z odpowiadającymi im punktami; na zbiorze analizowanym) $T'$ niedopasowanych do wzorca
$Q_X$	wzorzec regionu $X$ ( $X = F$ - twarz, $X = N$ - nos, $X = prn/sn/n/all/alr$ - otoczenia punktów charakterystycznych)
$T_X$	zidentyfikowany w danych region $X$ ( $X = F$ - twarz, $X = N$ - nos, $X = C$ - część środkowa twarzy, $X = L$ - lewa część nosa, $X = R$ - prawa część nosa, $X = prn/sn/n/all/alr$ - otoczenia punktów charakterystycznych)

---

# Spis rysunków

---

2.1	Podstawowy zestaw narzędzi do pomiarów antropometrycznych . . . . .	6
2.2	Punkty antropologiczne twarzy (widok z boku) [15]. . . . .	9
2.3	Punkty antropologiczne twarzy (widok z przodu) [15]. . . . .	10
3.1	Schemat skanera zbudowanego w oparciu o fotogrametrię. . . . .	13
3.2	Schemat skanera zbudowanego w oparciu o czas lotu światła. . . . .	14
3.3	Schemat skanera zbudowanego w oparciu o metodę skaningu laserem. . . . .	15
3.4	Schemat skanera opartego o interferometr . . . . .	15
3.5	Schemat skanera opartego o światło strukturalne. . . . .	15
3.6	Przykładowa chmura punktów powstała w wyniku skanowania powierzchni twarzy. . . . .	16
3.7	Ułożenie twarzy odpowiadające kolejnym skanom w bazie GavabDB [23]. . . . .	18
3.8	Przykładowy skan z bazy GavabDB (widok twarzy z przodu). . . . .	18
3.9	Przykładowy skan z bazy GavabDB (widok twarzy z prawego profilu). . . . .	19
3.10	Przykładowy obraz z bazy Mechatroniki (widoczna ucięta górną częścią twarzy). . . . .	20
3.11	Przykładowy obraz z bazy Mechatroniki (widoczne nieciągłości w modelu i punkty nie należące do twarzy). . . . .	20
3.12	Rozkład liczby punktów dla obrazów z bazy GavabDB. . . . .	21
3.13	Rozkład liczby punktów dla obrazów z bazy Mechatroniki. . . . .	22
3.14	Średnia liczba punktów w sąsiedztwie danego punktu w zależności od promienia sfery otaczającej dany punkt dla 10 skanów z 2 baz. . . . .	23
3.15	Odchylenie standardowe liczby punktów w sąsiedztwie danego punktu w zależności od promienia sfery otaczającej dany punkt dla 10 skanów z 2 baz. . . . .	23
4.1	Przykładowy histogram wartości indeksu kształtu względem iloczynu skalarnego wektorów w LSP. . . . .	28
4.2	Przykład budowania histogramu - <i>Shape Context</i> . . . . .	28
4.3	Zmodyfikowany układ współrzędnych stosowany w obrazach obrotu. . . . .	29
4.4	Ilustracja budowania obrazów obrotu dla różnych punktów obiektu [11]. . . . .	29
5.1	Ilustracja koncepcji hierarchicznej lokalizacji regionów. . . . .	38
5.2	Schemat struktury ogólnej zaprojektowanego systemu. . . . .	39

6.1	Zachowanie się filtrowania przez segmentację dla dwóch przykładowych plików z dwóch baz. . . . .	43
6.2	Wykres zależności czasu wstępnego filtrowania danych od liczby punktów obrazu (naniesiona prosta regresji). . . . .	44
6.3	Histogram ukazujący jaki ułamek punktów jest odrzucany w obrazach GavabDB. . . . .	44
6.4	Histogram ukazujący jaki ułamek punktów jest odrzucany w obrazach Mechatroniki. . . . .	45
7.1	Ilustracja poziomów zgodności promienia sąsiedztwa dla Mechatroniki. . . . .	49
7.2	Ilustracja poziomów zgodności promienia sąsiedztwa dla Gavab. . . . .	50
7.3	Zależność estymowanej gęstości powierzchniowej punktów $\sigma$ od promienia sąsiedztwa $r_e$ dla 10 obrazów Mechatroniki. . . . .	51
7.4	Zależność estymowanej gęstości powierzchniowej punktów $\sigma$ od promienia sąsiedztwa $r_e$ dla 10 obrazów Gavab. . . . .	52
7.5	Histogram obliczanego promienia otoczenia $r_n$ dla 50 obrazów z GavabDB. .	52
7.6	Histogram obliczanego promienia otoczenia $r_n$ dla 50 obrazów z Mechatroniki.	53
7.7	Schemat algorytmu estymacji kierunków wektorów normalnych. . . . .	54
7.8	Przykład działania algorytmu obliczającego kierunki wektorów normalnych.	55
8.1	Przykład losowego wyboru 150 punktów do generacji deskryptorów. . . . .	58
8.2	Przykład wyboru 150 punktów metodą opartą o K-średnich. . . . .	58
8.3	Schemat działania algorytmu octClustering. . . . .	60
8.4	Przykład podziału obrazu na 150 partycji algorytmem octClustering. . . . .	61
8.5	Przykład wyboru 150 punktów algorytmem octClustering. . . . .	61
8.6	Koszty dopasowania obrazów twarzy do samych siebie przy losowym wyborze deskryptorów. . . . .	62
8.7	Koszty dopasowania obrazów twarzy do samych siebie przy wyborze deskryptorów metodą opartą o k-średnich. . . . .	63
8.8	Koszty dopasowania obrazów twarzy przy losowym wyborze deksyptorów. .	64
8.9	Koszty dopasowania obrazów twarzy przy wyborze deskryptorów metodą opartą o k-średnich. . . . .	64
8.10	Koszty dopasowania obrazów twarzy przy wyborze deskryptorów metodą opartą o octClustering. . . . .	65
8.11	Porównanie kosztów (na jeden deskryptor) auto-dopasowania i dopasowania różnych twarzy. . . . .	65
9.1	Przykład wyniku działania fazy wstępного filtrowania danych (cara30 frontal1). . . . .	67
9.2	Przykład wyniku działania fazy wstępnego filtrowania danych (cara28 frontal1). . . . .	68
9.3	Wzorzec stosowany w wyborze segmentu twarzy (cara7 frontal1). . . . .	69
9.4	Schemat fazy wstępnego przetwarzania danych wraz z wyborem segmentu twarzy. . . . .	72
9.5	Przykład działania fazy wstępnego przetwarzania danych wraz z wyborem segmentu twarzy. . . . .	72

9.6 Zależność czasu wykonania fazy wstępnego przetwarzania względem liczby punktów obrazu. . . . .	73
9.7 Rozkład odległości od wzorca dla obrazów z bazy GavabDB. . . . .	75
9.8 Rozkład odległości od wzorca dla obrazów z bazy Mechatroniki. . . . .	75
9.9 Rozkład odległości od wzorca dla obrazów wygenerowanych losowo. . . . .	76
10.1 Przykład budowy wzorca do lokalizacji regionu. . . . .	78
10.2 Przykład działania algorytmu lokalizacji regionów. . . . .	81
10.3 Przykład działania klasyfikatorów dla klasyfikacji punktów twarzy. . . . .	82
10.4 Schemat algorytmu analizy twarzy. . . . .	83
10.5 Przykład działania algorytmu dla różnej liczby deskryptorów i różnych sieci. . . . .	86
10.6 Przykładowe wyniki analizy charakterystyk deskryptorów dla regionów. . . . .	88
10.7 Przykładowy obraz z grupy 1 zbioru testowego. . . . .	92
10.8 Przykładowy obraz z grupy 2 zbioru testowego. . . . .	92
10.9 Przykładowy obraz z grupy 3 zbioru testowego. . . . .	93
10.10 Czas hierarchicznej lokalizacji regionów względem liczby punktów obrazu. . . . .	93
10.11 Przykład działania algorytmu analizy na obrazie en face z GavabDB. . . . .	99
10.12 Przykład działania algorytmu analizy na obrazie en face z Mechatroniki. . . . .	99
10.13 Przykład działania algorytmu analizy na obrazie profilu z GavabDB . . . . .	100
11.1 Interfejs programu quark. . . . .	115
11.2 Górnny pasek narzędziowy programu quark. . . . .	115
11.3 Boczny panel konfiguracyjny quark - wstępne przetwarzanie. . . . .	116
11.4 Boczny panel konfiguracyjny quark - obliczanie deskryptorów. . . . .	116
11.5 Boczny panel konfiguracyjny quark - ustawienia dopasowywania. . . . .	117
11.6 Interfejs programu gui3dPtSelector. . . . .	118
11.7 Interfejs programu nProcessor. . . . .	119

---

# Spis tabelic

---

2.1	Zestawienie wartości parametrów twarzoczaszki dla osób powyżej 17 roku życia w okolicach Warszawy [20]. . . . .	11
2.2	Zestawienie wartości parametrów nosa dla osób powyżej 17 roku życia w okolicach Warszawy [20]. . . . .	11
3.1	Typy danych w bazie GavabDB. . . . .	17
3.2	Klasyfikacja danych z bazy Mechatroniki. . . . .	19
3.3	Statystyki liczby punktów w poszczególnych bazach obrazów 3D. . . . .	21
3.4	Przykładowe dane wybrane do analizy rozdzielczości obrazów. . . . .	22
7.1	Złożoność obliczeniowa poszczególnych kroków obliczania wektorów normalnych. . . . .	53
9.1	Wyniki dopasowywania obrazów testowych do wzorca segmentu twarzy dla różnych parametrów (filtrowanie metodą OR). . . . .	70
9.2	Sumaryczny koszt dopasowania obrazów testowych do wzorca dla wybranych parametrów (filtrowanie metodą OR). . . . .	71
9.3	Sumaryczny koszt dopasowania obrazów testowych do wzorca dla wybranych parametrów (filtrowanie metodą AND). . . . .	71
9.4	Sumaryczny koszt dopasowania obrazów testowych do wzorca dla wybranych parametrów (brak filtrowania). . . . .	71
10.1	Parametry fazy lokalizacji nosa. . . . .	88
10.2	Parametry fazy lokalizacji regionów na nosie. . . . .	89
10.3	Odległości między dwoma próbami oznaczenia punktu charakterystycznego nosa. . . . .	90
10.4	Koszt dopasowania (SCM) między dwoma próbami oznaczenia punktu charakterystycznego nosa. . . . .	90
10.5	Czasy obliczeń kolejnych faz hierarchicznej lokalizacji regionów dla obrazów GavabDB. . . . .	91
10.6	Jakość klasyfikacji regionu twarzy. . . . .	94
10.7	Odległości między środkami ciężkości wzorca regionu twarzy i wynikami klasyfikacji. . . . .	95
10.8	Sumaryczne koszty dopasowania otrzymane w fazie lokalizacji nosa. . . . .	95

10.9 Jakość klasyfikacji regionu nosa. . . . .	96
10.10 Odległości między środkami ciężkości wzorca regionu nosa i wynikami klasyfikacji. . . . .	96
10.11 Ułamek otoczenia punktów charakterystycznych zawarty w punktach zaklasyfikowanych jako należące do nosa. . . . .	97
10.12 Rozkład odległości między punktami charakterystycznymi, a środkami regionów w obrazach grupy 1 (GavabDB). . . . .	98
10.13 Rozkład odległości między punktami charakterystycznymi, a środkami regionów w obrazach grupy 2 (Mechatronika). . . . .	98