SPIS TREŚCI

1. Algorytmy rozpoznawania obrazu

1.1 Wstęp

Informacja zawarta w obrazach ma charakter przestrzenny (dwuwymiarowy) i relatywny (np. informacja o kontraście jest istotniejsza od absolutnych wartości pikseli). W przeciwieństwie do ludzi, dla komputerów jest ona niejasna. Obraz cyfrowy jest dla nich jedynie niezrozumiałym ciągiem liczb niosących informacje o chrominancji i luminancji w danym punkcie sceny. Takie nieustrukturyzowane dane w formie mapy pikseli do dalszej analizy wymagają zazwyczaj przetwarzania wstępnego. Dla potrzeb rozpoznawania obrazy przekształcane są do zwięźlejszej i bardziej informatywnej postaci opisującej ich zawartość. Uwzględniając różne właściwości obrazu można zbudować odmienne reprezentacje np. reprezentacja na podstawie cech niskopoziomowych, reprezentacja semantyczna, piramidalna reprezentacja przestrzenna itd.

Duża zmienność świata rzeczywistego wprowadza dodatkową zawiłość do skomplikowanego już procesu rozpoznawania. Przykładowo, to samo drzewo w zależności od pory roku diametralnie zmienia swój wygląd – od gołych gałęzi podczas zimy (nieraz przysypanych śniegiem), poprzez kwitnące pąki wiosną, aż do gęstych zielonych liści latem i żółto-czerwonych kolorów jesieni. Może być oglądane z odległej perspektywy lub w mocnym zbliżeniu, co również wpływa na jego postrzeganie. Mimo tych utrudnień człowiek z łatwością rozpozna je jako drzewo, podczas gdy komputerowa interpretacja może być bardzo nieoczywista. Inne aspekty stanowiące wyzwanie dla wizji komputerowej to m.in.:

- oświetlenie – różne warunki oświetleniowe mogą utrudniać poprawne rozpoznanie. Zasadniczo różnić się będą zdjęcia wykonane w dzień lub nocą. Efekty nasycenia (prześwietlenia) i niedoświetlenia konkretnych obszarów obrazu powodują utratę informacji o rzeczywistych wartościach natężenia (wartości wykraczające poza zakres dynamiki aparatu są obcinane do wartości minimalnej i maksymalnej). Cienie i refleksy świetlne niekorzystnie wpływają na identyfikację obiektów zwłaszcza metodami segmentacji i analizy regionów;

- zmienność wewnątrzklasowa – brak precyzyjnych definicji klas skutkuje dużymi różnicami pomiędzy ich instancjami (rys. 2.1a). Ponieważ klasy są bardzo ogólne reprezentacje obrazów używane w procesie klasyfikacji nie mogą być zbyt szczegółowe (wymagane są podejścia dobrze generalizujące wszystkie możliwe instancje danej klasy do podobnej postaci);

- zmienność międzyklasowa – z drugiej strony, mając na uwadze zmienność wewnątrzklasową, należy również zdawać sobie sprawę z podobieństw między klasami, które niekiedy są bardzo trudne do odróżnienia jak np. w przypadku klas „fortepiany” i „klawesyny” (rys. 2.1b). Rozróżnianie podobnych klas wymaga zatem reprezentacji obrazów o odpowiednim poziomie szczegółowości, co jest antagonistyczne wobec generalizacji pożądanej w poprzednim podpunkcie;

- przekształcenia geometryczne – w obrazach wejściowych często występują przekształcenia zaburzające ich geometrię, ale zachowujące pierwotną informację. Należą do nich translacje, obroty, skalowania czy rzuty perspektywiczne. W celu poprawy rezultatów algorytmy rozpoznawania powinny być na nie niewrażliwe;

- inne niekorzystne zjawiska takie jak: częściowe przesłanianie obiektów innymi elementami sceny, zmiana punktu widzenia sceny, różnorodne pozy w przypadku obiektów żywych, problemy na poziomie akwizycji obrazów np. szumy czy rozogniskowanie (nieostrość).

W technice wizyjnej analogowy obraz rzutowany na płaszczyznę światłoczułą przetwornika optoelektronicznego jest reprezentowany przez dwuwymiarową funkcję L(x,y), której argumenty x i y opisują powierzchniowe współrzędne punktu obrazu, zaś wartość funkcji określona jest przez poziom jasności obrazu (luminancję - w odróżnieniu od chrominancji, oddającej walor barwny).

Analiza obrazu przez system komputerowy wymaga przetworzenia z postaci analogowej na postać cyfrową – dokonuje się tego przez dyskretyzację i kwantyzację obrazu. Dyskretyzacja obrazu jest realizowana przez dwuwymiarowe próbkowanie w ściśle określonych miejscach przestrzeni (zazwyczaj w węzłach siatki prostokątnej). Kwantyzacja polega na podziale ciągłego zakresu wartości jasności na przedziały i przypisaniu każdemu punktowi wybranej wartości dyskretnej reprezentującej dany przedział.

W ten sposób funkcja L(x,y) o argumentach zmieniających się w sposób ciągły zostaje zamieniona na macierz L(m,n) o M wierszach i N kolumnach, której elementy zawierają skwantowane poziomy jasności. Elementy macierzy, w związku z liniowym charakterem pamięci komputera są ciągiem liczb. Obrazy poddawane algorytmom przetwarzania obrazów konwertowane są do skali szarości. Dlatego wartości macierzy przyjmują wartości od 0 do 255 dla skali szarości (reprezentacja odcieni szarości), lub od 0 do 1 dla postaci binarnej obrazu, gdzie zazwyczaj, 0 – kolor czarny, 1 – kolor biały). W przypadku obrazów kolorowych elementy macierzy przyjmują trzy wartości (w zależności od przestrzeni kolorów), wyjątkiem jest model przestrzeni RGBA, który posiada czwarty parametr – kanał „alpha", pozwalający na uzyskanie efektu przeźroczystości.

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix},$$

Którą zapisujmy jako ciąg liczb $p_{11}, p_{12}, p_{13} \ldots, p_{23}, p_{33}$ – elementy macierzy zapisywane wierszami od lewej do prawej. Dostęp do elementów macierzy obrazuje następujący wzór matematyczny

$$p_{xy} = P[y \cdot w + x].$$

gdzie w – szerokość obrazu (macierzy).

Do identyfikacji obrazów zazwyczaj stosuje się metody wykorzystujące pojęcie przestrzeni cech (ang. feature space). Jest to n-wymiarowa przestrzeń, w którą odwzorowywane są obrazy. Odwzorowanie następuje za pomocą ustalonego zestawu cech. Cechy to pewne właściwości obrazu dobrze odzwierciedlające jego zawartość. Ekstrakcja cech obrazu odbywa się bez rozumienia jego treści. Deskryptor cechy (ang. feature descriptor) jest numeryczną reprezentacją określonej cechy obrazu. Tę samą cechę można przekształcić na różne sposoby do postaci numerycznej otrzymując za każdym razem inny deskryptor. Zbiór wartości cech danego obrazu nazywany jest jego wektorem cech (ang. feature vector). Pod kątem rozpoznawania taki rezultat odwzorowania obrazu w przestrzeń cech niesie informację dużo bardziej zrozumiałą i użyteczną dla komputera niż pierwotna mapa pikseli.

1.1.1. Algorytmy ogólne – detekcja twarzy

1.1.1.1. SIFT- Scale Invariant Feature Transform

SIFT jest algorytmem bazującym na cechach, służącym zarówno do detekcji, jak i opisu cech lokalnych. Algorytmy takie najpierw znajdują punkty szczególne w danych obrazach. Następnie dopasowywują je parami i na podstawie ich położenia wyliczają wektor przemieszczenia. Jakość algorytmu w znacznym stopniu zależy od jakości znalezionych punktów szczególnych i ich opisu. Algorytm powinien wykrywać te punkty w odpowiednich miejscach niezależnie od zmian w naświetleniu, położeniu i orientacji obrazu, a także w pewnym stopniu od zmian punktu widzenia czy skali.

Algorytm ten jest obecnie najpowszechniej stosowany przy tworzeniu obrazów panoramicznych ze względu na jego bardzo dobre wyniki w porównaniu do innych tego typu algorytmów (Mikolajczyk & Schmid). Operuje na obrazach w skali szarości.

Algorytm SIFT można podzielić na cztery główne kroki:

1. Budowanie przestrzeni skalowej.

Na tym etapie wykrywane są ekstrema obrazka za pomocą różnic rozmyć Gaussowskich. Na tych ekstremach znajdują się potencjalne punkty kluczowe.

Podczas procesu przetwarzania danych z nieznanej sceny nie jesteśmy w stanie określić skali znajdujących się na nich obiektów. Rozwiązaniem tego problemu jest jednoczesne rozpatrywanie wszystkich możliwości, czyli konstrukcja tzw. piramidy przestrzeni skali (ang. scale space pyramid). Piramida jest podzielona na oktawy. Oktawa zawiera zdjęcia o tej samej rozdzielczości, ale wzrastającym odchyleniu standardowym filtru Gaussa. W kolejnych oktawach (poziomach piramidy) zmniejsza się wielkość obrazów.

Operacja tworzenia piramidy polega na zastosowaniu filtru Gaussa na obrazie i realizowana jest poprzez splot obrazu z operatorem Gaussa:

$$L(x, y, \sigma) = I(x, y) * G(x, y, \sigma)$$

gdzie:

L – nowy rozmyty obraz,

I – obraz wejściowy,

G – operator Gaussa

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

ze współczynnikiem skali (rozmycia) σ

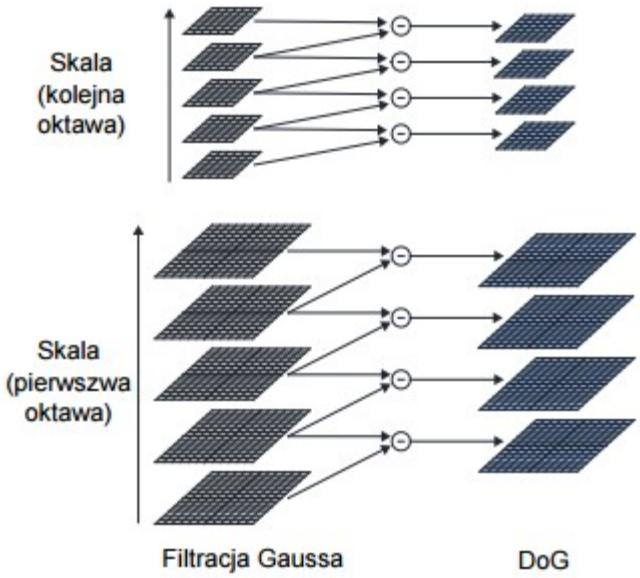
Splot wykonywany jest po zmiennych x i y ze zdefiniowaną wcześniej σ.

W kolejnym kroku obliczane są różnice rozmyć Gaussowskich dwóch kolejnych obrazków, tak zwany Difference of Gaussians (skrót: DoG). Powstaje nowa macierz, w której wartości 0 przyjmują piksele, gdzie obrazy mają taką samą wartość po rozmyciu

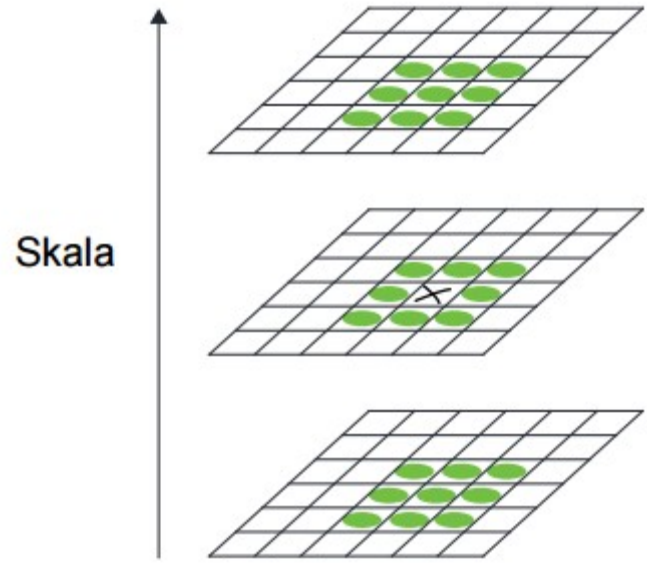
$$[G(x, y, k\sigma) - G(x, y, \sigma)] \cdot I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) = D(x, y, \sigma)$$

Optymalne wartości parametrów to: liczba oktaw = 4, liczba poziomów skali = 5, początkowa wartość σ = 1, 6, k = √ 2

Operacje opisane powyżej można przedstawić w uproszczonej wersji obrazkowej



W momencie kiedy DoG jest wyznaczone, zaczyna się etap wyszukiwania lokalnych ekstremów. Odbywa się to przez porównanie punktu z obrazu DoG z ich sąsiadami. Jeden piksel porównywany jest z 8 sąsiadami na tym samym poziomie skali oraz 9 na poprzednim i następnym poziomie skali (Rysunek 3.2). Jeżeli dany punkt jest lokalną ekstremą to jest to potencjalny punkt kluczowy.



2. Lokalizacja punktów kluczowych.

Wykrywane są punkty kluczowe wśród potencjalnych punktów kluczowych. Zostają również odrzucone te punkty kluczowe, które znajdują się blisko krawędzi zdjęcia.

Punkty kluczowe to niektóre elementy wśród potencjalnych punktów kluczowych. Potencjalnych punktów kluczowych jest bardzo dużo. Spowodowane jest to między innymi szumami na zdjęciu. Zostaną odrzucone te punkty potencjalne, które mało wyróżniają się wśród innych. W tym celu obliczany jest szereg Taylora w przestrzeni skali

$$D(\mathbf{x}) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T\frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}$$

gdzie

$$\mathbf{x} = [x, y, \sigma]^T.$$

Następnie wyznaczane jest położenie ekstremum, dzięki pochodnej funkcji D(x) i zostaje ona przyrównana do wartości 0.

$$\hat{\mathbf{x}} = -\frac{\partial^2 D^{-1}}{\partial \mathbf{x}^2}\frac{\partial D}{\partial \mathbf{x}}$$

Jeśli przesunięcie $\hat{\mathbf{x}}$ jest większe niż 0,5 w którymkolwiek z kierunków, ekstremum leży bliżej innego piksela. W takim przypadku próbkowany punkt jest zmieniany i na nim dokonuje się interpolacja. Wynikowe przesunięcie $\hat{\mathbf{x}}$ jest dodawane do położenia tego punktu, aby otrzymać interpolowane położenie ekstremum. Dodatkowo dużo punktów przy krawędzi obrazka może być traktowane jako punkty kluczowe. Punkty te są odrzucane, gdy spełniają warunek

$$det(\mathbf{H}) \leq 0 \text{ lub } \frac{Tr(\mathbf{H})^2}{det(\mathbf{H})} \geq \frac{(r+1)^2}{r}, \text{ gdzie } \mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \text{ oraz } r = 10.$$

3. Wyznaczanie kierunku punktu kluczowego.

Dzięki temu stadium zapewniona jest niezależność względem obrotu zdjęcia.

Na tym etapie wyznaczana jest orientacja punktu kluczowego. Wyznacza się ją dzięki lokalnym gradientom. Przy pomocy takiego zabiegu uzyskuje się niezależność punktów kluczowych od ich rotacji. W celu wyznaczenia orientacji punktu kluczowego obliczany jest kierunek i wielkość (moc) gradientu. Wyniki przedstawiane są w formie histogramu. Histogram posiada 36 przedziałów. Każdy przedział odpowiada zakresowi 10° . Moc i wielkość gradientu obliczana jest w sposób następujący:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$

$$\theta(x,y) = arctg(\frac{L(x,y+1) - L(x,y-1))}{L(x+1,y) - L(x-1,y)})$$

gdzie m(x, y) to wielkość gradientu, a θ(x, y) to kierunek gradientu wyrażony w stopniach. Następnie odcina się wartości na histogramie, które nie przekraczają 80% wartości najwyższego słupka. Może istnieć kilka kierunków dla jednego punktu na obrazie. Operacja ta jest powtarzana dla wszystkich punktów kluczowych na obrazie.

4. Tworzenie deskryptora punktu kluczowego.

W tym kroku tworzony jest wektor, który opisuje punkt kluczowy. Wektor ten zawiera dane, które były wyznaczone we wcześniejszych etapach.

Punkt kluczowy opisany jest wektorem. Jeżeli wymaga się porównania dwóch obrazów i znalezienia podobieństw pomiędzy nimi należy porównywać ze sobą deskryptory punktów kluczowych tych zdjęć. Im bardziej punkty kluczowe są do siebie podobne, tym wektory te powinny znajdować się bliżej siebie to znaczy, że odległość Euklidesowa pomiędzy wektorami powinna być mała. Porównywanie wielowymiarowych wektorów jest znanym problem związanym ze złożonymi obliczeniami. W celu przyspieszenia obliczeń rekomendowany jest algorytm Najlepszy Koszyk Pierwszy (ang. best-bin-first), który określa najbliżej położonego wektora z wysoką skutecznością i z użyciem ograniczonej ilości obliczeń. W pewnych sytuacjach zdarza się, że drugi najbliżej leżący wektor jest bardzo blisko pierwszego. W takim przypadku oblicza się stosunek odległości pierwszego wektora do drugiego. Jeżeli stosunek ten jest większy niż 0.8 to te punkty są odrzucane. Dzięki temu zabiegowi eliminowanych jest około 90% fałszywych dopasowani.

W momencie kiedy dwa zdjęcia mają wyznaczone punkty kluczowe wraz z ich deskryptorami można znaleźć cechy wspólne dla obydwu obrazków.

1.1.1.2. HOG-histogram of oriented gradients

[10] Deskryptory HOG należą do deskryptorów opisujących kształt i służą do znalezienia w obrazie konkretnego obiektu. Najogólniej rzecz ujmując, deskryptory HOG oparte są na idei zliczania występowania gradientów występujących w tej samej orientacji przestrzennej (pod pewnym kątem) w pewnym ściśle określonym fragmencie obrazu. Gradienty te są liczone w równomiernie rozmieszczonych komórkach (fragmentach obrazu). Dodatkowo, w celu polepszenia jakości wykrywania obiektów stosowana jest lokalna normalizacja kontrastu w nachodzących na siebie regionach.

Główną ideą deskryptorów HOG jest założenie, że wygląd obiektu może zostać opisany za pomocą rozkładu występowania krawędzi pod określonym kątem. Krawędzie występujące w obrazie są uzyskiwane poprzez obliczenie pochodnej kierunkowej wzdłuż odpowiedniego wektora, dlatego w literaturze traktującej o deskryptorach HOG mówi się zamiennie o gradientach.

Od strony implementacyjnej deskryptory obrazu HOG są uzyskiwane przez podzielenie obrazu wejściowego na małe fragmenty zwane komórkami i sporządzenie dla każdej komórki histogramu występowania orientacji krawędzi. Połączenie histogramów obliczonych dla wszystkich komórek stanowi deskryptor obrazu HOG. HOG jest odporny na wszystkie transformacje geometryczne z wyjątkiem rotacji. Oznacza to, że obraz, na którym wykryto dany obiekt, po poddaniu dozwolonej transformacji również będzie zawierał cechy HOG wskazujące na obecność tego obiektu.

Na rysunku przedstawiono realizację HOG (jedna komórka na blok, 16x16 pikseli w komórce, 8 kanałów) dla przykładowego obrazu.

a)

b)

Obliczanie gradientu

Pierwszym krokiem w obliczaniu deskryptorów HOG jest obliczenie gradientu. Gradient liczony jest w dwóch kierunkach – poziomie i pionie, poprzez filtrowanie obrazu wejściowego z zastosowaniem dwóch filtrów:

Filtru poziomego [-1,0,1]

Filtru pionowego $[-1,0,1]^T$

Działanie filtru polega na umiejscowieniu środka danego filtru w każdym pikselu obrazu wejściowego i obliczenie jego nowej wartości poprzez sumowanie trzech pikseli wyznaczonych przez filtr z uwzględnieniem jego wag.

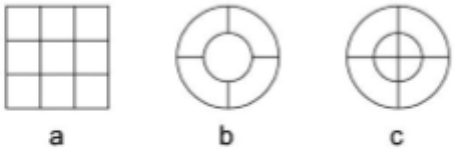
Obliczanie histogramu

Po obliczeniu gradientów dla każdej komórki tworzony jest histogram przedstawiający rozkład gradientów lub inaczej rozkład orientacji krawędzi. W zależności od tego czy kąty przedstawiane są ze znakiem lub bez, histogram przedstawia orientację krawędzi w zakresie 0 – 360° lub 0 – 180°. Histogram przedstawia wartość gradientu we wszystkich pikselach w komórce.

Konstrukcja bloków obrazu

Wspomniane komórki obrazu w celu normalizacji kontrastu są grupowane w większe przestrzenne bloki. Deskryptor obrazu HOG jest wektorem składającym się z obliczonych histogramów we wszystkich blokach. Warto podkreślić, że w oryginalnej implementacji bloki nachodzą na siebie, wobec tego pewne komórki występują więcej niż raz w ostatecznym deskryptorze.

Autorzy w proponują dwie geometrie przestrzenne bloków; bloki prostokątne R-HOG oraz bloki okrągłe C-HOG. Bloki prostokątne R-HOG składają się w siatkę o trzech parametrach: liczba komórek składających się na jeden blok, liczba pikseli w komórce oraz liczba kanałów (przedziałów klasowych) histogramu. W pracy najlepsze wyniki zostały uzyskane dla bloków składających się z dziewięciu komórek (3x3), każda komórka składała się z 6x6 pikseli, a liczba kanałów wynosiła 9. Komórki okrągłe C-HOG występują w dwóch postaciach. Pierwsza składa się z pojedynczej centralnie umieszczonej komórki, podczas gdy w drugiej komórki występują w równych odstępach kątowych. Na rys. X przedstawiona została przestrzenna geometria komórek R-HOG oraz C-HOG w dwóch wariantach.



Przestrzenna geometria komórek R-HOG (a), C-HOG centralnie rozmieszczonej (b) oraz C-HOG z równymi odstępami kątowymi (c).

Autorzy deskryptorów HOG doszli do wniosku, że najlepsze wyniki detekcji uzyskiwane są z zastosowaniem komórek C-HOG z czterema odstępami kątowymi i dwoma odstępami wzdłuż promienia jak na rys. c.

Normalizacja bloków

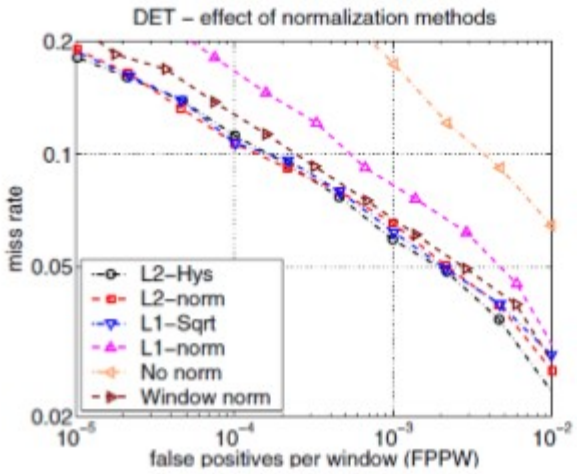
Deskryptory obrazu HOG odróżniają się od innych deskryptorów bazujących na krawędziach obiektów stosowaniem normalizacji po obliczeniu gradientów, zamiast normalizacji obrazu, z którego cechy są ekstrahowane. Normalizowane są poszczególne bloki, składające się z ustalonej liczby komórek w ustalonym ułożeniu przestrzennym. Stosowana normalizacja bazuje na k-normie, która dla n-elementowego wektora jest postaci:

$$\|x\|_k = \sqrt[k]{\sum_{i=1}^{n} x_i^k}.$$

Naturalnie, $\|x\|_{k=2}$ jest normą euklidesową. Wyróżnia się cztery sposoby normalizacji wektora cech HOG, z których każdy opiera się na współczynniku normalizującym w postaci:

1. L2-norm: $f = \frac{v}{\sqrt{\|v\|_2^2 + \varepsilon^2}}$
2. L2-hist: $f = \frac{v}{\sqrt{\|v\|_2^2 + \varepsilon^2}}$, $v \leq 0.2$
3. L1-norm: $f = \frac{v}{\|v\|_1 + \varepsilon}$
4. L1-sqrt: $f = \sqrt{\frac{v}{\|v\|_1 + \varepsilon}}$

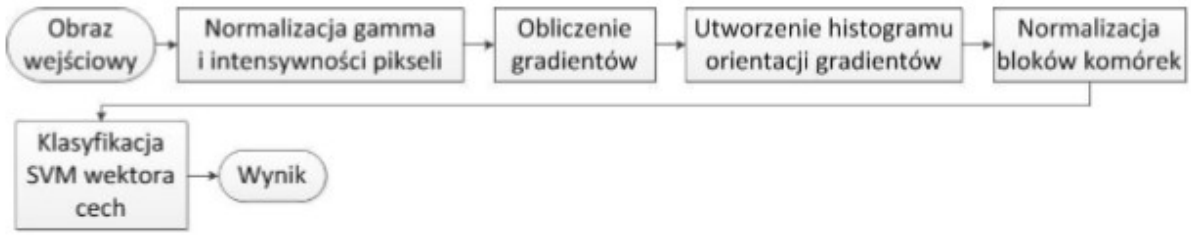
Normalizacje L2-norm, L2-hys oraz L1-sqrt cechują się skutecznością ulepszania działania deskryptorów HOG na podobnym poziomie, podczas gdy L1-norm działa nieco gorzej od pozostałych. Na rysunku przedstawiony został pozytywny wpływ normalizacji wektora cech na działanie detektora, wyrażony we współczynnikach skuteczności detekcji oraz fałszywych pozytywnych wykryć.



Wpływ normalizacji wektora cech obrazu HOG na skuteczność działania deskryptora

Sekwencja detekcji

Sekwencja detekcji obiektów z wykorzystaniem deskryptorów HOG została przedstawiona na schemacie z rysunku. Wstępne przetwarzanie obrazu wejściowego polegające na normalizacji intensywności pikseli i korekcji gamma jest opcjonalne, ponieważ nie wnosi znaczącej poprawy do jakości działania detektora. Obraz zawierający przedmiot detekcji jest poddawany procesowi obliczenia gradientów, tak jak to zostało opisane powyżej. Następnie następuje normalizacja wybraną metodą. Deskryptor obrazu wyliczony dla regionu wyznaczonego przez okno detektora jest następnie poddawany klasyfikacji, np. za pomocą SVM. Wynik klasyfikacji mówi o tym czy w danym miejscu na obrazie znajduje się. bądź też nie, wykrywany obiekt.



1.1.1.3. SURF- Speeded Up Robust Features

SURF jest algorytmem detekcji i opisu obrazu przez punkty charakterystyczne. Jest inwariantny względem obrotu i zmian skali, jednocześnie zachowuje wysoką powtarzalność detekcji punktów obrazu oraz odporność na jego zakłócenia. Zaprezentowany został przez Herberta Bay'a, Tinne Tuytelaars'a i Luca Van Gool. Opisuje rozkład odpowiedzi falki Haar'a (ang. Haar Wavelet) w sąsiedztwie punktów charakterystycznych. Wykorzystywane są tylko 64 wymiary, zmniejszając tym samym czas obliczeń cech.

Metodę zaproponowaną przez Bay'a, Tuytelaars'a i Van Gool można podzielić na trzy główne etapy. Pierwszy etap polega na znalezieniu punktów charakterystycznych (ang. interest points) w obrazie, takich jak narożniki, krawędzie czy skrzyżowania typu „T". Najważniejszą cechą tego etapu jest powtarzalność detekcji, czyli zdolność znalezienia tych samych

punktów charakterystycznych w zmieniających się warunkach otoczenia. Następny etap polega na stworzeniu wektora cech reprezentującego sąsiedztwo każdego punktu charakterystycznego. Deskryptor ten musi być reprezentatywny, a jednocześnie odporny na zaszumienia obrazu, błędy wykrycia oraz geometryczne i fotometryczne deformacje. W ostatnim etapie wektory deskryptora są dopasowywane między różnymi obrazami. Nie jest tu wykorzystywana informacja o kolorze. Dopasowywanie często opiera się na odległości między wektorami, np. odległości Mahalanobisa czy Euklidesa. Wymiar deskryptora ma bezpośredni wpływ na szybkość metody i jej dokładność.

Ze względu na wysoką precyzję, do lokalizacji punktów kluczowych używa się macierzy Hessego. Za jej pomocą wykrywa się cechy o typie „plamki" w miejscach, dla których wyznacznik macierzy osiąga maksimum. Na podstawie wyznacznika wybierana jest również odpowiednia skala. Macierz Hessego

$$\mathcal{H}(p,\sigma)$$

w punkcie $p = (x,y)$ o skali σ zdefiniowana jest następująco:

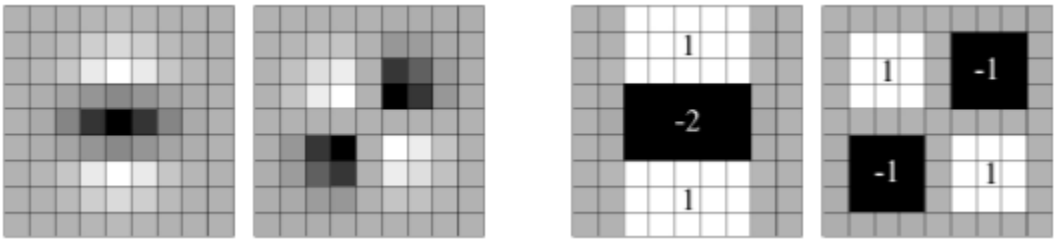
$$\mathcal{H}(p,\sigma) = \begin{bmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{xy}(p,\sigma) & L_{yy}(p,\sigma) \end{bmatrix}$$

gdzie $L_{xx}(p,\sigma)$ jest splotem drugiej pochodnej Gaussianu

$$\frac{\partial^2}{\partial x^2} g(\sigma)$$

z obrazem wejściowym *I* w punkcie *p.* Funkcja Gaussa jest optymalnym wyborem dla analizy w przestrzeni skali. W praktyce musi być jednak zdyskretyzowana i przycięta do rozmiaru filtra.

W algorytmie SURF, stosuje się aproksymację drugich pochodnych funkcji Gaussa (potrzebnych do wyznaczenia macierzy Hessego). W wyniku tej aproksymacji powstają odpowiednie maski filtrów (przedstawione na poniższym rysunku) umożliwiające wykorzystanie obrazów całkowitych do niezwykle wydajnego obliczenia splotu.

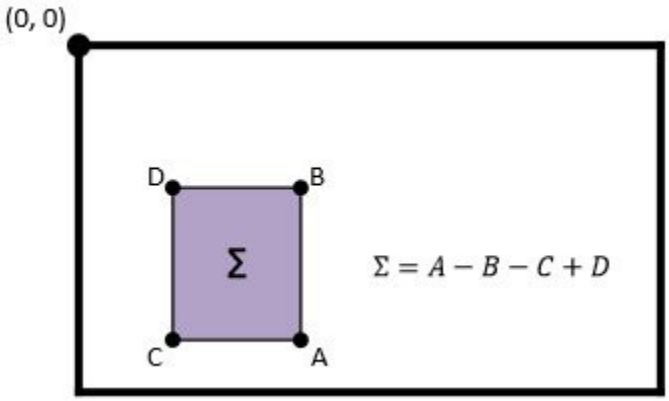


Na rysunku po lewej widać pochodne cząstkowe drugiego rzędu funkcji Gaussa. Po prawej stronie ich aproksymacje używane w algorytmie SURF

Wartość obrazu całkowitego $I_\Sigma(x,y)$ w punkcie (x,y) reprezentuje sumę wszystkich pikseli obrazu wejściowego I wewnątrz prostokąta wyznaczonego przez punkty (0 0) i (x,y):

$$I_\Sigma(x,y) = \sum_{i=0}^{i\leq x}\sum_{j=0}^{j\leq y} I(i,j).$$

Zastosowanie obrazów całkowitych umożliwia szybkie obliczanie filtracji konwolucyjnej (splotowej). Po wyznaczeniu obrazu całkowitego, obliczenie sumy intensywności wewnątrz dowolnego prostokąta wymaga jedynie trzech prostych operacji arytmetycznych (dodawania i odejmowania). Warto podkreślić, że czas tych obliczeń jest niezależny od rozmiaru prostokąta, co szczególnie przyspiesza obliczanie splotu dla masek o dużych rozmiarach.



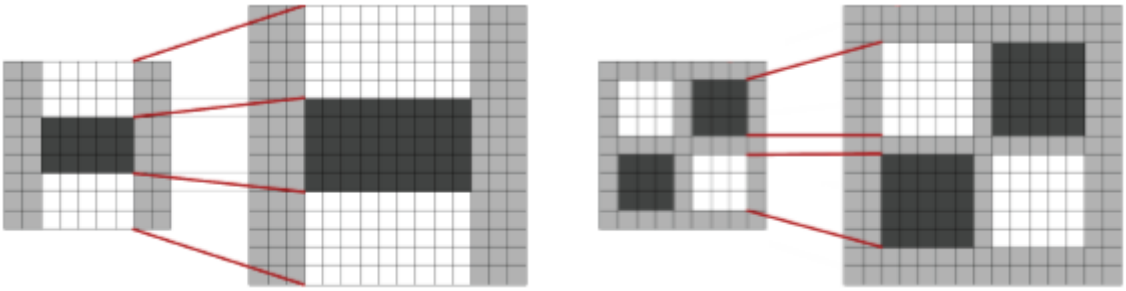
Maski filtrów o rozmiarach 9x9 pikseli przedstawione na powyższym rysunku odpowiadają Gaussianowi o odchyleniu standardowym σ = 1,2 i reprezentują najmniejszą skalę, dla której znajduje się cechy. Ich aproksymacje oznaczane będą symbolami D_{xx}, D_{yy}, D_{xy}. Stosując wprowadzone oznaczenia, można napisać:

$$\det(\mathcal{H}_{approx}) = D_{xx}D_{yy} - (\omega D_{xy})^2$$

Waga ω jest wprowadzona w celu odpowiedniego zbalansowania powyższego wyrażenia. Jej wartość wynosi 0,9.

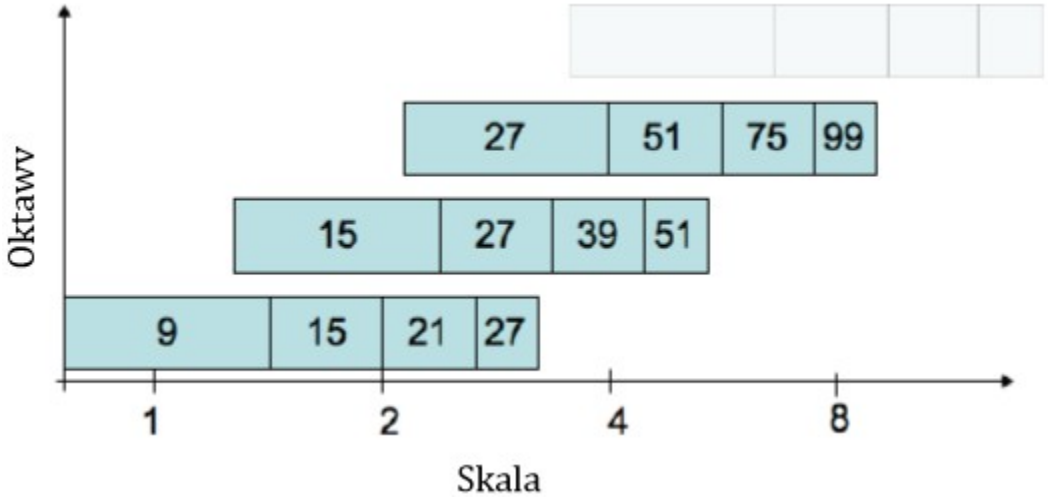
Ponieważ obrazy wejściowe występują w różnych skalach, konieczne jest znajdowanie cech o różnych rozmiarach. W tym celu używa się reprezentacji w przestrzeni skal, którą najczęściej implementuje się w postaci piramidy obrazów o malejącym rozmiarze. Obrazy są wielokrotnie wygładzane filtrem Gaussa i przepróbkowywane, w wyniku czego otrzymuje się obrazy o mniejszym rozmiarze odpowiadające wyższym poziomom piramidy (takie podejście stosowane jest między innymi w algorytmie SIFT). Dzięki zastosowaniu odpowiednich filtrów wraz z obrazami całkowitymi, zamiast używania tego samego filtru na wielokrotnie pomniejszanym obrazie, można posłużyć się coraz większym filtrem bezpośrednio na oryginalnym obrazie nie tracąc przy tym na prędkości filtracji. Z tego powodu w algorytmie SURF przestrzeń skal implementowana jest jako piramida filtrów o rosnącym rozmiarze.

Główną zaletą takiego podejścia jest wysoka wydajność obliczeniowa. Pierwszy poziom piramidy tworzy filtr 9x9, który odpowiada skali s = 1,2 (aproksymuje drugą pochodną Gaussianu o odchyleniu standardowym σ = 1,2). Kolejne poziomy otrzymuje się poprzez stopniowe powiększanie rozmiaru maski, jednocześnie zachowując specyficzną strukturę filtru. Filtry D_{xx} i D_{yy} składają się z trzech obszarów – jednego ujemnego i dwóch dodatnich. Centralny piksel tych filtrów odpowiada środkowi ujemnego obszaru. Aby zachować tą właściwość, każdy z obszarów musi być powiększony o parzystą liczbę pikseli (dzięki czemu wymiary obszarów pozostaną nieparzyste). Z tego powodu każdy z obszarów można powiększyć minimalnie o 2 piksele. Dla filtru składającego się z trzech obszarów oznacza to zwiększenie maski o 6 pikseli, z czego wynika minimalna różnica skali dwóch kolejnych poziomów piramidy.



Filtry D_{yy} (po lewej) i D_{xy} (po prawej) dla dwóch kolejnych poziomów skali (9x9 i 15x15)

Podobnie jak w algorytmie SIFT, przestrzeń skal dzielona jest na oktawy. Każda oktawa składa się z czterech poziomów skali. W sumie oktawa obejmuje przedział przestrzeni skal, w obrębie którego skala wzrasta dwukrotnie. Pierwszą oktawę rozpoczyna filtr 9x9, który odpowiada najmniejszej skali. Kolejne filtry pierwszej oktawy są powiększeniem poprzednika o 6 pikseli. Tym sposobem pierwsza oktawa składa się z filtrów o rozmiarach 9x9, 15x15, 21x21 i 27x27. Każdą następną oktawę rozpoczyna drugi filtr z oktawy poprzedniej. Ze wzrostem oktawy dwukrotnie zwiększa się przyrost rozmiaru filtra (w pierwszej oktawie filtry rosną o 6 pikseli, w drugiej oktawie o 12, w trzeciej o 24 itd.) oraz okres próbkowania obrazu (w pierwszej oktawie odpowiedź filtru obliczana jest dla każdego piksela, w drugiej oktawie dla co drugiego, w trzeciej dla co czwartego itd.). Zwiększenie okresu próbkowania obrazu zmniejsza czas obliczeń i odpowiada zmniejszaniu rozmiaru obrazów w tradycyjnym podejściu. Zgodnie z podanymi regułami druga oktawa składa się z filtrów o rozmiarach 15, 27, 39, 51, a trzeciej odpowiadają rozmiary 27, 51, 75, 99. Jeśli rozmiar obrazu nadal jest większy od powstałych tworzy się czwartą oktawę i kontynuuje analizę w przestrzeni skal.



Odpowiedź filtru jest normalizowana w stosunku do jego rozmiaru. Zapewnia to stałą wartość normy Frobeniusa dla każdego rozmiaru filtra. Dzięki temu filtry są znormalizowane względem skali i dodatkowe ważenie odpowiedzi nie jest potrzebne.

Przy pomocy piramidy filtrów dla wszystkich pikseli i skal obrazu oblicza się aproksymowany wyznacznik macierzy Hessego

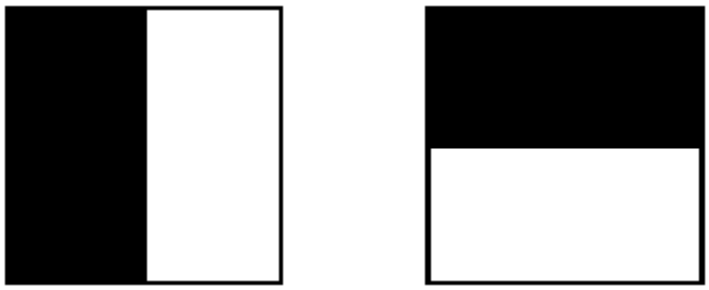
$\det(\mathcal{H}_{approx})$.

W celu ustalenia lokalnych maksimów wyznacznika usuwa się niemaksymalne punkty (ang. non-maximum suppression) w każdym 27 pikselowym otoczeniu w przestrzeni skali i obrazu [*x,y,s*] (przyjmuje ono postać sześcianu 3x3x3). Dokładne położenie maksimum jest ostatecznie interpolowane metodą zaproponowaną przez Browna i Lowe'a i stanowi poszukiwaną lokalizację punktu kluczowego. W przypadku algorytmu SURF interpolacja w przestrzeni skal jest szczególnie ważna z uwagi na relatywnie duże różnice w skali pomiędzy pierwszymi poziomami każdej oktawy.

Opisana dotychczas detekcja jest dopiero pierwszym krokiem w rozpoznawaniu obrazów. Aby zrobić użytek ze znalezionych cech trzeba w jakiś sposób dopasować je między sobą np. w celu wyznaczenia wspólnych obszarów dla dwóch różnych obrazów. Niezbędne jest odpowiednie opisanie punktu kluczowego na podstawie wyglądu jego otoczenia, które będzie niezmiennicze względem przekształceń geometrycznych lub różnych warunków oświetlenia, przy jednoczesnym zachowaniu właściwości rozpoznawczych. W tym celu opracowano deskryptor.

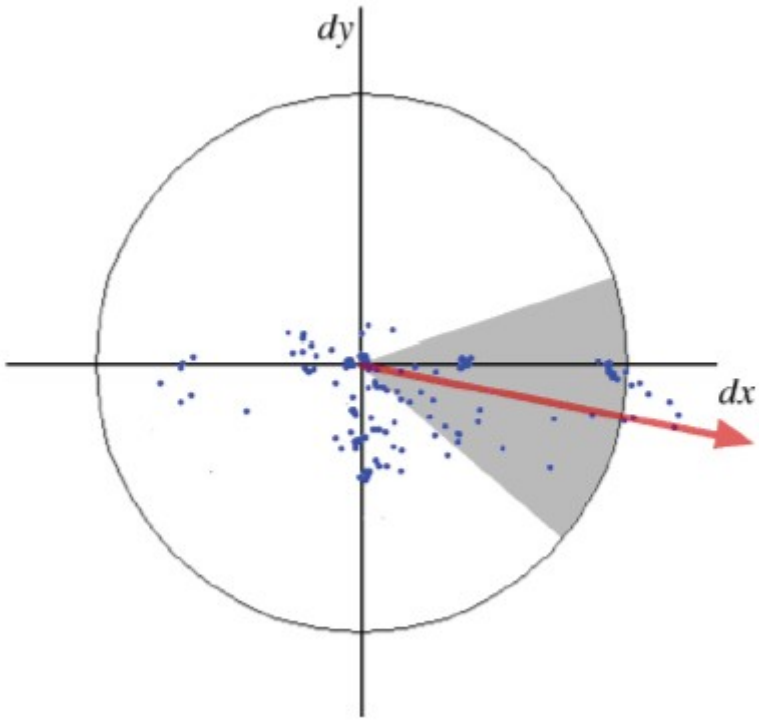
W celu uniezależnienia deskryptora od rotacji obrazu, pierwszym krokiem do opisu punktu kluczowego jest wyznaczenie jego orientacji. W związku z tym oblicza się odpowiedzi falki Haara w kierunku horyzontalnym i wertykalnym dla kolistego otoczenia o promieniu 6 wokół punktu kluczowego, gdzie jest jego skalą. Zarówno rozmiar falki jak i odstęp pomiędzy punktami, dla których oblicza się odpowiedzi, są zależne od skali i wynoszą odpowiednio 4 i . Stosowane filtry, których struktura odpowiada falce Haara, umożliwiają zastosowanie

obrazów całkowitych do przyspieszenia filtracji. Dzięki temu, obliczenie odpowiedzi filtru dla dowolnej skali wymaga jedynie sześciu operacji.



Filtry w kształcie falki Haara w kierunku horyzontalnym (po lewo) i wertykalnym (po prawo). Czarne obszary odpowiadają wadze -1, natomiast białe +1.

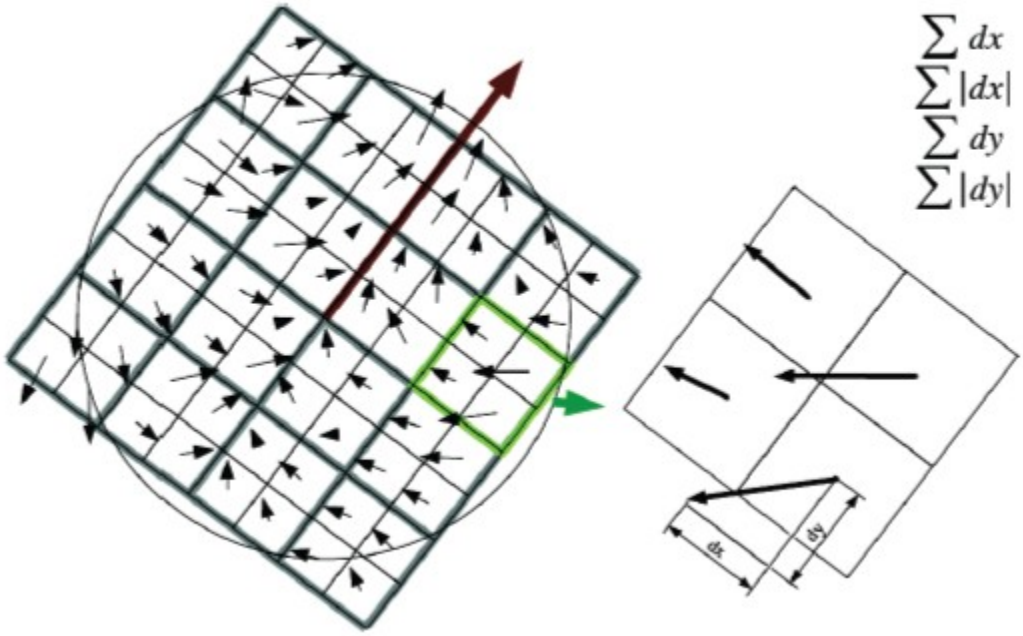
Analogicznie do algorytmu SIFT, odpowiedzi filtru dodatkowo waży się Gaussem ($\sigma=2s$) wyśrodkowanym w punkcie kluczowym. Wszystkie odpowiedzi obliczone dla pewnego otoczenia przedstawiane są w formie punktów w układzie współrzędnych, którego oś odciętych reprezentuje odpowiedz horyzontalną piksela, a oś rzędnych wertykalną. Główną orientację punktu kluczowego estymuje się w wyniku sumowania wszystkich odpowiedzi w obrębie przesuwnego okna o rozmiarze $\pi/3$.



Układ współrzędnych z zaznaczonymi reprezentującymi odpowiedzi falki Haara. Punkty w obrębie przesuwnego okna o rozmiarze $\pi/3$ (obszar zaznaczony na szaro) sumuje się otrzymując wektor lokalnej orientacji (czerwona strzałka)

Suma odpowiedzi horyzontalnych i wertykalnych (czyli suma współrzędnych punktów w utworzonym układzie współrzędnych) definiuje wektor lokalnej orientacji. Najdłuższy taki wektor spośród wszystkich pozycji okna określa orientację punktu kluczowego.

Dysponując orientacją punktu kluczowego, kolejnym krokiem jest stworzenie kwadratowego obszaru wokół punktu kluczowego o zgodnej z nim orientacji. Rozmiar tego obszaru jest zależny od skali i wynosi 20*s* x 20*s*. W celu uchwycenia informacji przestrzennej jest on dzielony na 16 mniejszych, kwadratowych podobszarów. Dla każdego z nich oblicza się odpowiedzi falki Haara (rozmiar filtra wynosi 2*s*) dla regularnie rozmieszczonych punktów na siatce 5x5. Dla uproszczenia oznaczeń, odpowiedzi w kierunku horyzontalnym nazywane będą d_x, a w kierunku wertykalnym d_y.



Kwadratowy obszar wokół punktu kluczowego podzielony na 16 podobszarów (niebieskie kwadraty). Jego orientacja musi być zgodna z orientacją punktu kluczowego (czerwona strzałka). Dla każdego podregionu obliczane są sumy odpowiedzi $\sum d_x, \sum d_y, \sum |d_x|$ $\sum |d_y|$ odpowiadające kierunkowi horyzontalnemu i wertykalnemu względem orientacji (prostopadłemu i równoległemu do orientacji). Cztery wartości obliczone dla każdego podobszaru łącznie tworzą 64-ro wymiarowy deskryptor

Należy pamiętać, że kierunek horyzontalny i wertykalny określa się w odniesieniu do orientacji punktu kluczowego. Aby zwiększyć odporność na zniekształcenia geometryczne oraz drobne przesunięcia, odpowiedzi d_x i d_y waży się Gaussem ($\sigma = 3.3s$) wyśrodkowanym w punkcie kluczowym. Następnie, dla każdego podobszaru sumuje się odpowiedzi d_x i d_y. Uzyskane sumy są pierwszymi 32-ma (dwie sumy dla 16 podregionów) elementami 64-ro wymiarowego deskryptora. Kolejne 32 elementy to sumy wartości bezwzględnych odpowiedzi – $|d_x|$ i $|d_y|$, które dostarczają informacji na temat polaryzacji zmian intensywności pikseli. W rezultacie struktura intensywności każdego podobszaru jest opisana

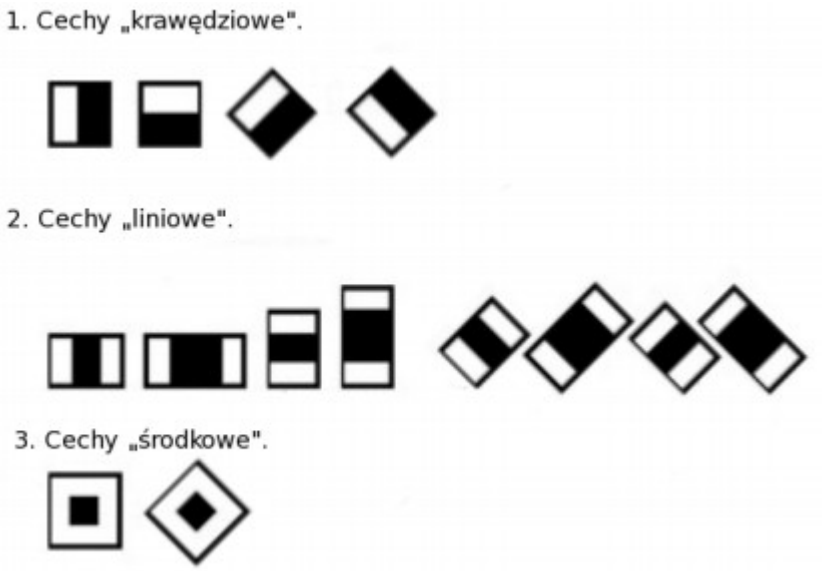
4-wymiarowym deskryptorem $\nu = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. Konkatenacja wektorów ν ze wszystkich 16 podobszarów tworzy 64-ro wymiarowy deskryptor punktu kluczowego.

1.1.1.4. HAAR – kaskadowe klasyfikatory Haara

Procedura detekcji twarzy klasyfikuje obrazy na podstawie prostych cech. Cechy te opierają się na prostokątnych obszarach obrazów cyfrowych. Ich nazwa wywodzi się z podobieństwa do falek Haara, dlatego też w literaturze przyjęło się używać tej nazwy. Podstawową zaletą operowania na cechach haaro-podobnych jest ich zdolność do zawierania wiedzy o opisywanych przez nie obszarach, takiej jak np. występowanie krawędzi lub linii. Dodatkową zaletą jest szybkość obliczania ich wartości.

Cechy haaro-podobne definiowane są przez:

- szablon cechy – każda cecha jest maską składającą się z dwóch lub trzech prostokątów
- współrzędne cechy względem okna przeszukiwania,
- rozmiar cechy – szerokość oraz wysokość.



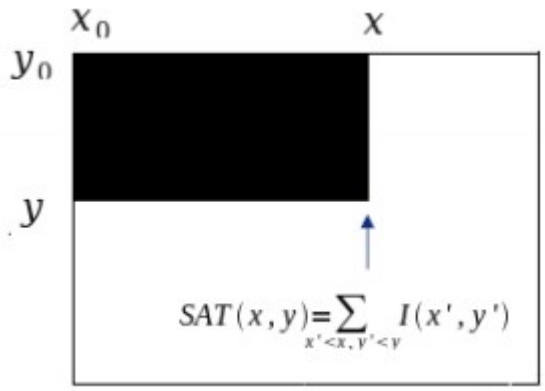
Dla wzorca o rozmiarze 24x24 piksele istnieje w przybliżeniu 180 000 tak zdefiniowanych cech haaro-podobnych (każda cecha jest maską na wzór jednego ze zdefiniowanych szablonów, znajduje się w określonej pozycji względem okienka i posiada określony rozmiar). Wartość każdej cechy jest obliczana jako różnica sumy poziomów szarości pikseli pokrywanych przez biały oraz czarny prostokąt oraz sumy poziomów szarości pikseli pokrywanych przez czarny prostokąt. Składniki tej różnicy posiadają wagi odwrotnie proporcjonalne do rozmiarów, dzięki czemu różnice wielkości dwóch obszarów są kompensowane.

Powtarzanie operacji obliczania wartości cechy haaro-podobnej przez sumowanie poziomów szarości obrazu doprowadziłoby do znacznego wydłużenia procesu detekcji. Celem jej przyspieszenia Viola wprowadził pojęcie obrazu scałkowanego (ang. integral image). W pierwszej kolejności jest obliczana wartość funkcji SAT (ang. Summed Area Table) oraz RSAT (ang. Rotated Summed Area Table) dla współrzędnych (x,y) każdego piksela, które przekształcają dany obraz w obraz scałkowany zgodnie ze wzorem:

$$SAT(x,y) = \sum_{x'<x, y'<y} I(x',y')$$

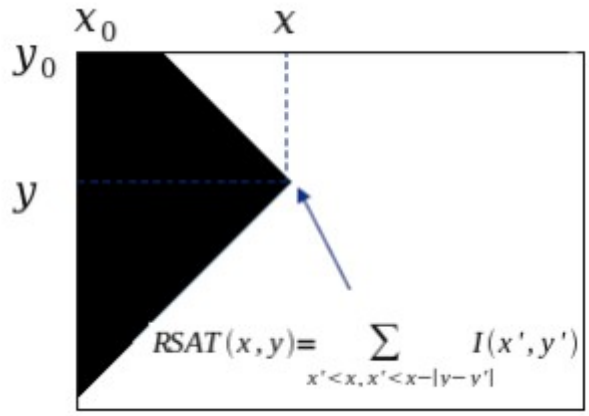
gdzie:

I(x, y') – jest wartością poziomu szarości piksela w punkcie *(x', y')*.

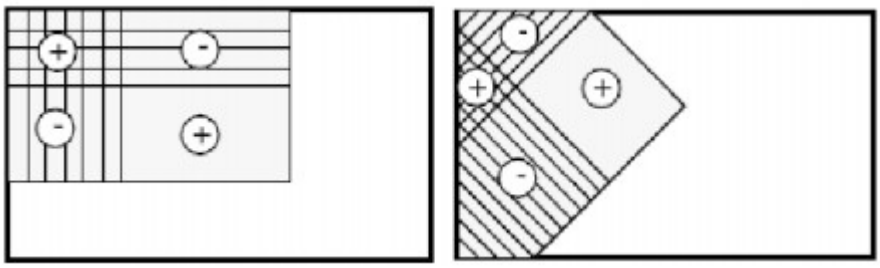


Analogicznie wyznaczana jest tablica RSAT:

$$RSAT(x,y) = \sum_{x'<x, x'<x-|y-y'|} I(x',y').$$



Dzięki zastosowaniu funkcji SAT oraz RSAT suma poziomów jasności pikseli obejmujących dowolny prostokątny obszar jest obliczana w stałym czasie poprzez proste dwie operacje dodawania oraz dwie operacje odejmowania.



W skład klasyfikatora cech haaro-podobnych wchodzi od kilku do kilkudziesięciu słabych klasyfikatorów wybranych w procesie boosting'u. Słaby klasyfikator jest drzewem decyzyjnym składającym się z kilku węzłów decyzyjnych (ang. split nodes) (wykazano, że dla więcej niż jednego węzła osiąga się lepsze wyniki, przy czym nie ma znaczącej różnicy w wynikach pomiędzy słabymi klasyfikatorami 2-, 3-, a nawet 4-węzłowymi). Z każdym węzłem związana jest jedna cecha haaro-podobna. W najprostszym przypadku, dla drzewa o wysokości 1, słaby klasyfikator wyraża wzór:

$$\begin{cases} h_i = +1, & f_i \geq \theta_i \\ h_i = -1, & f_i < \theta_i \end{cases},$$

gdzie:

h_i – wartość słabego klasyfikatora (+1 – „obiekt", −1 – „nie obiekt"),

f_{fi} – wartość cechy haaro-podobnej,

θ_i – wartość progowa i − tej cechy.

Wartość progowa θi jest dobierana w procesie budowania drzewa decyzyjnego tak aby zbudowane drzewo generowało jak najmniejszy błąd klasyfikacji wzorców uczących.

Kaskada klasyfikatorów

Celem wytłumaczenia działania kaskady klasyfikatorów, w niniejszym podrozdziale zostaną omówione użyte oznaczenia, powszechnie stosowane w klasyfikacji binarnej.

W wyniku testowania klasyfikatora otrzymuje się macierz pomyłek (ang. confusion matrix), z której można odczytać w ilu przypadkach model poprawnie sklasyfikował dane testowe, a w ilu się pomylił oraz jakiego typu były to pomyłki.

Orginalne klasy	Klasyfikacja	
	Obiekt	Nie-obiekt
Obiekt	TP	FN
Nie-obiekt	FP	TN

Znaczenie użytych w tabeli pomyłek symboli objaśnia się następująco:

- *TP* - liczba poprawnych klasyfikacji jako obiekt (ang. true positives),
- *FP* - liczba błędnych klasyfikacji jako obiekt (ang. false positives),
- *TN* - liczba poprawnych klasyfikacji jako nie-obiekt (ang. true negatives),
- *FN* - liczba błędnych klasyfikacji jako nie-obiekt (ang. false negatives).

Korzystając z powyższych oznaczeń używa się następujących współczynników opisujących działanie klasyfikatora:

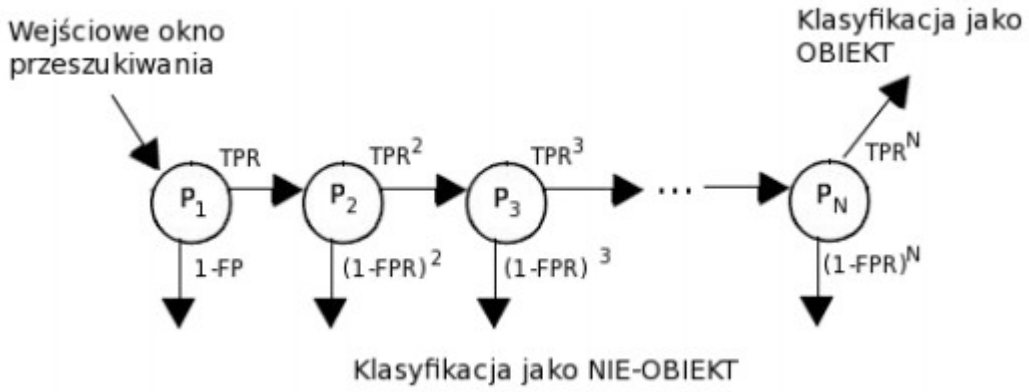
- *TPR* - współczynnik poprawnych klasyfikacji jako obiekt (ang. true positive rate)

$$TPR = \frac{TP}{TP + FN},$$

- *FPR* - współczynnik błędnych klasyfikacji jako obiekt (ang. false positive rate)

$$FPR = \frac{FP}{FP + TN}.$$

Celem przyspieszenia procesu detekcji zaproponowano wprowadzenie kaskady klasyfikatorów. Działanie kaskady przedstawiono na rysunku.



Na wejściu podawany jest analizowany obszar obrazu, jeśli klasyfikator pierwszego poziomu zaklasyfikuje obszar jako obiekt następuje analiza przez kolejny poziom, „przejście" wszystkich poziomów oznacza klasyfikację jako obiekt. „Odrzucenie" analizowanego obszaru na dowolnym poziomie oznacza klasyfikację jako nie obiekt.

Przyspieszenie detekcji osiąga się dzięki zastosowanemu sposobie konstrukcji kaskady. Każdy silny klasyfikator konkretnego poziomu jest uczony w celu osiągnięcia wysokiego współczynnika *TPR* oraz względnie wysokiego współczynnika *FPR* . Proces uczenia odbywa się sekwencyjnie, tak że kolejne poziomy korzystają z coraz mniej dystynktywnych słabych klasyfikatorów i potrzebują ich więcej aby osiągnąć zadane współczynniki. Proces detekcji polega na „przesuwaniu" po obrazie oknem przeszukiwania, które analizuje kaskada. Wysokie *TPR*, gwarantuje „przepuszczenie" przez każdy poziom prawie wszystkich obszarów zawierających obiekt, jednocześnie z każdym kolejny poziomem eliminowana jest część tła,

przez co najbardziej złożone, ostatnie poziomy kaskady analizują tylko niewielką część wejściowego obrazu.

Wartość współczynników klasyfikacji dla całej kaskady zależy od liczby użytych poziomów:

$$TPR_k = TPR^N$$

$$FPR_k = FPR^N,$$

gdzie:

TPR_k – współczynnik poprawnych klasyfikacji wytrenowanej kaskady,

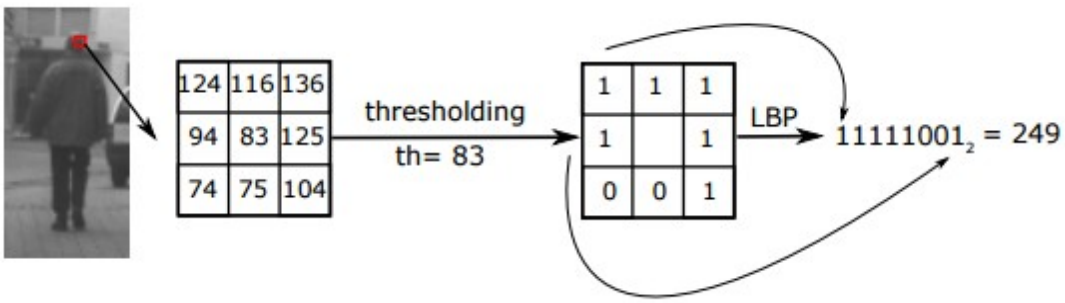
FPR_k – współczynnik błędnych klasyfikacji wytrenowanej kaskady,

N – liczba poziomów wytrenowanej kaskady.

1.1.1.5. LBP – Local Binary Pattern

Jest to metoda opisu lokalnych właściwości tekstury. Może on zostać wykorzystany do detekcji i lokalizacji twarzy. Jako zalety LBP podkreśla się niską złożoność obliczeniową i niewrażliwość na lokalne zmiany oświetlenia.

Wyznaczenie podstawowej wersji LBP dla danego piksela polega na binaryzacji otoczenia o rozmiarze 3x3 z progiem równym wartości elementu centralnego. Otrzymujemy w ten sposób 8-bitową liczbę binarną, która potem jest zamieniana na liczbę dziesiętną. Wyznaczenie LBP przedstawiono na rysunku.



Obraz twarzy po wyznaczeniu LBP wygląda następująco:

Wzorzec LBP jest określany jako jednorodny, jeżeli w zapisie binarnym występują maksymalnie dwie zmiany bitów z 0 na 1 lub 1 na 0. Przykładowo wzorce 00001111 (jedna zmiana), 11111001 (dwie zmiany) są jednorodne, a 10010010 (pięć zmian) jest niejednorodny. W przypadku analizy tekstur, najczęściej wykorzystuje się wzorce, które są jednorodne. W praktycznej realizacji wszystkie wzorce niejednorodne są wykluczane z dalszej analizy.

Modyfikacją jest obliczanie progu binaryzacji poprzez wykorzystanie w miejsce wartości piksela centralnego jednej z poniższych możliwości:

- dodanie stałego przesunięcia do progu
- wykorzystanie jako średniej z otoczenia 3x3
- wykorzystanie jako progu mediany z otoczenia 3x3
- wykorzystanie jako progu średniej wraz z odchyleniem standardowym z otoczenia 3x3

W metodologii detekcji obiektów z wykorzystaniem LBP jako deskryptora cech podstawowym elementem jest mechanizm ruchomego okna, które jest przykładane do każdej lokalizacji na analizowanym obrazie. Okno dzielone jest na K odrębnych bloków w których wyznaczany jest histogram:

$$h_k(i) = \sum_{x \in k} T(LBP == i),\ i = 0, 1 ... B - 1$$

gdzie h_k – histogram dla bloku k,

T(z) = 1 kiedy *z* jest prawdą, w przeciwnym przypadku *T(z)* = 0,

B – liczba przedziałów histogramu (w zależności od wykorzystywanego wariantu LBP wynosi 30, 59 lub 256).

Wartości histogramu normalizowane są do przedziału [0;1] poprzez podzielenie sum w przedziałach przez sumę wszystkich elementów

$$hn_k(i) = \frac{h_k(i)}{m \cdot n},\ i = 0, 1...B-1$$

gdzie: *m x n* rozmiar bloku.

Histogram dla danego bloku zapisywany jest w formie wektora:

$$V_k = [hn_k(0)\ hn_k(1)\ ...\ hn(k)_{(B-1)}]$$

Wektory ze wszystkich bloków tworzą wektor cech dla danego okna

$$F = [V_1\ V_2\ ...\ V_K]$$

Wyznaczony wektor cech może być teraz procesowany przy użyciu SVM (Support Vector Machine) lub innego algorytmu uczenia maszynowego do klasyfikacji obrazów.

1.1.2. Algorytmy precyzyjne – rozpoznawanie twarzy

1.1.2.1. Eigeenfaces- Algorytm twarzy własnych

EigenFace działa w oparciu o przechwytywanie zmian w kolekcji obrazów twarzy oraz wykorzystuje te informacje do kodowania i rozpoznawania twarzy. Ogólnie EigenFace jest zbiorem wektorów własnych macierzy kowariancji zbioru obrazów twarzy, w którym obraz złożony z N pikseli jest traktowany jako punkt (wektor) w N-wymiarowej przestrzeni. Pomysł wykorzystania składowych głównych do reprezentowania ludzkich twarzy został opracowany przez L. Sirovich i M. Kirby i wykorzystywany przez Turka i Pentlanda do wykrywania i rozpoznawania twarzy. Podejście EigenFace jest uważane przez wielu jako pierwszy rozdział w technice komputerowego rozpoznawania twarzy i używany jako podstawa większości systemów komercyjnych.

Głównymi cechami EigenFace sprawiającymi, że jest to metoda popularna są:

- Wyodrębnienie cech twarzy, które nie mogą być zauważone przez człowieka tak jak np. nos, oczy czy usta. Metoda bazuje na uchwyceniu statystycznych zmian pomiędzy obrazami twarzy.
- Skuteczna reprezentacja obrazów twarzy. Aby zmniejszyć złożoność obliczeń i przestrzeni, obraz każdej twarzy można przedstawić za pomocą niewielkiej liczby parametrów.

EigenFaces mogą być traktowane jako zespół cech, które charakteryzują globalne zmiany wśród obrazów twarzy.

Przed obliczeniem EigenFaces, zdjęcia twarzy są znormalizowane do linii oczu i ust, a następnie wszystkie obrazy są skalowane do takiego samego rozmiaru. EigenFaces zostają wyodrębnione z danych obrazu za pomocą analizy składowych głównych (PCA).

Komputerowe liczenie EigenFace:

- M obrazów o wymiarach [h x w] zostaje zapisanych jako wektory D (o wymiarach hw) i umieszczone w zbiorze

$$\{\Gamma_1, \Gamma_2, \ldots, \Gamma_M\}$$

  Wszystkie obrazy powinny być odpowiednio przeskalowane, a tło jednorodne lub usunięte

- Każda twarz różni się od średniej o wektor

$$\Phi_i = \Gamma_i - \Psi$$

  gdzie średnia twarz jest określona przez

$$\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i$$

- Macierz kowariancji $C \in \mathbb{R}^{DxD}$ jest zdefiniowana jako:

$$C = \frac{1}{M}\sum_{i=1}^{M}\Phi_i\Phi_i{}^T = AA^T \quad \text{, gdzie} \quad A = \{\Phi_1, \Phi_2, \ldots, \Phi_M\} \in \mathbb{R}^{D\times M}$$

- Ustalenie wektorów własnych C jest trudne dla obrazów o typowych rozmiarach,

  gdzie $D \gg M$. Aby efektywnie obliczyć wektory własne C można najpierw obliczyć

  wektory o wiele mniejsze M ×M macierzy A^TA . Wektory i wartości własne A^TA są definiowane jako:

$$V = \{v_1, v_1, \ldots v_\tau\}$$

  i

$$\Lambda = diag\{\lambda_1, \lambda_1, \ldots \lambda_\tau\}, \lambda_1 \geqslant \lambda_2 \geqslant \ldots \lambda_\tau > 0$$

gdzie τ jest wymiarem A .

Należy pamiętać, że wektory odpowiadające zerowym wartością własnym zostały odrzucone.

- Wartości własne i suma macierzy wektorów własnych C są Λ i

$U = AV\,\Lambda^{-1/2}$, gdzie $U=\{u_i\}$ jest zbiorem EigenFaces

Przykładowe twarze

Średnie twarzy i EigenFaces

W celu redukcji wymiaru wektora stosuje się analizę składowych głównych (PCA). PCA stara się znaleźć k „osi głównych”, które określają ortogonalny układ współrzędnych mogący przechwycić większość wariancji danych. Na przykład, biorąc pod uwagę macierz danych A, macierz kowariancji może być zapisana jako $C=AA^T$ natomiast główne składniki u_i ,

$$AA^T u_i = \lambda_i u_i$$

gdzie u_i są wektorami własnymi AA^T związanymi z wartością λ_i. Kiedy AA^T jest zbyt duży, aby możliwe było jego skuteczne rozłożenie, można ominąć to przez obliczenie wewnętrznej macierzy iloczynu A^TA

$$A^T A v_i = \lambda_i v_i$$

gdzie v_i są wektorami własnymi A^TA związanymi z wartością λ_i . należy zauważyć, że λ_i jest nieujemna, a

$$u_i = \lambda_i^{-0.5} A v_i$$

dla tych niezerowych wartości własnych

$$A = U \Delta V^T = \sum \delta_i u_i v_i^T$$

gdzie U i V są prostopadłe, przekątna macierzy Δ zawiera pojedynczą wartość δ_i , która może być pozytywna, zerowa, a nawet ujemna. U i V są wektorami własnymi

$$AA^T, A^T A, \delta_i^2 = \lambda_i$$

dla każdego i.

Algorytm Eigenfaces, wykorzystujący metodę głównych składowych, polega na wykonaniu następujących kroków, prowadzących do wyznaczenia etykiety dla rozpatrywanego obrazu:

Wartości głównych składowych zostają wyznaczone dla wszystkich obrazów znajdujących się do tej pory w bazie danych.

Wartości głównych składowych zostają wyznaczone dla próbki wejściowej.

Dla wyznaczonej próbki zostaje wyznaczony wektor najbliższy dla wartości obliczonych dla obrazów z bazy danych.

1.1.2.2. Fisherfaces- Algorytm twarzy dyskryminacyjnych

Algorytm Fisherfaces opiera się na liniowej analizie dyskryminacyjnej (LDA) i polega na wyznaczeniu wektora cech pozwalającego na rozdzielenie obiektów przynależnych do

różnych klas. W przypadku problemu rozpoznawania twarzy, przez klasy obiektów rozumiane są zbiory obrazów przedstawiające tego samego użytkownika. Problem sprowadza się do wyznaczenia wektora, który stanowi przybliżoną granicę między dwiema klasami obiektów, jednak można go uogólnić do problemu wieloklasowego. Wyznaczanie poszukiwanego

wektora rozpoczyna się od wyznaczenia wartości średniej μ wszystkich obiektów

znajdujących się w rozpatrywanym zbiorze, oraz wartości średniej μi wewnątrz

poszczególnych klas.

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

Gdzie N oznacza liczebność rozpatrywanego zbioru, x_i i-tym jest obiektem zbioru.

$$\mu_i = \frac{1}{|X_i|} \sum_{x_j \in X_i} x_j$$

Gdzie X_i jest klasą obiektów o indeksie i, dla i = 1, 2, … , n przy n oznaczającym liczbę klas.

Następnie wyznaczana jest macierz rozproszenia wewnątrzklasowego S_B oraz macierz

średniego rozproszenia wewnątrzklasowego S_W .

$$S_B = \sum_{i=1}^{n} N_i(\mu - \mu_i)(\mu - \mu_i)^T$$

$$S_W = \sum_{i=1}^{n} \sum_{x_j \in X_i} (x_j - \mu_i)(x_{j-} \mu_i)^T$$

Rozwiązanie problemu sprowadza się do znalezienia danych, które pozwolą na otrzymanie największej wartości określającej stosunek rozproszenia wewnątrzklasowego do średniego

wewnętrznego rozproszenia klas. Stąd algorytm poszukuje wektora d, dla którego poniższa

funkcja osiąga maksimum.

$$f(d) = \frac{d^T S_B d}{d^T S_W d}$$

W celu zmniejszenia złożoności problemu, stosuje się modyfikację powyższego kryterium,

wykorzystując macierz W złożoną z k głównych wektorów własnych wyznaczonych dla

macierzy kowariancji C wszystkich elementów zbioru.

$$f(d) = \frac{d^T W^T S_B W d}{d^T W^T S_W W d}$$

Znalezienie wektora d pozwala na wyznaczenie optymalnego kierunku rozdzielającego dwie

klasy obiektów

1.1.2.3. LBP- Algorytm lokalnych binarnych wzorców

W odróżnieniu od holistycznego podejścia w dwóch poprzednich przypadkach, algorytm LBPH wykorzystuje lokalne cechy przetwarzanych obiektów.

Dla każdego piksela wyznaczany jest ciąg binarny na podstawie porównania wartości, z każdym z sąsiadów. W przypadku, gdy jego wartość jest większa wtedy przyjmuje wartość 1,

a 0 w przypadku przeciwnym. Stąd dla każdego piksela wyznaczana jest wartość p-znakowego

binarnego ciągu, nazywanego lokalnym binarnym wzorcem. Wyznaczanie można przeprowadzić w otoczeniu o dowolnym promieniu. W ogólności wartość funkcji LBP dla

piksela o współrzędnych (x_c , y_c) wyznacza się następująco:

$$LBP(x_c, y_c) = \sum_{p=0}^{p-1} 2^p S(i_p - i_c)$$

Gdzie p jest rozmiarem rozpatrywanego sąsiedztwa o środku w punkcie (x_c , y_c) i jasności o

wartości i_c , a i_n jest wartością jasności dla n-tego punktu sąsiedztwa.

$S(x)$ jest funkcją znaku definiowaną następująco:

$$S(x) = \begin{cases} 1 \; dla \; x > 0 \\ 0 \; w \; p.p. \end{cases}$$

Działanie algorytmu polega na podzieleniu obrazu wejściowego na m różnych części i

wyznaczenie dla niego wartości LBP jasności pikseli. Następnie dla każdego regionu wyznaczany jest histogram wyliczonych wartości. Tak wyznaczone wartości są konkatenowane do postaci wektora.

Dla próbki wejściowej zostaje wyznaczony wektor histogramów, który następnie jest porównywany z wektorami obrazów znajdujących się w bazie danych. Przewidywana etykieta jest wyznaczana na podstawie etykiety wektora najbliższego sąsiada.

1.1.3. Algorytmy rozpoznawania, śledzenia sylwetki oraz detekcji upadku

Rozdział został przygotowany na podstawie rozprawy doktorskiej „Detekcja upadku i wybranych akcji na sekwencjach obrazów cyfrowych” mgr inż. Michał Kępski, Kraków 2016.

Wraz z rozwojem badań nad metodami detekcji upadku ukształtowały się wymagania stawiane tej technologii (Igual et al., 2013; Yu, 2008). W pracy (Igual et al., 2013) i pokrewnych określono największe wyzwania i potencjalne ograniczenia tej technologii. Według autorów do skutecznego wdrożenia metod detekcji upadku potrzebna jest:

- Wysoka skuteczność detekcji upadków w środowisku użytkowników z grupy docelowej – detekcja powinna charakteryzować się wysoką specyficznością i czułością. Mimo osiągnięcia wyników zbliżonych do pożądanych w warunkach laboratoryjnych, dokładność algorytmów w scenariuszach życia codziennego (ang. performance under real-life conditions) pozostaje kwestią otwartą, czego dowodzą prace (Bagalà et al., 2012; Igual et al., 2013).

- Użyteczność technologii – łatwość w obsłudze i maksymalna automatyzacja procesu inicjalizacji działania oraz minimalizacja czasu potrzebnego na obsługę systemu przez użytkownika są ważnym celem w procesie projektowania systemów telemedycznych. Należy uwzględniać potrzeby potencjalnych użytkowników, oraz ocenić ich potencjalną wiedzę na temat obsługi urządzeń elektronicznych. Opracowywaną technologię, mimo swojego potencjalnego wyrafinowania pod względem stosowanych metod, powinna charakteryzować prostota obsługi.

- Akceptacja użytkowników – jest powiązana ze skutecznością i użytecznością technologii (Kurniawan, 2008). Zbyt wysoka liczba fałszywych alarmów, bądź skomplikowana obsługa mogą być przeszkodą w skutecznym zastosowaniu metod w praktyce, ze względu na opór osób starszych. Ponadto zagadnienie prywatności jest kluczowe, gdyż osoby mogą czuć się monitorowane, co z kolei może mieć negatywny wpływ na ich jakość życia

Skuteczne rozpoznawanie akcji(takich jak upadek) w systemach wizyjnych wymaga zwykle wstępnego przetwarzania obrazów. Jakość rozpoznawania zależy nie tylko od poprawnego doboru cech oraz parametrów w oparciu o które działa klasyfikator, ale też wcześniejszego przygotowania danych (ang. preprocessing). Przykładowo, na pogorszenie wyników rozpoznawania akcji może wpływać niedokładna segmentacja postaci, czy też błędy w jej śledzeniu. Metody w systemie detekcji można podzielić na dwie grupy:

- metody działające przy inicjalizacji systemu – wykonywane jednorazowo podczas uruchomienia systemu lub przy zmianie pozycji kamery: detekcja podłogi, budowa modelu tła.

- metody działające w czasie rzeczywistym – wykonywane ciągle podczas działania systemu: odjęcie tła, detekcja postaci, śledzenie, aktualizacja modelu tła.

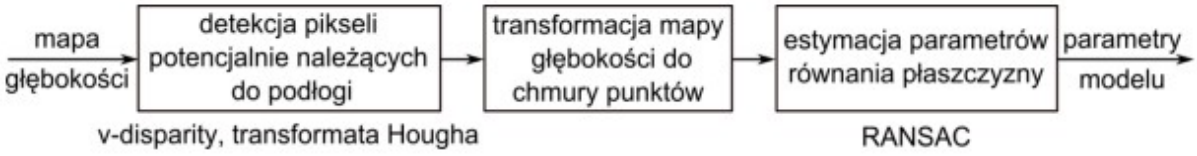
1.1.3.1. Algorytmy detekcji podłogi

Detekcja podłogi jest ważnym etapem wstępnym detekcji upadku, a jej wykorzystanie w systemie ma na celu:

Redukcję rozmiaru danych w celu zmniejszenia czasu przetwarzania, gdyż dzięki znajomości równania płaszczyzny, po pobraniu obrazu z urządzenia, można usunąć te elementy chmury punktów, które należą do podłogi.

Określenie cech opisujących postać, potrzebnych do detekcji upadku. Detekcja podłogi spełnia wymagania odnalezienia referencyjnego elementu sceny, względem którego będzie badany ruch osoby.

Przebieg algorytmu do detekcji podłogi można przedstawić w następujący sposób



Metodę v-dysparycji można zastosować także dla obrazów z sensora Kinect. Biorąc pod uwagę zasadę działania tego urządzenia oraz znajomość odległości od kamery, obliczyć można różnicę zobrazowania:

$$d = \frac{b \cdot f}{z}$$

gdzie z – odległość od kamery (w metrach), b – linia bazowa (w metrach), f – ogniskowa (w pikselach). Dla sensora Kinect długość linii bazowej wynosi 7,5 cm, zaś ogniskowa równa jest 580 pikseli. Poniższy rysunek przedstawia obraz v-dysparycji dla przykładowej mapy głębokości oraz korespondujący z nimi obraz RGB. Zakładając, że kamera umieszczona jest na wysokości powyżej 1 metra nad podłogą, początek odpowiadającego jej na obrazie v-disparity odcinka zawiera się w przedziale (20, 27) wartości dysparycji. Wyznaczenie położenia tego odcinka na obrazie, pozwoli dla każdej linii obrazu wejściowego (dysparycji, a w konsekwencji głębokości) znaleźć te piksele, które należą do podłogi.

Rysunek 2.12. Metoda v-*disparity* zastosowana dla danych z sensora *Kinect*, od lewej: obraz RGB sceny, mapa głębi, obraz v-dysparycji.

Detekcja linii na obrazie może być zrealizowana w oparciu o transformatę Hougha (Szeliski, 2010), która została opracowana do wykrywania regularnych kształtów na obrazach. Pozwala ona na wykrywanie prostych kształtów takich jak linie, okręgi czy elipsy z dużą odpornością na zakłócenia obrazu. Najpowszechniej używanym jej wariantem jest tzw. uogólniona transformata Hougha

Algorytm transformaty Hougha można przedstawić następująco:

1. Dokonaj kwantyzacji przestrzeni akumulatora, $\theta \in [0, \pi)$, $\rho \epsilon [-R, R]$

2. Dla każdego piksela obrazu wejściowego (xi , yi):

3. $\rho = xi \cos\theta + yi \sin\theta$, $\forall\theta \in [0, \pi)$

4. Zwiększ wartość akumulatora: $A(\rho, \theta) = A(\rho, \theta) + 1$

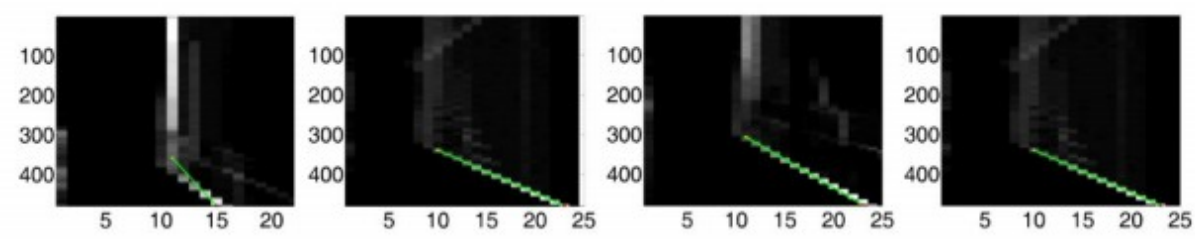
5. Dla przyjętego progu A_{thold} , każda wartość akumulatora spełniająca warunek

$A(\rho, \theta) > A_{thold}$ reprezentuje linię na obrazie wejściowym o parametrach (ρ, θ)

Biorąc pod uwagę charakterystykę obrazu v-dysparycji, przy detekcji linii metodą Hougha można dodać do oryginalnego algorytmu pewne modyfikacje, które wpłyną na poprawę skuteczności detekcji oraz na zmniejszenie nakładów obliczeniowych. Przede wszystkim, znając specyfikę obrazu v-disparity można znacznie ograniczyć przestrzeń akumulatora

zarówno w zakresie kąta θ jak i promienia wodzącego ρ. Kąt θ można, zgonie z pracą (Kępski

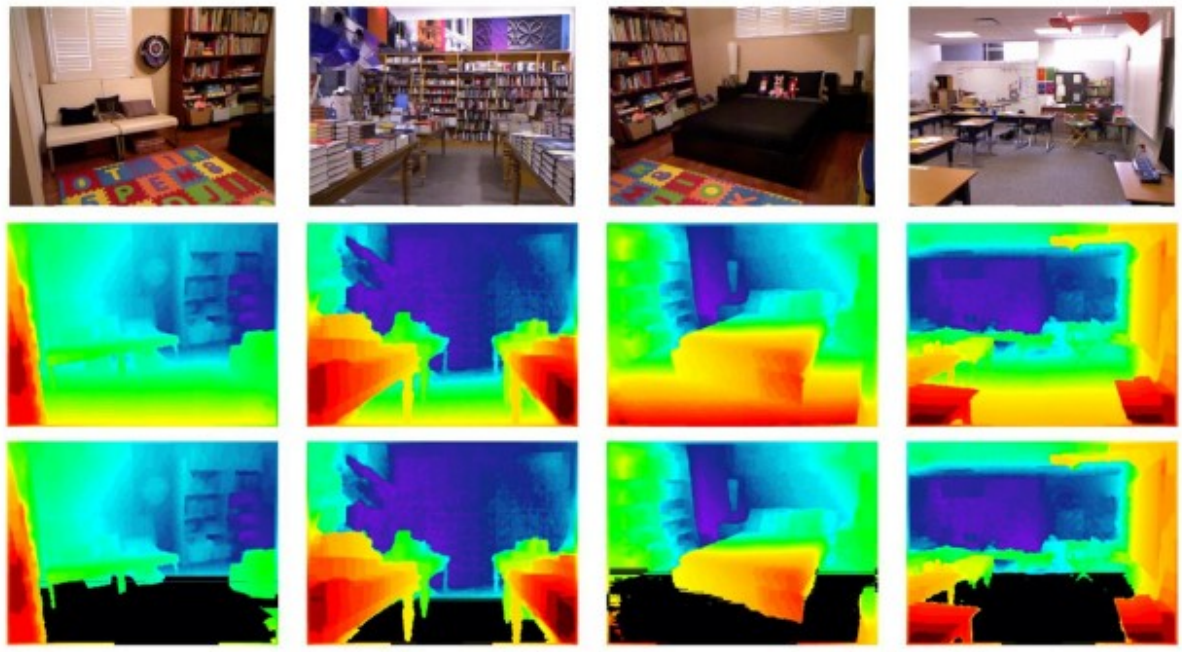
& Kwolek, 2013), ograniczyć do przedziału [0,30) stopni, co wpływa na zmniejszenie czasu obliczeń. Ponadto biorąc pod uwagę fakt, że obraz v-dysparycji jest swego rodzaju histogramem, można przeprowadzić detekcję linii bezpośrednio na nim (a nie na obrazie krawędzi).



Rysunek 2.14. Detekcja linii na obrazie *v-disparity* w oparciu o zmodyfikowaną transformatę Hougha.

Dzięki znajomości parametrów wykrytej prostej, można dla każdej linii obrazu wejściowego wyznaczyć wartość dysparycji, którą powinny posiadać piksele należące do płaszczyzny. Na podstawie tego kryterium można dokonać detekcji obszarów sceny, należących do podłogi.

Ewaluację metody v-disparity przeprowadzono na obrazach z publicznie dostępnej bazy danych NYU Depth Dataset V2 , która zawiera obrazy RGB-D wykorzystywane do oceny skuteczności algorytmów segmentacji sceny. Na rysunku 2.15 przedstawiono przykładowe rezultaty segmentacji.



Rysunek 2.15. Obrazy ilustrujące skuteczność wydzielania podłogi metodą v-dysparycji na obrazach z bazy danych NYU Depth Dataset V2. Od góry do dołu: obraz RGB, obraz głębi, obraz głębi z zaznaczonymi pikselami należącymi do podłogi. Ze względu na mały rozmiar obrazów, głębie przedstawiono w skali kolorów (czerwony – obiekty blisko kamery, niebieski – obiekty daleko od kamery).

Detekcja punktów sceny, które należą do podłogi, może okazać się niewystarczająca w kontekście percepcji sceny na potrzeby wydzielenia cech charakteryzujących osobę i jej ruch. Wygodniej jest traktować podłogę jako płaszczyznę i przedstawiać ją przy pomocy równania:

$ax + by + cz + d = 0$

Celem określenia wartości współczynników a, b, c, d należy zastosować metodę estymacji

parametrów modelu matematycznego na podstawie zbioru obserwacji. Algorytm RANSAC (Fischler & Bolles, 1981) (ang. Random Sample Consesus) jest ogólną metodą estymacji parametrów modelu na podstawie danych, zaprojektowany tak, aby radzić sobie z dużą liczbą wartości odstających od poszukiwanego modelu (ang. outliers). Algorytm ten stosuje technikę próbkowania, która pozwala na wygenerowanie potencjalnych rozwiązań (ang.

candidate solutions) wykorzystując jak najmniejszą liczbę obserwacji. Ze zbioru $D = \{d1, \dots,$

$dN\}$, do którego należą wszystkie dane wybierany jest losowo pewien podzbiór MSS, nazywany minimalnym zbiorem próbek (ang. minimal sample set). Następnie sprawdzana jest jaka część całego zbioru danych, oznaczona jako CS (ang. consensus set), jest zgodna z modelem $\mathcal{M}$, którego wektor parametrów Θ został oszacowany na podstawie podzbioru MSS. Pseudokod algorytmu można zapisać następująco:

1. **do**:
2. wybierz losowo podzbiór $MSS \epsilon D$, $|MSS| < |D|$
3. na podstawie MSS estymuj wektor parametrów Θ
4. dla każdej próbki $d \epsilon D$ wykonaj:
5. wyznacz błąd $e\mathcal{M}$, zdefiniowany jako odległość d do modelu $\mathcal{M}(\Theta)$

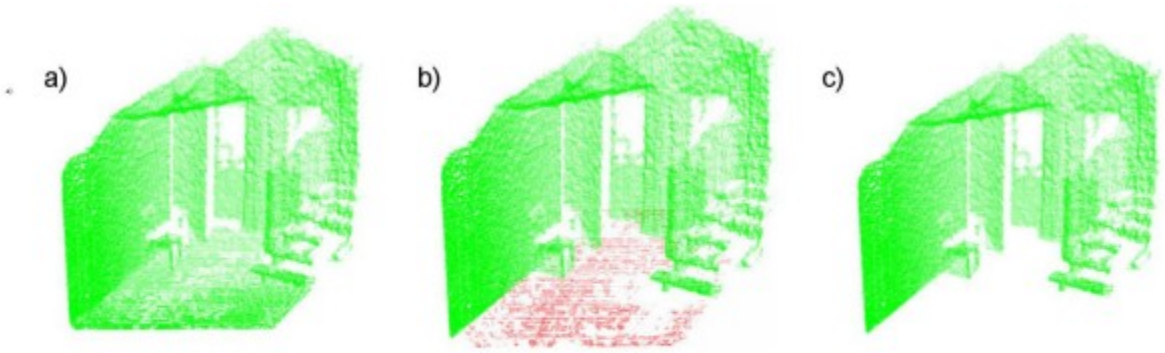
6. if $e_{\mathcal{M}}(d, \Theta) \leq \delta$ // δ jest predefiniowanym progiem $d \epsilon CS$

7. oblicz $|CS|$

8. **while** $|CS|$ $|D| < \tau$ // gdzie τ jest predefiniowanym progiem

Oprócz warunku stopu związanego z liczbą punktów należących do consensus set, można ograniczyć liczbę iteracji algorytmu, zakładając, że po pewnej liczbie iteracji h uzyskamy wystarczająco dobre rozwiązanie.

Na rysunku 2.16. przedstawiono chmury punktów przykładowego procesu wydzielania podłogi (korespondują one z mapą głębi z rysunku 2.12). Kolorem czerwonym zaznaczono te elementy chmury punktów (b), które są wynikiem segmentacji podłogi metodą v-dysparycji i transformaty Hougha. Porównując ją z chmurą punktów (a) można zauważyć, że segmentacja podłogi nie przebiegła w sposób zadowalający. Jak można zauważyć na wspomnianym rysunku, dane nie są pozbawione błędnie zakwalifikowanych do płaszczyzny punktów, zaś część właściwej płaszczyzny została pominięta. Nie mniej jednak, liczba outliers nie jest znacząca, a więc określone tym sposobem dane są dobrą hipotezą początkową dla algorytmu RANSAC. Chmura punktów (c), która uzyskana została w oparciu o estymację 42 parametrów równania płaszczyzny i usunięcie punktów do niej należących jest pozbawiona błędów segmentacji podłogi.



Rysunek 2.16. Wynik detekcji płaszczyzny podłogi przedstawiony za pomocą chmur punktów: a) wejściowa chmura punktów, b) wynik metody *v-disparity* i transformaty Hougha, c) wynikowa chmura punktów pozbawiona punktów należących do podłogi.

Do estymacji parametrów równania płaszczyzny można wykorzystać także inne metody (Tanahashi et al., 2001), np.: metodę najmniejszych kwadratów, metodę wartości i wektorów własnych (ang. eigenvalues) czy maximum-likelihood estimation. Metodę RANSAC należy wybrać ze względu na możliwość uzyskania dobrych wyników przy obecności znacznej liczby outliers. Wyznaczenie metodą v-disparity hipotezy początkowej dla algorytmu RANSAC pozwala na ograniczenie wpływu zaszumienia danych na dokładność procesu estymacji. Nie mniej jednak dla bardziej skomplikowanej sceny, przykładowo, jeśli znajdują się na niej przedmioty leżące na podłodze, wybór algorytmu RANSAC, jako bardziej złożonego (niż np. metoda najmniejszych kwadratów) może się okazać w pełni uzasadniony.

1.1.3.2. Budowa i aktualizacja modelu tła

Algorytmy detekcji tła i pierwszego planu (ang. background/foreground detection) są powszechnie wykorzystywane w systemach multimedialnych i znalazły szerokie zastosowanie m.in. w systemach monitoringu (detekcja i śledzenie obiektów), systemach interakcji człowiek-komputer, a także wykorzystywane są jako etap wstępnego przetwarzania przy rozpoznawaniu akcji i zachowań ludzi. Opracowano wiele metod detekcji tła i pierwszego planu działających w oparciu o dane z kamer RGB (Sobral & Vacavant, 2014). Do najprostszych metod zalicza się metody oparte na odjęciu tła (ang. background substraction). Metody te cechuje niewielki koszt obliczeniowy. Należą do najszybszych i dobrze sprawdzają się w systemach wbudowanych lub systemach czasu rzeczywistego wykorzystujących kilka kamer. Jednak metody te nie są pozbawione wad: często zawodzą przy zmianach oświetlenia, występujących cieniach itp. W praktyce najczęściej wykorzystuje się nieco bardziej zaawansowane modele rozkładu kolorów, takie jak mieszanina Gaussianów (ang. Mixture of Gaussians) (Zivkovic & van der Heijden, 2006), sieci neuronowe (Maddalena & Petrosino, 2008) czy też modele oparte o inne metody statystyczne (Wren et al., 1997; Yao & Odobez, 2007; Arsic et al., 2009).

Dzięki operowaniu na obrazach głębi, zamiast RGB, można uniknąć pewnych wad wspomnianych metod, w szczególności dotyczących nadmiernej czułości na zmiany oświetlenia sceny. W podstawowej wersji algorytmu, odejmowane są wartości pikseli obrazu tła od wartości pikseli obrazu z aktualnej klatki:

$$|D_t(x,y) - B_t(x,y)| > B_{thol}$$

gdzie D_t aktualna mapa głębi, B_t jest referencyjnym modelem tła, a B_{thol} to próg obcięcia,

poniżej którego piksele są traktowane jako tło.

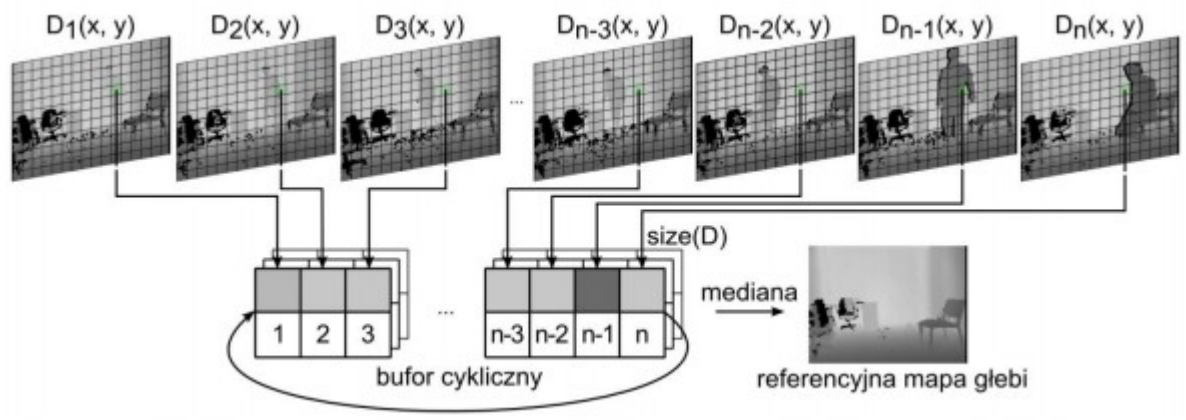
Do najpopularniejszych metod budowy modelu tego typu zaliczamy algorytmy, w których zastosowano filtry uśredniające (ang. mean filter) lub filtry medianowe (ang. median filter) w czasie. W przypadku zastosowania filtru uśredniającego dla każdego piksela tła wyznaczana jest średnia wartość na podstawie wartości pikseli z kilku obrazów. Operację tę można przedstawić w następujący sposób:

$$B_t(x,y) = \frac{1}{n}\sum_{i=1}^{n} D_{t-i}(x,y)$$

Do budowy modelu tła można wykorzystać także operację mediany (Arsic et al., 2009) dla pikseli z kilku poprzednich obrazów:

$$B_t(x,y) = median\{D_{t-i}(x,y)\},\ i\epsilon\{1,\dots,n\}$$

Schemat metody budowy modelu tła przedstawia poniższy rysunek.



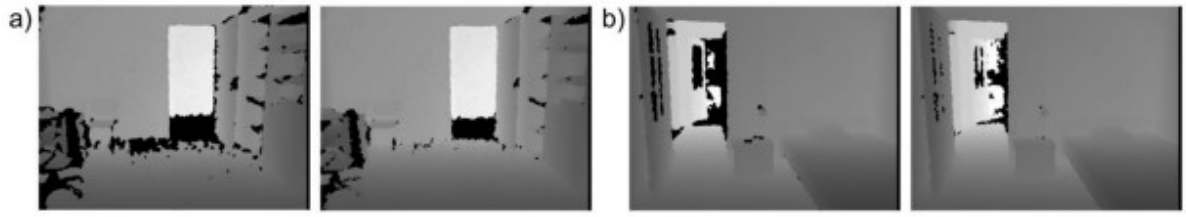
Zaletą omówionych metod jest szybkość działania, wadą zaś zwiększone wymagania pamięciowe, co wynika z konieczności zapamiętania n poprzednich map głębi. Metoda przedstawiona na powyższym rysunku służy do budowy modelu tła przy inicjalizacji działania systemu oraz późniejszej jego aktualizacji na wypadek wykrycia zmian na scenie. W procesie inicjalizacji, w celu poprawnego zbudowania modelu tła, scena powinna być pusta (pozbawiona osób i innych poruszających się obiektów). Można wówczas wykorzystać, następujące po sobie obrazy. W procesie aktualizacji tła, jeśli użytkownik przebywa na scenie, obrazy należy zapisywać do bufora co pewien czas, z większymi interwałami czasowymi, w ten sposób, aby uniknąć uwzględnienia pikseli należących do użytkownika w modelu tła. Operacja mediany pozwoli usunąć skrajne wartości pikseli, czego efektem będzie zbudowanie modelu bez osoby poruszającej się po scenie. Informacja o tym czy użytkownik porusza się czy nie, pozwoli na uniknięcie włączenia pozostającej w spoczynku osoby do modelu tła (Kwolek & Kępski, 2014)

Celem lepszego dopasowania algorytmu budowy modelu tła do specyfiki wynikającej z operowania na mapach głębi, należy dokonać istotnej zmiany w stosunku do klasycznej wersji algorytmu operującego na obrazach kolorowych. Urządzenie do akwizycji obrazów głębi, dla każdego piksela zwraca wartość odległości od kamery w milimetrach lub wartość 0 gdy nie można dokonać pomiaru głębokości (*nmd*, oznaczone kolorem czarnym na obrazach). Wynika to z niedoskonałości urządzenia. Modyfikacja algorytmu polega na pominięciu tych pikseli z n obrazów, dla których wartość wynosi 0. Pozwala to uniknąć sytuacji, gdy dla współrzędnych (x, y) na obrazie, w sekwencji n pikseli dominować będą piksele *nmd*, skutkiem czego byłoby umieszczenie tych pikseli w referencyjnym modelu tła. Zmodyfikowaną operację mediany pikseli z n poprzednich klatek można zatem przedstawić w następujący sposób:

$$B_t(x, y) = median\{D_{t\_i}(x,)\}, i\epsilon\{1, \dots, n\},$$

$$D_{t\_i}(x, y) \neq 0$$

Modyfikacja ta nie powoduje, że model tła będzie całkowicie pozbawiony pikseli o wartości zero, gdyż pewne obszary (obszary szklane, posiadające metaliczne powierzchnie, łatwo odbijające światło) będą zawsze reprezentowane na obrazie głębi jako zera. Nie mniej jednak, zastosowanie takiego podejścia pozwala na eliminację znacznej liczby zerowych pikseli, co z kolei powoduje powstanie gęstszej chmury punktów 3D modelu tła. Rezultaty tej modyfikacji przedstawiono na rysunku.



Rysunek 2.18. Pary modeli tła uzyskane klasyczną metodą mediany (po lewej stronie) oraz zmodyfikowaną metodą (obrazy po prawej stronie). Sekwencja a) zarejestrowana była przy dużym natężeniu światła słonecznego, zaś sekwencja b) przy niskim natężeniu światła.

Zastosowanie stałego interwału czasowego pomiędzy operacjami zapisu obrazów do bufora cyklicznego może prowadzić do sytuacji, w której fragment postaci użytkownika, poruszającego się ze zmienną prędkością po scenie lub przebywającego chwilowo w bezruchu, zostanie umieszczony w modelu tła. Ponadto można zaobserwować, że zazwyczaj wymagana jest modyfikacja jedynie fragmentu referencyjnej mapy głębi, gdyż zazwyczaj ingerencja użytkownika w otaczającą go scenę dotyczy jednego lub kilku obiektów tej sceny, pozostawiając pozostałe obiekty w stanie niezmienionym. Modyfikacja jedynie fragmentów modelu tła przedstawiających zmienione obszary sceny pozwala na zmniejszenie kosztu obliczeniowego poprzez znaczną redukcję liczby zbiorów pikseli, które należy poddać operacji mediany, a także pozwala zmniejszyć prawdopodobieństwo przypadkowego umieszczenia postaci w modelu tła. Modyfikację algorytmu budowy referencyjnej mapy głębi, polegającą na:

- Detekcji regionów zainteresowania algorytmu (ang. Regions of Interest, ROI), dzięki której regiony zainteresowania algorytmu są zdefiniowane jako obszary obrazu zawierające fragmenty sceny, które uległy modyfikacji (zob. Algorytm 3, linia 8 oraz rysunek 2.19d).

- Wykorzystaniu informacji o położeniu postaci w procesie doboru wartości głębokości umieszczanych w buforze. Informacja ta jest uzyskiwana w oparciu o segmentację metodą rozrostu obszarów (ang. region growing) oraz algorytmy śledzenia, opisane w kolejnym podrozdziale (zob. Algorytm 3, linia 5).

Zmodyfikowany algorytm budowy modelu tła można przedstawić w następujący sposób:

Algorytm 3 (Algorytm budowy modelu tła):

1. dany jest model tła $B(x, y)$, bufor $Q = \{D_{t-1}(x, y), D_{t-2}(x, y), \dots, D_{t-Qsize}(x, y)\}$

2. pobierz nową mapę głębi $D_t(x, y)$ (zob. rysunek 2.19c.)

3. wyznacz obraz

$$F_t(x,y) = \begin{cases} D_t(x,y), jeśli\ |D_t(x,y) - B(x,y)| \geq B_{thold} \\ 0, \quad jeśli\ |D_t(x,y) - B(x,y)| < B_{thold} \end{cases}$$

4. wyznacz na $Ft(x, y)$ pola powierzchni komponentów połączonych BLOB i określ ich liczbę n_{BLOB} // rysunek 2.19d.

5. przypisz obrazowi postaci $P_t(x, y)$ obszar BLOB o największym podobieństwie do $P_{t-1}(x, y)$ // wyznaczenie podobieństwa np. w oparciu o współczynniki kształtu

6. $\textbf{if } \frac{area(P_t(x,y))}{area(P_{t-1}(x,y))} > T_a \textbf{ or } n_{BLOB} > 1$

7. wyznacz $$F_{t-3}(x,y) = \begin{cases} D_{t-3}(x,y), jeśli\ |D_{t-3}(x,y) - B(x,y)| \geq B_{thold} \\ 0, \qquad jeśli\ |D_{t-3}(x,y) - B(x,y)| < B_{thold} \end{cases}$$

8. określ ROI, utwórz stos S, określ punkty startowe rozrostu obszarów (ang. seed)

na podstawie $Ft{-}3\ (x, y)$ oraz utwórz tablicę logiczną L i zainicjuj ją wartościami

false

9. $Pt\ (x, y) = RegionGrowing(Dt\ (x, y), seed)$

10. $D't\ (ROI) = Dt\ (ROI) - Pt\ (ROI)$ // rysunek 2.19f.

11. odłóż $D't\ (ROI)$ na stos S

12. L = L or $logical(D't\ (ROI))$ // rysunek 2.19g.

13. if dla każdego (x,y), $L(x, y) \neq false$

14. $B'\ (x, y) = median(S)$

15. return $B'\ (x, y)$

16. else

17. pobierz mapę głębi

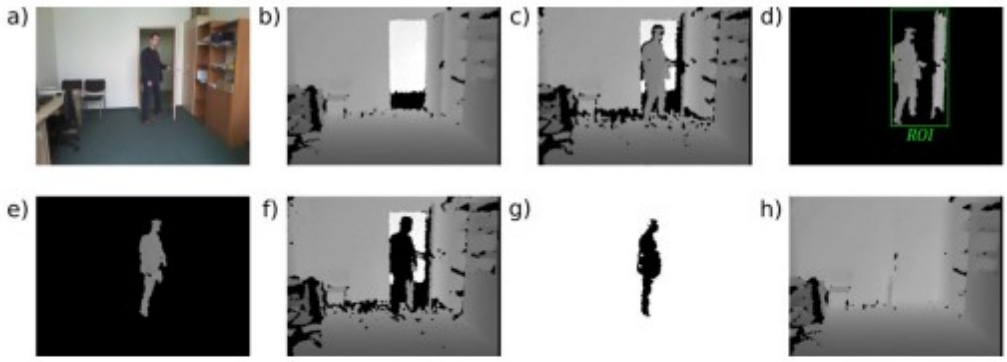
18. wyznacz punkty seed

19. **goto** 9

20. else

21. $B'(x, y) = B(x, y)$

22. return $B'(x, y)$ // rysunek 2.19h



Rysunek 2.19. Ilustracja wydzielania tła w oparciu o zmodyfikowany algorytm budowy tła. a) obraz RGB b) początkowy model tła $B(x, y)$ c) mapa głębi $D(x, y)$ d) obiekty BLOB pierwszego planu e) sylwetka postaci po operacji segmentacji f) mapa głębi $D'(x, y)$ z usuniętą sylwetką człowieka g) tablica logiczna L h) zmodyfikowany model tła $B'(x, y)$.

Pojawienie się nowego komponentu na obrazie wymaga sprawdzenia, czy nowym obiektem na scenie jest inny obiekt niż osoba. Aktualizacja modelu tła w takim wypadku byłaby niepotrzebna, co więcej, mogłaby doprowadzić do włączenia postaci do modelu tła. W razie konieczności aktualizacji tła, działanie algorytmu zilustrować można w następujący sposób:

- określ regiony zainteresowania (ROI), które zawierają zmodyfikowane elementy sceny (należące do zbioru połączonych komponentów),
- przygotuj stos S, na który odkładane będą mapy głębi służące do obliczenia nowego modelu tła (przy użyciu operacji mediany) oraz tablicę logiczną L, rozmiarem odpowiadającą regionom ROI,
- w każdej iteracji algorytmu dokonuj operacji segmentacji postaci (do inicjalizacji wykorzystywane jest położenie postaci przed momentem wykrycia zmiany na scenie, uzyskiwane dzięki przechowywaniu danych w buforze), mapę głębi

pozbawioną postaci odłóż na stos, zaś w tablicy logicznej L oznacz niezerowe piksele jako true,

- gdy dla wszystkich pikseli z ROI wartości w tablicy logicznej są prawdziwe (oznacza to, że dla każdego położenia piksela istnieje w stosie taki obraz, który posiada wartość różną od 0, a więc dla wszystkich pikseli można obliczyć wartości głębokości), wykonaj operację mediany na obrazach ze stosu S.

Przykładowe rezultaty ilustrujące działanie algorytmu dostępne są pod adresem http://fenix.univ.rzeszow.pl/~mkepski/demo/bg/

1.1.3.3. Wydzielenie postaci

Segmentacja obrazu (ang. image segmentation) jest procesem podziału obrazu na obszary, które odpowiadają poszczególnym widocznym na obrazie obiektom i są jednorodne pod względem pewnych wybranych własności. Celem segmentacji jest zazwyczaj oddzielenie poszczególnych obiektów wchodzących w skład obrazu i wyodrębnienie ich od tła, na którym występują, np. wydzielenie postaci znajdującej się na scenie. Większość metod segmentacji można zaliczyć do następujących kategorii (Skarbek & Koschan, 1994; Szeliski, 2010):

- Metody działające w oparciu o histogram. Segmentacja polega na doborze progu na podstawie histogramu obrazu, zaś jej wynikiem jest obraz binarny. Histogram może być zbudowany w dziedzinie koloru, jasności lub głębi. Niskie nakłady obliczeniowe pozwalają na zastosowanie takich metod w przetwarzaniu danych video

- Metody krawędziowe wykorzystują fakt, że granice regionów i krawędzie w obrazie są ściśle powiązane. Krawędzie, które zostały zidentyfikowane na obrazie (przy zastosowaniu np. gradientu) muszą być połączone w celu uformowania zamkniętej krzywej otaczającej obszary jednorodne. Metody aktywnych konturów pozwalają na wykrycie krawędzi otaczających regiony o podobnych własnościach (Szeliski, 2010).

- Metody obszarowe:
  - oparte o rozrost obszarów (ang. region growing) – polegają na poszukiwaniu elementów o podobnych właściwościach i łączeniu ich ze sobą,
  - metody podziału i łączenia (ang. split & merge).

Istnieją także metody hybrydowe wykorzystujące więcej niż jedną z powyższych metod, np. łączące rozrost obszarów z informacją o przebiegu krawędzi. Osobną grupą są techniki wykorzystujące metody uczenia maszynowego.

Ze względu na ograniczone możliwości platform mobilnych zaleca się wydzielenie sylwetki człowieka w sekwencji obrazów głębokości. Metoda ta polega na wykorzystaniu uprzednio

zbudowanego modelu tła do wydzielenia postaci. Piksel obrazu o głębokości D_t i

współrzędnych (x, y) uznawany jest za należący do tła, jeśli wartość jego głębokości nie różni

się od wartości głębokości piksela modelu tła B_t o tych samych współrzędnych o więcej niż

pewien predefiniowany próg B_{thold} dobrany eksperymentalnie. Algorytm można przedstawić

następująco:

Algorytm 4 (Algorytm wydzielania sylwetki człowieka):

1. Pobierz nową mapę głębi $D_t(x, y)$
2. Dla każdego piksela:
3. If $|D_t(x, y) - B_t(x, y)| > B_{thold}$
4. $F_t(x, y) = D_t(x, y)$
5. else:
6. $F_t(x, y) = 0$

Wynik operacji różnicy obrazów przedstawiono na poniższym rysunku. Jak można zaobserwować obraz wynikowy jest zaszumiony, lecz postać jest dość dobrze widoczna. Konieczne jest zatem przeprowadzenie operacji usunięcia szumu i wyodrębnienia z obrazu komponentów reprezentujących obiekty znajdujące się na scenie.

Rysunek 2.20. Operacja różnicy obrazów głębi. Od lewej: obraz tła, aktualny obraz głębi, różnica obrazów (zilustrowana w oparciu o obraz binarny).

Algorytm przetwarza wejściowy obraz binarny w ten sposób, że połączonym ze sobą obszarom pikseli nadawana jest wspólna etykieta. Można to przedstawić jako transformację

obrazu binarnego B w obraz symboliczny S, taką że:

- wszystkie elementy obrazu binarnego B mające wartość tła, również będą należały do tła na obrazie S,
- każdy maksymalny podzbiór elementów obrazu binarnego B niebędących tłem, otrzyma unikalną etykietę (reprezentowaną przez kolejne liczby naturalne) na obrazie symbolicznym S.

Sąsiedztwo pikseli może być 4-elementowe i dla piksela o współrzędnych (u, v) może być

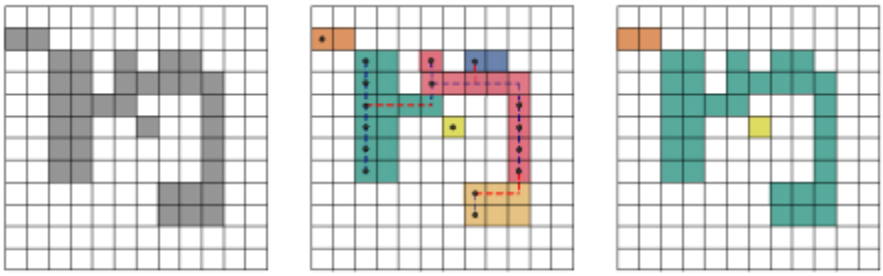
zdefiniowane w następujący sposób:

$$\{(u-1,v),(u+1,v),(u,v-1),(u,v+1)\}$$

lub 8-elementowe i zdefiniowane w następujący sposób:

$$\begin{Bmatrix} (u-1,v),(u+1,v),(u,v-1),(u,v+1), \\ (u-1,v-1),(u+1,v-1),(u+1,v+1),(u-1,v+1) \end{Bmatrix}$$

Większość stosowanych implementacji tej metody opiera się na algorytmie dwuprzebiegowym (ang. two-pass), w którym podczas pierwszego przebiegu pikselom nadawane są tymczasowe etykiety, zaś etykiety regionów sąsiadujących, które są przeznaczone do połączenia są przechowywane w tablicy. W trakcie lub po zakończeniu pierwszego przebiegu, dokonywane jest łączenie obszarów zazwyczaj za pomocą algorytmu union-find. Etykietowanie komponentów o 4-elementowym sąsiedztwie na przykładowym obrazie przedstawia rysunek.



Rysunek 2.21. Etapy etykietowania komponentów połączonych, od lewej: oryginalny obraz binarny, pierwszy etap etykietowania (przebieg horyzontalny) z zaznaczonymi regionami przygotowanymi do połączenia, obraz wynikowy po operacji połączenia sąsiadujących obszarów.

Metodę etykietowania komponentów połączonych można zastosować do usunięcia szumu i wydzielenia obiektów pierwszego planu na mapach głębi. Wprowadzając modyfikację metody, polegającą na łączeniu ze sobą we wspólny komponent tylko tych pikseli, których

wartość głębi różni się nie więcej niż pewien próg Th, wyznaczany eksperymentalnie.

Pozwala to na uniknięcie sytuacji, w której komponenty reprezentujące postać zostały by obarczone szumem lub złączone z innymi obiektami, które mimo sąsiedztwa na obrazie, oddalone są w przestrzeni 3D. Połączone komponenty o odpowiednio dużej liczbie pikseli uznawane są za obiekty pierwszego planu (zob. rysunek 2.22). Gdy liczba komponentów jest większa niż 1, co zazwyczaj oznacza zmianę tła poprzez interakcję użytkownika z otoczeniem,

dokonywana jest adaptacja tła z wykorzystaniem algorytmu opisanego w poprzednim podrozdziale (zob. algorytm 3)

Rysunek 2.22. Działanie metody wydzielania postaci zilustrowane na kilku obrazach z sekwencji. Od góry: obraz kolorowy, mapa głębi, obraz symboliczny pierwszego planu zawierający etykiety obiektów.

Sukces metody aktualizacji tła zależy w dużej mierze od odpowiedniego doboru sekwencji obrazów, na których zostanie wykonana operacja mediany. W idealnym, pożądanym przypadku, pomimo obecności użytkownika na scenie obraz tła zostanie utworzony i uwzględniał będzie całą scenę z pominięciem użytkownika. Obrazy muszą być dobierane z odpowiednim interwałem czasowym, co więcej interwał ten nie zawsze powinien być stały. Należy także wziąć pod uwagę kontekst, w jakim znajduje się osoba na scenie. Osoba może poruszać się z różną prędkością lub przebywać w spoczynku. Wówczas należy pominąć okresy nieaktywności osoby, a więc sytuacje w których cała sylwetka zostałaby zaliczona do nowego modelu tła. W podjęciu decyzji, które obrazy należy uwzględnić przy wyznaczaniu mediany, może być pomocne śledzenie osoby na obrazach głębi.

Zalecane jest powiązanie metody etykietowania połączonych komponentów z algorytmem Mean-shift. Jest to nieparametryczny estymator funkcji gęstości prawdopodobieństwa. Idea

tego algorytmu polega na traktowaniu punktów w d-wymiarowej przestrzeni jak rozkład

prawdopodobieństwa, której gęste regiony korespondują z maksimami funkcji prawdopodobieństwa. Algorytm ten mając do dyspozycji punkt początkowy iteracyjnie wyznacza wektor przesunięcia w kierunku maksimów funkcji rozkładu prawdopodobieństwa.

Mając n punktów x_k, $k = 1, \ldots, n$ w d-wymiarowej przestrzeni $\mathbb{R}^d$ wektor przesunięcia

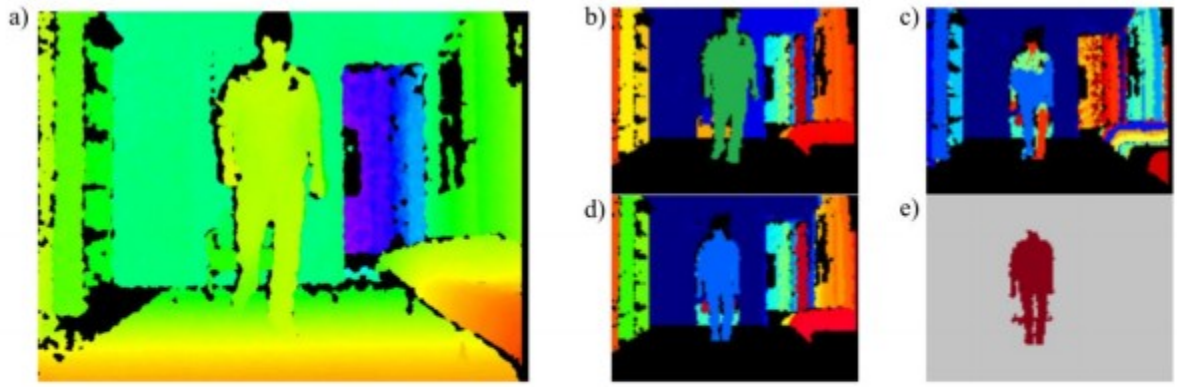
określany jest na podstawie następującej zależności:

$$m(x) = \frac{\sum_{k=1}^{n} x_k g\left(\left\|\frac{x - x_k}{h}\right\|^2\right)}{\sum_{k=1}^{n} g\left(\left\|\frac{x - x_k}{h}\right\|^2\right)} - x$$

gdzie g jest pewną funkcją jądra, a h parametrem wygładzania. Dzięki temu Mean-shift

dokonuje klasteryzacji d-wymiarowej przestrzeni poprzez przypisywanie każdemu punktowi

wartości lokalnego maksimum korespondującej funkcji prawdopodobieństwa. Każdemu pikselowi obrazu przypisano wektor 3D określający jego położenie w przestrzeni. Następnie, po określeniu klastrów pikseli na obrazie, dokonano połączenia tych obszarów, które posiadały zbliżone wartości średniej głębokości, a więc analogicznie jak podczas drugiego przebiegu operacji etykietowania połączonych komponentów. Dzięki temu obiekty, wstępnie podzielone na kilka obszarów, mogły być połączone w jeden spójny region, korespondujący z rzeczywistą reprezentacją obiektu zainteresowania w przestrzeni 3D. Efekty tych operacji zilustrowano w sposób poglądowy na rysunku.



Rysunek 2.23. Wydzielenie postaci: a) mapa głębi, b), c) rezultat segmentacji Mean-shift, d) efektywniejsza segmentacja osoby metodą Mean-shift w połączeniu z metodą sąsiadujących komponentów, e) wydzielona postać.

W celu skrócenia czasu segmentacji należy dokonać segmentacji na przeskalowanych obrazach do rozmiaru 128x96 pikseli.

Algorytm rozrostu obszarów (ang. region growing) jest jednym z algorytmów segmentacji polegającym na podziale obrazu na obszary, w zależności od pewnego zbioru punktów startowych (seed points). Algorytm ten opiera się na założeniu, że piksele w obrębie danego obszaru charakteryzują się podobną wartością intensywności. Wejściem algorytmu jest obraz

oraz zbiór punktów startowych pogrupowany w n podzbiorów: A_1, A_2, A_3, … , A_n . Podzbiory te

mogą zawierać jeden lub więcej punktów. W kolejnych iteracjach algorytmu następuje rozrost początkowych obszarów, poprzez dołączanie pikseli o podobnych wartościach intensywności. Algorytm ten znajduje, taką segmentację obrazu, w której każdy element

obrazu należy do dokładnie jednego obszaru powstałego z rozrostu podzbioru pikseli A_i ,

mając na względzie kryterium homogeniczności. W praktyce, obszary dołączane są do tego podzbioru, dla którego różnica δ pomiędzy wartością intensywności dla piksela a średnią intensywnością regionu jest najmniejsza. Wadą oryginalnego algorytmu jest silna zależność wyniku od kierunku przetwarzania pikseli, co szczególnie zauważalne jest, gdy obraz zawiera dużo małych obszarów o podobnej średniej intensywności pikseli. Jest to wynikiem zastosowania posortowanej listy pikseli, w której przechowywane są nieprzydzielone piksele w kolejności od najmniejszej do największej wartości δ. W kolejnych iteracjach algorytmu wartości δ pikseli znajdujących się na liście nie są aktualizowane, pomimo możliwej zmiany średnich wartości intensywności rozrastających się regionów. W pracy (Mehnert & Jackway, 1997) zaproponowano modyfikację algorytmu, polegającą na zastosowaniu kolejki priorytetowej (ang. priority queue, PQ) posortowanej rosnąco względem wartości δ i kilku stosów pikseli przechowujących piksele o wartości δ, zależnej od pozycji stosu w kolejce PQ. Podczas usuwania pikseli o najwyższym priorytecie (a w konsekwencji najmniejszej wartości δ) przetwarzany jest nie jeden piksel, a cały stos. Zastosowanie mapy kolejek, indeksowanej ma podstawie różnicy głębokości między dołączanymi pikselami a rozrastającym się regionem, zamiast sortowanej kolejki priorytetowej pozwala na uniknięcie konieczności sortowania w kolejnych iteracjach algorytmu.

W poniższej tabeli przedstawiono zestawienie oznaczeń wykorzystanych przy zapisie zmodyfikowanego algorytmu rozrostu obszaru. Następnie przedstawiono pseudokod zaproponowanej metody rozrostu obszaru (Algorytm 5).

Oznaczenia	
REGION_MEAN	Średnia wartość głębi rozrastającego się regionu
δ_{thold}	próg różnicy głębokości, wartość dobierana eksperymentalnie
NHQ	*Neighbours holding queue* – kolejka zawierająca wskaźniki do pikseli sąsiadujących z regionem rozrastającym się
QM	Mapa kolejek indeksowana na podstawie różnicy głębokości. Kluczem elementu jest δ, a wartością kolejka przechowująca piksele posiadające wartość δ.
FQ	Kolejka o najmniejszej wartości δ w danej iteracji
Etykiety	
NOT_LABELED	Piksel nie został odwiedzony
IN_NQUEUE	Piksel znajduje się w NHQ
IN_QUEUE	Piksel znajduje się w QM
LABELED	Piksel został przydzielony do rozrastającego się regionu

Algorytm 5 (Zmodyfikowany algorytm rozrostu obszarów):

1. Nadaj etykiety pikselom regionu początkowego, oblicz REGION_MEAN, piksele sąsiadujące z regionem początkowym wstaw do NHQ, nadaj im etykietę IN_NQUEUE
2. **while** NHQ i QM są niepuste powtarzaj:
3. **while** NHQ jest niepuste powtarzaj:
4. usuń piksel z NHQ 5. oblicz jego δ
5. **if** $\delta > \delta_{thold}$:
6. **continue**
7. wstaw do odpowiedniej kolejki w QM (o indeksie δ)
8. nadaj mu etykietę IN_QUEUE
9. **if** QM jest niepuste:
10. usuń FQ z QM
11. **while** FQ jest niepuste:
12. usuń piksel z FQ
13. nadaj mu etykietę LABELED
14. dodaj jego sąsiednie piksele NOT_LABELED do NHQ i nadaj im etykietę IN_NQUEUE
15. zaktualizuj REGION_MEAN o wartości pikseli usuniętych z FQ

W celu uzyskania dokładniejszej segmentacji postaci oraz uniknięcia możliwości włączenia do komponentu reprezentującego postać pikseli do niej nienależących, szczególnie w przypadku,

w którym osoba dokonuje interakcji z otoczeniem, można wykorzystać detektor postaci oparty o histogramy gradientów i klasyfikator SVM. W przypadku nagłego wzrostu pola powierzchni wydzielonego komponentu, podejście zaprezentowane w pracy (Kępski & Kwolek, 2014a) pozwala na dokonanie detekcji jedynie w obszarze komponentu, a nie na całej mapie głębi. Takie rozwiązanie jest korzystne obliczeniowo, szczególnie w kontekście implementacji systemu na platformie obliczeniowej z procesorem o architekturze ARM. Detekcja postaci dostarcza informacji o położeniu postaci, która może być wykorzystana do ponownej inicjalizacji algorytmu rozrostu obszarów.

1.1.3.4. Śledzenie głowy osoby

Problem śledzenia można przedstawić jako zagadnienie estymacji nieznanego stanu pewnego dynamicznego systemu. Stan ten można przedstawić w uproszczeniu jako zbiór wszystkich czynników mogących mieć wpływ na dany system w przyszłych jednostkach czasu (Thrun et al., 2005). Stan oznaczamy jako x, natomiast stan systemu w konkretnym czasie t jako x_t. Stan ten może być jedynie estymowany na podstawie danych pomiarowych, oznaczonych jako z. Klasycznym przykładem algorytmu estymacji ukrytego stanu systemu jest filtracja Bayesa.

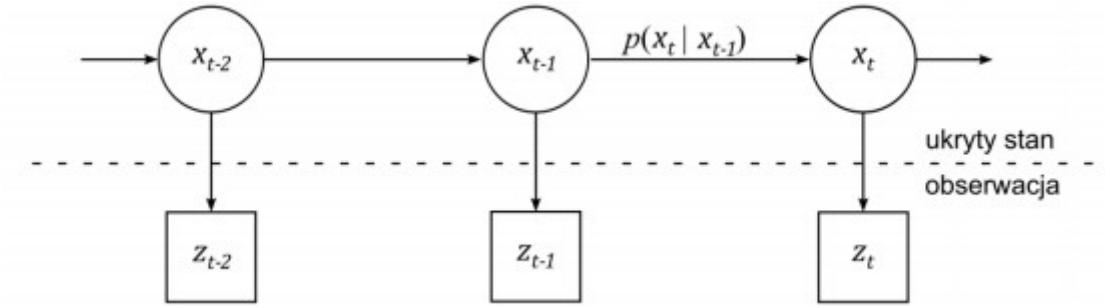
Filtry Bayesa stosuje się zakładając, że stochastyczny proces spełnia kryteria procesu Markowa, co oznacza że stan x_t zależy bezpośrednio od stanu procesu w poprzedzającej chwili czasu:

$$p(x_t | x_{1:t-1}) = p(x_t | x_{t-1})$$

gdzie p jest warunkowym rozkładem prawdopodobieństwa. Założenia modelu Markowa dotyczą także niezależności poszczególnych pomiarów z:

$$p(z_t | x_{1:t}) = p(z_t | x_t)$$

Sieć Bayesa została przedstawiona na rysunku.



W metodach bayesowskich wiedza o aktualnym stanie systemu w chwili czasu t

reprezentowana jest za pomocą warunkowego rozkładu prawdopodobieństwa a posteriori

$p(x_t \mid z_t)$ zmiennej stanu x_t. Filtry Bayesa rozwiązują problem estymacji stanu systemu

poprzez zastosowanie następujących po sobie mechanizmów predykcji i korekcji. Załóżmy, że

poprzedni rozkład prawdopodobieństwa $p(x_{t-1} \mid z_{t-1})$ jest znany w czasie t. Etap predykcji

polega na określeniu rozkładu prawdopodobieństwa bez znajomości nadchodzącego w czasie

t pomiaru z_t :

$$p(x_{t-1} \mid z_{1:t-1}) \rightarrow p(x_t \mid z_{1:t-1})$$

Etap ten można przedstawić za pomocą równania Chapmana-Kołmogorowa:

$$p(x_t \mid z_{1:t-1}) = \int p(x_t \mid x_{t-1}) p(x_{t-1} \mid z_{1:t-1}) dx_{t-1}$$

Etap korekcji polega na określeniu rozkładu prawdopodobieństwa przy znajomości pomiaru

z_t . Etap ten następuje po wykonaniu etapu predykcji:

$$\{p(x_t|\, z_{1:t-1}), z_t\} \rightarrow p(x_t|\, z_{1:t})$$

$$p(x_t|\, z_{1:t}) = p(z_t|\, x_t)p(x_t|\, z_{1:t-1})$$

Sekwencyjną aktualizację warunkowego rozkładu prawdopodobieństwa a posteriori można przedstawić następująco:

$$p(x_t|\, z_{1:t}) = p(z_t|\, x_t) \int p(x_t|\, x_{t-1})p(x_{t-1}|\, z_{1:t-1})dx_{t-1}$$

Do najczęściej spotykanych metod wyznaczania estymaty wektora stanu modelu dynamicznego systemu opartych o sieci Bayesa, można zaliczyć filtr Kalmana (ang. Kalman Filter) (Kalman, 1960), rozszerzony filtr Kalmana (ang. Extended Kalman Filter) (Ljung, 1979) oraz filtr cząsteczkowy (ang. Particle Filter) (Gordon et al., 1993).

Do oszacowania bieżącego stanu systemu filtr Kalmana wymaga jedynie znajomości stanu poprzedniego oraz wektora obserwacji. W procesie filtracji można wyróżnić dwie fazy: predykcji oraz uaktualniania. Podczas predykcji na podstawie bieżącego stanu systemu przewidywane są przyszłe wartości wektora stanu, natomiast na etapie uaktualniania wykorzystuje się bieżące obserwacje, co z kolei prowadzi do poprawy przewidywanego wektora stanu. Filtr Kalmana jest optymalnym estymatorem, ponieważ przy konkretnych założeniach może spełniać pewne kryterium, np. minimalizacji błędu średniokwadratowego estymowanych parametrów. Rozszerzony filtr Kalmana jest przykładem modyfikacji oryginalnego algorytmu, pozwalającym na zastosowanie go w systemach dyskretnych. Największą wadą rozszerzonego filtru Kalmana jest to, że poddanie zmiennej losowej o rozkładzie normalnym nieliniowym transformacjom spowoduje zmianę rozkładu na inny niż normalny. Algorytm ten jest prostym estymatorem stanu, który poprzez linearyzację przybliża bayesowskie reguły filtracji. Ponadto metoda ta nie nadaje się szczególnie dobrze do śledzenia wielu hipotez jednocześnie. Pomimo wspomnianych wad, metoda ta jest szeroko stosowana, ze względu na swoją prostotę i stosunkowo niewielką złożoność obliczeniową,

która wynosi $O(k^{2,8} + n^2)$, gdzie k jest długością wektora z_t, a n liczbą wymiarów przestrzeni

stanu x_t.

Filtr cząsteczkowy jest nieparametryczną implementacją filtracji bayesowskiej. Główną ideą jest reprezentacja rozkładu prawdopodobieństwa a posteriori za pomocą skończonego zbioru

próbek wylosowanych na podstawie danego rozkładu. Estymatorem miary

prawdopodobieństwa p jest rozkład empiryczny o postaci:

$$p(x) = \frac{1}{N}\sum_{i=1}^{N} \delta(x - x^{(i)})$$

gdzie δ jest deltą Diraca, natomiast ciąg jest $\{x^{(i)}\}_{i=1}^{N}$ ciągiem niezależnych próbek, zwanych

cząstkami. Każda cząstka jest instancją stanu w czasie t, a więc hipotezą na temat

prawdziwego, ukrytego stanu w czasie t. W praktyce do wygenerowania próbki stosuje się

metody Monte Carlo wykorzystujące łańcuchy Markowa (MCMC) lub rozkład proponowany (funkcję istotności). Liczba cząstek jest zazwyczaj duża i zależy od wymiarów wektora stanu.

Istnieją implementacje, w których liczba cząstek jest funkcją zmiennej t lub innych wartości

związanych z rozkładem prawdopodobieństwa p. Metoda oparta na funkcji ważności polega

na zastąpieniu rozkładu p rozkładem q o podobnych właściwościach oraz późniejszym

wykorzystaniu go do generacji próbki niezależnych, ważonych zmiennych losowych $\{x^{(i)}$,

$i \quad N \qquad\qquad\qquad\qquad i$
$w^{(\,)}\}_{=1}$. Wagi $w^{(\,)}$ odzwierciedlają prawdopodobieństwo, że próbki pochodzą z rozkładu p.
$\quad i$

Wybór funkcji istotności jest istotnym krokiem, gdyż od niej zależy efektywność filtru. W najprostszej wersji filtru wykorzystywana jest sekwencyjna funkcja ważności (ang. Sequential Importance Sampling – SIS), w której wagi wyznacza się z następującej zależności:

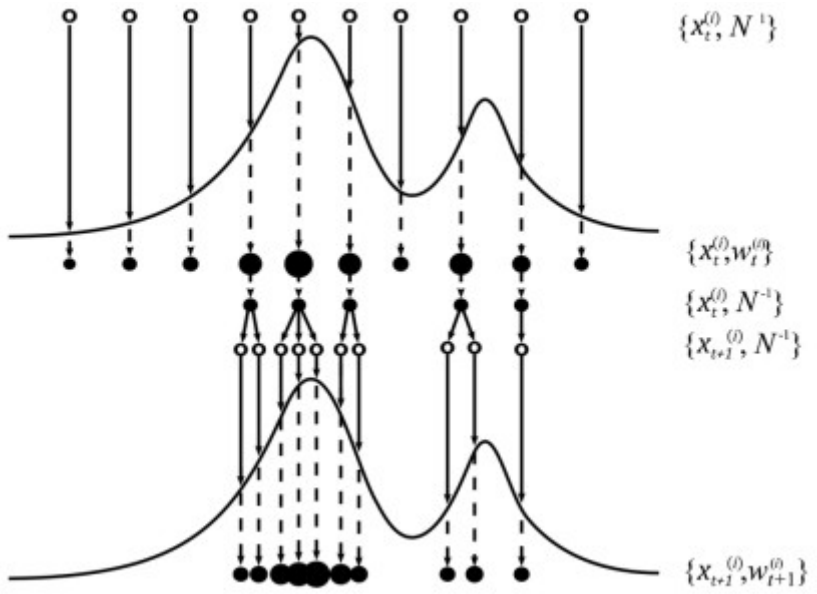
$$w_t^{(i)} = \frac{p(x_t^{(i)}|z_t)}{q(x_t^{(i)}|z_t)} \propto w_{t-1}^{(i)} \frac{p(z_t|x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)})}{q(x_t^{(i)}|x_{t-1}^{(i)}, z_t)}$$

Aby zapobiec degeneracji próbek, czyli spadkowi wartości wag do wartości zaniedbywalnych, stosuje się mechanizm przepróbkowania (ang. resampling). Polega on na zastąpieniu cząstek

z niewielkimi wagami nowym zbiorem cząstek N z jednakowymi wagami:

$$\left\{x_t^{(i)}, w_t^{(i)}\right\} \rightarrow \left\{x_t^{(i)}, \frac{1}{N}\right\}$$

Dzięki temu algorytm skupia się na tych obszarach przestrzeni stanu, w których cząstki stosunkowo trafnie odwzorowują stan systemu. Omawiany algorytm jest rozszerzoną wersją algorytmu SIS i znany jest w literaturze pod nazwą SIR (ang. Sampling Importance Resampling). Schemat jego działania przedstawiono na rysunku.



Metoda ta polega na przeprowadzeniu następujących po sobie kroków predykcji, korekty oraz ponownego próbkowania. Dzięki temu zapobiega się propagowaniu nieistotnych cząstek

do kolejnych iteracji. Algorytm filtru cząsteczkowego SIR można przedstawić w następujący sposób:

Algorytm 4:

1. $\left\{x_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{N} = \mathrm{PF}\left(\left\{x_{t-1}^{(i)}, w_{t-1}^{(i)}\right\}_{i=1}^{N}, z_t\right)$
2. **Dla** i = 0, 1, …, N
3. $x_t^{(i)} \sim p\left(x_t^{(i)} \middle| x_{t-1}^{(i)}\right)$
4. $w_t^{(i)} = p\left(z_t \middle| x_t^{(i)}\right)$
5. $W_s = \sum_{i=1}^{N} w_t^{(i)}$
6. **Dla** i = 0, 1, …, N
7. $w_t^{(i)} = \frac{w_t^{(i)}}{W_s}$
8. $\left\{x_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{N} = \mathrm{Resampling}\left(\left\{x_t^{(i)}, w_t^{(i)}\right\}_{i=1}^{N}\right)$

W proponowanym rozwiązaniu głowa postaci jest śledzona przy wykorzystaniu filtru cząsteczkowego. Kształt głowy postaci zamodelować można za pomocą elipsoidy. Model obserwacji łączy w sobie wartości funkcji dopasowania pomiędzy elipsoidą a chmurą punktów. Charakterystyka danych pochodząca z czujnika głębi jest taka, że odległość pomiędzy sąsiednimi elementami chmury punktów jest niewielka dla obiektów znajdujących się blisko kamery i rośnie wraz z nią. Obiekty znajdujące się dalej od kamery reprezentowane są w postaci warstwic, co powoduje częściową utratę informacji o ich rzeczywistym kształcie względem osi z układu kamery (głębokości). Omawiane zjawisko dotyczy szczególnie wszelkiego rodzaju krzywizn. W celu określenia dopasowania modelu do chmury punktów, w pierwszej kolejności wyznaczana jest odległość punktu (x, y, z) do elipsoidy za pomocą następującej zależności:

$$d = \sqrt{\frac{(x-x_0)^2}{a^2} + \frac{(y-y_0)^2}{b^2} + \frac{(z-z_0)^2}{c^2}} - 1$$

gdzie a, b, c to półosie elipsoidy. Stopień przynależności punktu do elipsoidy określany jest w

następujący sposób:

$$m = 1 - \frac{d}{d_{thold}}$$

gdzie d_{thold} jest progiem ustalonym eksperymentalnie. Funkcja dopasowania modelu głowy

postaci do chmury punktów określona jest więc następująco:

$$f_1 = \sum_{(x,y,z)\epsilon S} m(x,y,z)$$

gdzie S(x_0, y_0, z_0) oznacza zbiór punktów należących do głowy postaci i jej otoczenia.

Model obserwacji, wykorzystujący funkcje dopasowania 2D i 3D, zdefiniowany jest za pomocą zależności:

$$p(\mathrm{x}_t^{(i)}|\mathrm{z}_t^{(i)}) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{f_1 \cdot f_2}{2\sigma^2}\right)$$

Pomiędzy jednostką czasu $t-1$ a t wszystkie cząstki są propagowane w przestrzeni stanu

zgodnie z modelem lokomocji

$$\mathrm{x}_t^{(i)} = \mathrm{x}_{t-1}^{(i)} + \delta, \qquad \delta = N(0,\Sigma)$$

Na rysunku przedstawiono przykładowe obrazy ilustrujące śledzenie głowy postaci dla sekwencji z kamerą statyczną za pomocą proponowanej metody i filtru składającego się z 500 cząsteczek. Wartości δ zostały dobrane eksperymentalnie.

Wektor stanu składa się z pięciu elementów i zawiera współrzędne 3D środka elipsoidy (x_0, y_0, z_0) oraz wartości dwóch kątów obrotu modelu głowy (pitch, roll). Jak można zaobserwować na rysunku ewaluacja śledzenia była przeprowadzona podczas interakcji użytkownika z obiektami sceny. W rozwiązaniu opartym jedynie o model tła doszłoby do sytuacji, w której model tła stawał się nieaktualny i na obrazie pierwszego tła pojawiłyby się inne obiekty oprócz użytkownika. Dzięki zastosowaniu śledzenia głowy osoby, uzyskano informację o położeniu istotnej części ciała człowieka. Wykorzystanie wspomnianej informacji może znacząco polepszyć skuteczność detekcji upadku.

1.1.3.5. Algorytm, deskryptory upadku

W literaturze (Noury et al., 2008) zaproponowano podział aktywności człowieka związanej z typowym upadkiem, na następujące fazy:

1. normalna czynność dnia codziennego, taka jak chodzenie, wstawanie z krzesła czy łóżka, itp.,
2. faza krytyczna, w której człowiek znajduje się w niekontrolowanym ruchu przypominającym spadek swobodny, po którym następuje uderzenie (ang. impact),
3. faza po upadku, zazwyczaj związana z leżeniem na podłodze
4. faza próby powrotu do normalnej postawy, która może zakończyć się niepowodzeniem lub w ostateczności może do niej nie dojść, np. w wyniku urazu lub utraty przytomności.

Część prac badawczych wykorzystuje założenie, że upadek kończy się w większości przypadków w pozycji leżącej (lub rzadziej siedzącej) i dokonuje detekcji w oparciu o orientację ciała człowieka (ang. posture).

Proponowany algorytm detekcji upadku można podzielić na kilka etapów. Pierwszym jest akwizycja danych z kamery głębi. Mapy głębi zostają poddane operacjom wstępnego przetwarzania (filtracji) i zostają zapisane do bufora cyklicznego. Przechowywanie map w buforze jest niezbędne do wykonania operacji budowy i aktualizacji modelu tła. W kolejnym kroku zostaje wykonana operacja wydzielenia pierwszego planu i etykietowania połączonych komponentów. Przypadek pojawienia się więcej niż jednego połączonego komponentu na obrazie różnicy aktualnej mapy głębi i modelu tła traktowany jest jako zmiana elementów sceny (np. poprzez interakcję użytkownika z otoczeniem). System rozpoczyna procedurę aktualizacji modelu tła. W sytuacji braku zmian otoczenia użytkownika realizowana jest

procedura detekcji upadku. Następnie dokonywane jest wydzielenie cech sylwetki postaci na obrazach głębi oraz poddanie ich klasyfikacji. Gdy wykryto upadek, system zgłasza alarm, w przeciwnym razie kontynuowane jest działanie algorytmu dla nowych danych.

Detekcję upadku można rozpatrywać jako problem klasyfikacji dwuklasowej. Przygotowanie odpowiedniego klasyfikatora wymaga opracowania zbioru cech postaci wydzielanych z map głębi oraz wyboru cech istotnych z punktu widzenia klasyfikacji. Wybór cech jest kluczowym problemem w dziedzinie widzenia i uczenia maszynowego i w dużym stopniu wpływa na skuteczność klasyfikacji oraz wynik działania systemu wizyjnego (Dollar et al., 2007). Cecha o dużej sile dyskryminacyjnej powinna:

- nieść dużą ilość informacji (ang. informative),
- być niewrażliwa na szum lub określony zbiór transformacji,
- wyznaczanie powinno być zrealizowane niskim kosztem obliczeniowym,
- charakteryzować się niewielką wariancją wewnątrz klasy decyzyjnej oraz dużą wariancją pomiędzy klasami.

W kontekście systemu detekcji upadku opartego o obrazy głębi można wyróżnić także inne pożądane cechy:

- odporność na szum charakterystyczny dla sensora głębi,
- odporność na przesłonięcia sylwetki czy złączenia sylwetki z elementami sceny,
- czas wyznaczania pozwalający na działanie systemu w czasie rzeczywistym.

Ze zbioru cech proponuje się wybrać następujące deskryptory:

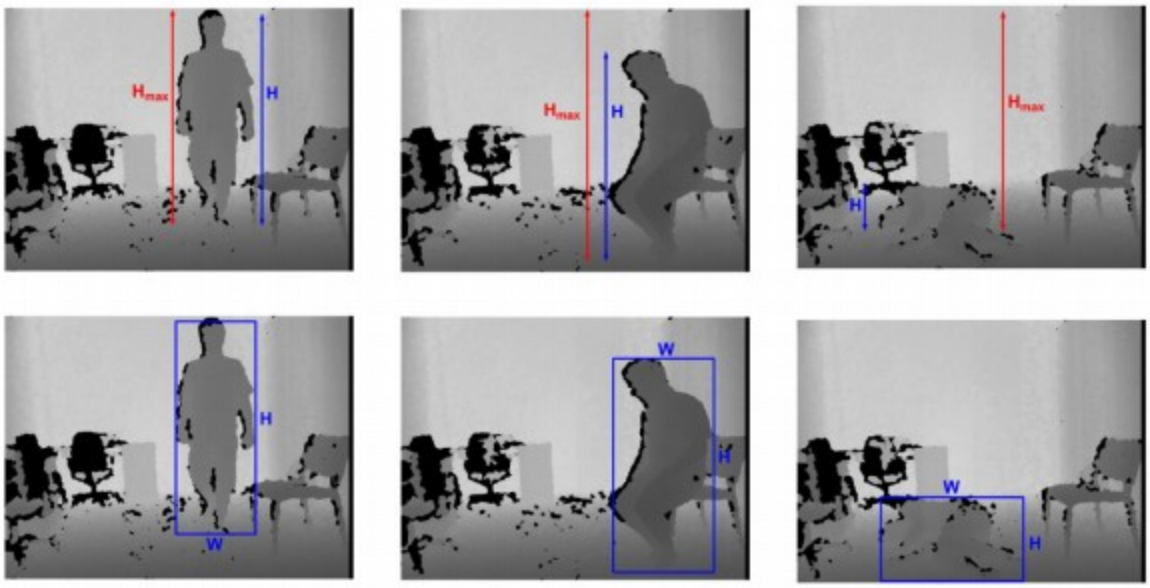
- H/W – stosunek wysokości do szerokości wydzielonej postaci, wyznaczony na podstawie map głębi (rysunek 3.6),
- H/H_{max} – stosunek wysokości wydzielonej postaci w danej klatce obrazu do jej rzeczywistej wysokości w postawie wyprostowanej, wyznaczony na podstawie chmury punktów,
- *D* – odległość środka ciężkości postaci od płaszczyzny podłogi wyrażona w milimetrach,

- $max(\sigma_x, \sigma_z)$ – maksymalne odchylenie standardowe wartości punktów należących

  do postaci od jej środka ciężkości, wzdłuż osi X i Z układu współrzędnych kamery Kinect,

- $P40$ – stosunek liczby punktów należących do postaci, leżących w

  prostopadłościanie o wysokości 40 cm, umieszczonym nad podłogą, do liczby wszystkich punktów należących do postaci (rysunek 3.7)



Rysunek 3.6. Cechy H/W oraz H/H_{max} przedstawione dla różnych sylwetek postaci: stojącej, siedzącej i leżącej.

Cecha H/W jest często spotykana w literaturze w kontekście detekcji upadku (Mastorakis &

Makris, 2012; Anderson et al., 2006; Liu et al., 2010). Gdy osoba stoi, stosunek wysokości do szerokości prostokąta jest duży, zazwyczaj większy od 1. Gdy osoba znajduje się w pozycji leżącej, wartość ta jest znacznie mniejsza. Jednak detekcja upadku jedynie w oparciu o tą cechę charakteryzuje się skutecznością około 80%, jak przedstawiono w pracy (Stone & Skubic, 2014). Cecha ta może być podatna na przysłonięcia i prowadzić do błędów klasyfikacji drugiego rodzaju, gdy część osoby po upadku znajduje się poza polem widzenia kamery. Ponadto, ten sam błąd może wystąpić, gdy osoba zostanie niezbyt precyzyjnie wydzielona, np. w wyniku zmiany położenia obiektów na scenie w momencie upadku

Miary $H/Hmax$ oraz D uzależnione są przede wszystkim od wysokości sylwetki postaci w

danej chwili. Cecha D została już przedstawiona w literaturze (Rougier et al. 2011), jednak jej

niewielka skuteczność wynikała z zastosowania jej samodzielnie oraz wyborze prostego

sposobu wnioskowania o upadku w oparciu o próg wartości. Cechy $max(\sigma x, \sigma z)$ oraz $P40$,

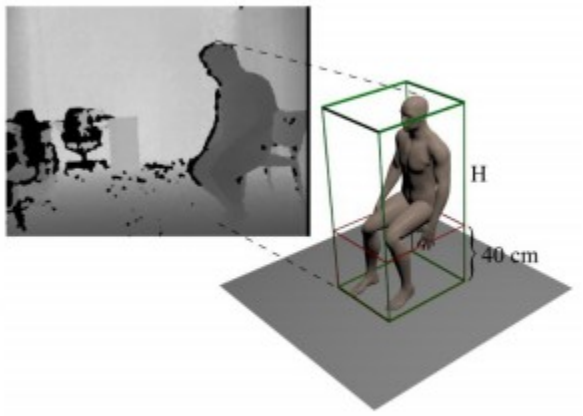
nie były prezentowane uprzednio w żadnych pracach naukowych innych zespołów badawczych. Zostały one zaproponowane celem odróżnienia od upadku akcji, w których osoba schyla się po przedmiot, kuca lub wykonuje podobne czynności. Duże wartości

odchylenia standardowego punktów należących do postaci względem osi x lub y, są

charakterystyczne dla leżącej sylwetki. Istotne jest to, że odchylenie standardowe wyznaczane jest na podstawie chmury punktów zamiast mapy głębi. Dzięki temu można uniezależnić wartość omawianego atrybutu od orientacji leżącej osoby względem kamery.

Cecha $P40$ ma za zadanie odróżniać sytuację, w których postać znajduje się blisko podłogi,

wydzielonej wcześniej algorytmami v-dysparycji i RANSAC.



Rysunek 3.7. Cecha P_{40} przedstawiona na mapie głębi i odpowiadającym mu modelu postaci 3D.

1.1.3.6. Klasyfikatory

Z przeglądu literatury wynika, że w pracach dotyczących detekcji upadku najczęściej wykorzystywane są drzewa decyzyjne, algorytm SVM oraz metoda k-najbliższych sąsiadów (ang. *k-Nearest Neighbor*, k-NN). Stosunkowo nieliczne są prace, w których wykorzystywane są podejścia oparte o logikę rozmytą czy też inne metody. Tym niemniej, potencjał aplikacyjny wielu z nich, a w szczególności metod opartych o *soft-computing*, czy też zespoły klasyfikatorów (Woźniak et al., 2014) może być znaczący. Proponuje się aby detekcja upadku realizowana była w oparciu o klasyfikator k-NN, SVM.

Metoda k-najbliższych sąsiadów (Cover & Hart, 1967) jest nieparametryczną metodą wykorzystywaną często w zadaniach klasyfikacji. Metoda ta opiera się na określaniu odległości między próbką testową, a zbiorem próbek uczących. Odległość euklidesowa próbki

x_i zawierającej p-elementowy wektor cech ($x_{i1}, x_{i2}, \dots, x_{ip}$) od próbki x_l określona jest

następująco:

$$d(x_i, x_l) = \sqrt{(x_{i1} - x_{l1})^2 + (x_{i2} - x_{l2})^2 + \cdots + (x_{ip} - x_{lp})^2}$$

Klasyfikacja dla k = 1, przebiega na przypisaniu próbce testowej x_i etykiety klasy jej

najbliższego sąsiada. Gdy k > 1, próbka testowa otrzymuje etykietę klasy, która najczęściej występuje wśród jej k-sąsiadów. Metoda ta należy do tzw. leniwych technik klasyfikacji (ang. lazy learning), które nie generują modelu (hipotezy) dla funkcji docelowej, lecz w momencie klasyfikacji próbek testowych odpowiedź klasyfikatora jest uzyskiwana na podstawie próbek trenujących. Do zalet tego typu technik można zaliczyć zdolność do klasyfikacji problemów o zmieniającym się zbiorze treningowym bez konieczności ponownej budowy modelu. Jednak strategia wyznaczania odległości pomiędzy próbkami na etapie klasyfikacji obarczona jest pewnym kosztem obliczeniowym. Najprostsze metody *brute force* polegają na obliczaniu

odległości między wszystkimi próbkami, co prowadzi do złożoności obliczeniowej $O[DN^2]$ dla

N próbek o liczbie wymiarów D. W celu ograniczenia kosztu obliczeniowego algorytmu

klasyfikacji podjęto próby implementacji algorytmu k-NN w oparciu o różne struktury danych, m. in. kd-drzewa czy inne rodzaje drzew binarnych (Bentley, 1975). Wcześniejsza budowa drzewa pozwala na uniknięcie potrzeby wyznaczania wszystkich odległości potrzebnych do wybrania k sąsiadów, co w konsekwencji pozwala na zmniejszenie złożoności obliczeniowej

wyszukiwania sąsiadów do $O[DN \log N]$ lub mniejszej (Arya et al., 1998).

Podstawową ideą algorytmu SVM jest znalezienie hiperpłaszczyzny pozwalającej na separowanie danych w możliwie optymalny sposób (Cortes & Vapnik, 1995). Jeśli w zadaniu klasyfikacji binarnej zbiór danych jest liniowo separowalny, wówczas istnieje co najmniej jedna hiperpłaszczyzna:

$$\hat{g}(x) = w^T x + w_0 = 0$$

pozwalająca na rozgraniczenie wektorów danych należących do różnych klas. Margines hiperpłaszczyzny rozdzielającej można zdefiniować w następujący sposób:

$$m = (d^+ + d^-)$$

gdzie d^+ i d^- to odległości między hiperpłaszczyzną rozdzielającą a najbliższym punktem z

klasy reprezentującej dodatnie i ujemne przykłady, odpowiednio. Intuicyjnie, za lepsze można uznać te płaszczyzny, które przebiegają możliwie daleko od obiektów obydwu klas, zapewniające jak największy margines separacji. Hiperpłaszczyzna taka zapewnia najlepszą zdolność klasyfikatora do uogólniania i jest nazywana optymalną hiperpłaszczyzną rozdzielającą (ang. optimal separating hyperplane, OSH). Klasyfikator liniowy, w którym hiperpłaszczyzną rozdzielającą jest OSH nazywany jest liniowym klasyfikatorem SVM. Problem poszukiwania hiperpłaszczyzny z maksymalnym marginesem, a więc wyznaczenia

wektora wag w i stałej $w0$, jest problemem optymalizacji kwadratowej, dla którego

przedstawiono efektywne rozwiązania w pracy (Cristianini & Shawe-Taylor, 1999). Dla problemu nieseparowalnego liniowo znalezienie optymalnej hiperpłaszczyzny polega na

wprowadzeniu zmiennych osłabiających $\{\xi_i\}_{i=1}^{n}$, które są miarą odchylenia danego wektora uczącego od przypadku liniowej separowalności. Wprowadzenie w procesie optymalizacji do funkcji celu sumy składnika, uwzględniającego wartości zmiennych osłabiających oraz

parametru C, w postaci:

$$c_o = C\left(\sum_{i=1}^{n} \xi_i\right)$$

pozwala na uwzględnienie kosztu c_o związanego z błędną klasyfikacją. Dzięki dobraniu

odpowiedniej wartości parametru C możliwe jest uzyskanie kompromisu pomiędzy liczbą

błędnie sklasyfikowanych próbek, a szerokością marginesu separacji. Należy wspomnieć, że oprócz liniowego klasyfikatora SVM istnieją jego nieliniowe odmiany wykorzystujące transformację przestrzeni wejściowej za pomocą nieliniowego przekształcenia (tzw. funkcji jądra).

1.1.3.7. Wyniki badań

W rozprawie doktorskiej „Detekcja upadku i wybranych akcji na sekwencjach obrazów cyfrowych" badania zrealizowano na sekwencjach obrazów głębi ze zbioru danych UR Fall Dataset. Przygotowane sekwencje zawierają upadki oraz typowe czynności dnia codziennego, które wykonane były przez 6 osób, zarówno w środowisku laboratoryjnym jak i domowym. W tabeli zestawiono wykonane czynności z podziałem na upadki i czynności dnia codziennego (ADLs). Zrealizowano upadki rozpoczynające się zarówno od pozycji stojącej jak i siedzącej. Warto podkreślić, że w większości prac dotyczących detekcji upadku rozpatruje się jedynie upadki z pozycji stojącej. Tym niemniej upadki z pozycji siedzącej mogą stanowić duży odsetek upadków, w szczególności w środowisku domowym. Ogólna liczba sekwencji wynosi 70, spośród których 30 prezentuje upadki.

	typ akcji	liczba akcji
upadek	z pozycji stojącej	15
	z pozycji siedzącej	15
ADLs	siadanie	10
	leżenie	10
	kucanie	10
	schylanie	10

Z przeglądu literatury wynika (Igual et al., 2013), że wiele systemów do detekcji upadku wykorzystuje jedynie informację o pozie leżącej. Celem przebadania użyteczności tego podejścia, w oparciu o przygotowany zestaw deskryptorów dokonano klasyfikacji pozy osoby. Czułość, swoistość, dokładność i precyzję klasyfikatora określono w oparciu o zestaw cech omówiony w podrozdziale Deskryptory upadku. Wspomniane parametry określono w oparciu

o następujące cechy: H/W, H/H_{max}, D, $max(\sigma_x, \sigma_z)$, P_{40}. W tabeli zebrano uzyskane wyniki

klasyfikacji pozy osoby. Jak zaobserwować można, osiągnięto wysoką czułość i swoistość systemu, jednak wyniki klasyfikacji nie są pozbawione błędów pierwszego i drugiego rodzaju. Co więcej, w praktycznym zastosowaniu, wspomniane błędy przełożyłyby się na wystąpienie fałszywych alarmów oraz pominięcie niektórych upadków.

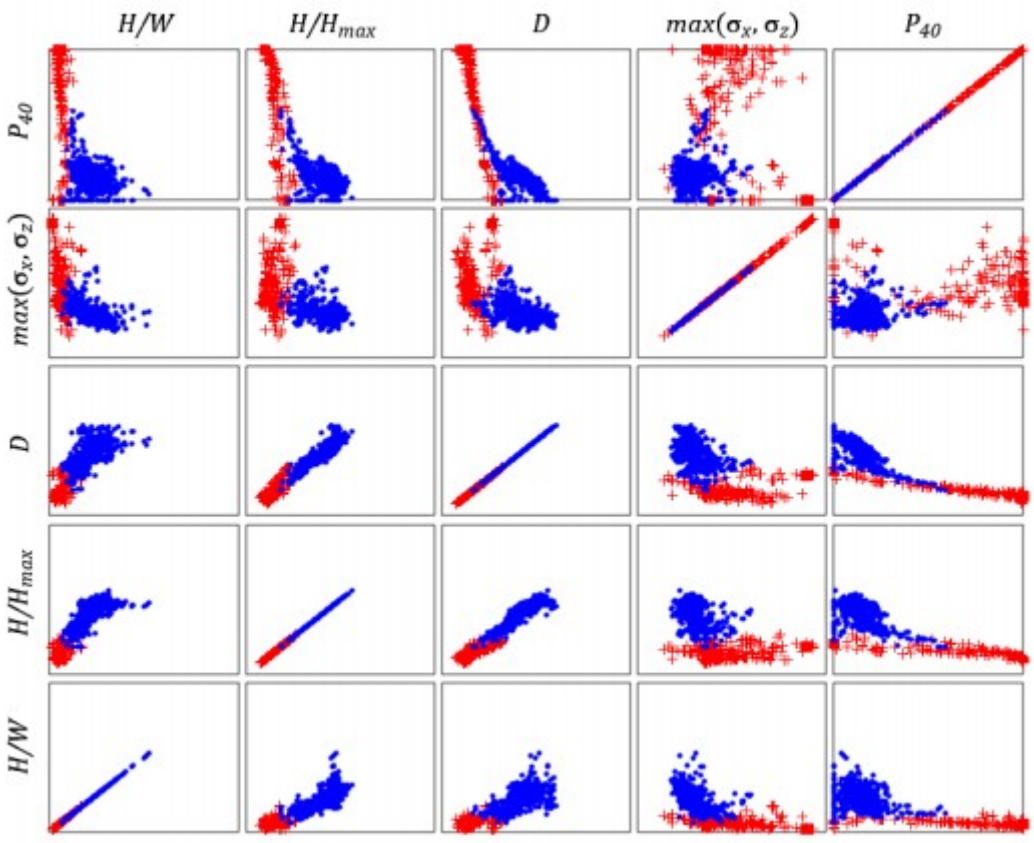
			Rzeczywiste pozy osoby		
			osoba w pozie leżącej	osoba w pozie nieleżącej	
Przewidywane pozy osoby	k-NN(3)	osoba w pozie leżącej	898	6	Dokładność = 99,55%
		osoba w pozie nieleżącej	5	1516	Precyzja = 99,34%
			Czułość = 99,45%	Swoistość = 99,61%	

W algorytmie przedstawionym w rozdziale Algorytm, deskryptory upadku w bloku odpowiedzialnym za klasyfikację wykorzystano klasyfikatory k-NN oraz SVM. Celem oceny wskaźników jakościowych zrealizowano badania eksperymentalne, których wyniki zestawiono w tabeli. W omawianej tabeli zamieszczono także wyniki uzyskane przez reprezentatywne i powszechnie przywoływane w literaturze metody.

		k-NN	*UFT* (Bourke et al., 2007)	*LFT* (Bourke et al., 2007)
Wyniki	**Dokładność**	90,00%	88,57%	78,57%
	Precyzja	81,08%	78,95%	68,29%
	Czułość	100%	100,00%	93,33%
	Swoistość	82,5%	80,00%	67,50%

Detekcja upadku jedynie w oparciu o klasyfikację pozy w jakiej znajduje się osoba (zob. wyniki uzyskiwane przez k-NN), prowadzi do niezadowalającej swoistości i precyzji metody, mając na względzie praktyczne zastosowania w systemach nieprzerwanego monitoringu osoby.

Jedną z przyczyn dla których klasyfikator oparty jedynie o cechy obrazów nie uzyskuje wyższych wskaźników jakości detekcji upadku jest to, że cechy, które są powszechnie wykorzystywane w systemach do detekcji upadku charakteryzują się dużą wariancją wewnątrzklasową przy małej wariancji międzyklasowej. Jak można zaobserwować na poniższym rysunku, na którym zilustrowano grupowanie się cech dla obrazów wykorzystywanych w badaniach nad klasyfikacją pozy, cechy zaproponowane w niniejszej pracy mają znaczącą siłę dyskryminującą.



1.1.3.8. Rozwiązanie techniczne

W poniższej tabeli przedstawiono zestawienie wybranych systemów detekcji upadku z podziałem na trzy grupy, którym literatura naukowa poświęca najwięcej uwagi: inercyjne czujniki ruchu, systemy wizyjne oraz systemy wizyjne oparte o kamery głębi. Ze względu na obszerność literatury przedmiotu w tabeli ujęto jedynie kilka prac dla każdej z grup, a bardziej szczegółowy przegląd można znaleźć w pracach (Igual et al., 2013; Webster & Celik, 2014). Celem umożliwienia analizy porównawczej wyników, przy wyborze prac przyjęto pewne kryteria. W omawianym zestawieniu zaprezentowano tylko te prace, które opisują jaki był scenariusz wykonywanych eksperymentów: liczba osób, liczba wykonanych akcji (z podziałem na upadki i czynności życia codziennego). W zastawieniu pominięto wyniki, dla których liczba eksperymentów lub osób wykonujących akcje była mała (1 - 3 osoby). W grupie systemów opartych o inercyjne czujniki ruchu przedstawiono rezultaty uzyskane w pracy porównawczej (Bagalà et al., 2012), co pozwala na rzetelną ocenę skuteczności metod detekcji upadku - eksperymenty odbywały się w oparciu o te same dane, uzyskane w środowisku niekontrolowanym, poza laboratorium. W grupie systemów wizyjnych, ze względu na brak odpowiednich eksperymentów porównawczych, przedstawiono wyniki uzyskane przez poszczególne zespoły badawcze.

Autor	Rok	Użyte cechy	Liczba osób	Wyniki
Detekcja upadku w oparciu o inercyjne czujniki ruchu				
Chen et al.	2005	wartość przyspieszenia, zmiana orientacji		*accuracy:* 93,7% *sensitivity:* 76,0% *specificity:* 94,0%
(Bourke et al.)	2007	wartość przyspieszenia	15 osób 32 upadki 1170 ADLs wiek: 66.4 ± 6.2 lat	*accuracy:* 21,3% *sensitivity:* 100% *specificity:* 19,0%
(Kangas et al.)	2009	wartość przyspieszenia, prędkość, położenie ciała człowieka po upadku		*accuracy:* 96,7% *sensitivity:* 28,0% *specificity:* 98,0%
(Bourke et al.)	2010	wartość przyspieszenia, prędkość, położenie ciała człowieka po upadku		*accuracy:* 96,3% *sensitivity:* 83,0% *specificity:* 97,0%

Detekcja upadku w oparciu o kamery RGB				
Lee & Mihailidis	2005	cechy geometryczne sylwetki człowieka (obwód i oś główna), prędkość środka ciężkości sylwetki	21 osób 126 upadków 189 ADLs wiek: 20 - 40 lat	*sensitivity*: 93,9% *specificity*: 80,5%
Miaou et al.	2006	cechy geometryczne sylwetki (wysokość, szerokość)	20 osób 33 upadki 27 ADLs wiek:brak danych	*accuracy*: 81,0% *sensitivity*: 90,0% *specificity*: 86,0%
Liu et al.	2010	szerokość i wysokość prostokątnego obrysu sylwetki	15 osób 45 upadków 45 ADLs wiek: 24 - 60 lat	*accuracy*: 84,4%
Shoaib et al.	2011	odległości między elementami sylwetki (głowa, nogi), kontekst sceny	5 osób 43 upadki 30 ADLs wiek: brak danych	*sensitivity*: 97,7% *specificity*: 90,0%
Detekcja upadku w oparciu o kamery głębi (w nawiasach podano dokładność metody według pracy (Stone & Skubic, 2014))				
Zhang et al.	2012	wysokość postaci w 3D	5 osób 200 sekwencji (upadki + ADLs) wiek: brak danych	*accuracy*: 94,0%
Mastorakis & Makris	2012	obrys sylwetki w 3D	8 osób 48 upadków 136 ADLs wiek: brak danych	*accuracy*: 100% (*sensitivity*: < 80%)
Planinc & Kampel	2012	orientacja ciała w 3D na podstawie modelu szkieletu postaci	4 osoby 40 upadków 32 ADLs wiek: brak danych	*accuracy*: 95,8% (*sensitivity*: < 60%)
Stone & Skubic	2014	orientacja ciała w 3D oraz orientacja ciała	16 osób 454 upadki wiek: brak danych	*sensitivity*: z pozycji: stojącej: 98,0% siedzącej: 70,0%

Analizując rezultaty badań przedstawione w tabeli zauważyć można, że uzyskanie wysokiej czułości detekcji przy pomocy inercyjnych czujników ruchu wiąże się z wysokim odsetkiem

wyników fałszywie pozytywnych. Najprostsze algorytmy detekcji upadku, działające w oparciu o wartość przyspieszenia, poddane ciągłej, długotrwałej ewaluacji generują od 22 do 85 fałszywych alarmów na dobę (Bagalà et al., 2012), co jest wartością nie do zaakceptowania przez potencjalnych użytkowników tej technologii. Detekcja upadku w oparciu o obraz RGB charakteryzuje się mniejszą skutecznością niż w przypadku kamer głębi. Jednak opracowane dotychczas algorytmy nie są pozbawione wad i ich skuteczność maleje dla trudniejszych scenariuszy. Przykładowo w pracy (Stone & Skubic, 2014) uzyskano wysoką czułość dla upadków z pozycji stojącej, jednak detekcja zdarzeń rozpoczynających się z pozycji siedzącej okazała się być problematyczna.

2 Biblioteki

2.1 Serverowe (API)

(http://blog.mashape.com/list-of-14-image-recognition-apis/)

2.2 Lokalne (device side)

(https://www.quora.com/What-are-the-best-face-detection-APIs)

2.2.1 Komercyjne

2.2.1.1 Luxand FaceSDK

Biblioteka dostarczająca API do wyodrębniania szablonów twarzy i dopasowywania ich.

FaceSDK jest wysokowydajnym, wieloplatformowym rozwiązaniem do rozpoznawania twarzy. Kompatybilna z 32 i 64-bitowymi środowiskami desktopowymi oraz platformami mobilnymi (iOS, Android). Biblioteka wykrywa całą twarz oraz indywidualne cechy twarzy. Rozpoznaje twarz ze statycznego obrazu lub z video. Posiada innowacyjny system samo uczenia.

Bibliotekę można wykorzystać do kontroli dostępu opierającej się wyłącznie o osobiste właściwości biometryczne (cech konkretnej twarzy) z pominięciem innych metod uwierzytelniania (np. PIN).

Zaletą biblioteki jest to, że automatycznie wykrywa twarze pojawiające się na video. rejestrując ich kompletną informację biometryczną uchwyconą z różnych ujęć wraz z wyrazami twarzy. Oznaczanie osoby wymaga tylko nadania jej identyfikatora. Po takiej operacji system automatycznie zidentyfikuje osobę w przeszłych, bieżących i przyszłych video.

W odróżnieniu od innych bibliotek nie proces uczenia nie jest wymagany, zbieranie zdjęć twarzy osoby, która ma być rozpoznana, z różnych kątów i w różnych ekspozycjach nie jest konieczne.

Biblioteka jest zaprojektowana aby równie wydajna w różnych warunkach oświetlenia. Działa równie dobrze przy świetle dziennym, świetlówkowym i żarówkowym. Przy testowaniu bazy FRGC, biblioteka prawidłowo zidentyfikowała osoby w 93,9% przy fałszywych wynikach pozytywnych 0,1%.

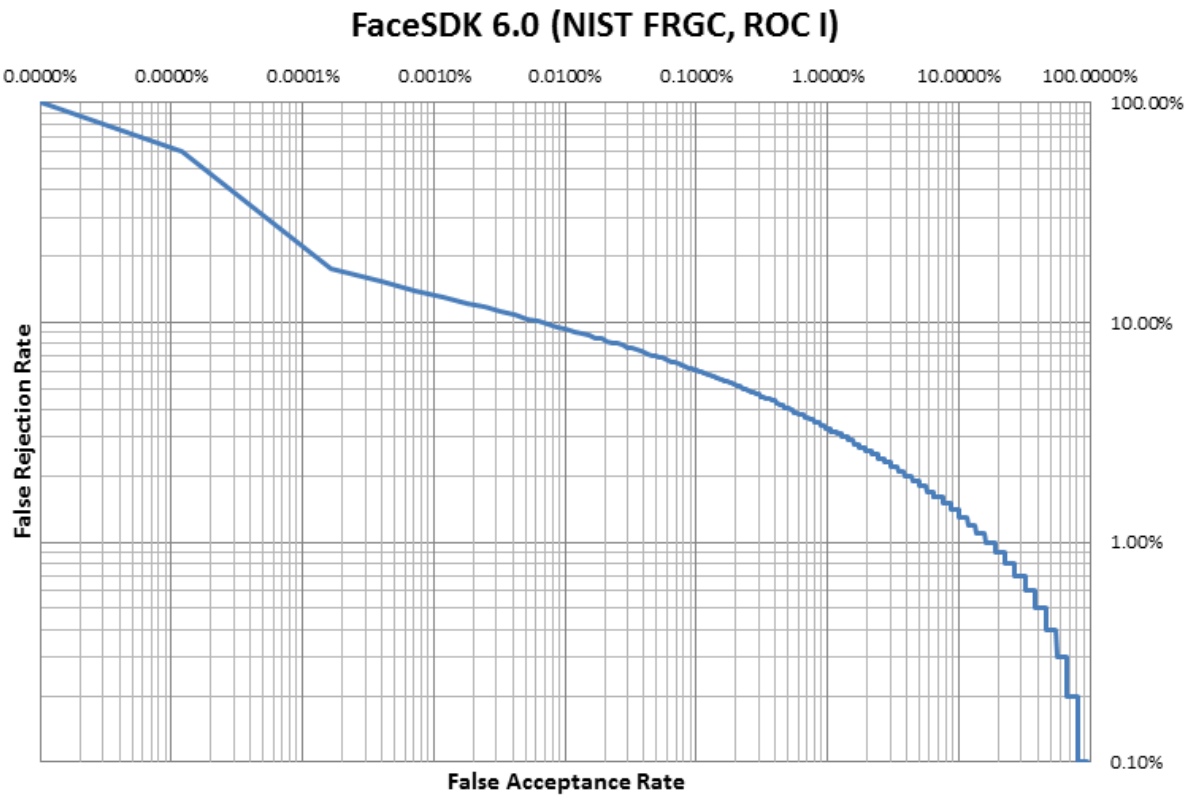
Specyfikacja:

Detekcja twarzy:

- skuteczny algorytm wykrywania twarzy en face
- wykrywanie wielu twarzy na jednym obrazie
- wykrywanie twarzy przy pochyle od-30 do +30° oraz przy obrocie od-30 do +30°
- szybkość „werbowania”:
    - kamera internetowa, przy pochyle ±15°: 0,003456 s (289FPS) (INTEL Core i7 4850HQ), 0,006420 s (156 FPS) (iOS), 0,020 sy (Android)
    - kamera IP, przy pochyle ±15°: 0,07571 s (Intel), 0,22235 s (iOS), 0,642 s (Android)
- po wykryciu zwracane są współrzędne x,y środka wykrytej twarzy, jej szerokość i kąt pochyłu
- łatwa konfiguracja

Rozpoznawanie/porównywanie twarzy:

- porównywanie dwóch twarzy z podanymi FAR (False Acceptance Rate) i FRR (False Rejection Rate)
- szybkość rozpoznawania:
    - kamera internetowa, przy użyciu FSDK_DetectEyes/FSDK_GetFaceTemplateUsingEyes: 0,01455 s (69 FPS) (Intel), 0,01311 s (76 FPS) (Intel Xeon), 0,03398 s (29 FPS)(iOS), 0,091 s (11 FPS) (Android)
    - kamera internetowa, przy użyciu FSDK_GetFaceTemplateInRegion (wyższa skuteczność): 0,014603 s (68 FPS) (Intel), 0,013035 s (76 FPS) (Intel Xeon), 0,033867 s (29 FPS) (iOS), 0,097 s (10 FPS) (Android)
- rozmiar szablonu 13 kb
- szybkość wykrywania na sekunde:
    - szablon z pojedyncza cech: 77073 (Intel), 96899 (Intel Xeon), 80671 (iOS), 12080 (Android)
    - szablon z wieloma cechami: 311526 (Intel), 442635 (Intel Xeon), 15440 (iOS), 45353 (Android)
- zwracana jest informacja o poziomie podobieństwa twarzy
- wykres ROC



2.2.1.1.1 Wymagania

Biblioteka wspiera następujące platformy:

- Windows 2000/XP/2003/Vista/2008/2012, Windows 7, Windows 8, Windows 10
- Linux (RHEL 5+, CentOS 5+ and other)
- Mac OS X 10.5+ x86_64
- iOS 5.0+, armv7/x86 (iPhone 3GS+, iPad 1+, simulator)
- iOS 7.0+, arm64/x86_64(iPhone 5S+, iPad Air+, iPad mini retina+, simulator)
- Android 4.0+ (platform version 14+), armv7 (armeabi-v7a)/x86

Aby uzyskać większą wydajność zalecany jest procesor Intela.

Minimalne wymagania systemowe:

- procesor 1GHz
- 256 MB RAM

Zalecane wymagania systemowe:

- procesor Intel Core i7 lub Xeon
- 2 GB RAM
- kamera internetowa kompatybilna z DirectShow
- kamera IP z interfejsem MJPEG (np. kamery IP AXIS)

Należy zwrócić uwagę, że funkcje kamery internetowej są dostępne tylko dla platform z Windowsem. Kamery IP są dostępne na wszystkich platformach.

2.2.1.1.2 Język/Środowisko

API wspiera następujące środowiska: Microsoft Visual C++, C#, Objective C, VB.NET, VB6, Java, Delphi, C++Builder, Xcode (iOS), Eclipse ADT (Android) i Android Studio (Android).

Każda metoda jest opisana i użyta w przykładzie, który można skopiować do swojego projektu.

2.2.1.1.3 Dostępne implementacje algorytmów

Brak danych.

2.2.1.1.4 Użycie w komercyjnych produktach

W wielu, brak danych jakie.

2.2.1.1.5 Dostępne algorytmy do preprocessingu

Brak danych.

2.2.2 Darmowe

2.2.2.1 OpenCV

Biblioteka funkcji wykorzystywanych podczas obróbki obrazu, oparta na otwartym kodzie, zapoczątkowana przez Intela. Autorzy jej skupiają się na przetwarzaniu obrazu w czasie rzeczywistym.

OpenCV składa się z pięciu podstawowych komponentów:

CV oraz CVaux – komponenty zawierające funkcję transformacji, filtracji oraz konwersji przestrzeni obrazów, funkcję analizy obrazów takie jak selekcja, operacje morfologiczne, detekcję krawędzi oraz obsługę histogramów, detekcję obiektów, kalibrację kamery, rekonstrukcję sceny 3D i inne,

MLL – Machine Learning Library, jak sama nazwa wskazuje zawiera funkcje tworzenia klasyfikatorów bazujących na statystyce odgrywających znaczącą rolę w uczeniu maszyn sposobu detekcji,

HighGUI – zawiera metody akwizycji i zwalniania obrazów, sekwencji wideo, narzędzia tworzenia interfejsu okienkowego, suwaków, obsługi myszy etc.

CxCore – Podstawowy komponent biblioteki zawiera operacje na tablicach, algebrę macierzy, funkcje matematyczne, transformacje Fouriera, wsparcie dla plików XML, narzędzia rysowania obiektów 2D i inne.

2.2.2.1.1 Wymagania

Aby stworzyć niektóre fragmenty oprogramowania dotyczące zagadnień wejścia sygnału (z kamer) konieczne jest zainstalowanie bibliotek SDK DirectShow. Biblioteki te można znaleźć w podkatalogu Samples\Multimedia\DirectShow\BaseClasses Platformy SDK Microsoftu; po ściągnięciu należy je odpowiednio zainstalować, aby można było tworzyć oprogramowanie w OpenCV. Ponadto w celu korzystania z mechanizmów wielowątkowych konieczna jest instalacja biblioteki TBB Intela.

Wadą są dość duże wymagania sprzętowe - do płynnego działania bibliotek najczęściej będzie nam potrzebny komputer PC.

2.2.2.1.2 Język/Środowisko

Biblioteka ta jest wieloplatformowa, można z niej korzystać w Mac OS X, Windows jak i Linux.

Biblioteka została stworzona w języku C, lecz istnieją nakładki umożliwiające korzystanie z niej również w językach C++, C#, Python, Java (np. Emgu CV, JavaCV).

2.2.2.1.3 Licencja

OpenCV jest bezpłatną, open-source'ową biblioteką do użytku komercyjnego i edukacyjnego, bez konieczności udostępniania swoich projektów opartych o jej implementację.

2.2.2.1.4 Dostępne implementacje algorytmów

W bibliotece dostępne są algorytmy:

- Eigenfaces
- Fisherfaces
- LBP

Opis użycia algorytmów można znaleźć pod tym linkiem
http://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutorial.html

Literatura

1. N. Dalal, B. Triggs, „Histograms of oriented gradients for human detection”, Proceedings of IEEE Conference Computer Vision and Pattern Recognition, San Diego, USA, 2005.
2. 2. P. Harrington, “Machine Learning in Action”, Manning Publications, New York, 2012.
3. http://rab.ict.pwr.wroc.pl/~mw/Stud/Dypl/morynicz/inz.pdf
4. Tomasz MARCINIAK, Adam DĄBROWSKI „Porównanie i ocena skuteczności detekcji i rozpoznawania twarzy w sekwencjach wideo „
5. Łukasz Szpak „Klasyfikacja obrazów bazująca na autorskiej reprezentacji hipsometrycznej„
6. Jan MATUSZEWSKI „System inteligentny rozpoznawania znaków drogowych”
7. Michał KOWALCZYK „SYSTEM DETEKCJI I ROZPOZNAWANIA TWARZY”
8. Aleksandra Ucińska „Rozpoznawanie twarzy - aplikacja w architekturze Klienta Natywnego”
9. mgr inż. Michał Kępski „DETEKCJA UPADKU I WYBRANYCH AKCJI NA SEKWENCJACH OBRAZÓW CYFROWYCH”
10. MARCIN KMIEĆ „WYKRYWANIE NIEBEZPIECZNYCH PRZEDMIOTÓW W AUTOMATYCZNIE ANALIZOWANYCH SEKWENCJACH WIDEO”