System automatycznych pomiarów rynometrycznych (6)

Tomasz Kuśmierczyk

Studenckie Koło Naukowe Cybernetyki, Politechnika Warszawska

Streszczenie: Celem projektowanego systemu jest analiza i rozpoznawanie obrazów trójwymiarowych twarzy. Wykorzystując dostępne metody analizy i narzędzia algorytmiczne, dąży się do pozyskania z danych pochodzących ze skanerów 3D informacji dotyczących wymiarów nosa. Artykuł prezentuje innowacyjny pomysł zastosowania deskryptorów punktów w połączeniu z metodami uczenia maszynowego do detekcji wybranych obszarów w skanach 3D. W pracy opisano algorytmu i omówiono różne jego aspekty, pokazano również praktyczne rezultaty zastosowania algorytmu do analizy twarzy.

Słowa kluczowe: spin image, uczenie maszynowe, rozpoznawanie twarzy, chmura punktów

Wstęp

Poprzedni artykuł był poświęcony opisowi praktycznego zastosowania deskryptorów punktów. Pokazano, jak wykorzystując to narzędzie uzyskać praktyczne rezultaty w rozpoznawaniu obrazów. Pominięto jednak kilka istotnych, czasem wręcz kluczowych, aspektów praktycznej realizacji systemu. Zostaną one wyjaśnione w tej części. Opisane algorytmy są użyteczne nie tylko w kontekście opisywanego systemu, ale są podstawowe w przetwarzaniu i rozpoznawaniu obrazów 3D.

Pokazana zostanie najlepsza znana metoda [1] pozwalająca na estymację wektorów normalnych do powierzchni w danych trójwymiarowych. Wyjaśnione zostanie działanie algorytmu redukcji wymiarowości danych PCA. Pokazane zostaną wyniki praktycznych eksperymentów.

Dalej opisany zostanie algorytm filtracji danych, który znalazł swoje zastosowanie zarówno przy wstępnym przetwarzaniu danych jak i przy przygotowywaniu danych uczących. Koncepcyjnie jest on bardzo zbliżony do algorytmu klasteringu danych – DbScan.

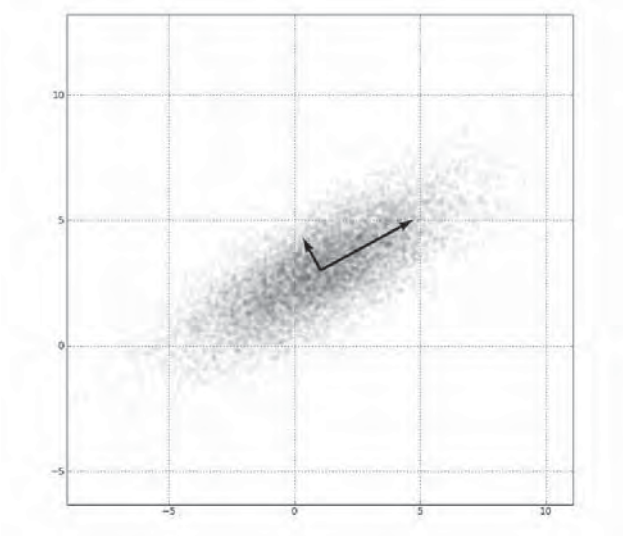
Trzecia część artykułu poświecona zostanie procedurze wyboru podzbioru punktów z chmury punktów. Opisano trzy algorytmy o różnych własnościach. Ich celem jest uzyskanie jak najbardziej równomiernego rozkładu wybranych punktów na całej chmurze przy jednocześnie niezbyt dużym koszcie obliczeniowym.

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. Równoważnym do powyższego problemu jest wyznaczenie płaszczyzny stycznej do powierzchni w punkcie. Mając ją wyznaczona można w łatwy sposób wyznaczyć kierunek do niej prostopadły. Najlepszą metodą [1] służącą do tego celu jest PCA (ang. *Principal Component Analysis*).

PCA jest metodą statystycznej analizy danych, 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 nowej bazy. Wybierając kolejne kierunki uzyskujemy najpierw kierunek największego rozrzutu danych, później drugiego w kolejności itd. Wybierając pierwszych m ($m < k$) kierunków i rzutując dane k-wymiarowe do m-wymiarowej podprzestrzeni uzyskuje się redukcję wymiarowości, której własnością jest to, że minimalizuje ona traconą informację o rozkładzie danych w przestrzeni. Implementację w środowisku MATLAB i ilustrację działania PCA zaprezentowano poniżej.

```
function [features trmx] = pca(features, comp_count)
% features - cechy (w wierszach kolejne probki, w kolumnach cechy)
% comp_count - do ilu cech zredukowac
  mu = mean(features);                                  %obliczenie sredniej
  features = features - repmat( mu, size(features,1), 1);  %odjecie sredniej
  cmx = cov(features);                                  %kowariancja
  [evec eval] = eig(cmx);                               % wektory i wartości własne
  eval = sum(eval);                                     %zamiana macierzy na wektor
  [eval evid] = sort(eval);                             %sortowanie po wartościach wlasnych
  eval = eval(size(eval,2):-1:1);                       %odwrocenie kolejnosci wartosci wlasnych
  evec = evec(:, evid(size(eval,2):-1:1) );             %posortowanie w odwroconej kolejnosci wektorow wlasnych
  trmx = evec(:, 1:comp_count);                         %macierz konwersji
  features = features * trmx;                           %rzutowanie do podprzestrzeni
end
```



Rys. 1. Ilustracja wyznaczania przez PCA kierunków największej wariancji danych [2]

Fig. 1. PCA used for distinction of maximum-data-spread directions [2]

W opisywanym rozwiązaniu wykorzystano PCA do wyznaczenia płaszczyzny stycznej do danych w danym punkcie p. Mając płaszczyznę styczną, wyznaczoną przez 2 wektory - kierunki największej wariancji danych: w i v; kierunek wektora normalnego n wyznaczyć można jako iloczyn kartezjański:

$$n = w \times v.$$

Problem sąsiedztwa

W celu estymacji płaszczyzny stycznej (ozn. Tan_p) do powierzchni obiektu danego jako chmura punktów, brane jest pewne otoczenie punktu p. Przyjęto otoczenie sferyczne:

$$O(p) = \{q: ||p - q|| < r_n\}$$

Im otoczenie O jest mniejsze tym lepszym jego przybliżeniem jest płaszczyzna. Jej położenie wyznaczają dwa kierunki największego rozrzutu punktów, które obliczone mogą zostać właśnie z wykorzystaniem PCA.

Stosując metodę PCA z tak zdefiniowanym otoczeniem punktu należy rozważyć 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.
2. Zbyt duże otoczenie powoduje, ze zatracana jest „lokalność".

Należy dobrać taką wartość r_n, aby zredukować negatywny wpływ obu czynników. W przypadku ogólnym może być ona różna dla różnych regionów chmury punktów np. dla dużych płaskich obszarów powinna być ona większa. Dla obszarów o dużym urozmaiceniu bardzo mała. Niewątpliwie taka analiza istotnie zwiększa złożoność obliczeniową, dlatego w opisywanym rozwiązaniu przyjęto rozwiązanie uproszczone z wartością r_n stałą dla wszystkich punktów chmury.

Eksperymenty pokazały rozbieżność optymalnej wartości promienia dla różnych danych testowych (przypomnijmy, że testy odbywają się na dwóch bazach obrazów: GavabDB i pozyskanej z wydziału Mechatroniki). Wynika to z różnej rozdzielczości obrazów w obu bazach. W bazie GavabDB dla tego samego promienia otoczenia uzyskuje sie znacznie mniej punktów. W celu uzależnienia wartości promienia sąsiedztwa (otoczenia) od gęstości punktów w obrazie zaproponowano metodę estymacji gęstości powierzchniowej punktów.

Estymacja gęstości powierzchniowej punktów

W celu oceny gęstości powierzchniowej punktów s (liczby punktów przypadających na jednostkę powierzchni) w obrazach zaproponowano algorytm zrandomizowany:

1. losowo wybierz podzbiór P punktów obrazu;
2. dla każdego z wybranych punktów p w P, policz liczbę punktów sąsiednich q w sferycznym otoczeniu p: $k_p = \#\{q: ||p - q|| < r_e\}$;
3. oblicz estymacje wg wzoru:

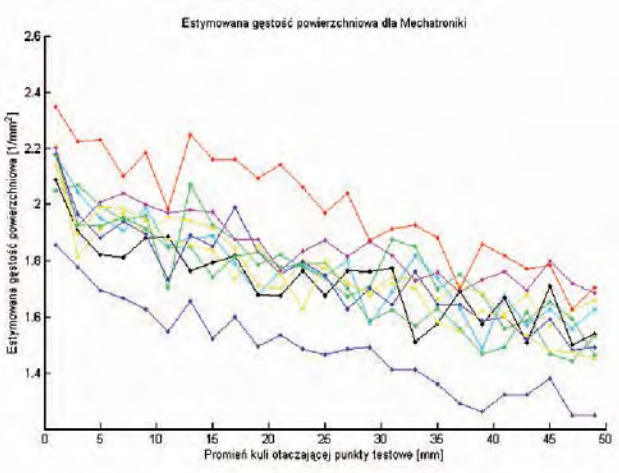
$$s = \frac{średnia_p\left(k_p\right)}{\pi r_e^2}$$

Tak obliczoną wartość s wykorzystać można do obliczania wartości promienia sąsiedztwa używanego w PCA:

$$r_n = \sqrt[2]{\frac{E}{s\pi}}$$

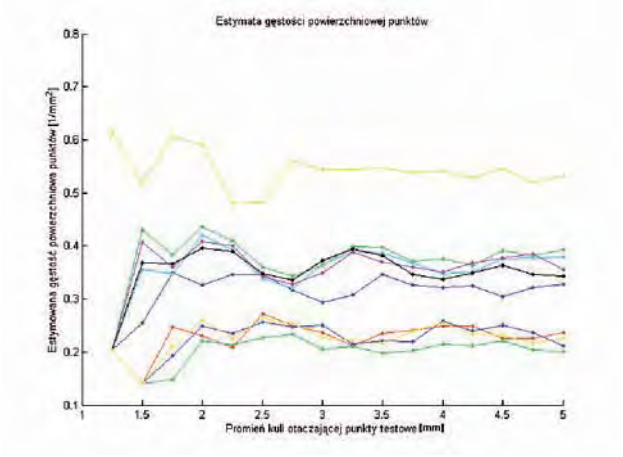
gdzie: r_e – stała, promień obszaru estymacji (np. w eksperymentach przyjęto $r_e = 2$mm); E – stała, pożądana liczba punktów, które powinny się znaleźć w sąsiedztwie punktu p przy wykonywaniu PCA (w eksperymentach przyjęto 20).

Opracowany algorytm obliczania gęstości powierzchniowej przetestowano na 10 obrazach z bazy GavabDB i 10 obrazach z bazy Mechatronika. Przeanalizowano zależność uzyski-



Rys. 2. Zależność estymowanej gęstości powierzchniowej punktów s od promienia otoczenia r_e dla 10 obrazów z bazy Mechatroniki

Fig. 2. Estimated density of points ***s*** vs. neighborhood radius r_e calculated on 10 images (Mechatronika database)



Rys. 3. Zależność estymowanej gęstości powierzchniowej punktów *s* od promienia otoczenia r_e dla 10 obrazów z bazy GavabDB

Fig. 3. Estimated density of points *s* vs. neighborhood radius r_e calculated on 10 images (GavabDB database)

wanego wyniku od wartości r_e. Wyniki pokazują poniższe wykresy. Potwierdzona została uwaga o większej gęstości punktów w modelach w drugiej z baz. Dodatkowo na wykresach widoczne są dwa problemy jakie rozważyć należy dobierając parametr r_e dla algorytmu:

- wraz ze wzrostem r_e wartość *s* maleje. Spowodowane jest to tym, że część losowanych punktów znajduje się blisko brzegu analizowanego obiektu. Zwiększając promień sąsiedztwa coraz większy obszar sfery nie zawiera w ogóle punktów powierzchni.
- dla zbyt małych wartości r_e uzyskiwany wynik jest niestabilny. Powodem tego jest, to, że w otoczeniu jest niewielka liczba punktów i pojawienie się każdego kolejnego przy zwiększaniu r_e wpływa istotnie na wynik.

Korekta zwrotów wektorów normalnych

Stosując metodę PCA można wyznaczyć kierunki wektorów normalnych do powierzchni. Nie jest jednak możliwe ustalenie zwrotów (orientacji powierzchni): do wewnątrz/na zewnątrz. Jeden z najprostszych znanych sposobów ich unifikacji polega na tym, ze zwrot wektora n_p jest korygowany względem kierunku od punktu ciężkości chmury punktów do punktu p (w którym obliczany był wektor n_p). Algorytm jest następujący:

1. oblicz wektor pomocniczy k skierowany od punktu ciężkości obrazu do punktu p: $k = p - centroid(Obraz)$ gdzie: $centroid(Obraz)$ – środek ciężkości obrazu.
2. oblicz znak iloczynu skalarnego między k i n_p: $znak = sign(k \cdot n_p)$.
3. jeśli $znak = -1$ to odwróć wektor normalny: $n_p = -n_p$

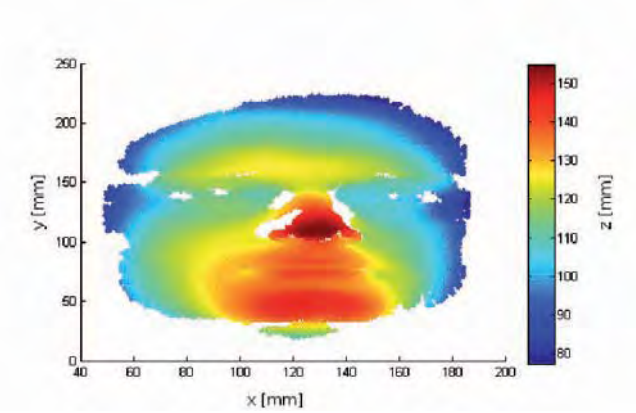
Pełny schemat wyznaczenia wektorów normalnych

Pełna wersja algorytmu obliczającego wektory normalne składa się z 4 kroków. Zaprezentowana została na poniższym schemacie. Na wejściu podawane są chmura N punktów i lista M punktów dla których budowane będą w dalszych fazach deskryptory. W wyniku działania metody obliczanych jest odpowiadające im M wektorów normalnych.

W pierwszym kroku wykonywana jest powyższa procedura estymacji gęstości powierzchniowej i obliczenia promienia r_n. Tak obliczony promień używany jest następnie do wyznaczenia sąsiedztwa podanych M punktów. Dla każdego punktu p z M punktów wybierany jest zbiór jego sąsiadów q. W oparciu o nie wykonywana jest analiza PCA i następnie wyznaczany jest kierunek wektora normalnego do powierzchni w punkcie p. Na koniec, w kroku 4 wykonywana jest unifikacja wektorów normalnych.

Rys. 4. Schemat algorytmu estymacji kierunków wektorów normalnych

Fig. 4. Estimation of normal vectors flow chart

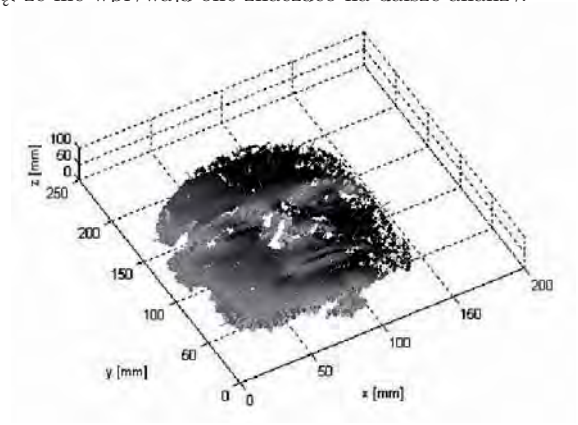


Rys. 5. Ilustracja działania estymacji wektorów normalnych – obraz wejściowy

Fig. 5. Estimation of normal vectors – input image

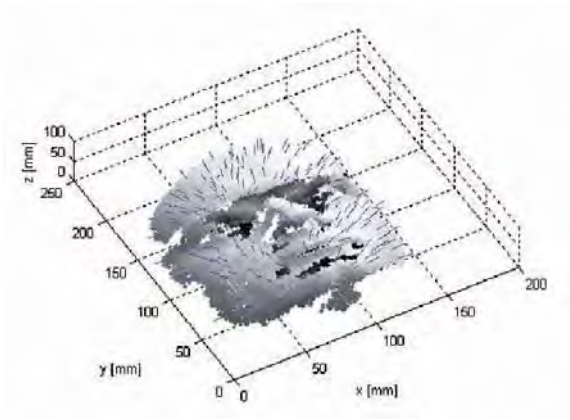
Działanie algorytmu ilustrują poniższe rysunki. Na pierwszym z nich zaprezentowano obraz 3D twarzy, który podawany jest działaniu algorytmu (chmura N punktów). Kolejny przedstawia wynik działania pierwszych trzech faz działania procedury. Ostatni prezentuje wynik końcowy. Obli-

czone wektory normalne wykorzystano do obliczenia świateł w wizualizowanych danych. Na drugim obrazku widać błędnie obliczone oświetlenie co spowodowane jest tym, że część wektorów normalnych skierowanych jest do wewnątrz, a część na zewnątrz. Jak widać na ostatnim obrazku 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.



Rys. 6. Ilustracja działania estymacji wektorów normalnych – obraz przed korektą

Fig. 6. Estimation of normal vectors – before correction



Rys. 7. Ilustracja działania estymacji wektorów normalnych – obraz po korekcie

Fig. 7. Estimation of normal vectors – after correction

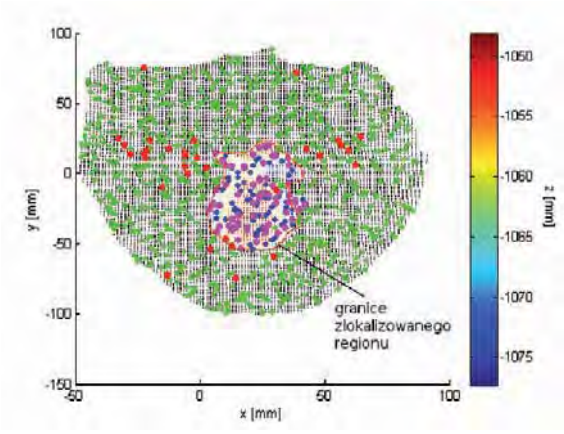
Filtracja danych

Opisana metoda filtracji danych zastosowana może zostać do wstępnego przetwarzania danych celem usunięcia błędnych punktów. Może też zostać użyta do korekty błędnie dopasowanych deskryptorów w algorytmie lokalizacji regionów opisanych w poprzedniej części. Koncepcyjnie przypomina ona bardzo algorytm DBScan, a zaproponowana została w [3]. Składa się ona z 2 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: $||p\text{-}q|| < D$. 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 wielkość (liczba punktów) jest mniejsza od wartości progowej: $|Grupa| < S$. W założeniu, w ten sposób, powinny zostać usunięte pojedyncze błędne punkty i grupy punktów.

W wersji oryginalnej parametry D i S były wartościami stałymi. Takie podejście jednak uzależnia zachowanie się algorytmu od rozdzielczości 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. Również wartość S można obliczać jako procent liczby wszystkich punktów analizowanej chmury.

Oczywistym jest użycie powyższego algorytmu do usunięcia niepoprawnych, nie połączonych z innymi, punktów w danych wejściowych. Mniej oczywiste jest zastosowanie do filtracji danych uczących w algorytmie lokalizacji regionów. Zilustrowane zostało ono na poniższym rysunku. Początkowo punkty (ich deskryptory) w kolorze czerwonym dopasowane zostały do punktów wzorca. Jednak nie są one połączone z punktami oznaczonymi na niebiesko – tymi które faktycznie należą do poszukiwanego regionu. W wyniku działania powyższego algorytmu zostają one odrzucone. Należy zmienić ich etykietowanie.



Rys. 8. Ilustracja użycia filtracji w algorytmie lokalizacji regionów

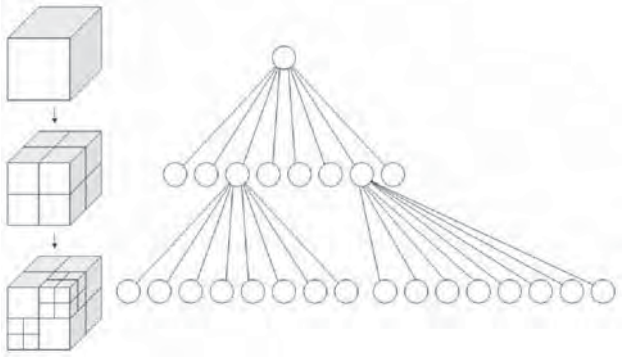
Fig. 8. Data filtering applied for region localization algorithm

Przykład pokazuje, że algorytm pozwala wyeliminować dużą część błędów i dzięki temu narzuca mniejsze wymagania na późniejsze wykorzystanie tak przygotowanych danych.

Wybór punktów

Przy zastosowaniu deskryptorów do dopasowywania obiektu (podzbioru punktów w obrazie 3D) do wzorca, kwestią podstawową jest wybór punktów dla których zostaną obliczone wektory normalne, a później deskryptory punktów. Ważne dla rozpoznawania obrazów jest żeby miały one wyróżniającą charakterystykę. Jedną z możliwości jest wybór punktów o maksymalnej krzywiźnie. Problemem jest koszt obliczania krzywizny. Znacznie prostszym rozwiązaniem, a również dającym dobre wyniki jest taki wybór punktów aby były one w miarę równomiernie rozłożone na powierzchni chmury punktów. Rozważyć można 3 sposoby rozwiązania tego problemu:

1. W sposobie nr 1 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.
   Dla poprawy rozkładu punktów na powierzchni chmury punktów zastosować można wstępne pogrupowanie punktów na podstawie ich położenia. Następnie wybrać należy reprezentanta z każdej z tak powstałych grup. Takie rozwiązanie zapewnia, że niezależnie od zależności między indeksami punktów a ich położeniem w przestrzeni, uniknie się sytuacji, kiedy wybrane zostają punkty położone bardzo blisko siebie, a powstają regiony, z których nie zostaje wybrany żaden reprezentant.
2. Pierwszą metodą grupowania punktów jest algorytm k-średnich. Analizowany obraz 3D dzielony jest losowo na M grup. Następnie dla każdej grupy znajdowany jest centroid. W kolejnym kroku punkty z chmury są przyporządkowywane do najbliższego centroidu i dla nowego przyporządkowywania ponownie wyliczane są centroidy. Algorytm powtarza się. Warunkiem stopu jest ustabilizowanie się położenia centroidów. Znany jest dowód zbieżności algorytmu.
3. Wadą algorytmu k-średnich jest duży koszt obliczeniowy: $O(N^{3M})$. Podobnie dobre rezultaty dla dużych wartości M można uzyskać znacznie tańszym kosztem. Aby z jednej strony zredukować koszt obliczeniowy, a jednocześnie uzyskać dobry rozkład wybieranych punktów, w [4] zaproponowano algorytm grupowania, oparty o koncepcję drzew ósemkowych.



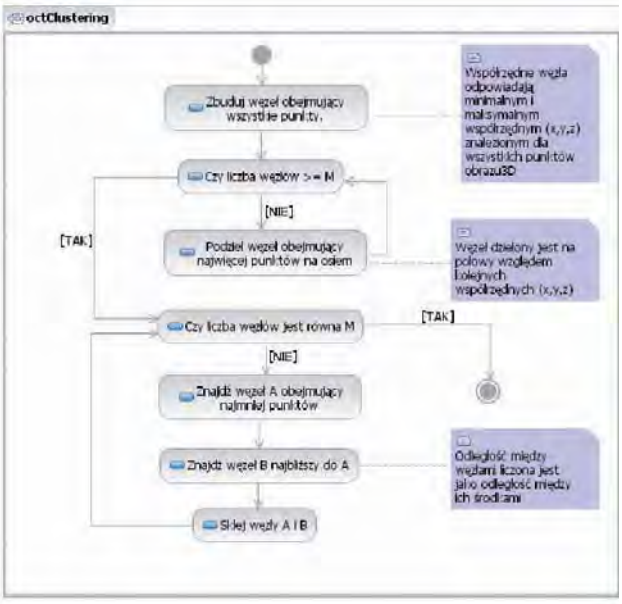
Rys. 9. Drzewo ósemkowe: po lewej – rekursywny podział sześcianu, po prawej – drzewo ósemkowe odpowiadające podziałom [5]

Fig. 9. Octree – on the left: recursive split of the cube; on the right: tree [5]

Opis drzew ósemkowych zaczerpnięty z [5]:

„Drzewo ósemkowe (ang. *octree*) to stosowana w grafice komputerowej struktura danych będąca drzewem, używana do przestrzennego podziału trójwymiarowej przestrzeni na mniejsze, regularne części.

Konstrukcja drzewa ósemkowego polega na otoczeniu całości sceny trójwymiarowej sześcianem (lub prostopadłościanem), który następnie dzielony jest na osiem mniejszych, a następnie każdy z nich na osiem kolejnych itd. – proces podziału ma charakter rekurencyjny."



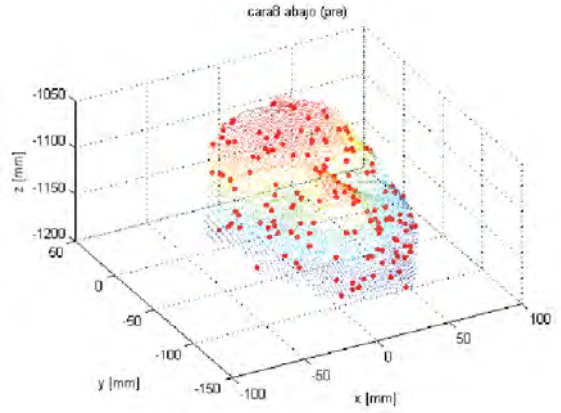
Rys. 10. Schemat algorytmu wyboru podzbioru punktów w oparciu o drzewa ósemkowe

Fig. 10. Flow chart of the algorithm of sampling cloud of points using octrees

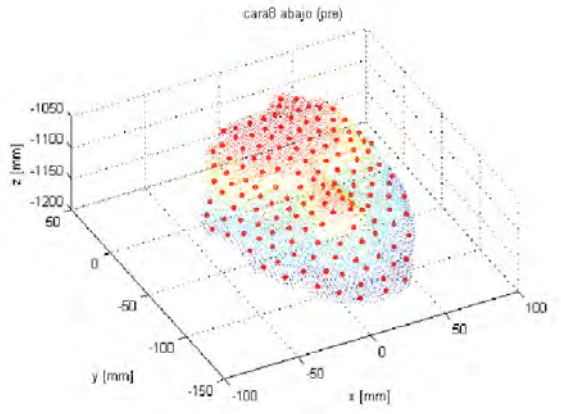
Zaproponowany algorytm działa w sposób bardzo zbliżony do generacji drzew ósemkowych. Najpierw wszystkie punkty chmury otaczane są przez pojedynczy prostopadłościan. Następnie sprawdzany jest warunek stopu, tj. czy osiągnięto już co najmniej M węzłów drzewa (i zawartych w nich M grup punktów). Jeśli nie, to z wszystkich dostępnych węzłów wyszukiwany jest ten, który zawiera najwięcej punktów (chcemy osiągnąć podział na grupy w przybliżeniu równych ślicznościach) i jest on dzielony na 8 węzłów. Podział następuje w środkach każdego z boków prostopadłościanu.

W drugiej fazie algorytmu mamy co najmniej M węzłów drzewa. Może ich być nieznacznie (mniej niż M+8) więcej. Ponieważ chcieliśmy ich mieć dokładnie M, to następuje redukcja liczności. Wybierany jest węzeł o najmniejszej liczbie zawartych w nim punktów. Następnie znajdowany jest węzeł o środku położonym najbliżej do jego środka. Węzły są sklejane. W wyniku tego kroku może nastąpić zaburzenie struktury drzewa, jednak dla docelowego zastosowania nie ma to większego znaczenia.

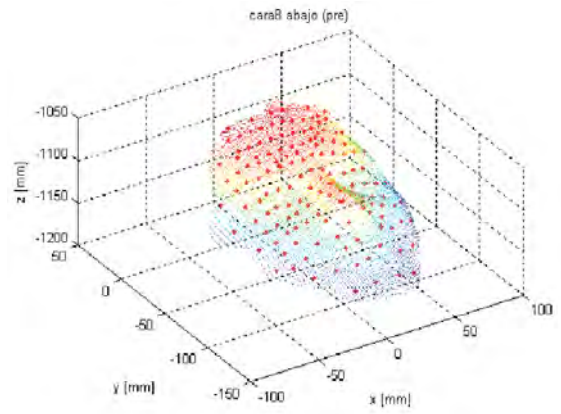
Wyniki działania poszczególnych metod wyboru punktów przedstawiają rysunki. Widać że dla użycia wersji 1 deskryptory zostały rozmieszczone losowo po powierzchni obrazu. Nie rozkładają się one jednak równomiernie. Występują



Rys. 11. Przykład działania algorytmu wyboru punktów: wersja nr 1
Fig. 11. Sampling of points algorithm: version 1



Rys. 12. Przykład działania algorytmu wyboru punktów: wersja nr 2
Rys. 12. Sampling of points algorithm: version 2



Rys. 13. Przykład działania algorytmu wyboru punktów: wersja nr 3
Fig. 13. Sampling of points algorithm: version 3

obszary o większym, jak i o mniejszym zagęszczeniu wybranych punktów. Dla wersji z k-means rozkład punktów jest bardzo dobry. Pamiętać należy jednak, że zostało to okupione długim czasem obliczeń. Ostatni z rysunków przedstawia wyniki działania algorytmu bazujacego na grupowaniu z wykorzystaniem drzew ósemkowych. Osiągnięto kompromis między jakością rozmieszczenia punktów na powierzchni a złożonością obliczeniową.

Bibliografia

1. Klasing K., Althoff D., Wollherr D., Buss M.: *Comparison of Surface Normal Estimation Methods for Range Sensing Applications ICRA 09,* IEEE Conference on Robotics and Automation, p. 3206–3211.
2. [http://en.wikipedia.org/wiki/Principal_component_analysis] – *Principal component analysis.*
3. Sitnik R.: *A fully automatic 3D shape measurement system with data export for engineering and multimedia systems. PhD thesis*, Politechnika Warszawska, Warszawa, 2002.
4. Kuśmierczyk T.: *Deskryptory punktów w analizie morfologicznej obrazów trójwymiarowych twarzy.* Praca inżynierska, Politechnika Warszawska, Warszawa, 2010.
5. [http://pl.wikipedia.org/wiki/Drzewo_%C3%B3semkowe] – *Drzewo ósemkowe.*

Automatic nose measurement system, part 6

Abstract: The purpose of designed system is to analyze and recognize three-dimensional face images. Using known techniques, algorithms and tools I am aiming to retrieve nose parameters directly from 3D scans. Current part is devoted to describe three important procedures that greatly help in 3D images recognition. First of them is estimation of normal vectors using Principal Component Analysis. Another one is bad points' filtering. Using direct and indirect connectivity points are clustered into sub-clouds and then verified. Last procedure is used to sample cloud of points. Shown algorithms allows for obtaining samples of desired characteristics.

Keywords: PCA, normal vector estimation, filtering, resampling, cloud of points

Tomasz Kuśmierczyk

Ukończył studia pierwszego stopnia na Wydziale Elektroniki i Technik Informacyjnych PW na kierunku Informatyka. Kontynuuje studia na poziomie magisterskim na Wydziale Matematyki, Informatyki i Mechaniki UW. Jednocześnie studiuje na Wydziale Fizyki UW na kierunku Fizyka. Zainteresowania naukowe: sztuczna inteligencja i uczenie maszynowe w zastosowaniu do analizy i rozpoznawania obrazów.

e-mail: tk290810@okwf.fuw.edu.pl