POLITECHNIKA WROCŁAWSKA
WYDZIAŁ ELEKTRONIKI

KIERUNEK: Automatyka i Robotyka (AiR)
SPECJALNOŚĆ: Robotyka (ARR)

PRACA DYPLOMOWA MAGISTERSKA

Wykrywanie i parametryzacja ruchu obiektów na podstawie sekwencji obrazów

Detection and parametrization of objects' motion on the basis of movie sequence

AUTOR:
Krzysztof Charusta

PROWADZĄCY PRACĘ:
dr inż. Marek Wnuk, I-6

OPIEKUN:
dr inż. Marek Wnuk, I-6

OCENA PRACY:

WROCŁAW 2007

Pracę dedykuję rodzicom.

Rozdział 1

Wstęp

Robotyka jest dziedziną nauki interdyscyplinarną, korzystającą z wielu dziedzin wiedzy począwszy od mechaniki, poprzez elektronikę i informatykę na sztucznej inteligencji kończąc. Gdzieś pomiędzy znajduje się dziedzina przetwarzania obrazów, której rozwój jest coraz większy. Wraz z upowszechnieniem urządzeń cyfrowej akwizycji obrazu i zwiększaniem mocy obliczeniowej, co ogólnie można nazwać mianem postępu technicznego, również robotycy, a może przede wszystkim robotycy, sięgają po systemy cyfrowej analizy obrazu. Można bez problemu wymienić szereg projektów robotycznych wykorzystujących systemy wizyjne, nierzadko autonomiczne. Dość wspomnieć, że lądownik Mars Pathfinder wylądował na czerwonej planecie z pomocą systemu optycznej korekcji trajektorii lądowania.

Analiza obrazu stwarza możliwość nauczenia maszyny postrzegania otoczenia w sposób podobny do ludzkiego. Obok statycznych własności obrazu jak jasność czy głębia wyróżniamy również dynamiczne, znacznie bardziej istotne, bo wzbogacające wiedzę systemu wizyjnego o zmiany zachodzące w otoczeniu. Należy pamiętać, że obraz niesie ze sobą ilość informacji zbyt dużą do całościowej analizy. Z tego względu właściwa, selektywna i skuteczna analiza informacji jest istotna. Wykrywanie ruchu jest właśnie taką analizą.

1.1 Cele i założenia

Temat niniejszej pracy wychodzi naprzeciw zagadnieniom współczesnej robotyki, jego opracowanie ma na celu przybliżenie możliwości wykrywania ruchu w zastosowaniach robotycznych. Głównym celem pracy było zapoznanie się z metodami wykrywania ruchu obiektów przez systemy wizyjne pod kątem zastosowań robotycznych. Inspiracją do pracy był projekt autonomicznego systemu wizyjnego pracującego pod zarządzaniem mikrokontrolera. Niezbędne więc było poszukiwanie metody efektywnej obliczeniowo oraz łatwej w implementacji na mikrokontrolery. Temat pracy zakłada przetwarzanie przez system wizyjny obrazu w celu wykrywania poruszających się obiektów i parametryzację tego ruchu, jednak w trakcie zgłębiania wiedzy o zagadnieniu wykrystalizowały się bardziej spójne cele i założenia dotyczące systemu wizyjnego:

1. system wizyjny będzie oparty na kamerze internetowej (z uwagi na niską cenę i dostępność),

2. system wizyjny ma być odporny na niewielką liczbę klatek i zaszumiony obraz,

3. system wizyjny będzie umożliwiał wykrywanie i parametryzację ruchu obserwowanych obiektów,

4. stworzony system wizyjny będzie umożliwiał badanie zaproponowanych metod i testowanie ich przydatności na rzeczywistych sekwencjach,
5. obserwowany obiekt będzie robotem mobilnym,
6. wybrana metoda estymacji ruchu będzie elastyczna, umożliwiająca pracę w niskiej rozdzielczości, z zaszumionym obrazem,
7. wybrana metoda będzie efektywna obliczeniowo,
8. nie jest potrzebne wykrywanie ruchu subpikselowego.

1.2 Zawartość pracy

Praca poświęcona jest wykrywaniu ruchu w sekwencjach obrazów. W kolejnych rozdziałach zostały przedstawione ogólne podejście do zagadnienia oraz szczegółowa analiza obejmująca metody realizujące przedstawione wyżej cele i założenia.

Rozdział 2 zawiera przegląd metod wykrywania ruchu obiektów. Na początku (2.2) przedstawiono istotne problemy związane z przetwarzaniem obrazu w celu wykrywania ruchu, następnie zostały przedstawione i scharakteryzowane metody gradientowe (2.3.1), częstotliwościowe (2.3.2), korelacyjne (2.3.3) i oparte na obrazie historii ruchu (2.3.4). Następnie, powyższe grupy metod zostały porównane pod kątem przedstawionych celów - rozdział 2.4, a wybrana metoda dokładnie opisana (2.5, 2.6). Kolejny rozdział (3) to opis stworzonego systemu wizyjnego i implementacji wybranych metod wykrywania ruchu. Przedostatni rozdział to przeprowadzone badania i testy (4). W ostatnim rozdziale (5) podsumowano zrealizowany projekt wraz z najistotniejszymi wnioskami. Dodatek A zawiera pomoc użytkownika dla stworzonych programów.

Rozdział 2

Podstawy teoretyczne

2.1 Pole ruchu a przepływ optyczny

Wykrywanie ruchu obiektów i jego parametryzacja na podstawie sekwencji obrazu jest procesem wyznaczania wektora ruchu pomiędzy dwoma obrazami z sekwencji, występującymi po sobie. Przetwarzanie obrazu zyskało na popularności w ostatnich czasach, ponieważ powszechny stał się obraz cyfrowy. Ze względu na rosnącą z każdym rokiem moc obliczeniową komputerów, przetwarzanie obrazu staje się coraz łatwiejsze. Praktycznie każdy dysponujący domowym komputerem i kamerą internetową może próbować swoich sił w cyfrowym przetwarzaniu obrazu. Poniżej zostaną przedstawione podstawowe pojęcia dotyczące zagadnienia.

Pole ruchu(*ang. motion field*) jest pojęciem czysto geometrycznym. Definiowane jest jako zrzutowanie przestrzeni rzeczywistych trójwymiarowych wektorów na płaszczyznę obrazu. Operacja ta daje pogląd o składowej wektora ruchu obiektów równoległej do płaszczyzny obrazu. W związku z tym odtworzenie pierwotnej informacji – oryginalnych wektorów, jest niemożliwe.

Przepływ optyczny(*ang. optical flow*) jest również polem wektorowym, zawierającym informacje pozwalające przekształcić obraz pierwszy w obraz drugi, zgodnie z kierunkiem przepływu. Przepływ optyczny nie jest ściśle zdeterminowany, więc istnieje wiele pól przepływu pozwalających uzyskać obraz drugi z obrazu pierwszego.

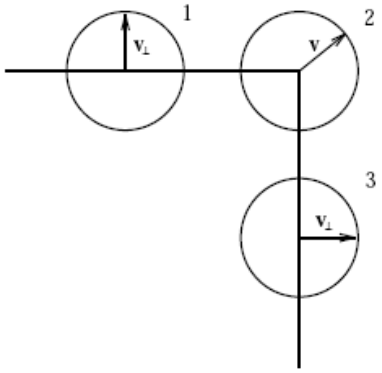
2.2 Problemy estymacji przepływu optycznego

W poniższej części zostaną przedstawione najważniejsze problemy z jakimi należy się zmierzyć podczas prób wyznaczania przepływu optycznego. Wszystkie pociągają za sobą pewne założenia idealizujące obraz obserwowany. Aby spełnić wszystkie założenia należałoby obserwować sztuczny obraz, który trudny jest do osiągnięcia w rzeczywistości.

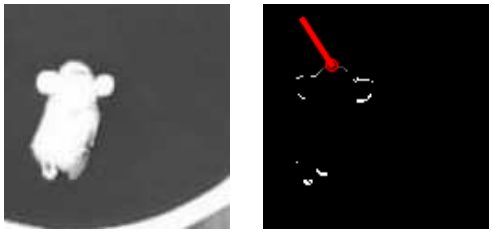
- Znalezienie wartości prędkości optycznej w najlepszym przypadku odzwierciedla jedynie składową ruchu obiektu prostopadłą do płaszczyzny obrazu. Poza tym, znalezienie wektora przepływu nie gwarantuje poprawnego wykrycia wektora ruchu. Może się zdarzyć, że w wyniku ruchu obrotowego, nagłych zmian oświetlenia sceny lub innych nieprzewidzianych zjawisk wektor zostanie wskazany nieprawidłowo.

- Problem szczelinowy(*ang. aperture problem*). Zjawisko występuje, gdy lokalne zmiany jasności obrazu mają charakter jednowymiarowy, wtedy wyznaczenie pełnego wektora prędkości jest niemożliwe [18]. Doskonale to obrazuje rysunek 2.1, gdzie w szczelinach 1 i 3 możemy zaobserwować prędkości optyczne jedynie w kierunkach normalnych do

krawędzi kwadratu. Przez szczelinę 2 umieszczoną w narożniku możemy zmierzyć pełną prędkość.



Rysunek 2.1: Problem szczelinowy.



a) b)

Rysunek 2.2: Przykład wykrywania ruchu i problem szczelinowy dla obiektu jednolitego koloru(Metoda Horna i Shuncka [5]), a) klatka oryginalna, b) wykryty ruch w miejscu pojawiania się i zanikania obiektu (sekwencja *baran1.avi*).

- Brak teksturowania lub przeźroczystość obiektu może prowadzić do błędnego wykrycia ruchu lub tylko częściowego – na konturach, w miejscy pojawiania się i znikania obiektu na obrazie. W ekstremalnym wypadku możemy wyobrazić sobie obracającą się kulę o jednolitym kolorze.
- Zmienne oświetlenie może doprowadzić do wykrycia pozornego ruchu niezależnie od zastosowanej metody. Można w tym wypadku kompensować oświetlenie jeśli kamera jest statyczna lub posługiwać się kolorem obrazu w dziedzinie HSV, co poprawia odczyt.
- Specyficzne teksturowanie obiektu może być problematyczne. Jeśli wyobrazimy sobie obracający się walec z naniesionymi spiralnie pasami i oś obrotu będzie pionowa, to zwrot wektora prędkości pozornej (obserwowanej) będzie równoległy z osią pionową.
- Zachodzenie obiektów na siebie, powoduje znikanie i pojawianie się fragmentów obrazu lub niespodziewane zmiany jasności. W tym przypadku wskazanie wektorów prędkości może być kłopotliwe.

Jak widać wyznaczenie przepływu optycznego nie jest możliwe bez pewnym założeń i uproszczeń, stąd postulaty idealizujące:

- jednolite oświetlenie sceny i obiektu,

- obiekt o niejednolitym wzorze - teksturowany,
- obiekt nieprzeźroczysty i niezlewający się z tłem,
- idealne rozproszenie światła na powierzchni obiektu,
- płaszczyzna przemieszczenia obiektu równoległa do powierzchni obrazu.

2.3 Przegląd metod

Wykrywanie ruchu można realizować na wiele sposobów. Cechą wspólną wszystkich metod jest wykrywanie pola przepływu optycznego. Generalnie metody można podzielić na gradientowe, częstotliwościowe, korelacyjne. Choć różnią się one podejściem, można wydzielić podstawowe etapy estymacji:

1. filtrowanie oraz wygładzanie wstępne filtrami dolno- i pasmowo-przepustowymi w celu wyodrębnienia interesującego sygnału – zwiększenia odstępu sygnału od szumu SNR(*ang. Signal to Noise Ratio*),
2. uzyskanie podstawowych cech obrazu, t.j. pochodne lub powierzchnie korelacji,
3. integracji uzyskanych cech i wartości w celu stworzenia dwuwymiarowego obrazu.

W kolejnych podrozdziałach zostaną przedstawione i omówione pokrótce wspomniane grupy metod.

2.3.1 Metody gradientowe

Bodaj najstarszą i wciąż bardzo popularną i rozwijaną metodą estymacji optycznego przepływu jest metoda gradientowa, w której przepływ optyczny wyznaczany jest w oparciu o pochodne przestrzenne i czasowe obrazu. Konieczne jest założenie ciągłość dziedziny, co pozwala na różniczkowanie. Metody z tej grupy dzielone są ze względu na rząd wykorzystywanych pochodnych, a najczęściej spotykane są metody pierwszego rzędu.

Metody pierwszego rzędu opierają się na założeniu niezmienności jasności punktu zrzutowanego na płaszczyznę obrazu, co opisuje wzór:

$$\frac{dI(\overrightarrow{x},t)}{dt} = 0, \tag{2.1}$$

gdzie $I(\overrightarrow{x},t)$ jest funkcją intensywności obrazu w punkcie $\overrightarrow{x}$ i w chwili t. Powyższe założenie wynika bezpośrednio z definicji przepływu optycznego danej wzorem:

$$I(\overrightarrow{x},t) \approx I\left(\overrightarrow{x} + \overrightarrow{\delta x}, t + \delta t\right), \tag{2.2}$$

Spełnienie warunku (2.1) jest prawdziwe w przypadku jednolitego oświetlenia i niewielkiej wartości przemieszczenia. Z rozwinięcia Taylora prawej strony równania (2.2) otrzymujemy:

$$I\left(\overrightarrow{x} + \overrightarrow{\delta x}, t + \delta t\right) = I(\overrightarrow{x},t) + (\nabla I)^T \overrightarrow{\delta x} + \delta t I_t + O^2, \tag{2.3}$$

gdzie $\nabla I = (I_x, I_y)^T$ i I_t to pochodna rzędu pierwszego funkcji intensywności $I(\overrightarrow{x},t)$ względem t, pochodne wyższych rzędów są zawarte w O^2. Ostatecznie, po uwzględnieniu (2.2) i odjęciu

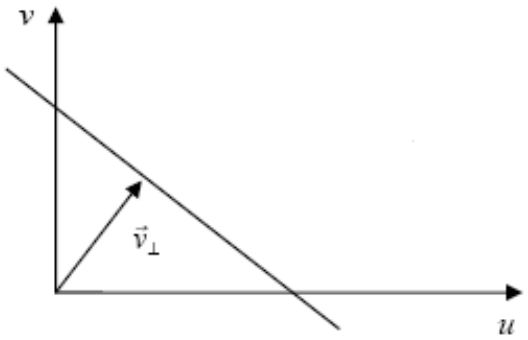
od obu stron równania wyrazu $I(\overrightarrow{x},t)$, pominięciu O^2 i podzieleniu przez δt otrzymujemy równanie ograniczenia gradientu, zwane również równaniem przepływu optycznego:

$$(\nabla I)^T \overrightarrow{v} + I_t = 0, \tag{2.4}$$

gdzie $\overrightarrow{v} = (u,v)^T$ jest prędkością przepływu optycznego, a wyrażenie $\nabla I = (I_x(x,t), I_y(x,t))^T$ nazywane jest planarnym gradientem intensywności. Równanie (2.4) nie jest wystarczające do określenia obu składowych prędkości $\overrightarrow{v}$, ponieważ niewiadome są dwie składowe, a dysponujemy pojedynczym równaniem liniowy. W rzeczywistości równanie (2.2) definiuje jedynie prostą ograniczającą w przestrzeni prędkości optycznej (rys. 2.3). Prostą otrzymujemy licząc wektor normalny prędkości optycznej:

$$\overrightarrow{v}_\perp = \frac{-I_t \nabla I}{||\nabla I||_2^2} \tag{2.5}$$

Wyznaczenie pełnej prędkości wymaga przyjęcia dalszych założeń.



Rysunek 2.3: Prosta ograniczająca prędkość optyczną.

W przypadku metod gradientowych rzędu drugiego jako ograniczenie prędkości optycznej wykorzystywana jest pochodna rzędu drugiego – Hesjan funkcji intensywności $I(\overrightarrow{x},t)$:

$$\begin{bmatrix} I_{xx}(\overrightarrow{x},t) & I_{yx}(\overrightarrow{x},t) \\ I_{xy}(\overrightarrow{x},t) & I_{yy}(\overrightarrow{x},t) \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{bmatrix} I_{tx}(\overrightarrow{x},t) \\ I_{ty}(\overrightarrow{x},t) \end{bmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{2.6}$$

Powyższe równanie wynika z założenia bazowego dla metod drugiego rzędu, że gradient intensywności, dany wzorem (2.7) jest stały.

$$\frac{d\nabla I(\overrightarrow{x},t)}{dt} = 0 \tag{2.7}$$

Ograniczenie (2.7) jest silniejsze niż ograniczenie (2.4) co prowadzi do zmniejszenia zakresu zastosowań tych metod zwłaszcza dla ruchu obrotowego czy skalowania [11].

Idealny obraz wejściowy dla metod gradientowych posiada następujące cechy:

- liniowy charakter zmian intensywności,
- prędkość optyczna nie przekraczająca jednego punktu na ramkę.

Dwie najbardziej znane metody gradientowe to metoda Horna i Schuncka(HS) [5] oraz Lucasa i Kanade (LK) [6], są one reprezentantami odpowiednio podejścia globalnego i lokalnego w metodach gradientowych.

Metoda Horna i Schuncka

Metody globalne dodają do ograniczenia (2.4) dodatkowe ograniczenie, zazwyczaj na gładkość. Metoda HS jako pierwsza wykorzystywała ograniczenie z góry szybkości zmian prędkości przepływu optycznego. W połączeniu z ograniczeniem (2.4) dawała funkcjonał:

$$\int_D \left((\nabla I)^T \overrightarrow{v} + I_t\right)^2 + \lambda^2 \left(||\nabla u||_2^2 + ||\nabla v||_2^2\right) d\overrightarrow{x}, \tag{2.8}$$

który minimalizowany na przestrzeni dziedziny D, pozwalał odnaleźć prędkości dla poszczególnych punktów [18]. We wzorze (2.8) λ jest współczynnikiem wagowym członu ograniczającego gładkość.

Metoda Lucasa i Kanade

W metodzie lokalnej opracowanej przez Lucasa i Kanade [6] do minimalizacji, dającej szukaną wartość prędkości optycznej, wykorzystuje się prędkości normalne z sąsiedztwa, co daje poszukiwaną wartość prędkości. W metodach tych zakłada się, że punkty sąsiednie poruszają się w tym samym kierunku. W przypadku LK funkcjonał ma postać:

$$\sum_{\overrightarrow{x} \in R} W^2(\overrightarrow{x}) \left[(\nabla I(\overrightarrow{x},t))^T \overrightarrow{v} + I_t(\overrightarrow{x},t)\right]^2, \tag{2.9}$$

gdzie $W(\overrightarrow{x})$ oznacza *funkcję okna* zmniejszającą znaczenie dalszego sąsiedztwa.

Metody gradientowe nie będą dalej wykorzystywane, dalsze szczegóły można znaleźć w [6, 5, 11, 18].

2.3.2 Metody częstotliwościowe

Kolejną grupą algorytmów są metody wyznaczania przepływu optycznego w oparciu o filtry częstotliwościowe, które są czułe na kierunek ruchu. Jedną z ważniejszych cech tych metod jest mechanizm wykrywania ruchu operujący na czasowo-przestrzennym rozkładzie energii w przestrzeni fourierowskiej, który daje możliwość wykrywania ruchu na obrazie, niewykrywalnego przez zwykłe dopasowanie [18].

Dla przykładu, losowy układ punktów może być trudny do wychwycenia metodą gradientową lub korelacyjną, natomiast w dziedzinie transformaty Fouriera, wyjściowy parametr – ukierunkowana energia, może być łatwo wyznaczona w celu określenia kierunku.

Transformata Fouriera przesuwającego się dwuwymiarowego sygnału intensywności obrazu (2.2) ma postać:

$$\hat{a}\left(\overrightarrow{k},\omega\right) = \hat{I}_0\left(\overrightarrow{k}\right)\delta\left(\overrightarrow{v}^T\overrightarrow{k} + \omega\right), \tag{2.10}$$

gdzie $\hat{I}_0(\overrightarrow{k})$ to transformata Fouriera intensywności obrazu $I(\overrightarrow{x},0)$. δ oznacza deltę Diraca, $\overrightarrow{k} = (k_x,k_y)^T$ częstość przestrzenną i ω to częstotliwość czasowa.
Prowadzi to do równania ograniczenia przepływu optycznego w dziedzinie częstotliwościowej:

$$\overrightarrow{v}^T\overrightarrow{k} + \omega = 0. \tag{2.11}$$

Widzimy, że prędkość przesuwającego się obiektu w przestrzeni dwuwymiarowej jest funkcją jej czasowo-przestrzennej częstotliwości i tworzy płaszczyznę w obrazie źródłowym dla przestrzeni fourierowskiej [18].

Metody częstotliwościowe można generalnie podzielić na bazujące na filtrach częstotliwościowych – analizie rozkładu energii w przestrzeni fourierowskiej (metoda Adelsona i Bergena), oraz metody bazujące na analizie fazy sygnału w tej przestrzeni (metoda Fleeta i Jepsona).

Metody bazujące na filtrach częstotliwościowych

Adelson i Bergen zaproponowali grupę algorytmów wykorzystujących fakt, że rozpoznawanie ruchu na obrazie jest równoznaczne z wykorzystaniem czasowo-przestrzennej orientacji. Aby to osiągnąć często wykorzystywany jest filtr Gabora, umożliwiający ekstrakcję czasowo-przestrzennej energii w przestrzeni fourierowskiej. Filtr Gabora jest funkcją Gaussa pomnożoną przez funkcję sinusoidalną, więcej szczegółów można znaleźć w [18].

Techniki bazujące na filtrach częstotliwościowych są też często przedstawiane jako model ludzkiego sposobu postrzegania ruchu [18].

Metody bazujące na filtrach fazowych

Drugą wspomnianą podgrupą metod częstotliwościowych są metody bazujące na filtrach fazowych, nazwanych tak, ponieważ prędkość definiuje się w kategoriach zmian fazy sygnału. Są to sposoby opracowane przez Fleeta i Jepsona oparte na pasmowo-przepustowych, dostrojonych do prędkości, filtrach fazowych Gabora, przekształcających sygnał wejściowy zgodnie ze skalą, prędkością i orientacją. Metoda definiuje *prędkość składową* jako chwilową zmianę obwiedni fazy na wyjściu filtru [11]. Każdy filtr w wyniku daje wielkość zespoloną postaci:

$$R(\overrightarrow{x},t) = \rho(\overrightarrow{x},t)e^{i\phi(\overrightarrow{x},t)}, \tag{2.12}$$

gdzie $\rho(x,t)$ i $\phi(x,t)$ to amplituda i faza sygnału wyjściowego. Prędkość normalna obwiedni fazy sygnału jest dana wzorem:

$$\overrightarrow{v_\perp} = \frac{-\phi_t(\overrightarrow{x},t)\nabla\phi(\overrightarrow{x},t)}{||\nabla\phi(\overrightarrow{x},t)||_2^2}, \tag{2.13}$$

gdzie $\phi_t(\overrightarrow{x},t)$ jest pochodną po czasie z fazy, natomiast $\nabla\phi(\overrightarrow{x},t)$ jest gradientem przestrzennym. Gradient fazy można wyznaczyć z wzoru:

$$\nabla\phi(\overrightarrow{x},t) = \frac{Im[R^*(\overrightarrow{x},t)\nabla R(\overrightarrow{x},t)]}{|R(\overrightarrow{x},t)|^2}, \tag{2.14}$$

gdzie R^* jest zespolonym sprzężeniem $R(x,t)$,$\nabla R(x,t)$ jest gradientem $R(x,t)$, a Im oznacza część urojoną liczby zespolonej. Autorzy tej metody powiązali prędkość ze zmianą fazy ponieważ bardziej stabilna niż zmiana amplitudy kiedy małe odchylenia od kierunku translacji pojawiają się regularnie na trójwymiarowym obrazie.

Filtry fazowe są bardziej efektywne niż metody gradientowe (wspomniane w rozdziale 2.3.1) lub oparte na analizie rozkładu energii w przypadku zmiennego oświetlenia sceny. Intuicyjnie można to pojąć patrząc na sygnał $Acos(\omega t + \phi)$. Zmiana amplitudy zmieni wartość pochodnej lub energii w przestrzeni częstotliwości, natomiast faza pozostanie bez zmian. Metody oparte na filtrach fazowych są również bardziej czułe przy określeniu prędkości obiektów przeźroczystych lub przesłanianych.

2.3.3 Metody korelacyjne

W tej metodzie definiujemy przemieszczenie wyznaczane na podstawie najlepszego dopasowania regionów obrazów następujących po sobie w czasie. Dopasowanie najczęściej rozumiane jest jako maksymalizacja pewnej miary podobieństwa. W szczególności współczynnik korelacji pomiędzy dwiema funkcjami f i g można zdefiniować jako:

$$\int_D f\left(\overrightarrow{x} + \overrightarrow{\delta x}\right) g\left(\overrightarrow{x}\right) d\overrightarrow{x}. \tag{2.15}$$

Znalezienie δx, które maksymalizuje (2.15) pozwala znaleźć przemieszczenie pomiędzy f i g.

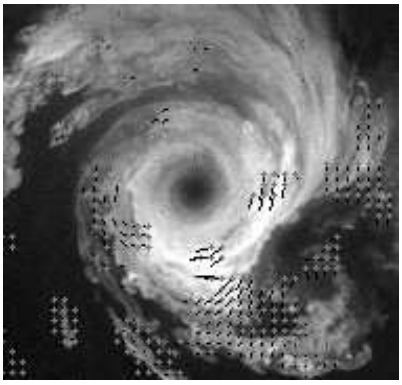
Kryteria korelacji mogą przybierać różne formy, t.j.: **korelacji bezpośredniej**, gdzie wartości odpowiednich punktów w obszarze okna są przez siebie mnożone, a następnie sumowane; **korelacji znormalizowanej średniej**, gdzie od wartości punktów odejmowana jest średnia wartość intensywności wyliczona dla danego okna, a następnie są one mnożone i sumowane; **sumy kwadratów różnic**, gdzie sumowane są kwadraty różnic intensywności odpowiednich pikseli.

Najczęściej, te metody są odpowiednie gdy obraz jest zaszumiony, dysponujemy ograniczoną ilością klatek lub występuje aliasing przestrzenny lub czasowy podczas akwizycji danych. W tych wypadkach wyliczenie z odpowiednią precyzją pochodnych przestrzennych i czasowych może być niemożliwe, co czyni bezużytecznymi zarówno metody gradientowe jak i częstotliwościowe [11].

Podobnie jak w przypadku wcześniejszych metod tak i tutaj możemy podzielić metody korelacyjne na grupy. Pierwsze bazują na korelacji fragmentów obrazu. Metody z drugiej grupy korelują pewne wybrane, szczególne cechy obrazu.

Korelacja fragmentów obrazu

W przypadku korelacji obrazu należy założyć, że sąsiednie punkty należą do tego samego obiektu, założenie to jest w przybliżeniu spełniane również w przypadku skalowania lub obrotu, jeśli rozpatrujemy części widocznego obiektu. Rozmiar i położenie obszaru poszukiwań wpływa na skuteczność, wydajność i zakres zastosowań danej metody. Jest to metoda bardzo wszechstronna.



Rysunek 2.4: Wykrywanie ruchu metodą dopasowania bloków przy ruchu obrotowym.

Najbardziej rozpowszechnioną metodą korelacyjną jest algorytm dopasowywania bloków BMA(*ang. Block Matching Algorithm*) szeroko spotykana w kompresji wideo.

Korelacja cech

Ciekawą odmianą jest korelacja cech charakterystycznych obrazu. Metoda ta bazuje na analizie obrazu pod kątem cech wyższego poziomu, a następnie opisaniu uzyskanych cech za pomocą struktur geometrycznych i teksturowania. Zakłada się, że struktury są łatwe do odnalezienia na obrazie referencyjnym nawet w przypadku ruchu obrotowego lub skalowania.

Z tego typu wykrywania korzysta kompresja wideo drugiej generacji, gdzie poprzez analizę systemu wizyjnego człowieka, wysnuto wnioski, że:

- informacja krawędziowa i konturowa jest istotna dla HVS*(ang. Human Vision System*) i percepcji,

- informacja o teksturze jest względnie ważna,
- obraz wielu naturalnych obiektów można przybliżać przez fraktale deterministyczne [17].

Pozwoliło to na zaproponowanie koncepcji kodowania ruchu kontur i tekstury. Trzeba zaznaczyć, że ta dziedzina jest młoda i wciąż rozwijana i w wypadku kiedy cecha jest skomplikowana to jej analiza może powodować duże obciążenie obliczeniowe. Przykład metody działającej na podobnej zasadzie można znaleźć w [16].

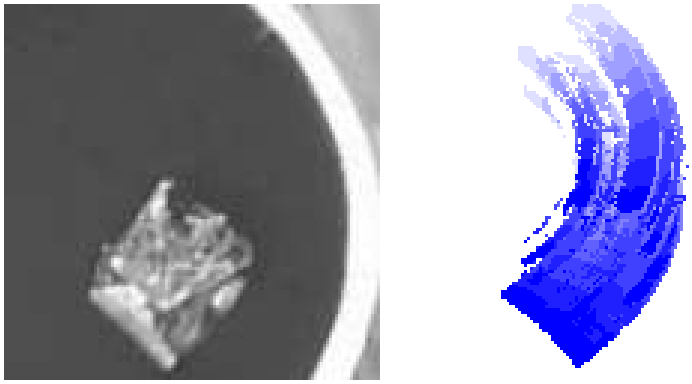
Metody korelacyjne we współczesnych zastosowaniach

W ostatnich latach, wraz ze wzrostem mocy obliczeniowej komputerów, sekwencje wideo na stałe zagościły na dyskach komputerów. Ekspansja internetu doprowadziła do do rozwoju metod kompresji wideo i opracowania coraz szybszych metod kodowania. Wyznaczanie przepływu optycznego jest obecne w kompresji MPEG -1/2/4 oraz H.261/263/264, gdzie obraz jest kodowany metodą *inter*, która bazuje na założeniu, że klatki występujące obok siebie w czasie są podobne, więc można zakodować jedynie część z nich, a resztę odtworzyć mając wektory przemieszczeń fragmentów obrazu, swoiste pole optycznego przepływu [17].

Kompensacja ruchu w kodowaniu jest bardzo czasochłonna i może zabierać nawet 41%, w przypadku kodowania H.264, potrzebnej mocy obliczeniowej [9], dlatego dużo uwagi poświęcono opracowaniu szybkich metod estymacji. Najczęściej stosowany jest algorytm dopasowywania bloków i jego szybkie sprzętowe implementacje.

2.3.4 Metoda budowania obrazu historii ruchu

Kolejną metodą estymacji ruchu jest budowanie obrazu MHI (*ang. Motion History Image*) - obrazu historii ruchu [7]. W tym podejściu poruszający się obiekt pozostawia, w każdej próbkowanej chwili czasowej, sylwetkę (lub różnicę sylwetek) rzutowaną na płaszczyznę obrazu, której wartości pikseli odpowiadają aktualnej chwili czasowej. Taka sylwetka uaktualnia obraz historii MHI. Kierunek przemieszczania określany jest na podstawie gradientu obrazu MHI.



a) b)

Rysunek 2.5: Tworzenie historii MHI dla sekwencji *ukazka2.avi*. a) Ostatnia klatka obrazu, b) obraz historii ruchu MHI.

Reprezentacja za pomocą obrazu MHI ma tą zaletę, że pewien zakres ramek obrazu zawierających ruch może być zawarty w pojedynczym obrazie, co jest istotne kiedy chcemy obserwować ruch wolnozmienny, np.: ruch ludzki, co więcej pamięć obrazu MHI można regulować [7]. Poza tym, metodę tą cechuje dość dużą szybkością działania w porównaniu z np.: metodami gradientowymi, co pokazano w rozdziale 2.4.1.

2.3.5 Podejście hierarchiczne

Należy jeszcze wspomnieć o technice wspomagającej wszystkie wymienione metody. Podejście hierarchiczne polega na dekompozycji obrazu, pozwala uzyskać kilka obrazów o różnych rozmiarach (rozdzielczościach), na których jest prowadzona estymacja ruchu. Dekompozycja realizowana jest przez interpolacje obrazu (resampling), które powoduje zmniejszenie jego rozmiarów, bez ryzyka powstania zniekształceń, które pojawiają się podczas zmniejszaniu rozdzielczości bez filtrowania. Dzięki tej metodzie możliwe jest wykrywanie nawet intensywnego ruchu, gdy proces jest kosztowny obliczeniowo.

2.4 Wybór metody

W części 2.3 niniejszej pracy zostały przedstawione metody estymacji przepływu optycznego. Poszczególne algorytmy wyznaczania ruchu były tworzone pod kątem różnych zastosowań, ich pomysłodawcy opierali się na różnych założeniach, tak aby optymalnie wykorzystać dostępną informację o ruchu widocznym na obrazie. Akceptacja tych założeń wiąże się z ograniczeniami, którym należy się podporządkować aby dana metoda działała poprawnie i dawała oczekiwane wyniki. Nie ma algorytmów wolnych od ograniczeń, a jedynie bardziej i mniej elastyczne.

Należałoby wybrać kryteria, pozwalające na wyszczególnienie odpowiedniej metody estymacji ruchu.

- **Wydajność** w najprostszej formie rozumiana jako czas obliczeń. Jednak rozsądne wydaje sie uwzględnienie takiego aspektu jak opóźnienie pomiędzy akwizycją danych i otrzymaniem rezultatów, występujące w przypadku metody historii ruchu. Czas jest szczególnie istotny w przypadku zastosowań w aplikacjach czasu rzeczywistego.

- **Jakość** uzyskiwanego pola wektorowego definiowana jako: jego spójność, potrzebna przy dalszej obróbce (np.: segmentacji), odzwierciedlenie rzeczywistych przemieszczeń.

- **Elastyczność** jest kolejną cechą, która daje porównanie poszczególnych metod. Wszechstronność jest określana jako możliwość zastosowana w różnych warunkach zarówno ekspozycji, zakłóceń jak i rodzaju przemieszczeń.

W pracy [11] przeprowadzono porównanie metod na obrazach pół-syntetycznych i syntetycznych. Poniżej przytoczono najważniejsze wnioski. Głównymi trudnościami metod gradientowych jest ciągłość czasowo-przestrzenna oraz właściwy wybór metody różniczkowania numerycznego wrażliwego na zaszumiony obraz niepozbawiony efektu aliasingu. Spośród metod gradientowych, autorzy pracy [11], proponują metody gradientowe lokalne jako wydajniejsze obliczeniowo i bardziej efektywne oraz pierwszego rzędu jako mniej podatne na ruch obrotowy i skalowanie.

Kolejna grupa to metody częstotliwościowe, które analizowane w [11] nie dały dobrych rezultatów zarówno pod względem efektywności wydajności, jednak są rozsądne w wykrywaniu ruchów nieuchwytnych innymi metodami.

Metody korelacyjne określane są w pracy [11] jako generalnie mniej skuteczne w przypadku niewielkiego ruchu z uwagi na niemożność określania prędkości subpikselowych. Trzeba jednak zaznaczyć, ze testy prowadzone przez autorów wspomnianej pracy były robione na obrazach syntetycznych lub pół-syntetycznych, pozbawionych szumów, co premiowało metody gradientowe. Autorzy wspominają również, że metody korelacyjne dają lepsze rezultaty dla większych prędkości. W metodach korelacyjnych ograniczeniem jest ciągłości sąsiedztwa.

2.4.1 Wstępne testy metod

Dla potwierdzenia wniosków literaturowych przeprowadzono wstępne badania metod obliczania przepływu optycznego. W testach wykorzystano funkcje biblioteki OpenCV, o której szerzej traktuje część 3.7. Porównano metody gradientowe: globalną HS, lokalną LK, metodę wyznaczania obrazu MHI oraz korelacyjną dopasowywania bloków (*ang. Block Matching*). Wszystkie funkcje opisane są w pomocy biblioteki OpenCV [1]. Badania przeprowadzono na dwóch sekwencjach testowych *ukazka2.avi* oraz *baran1.avi*. Testy wykazały, że najszyb-

Metoda	HS	LK	MHI	BM
sekwencja	ukazka2.avi			
kl/s	9.5	20.4	27.4	27
sekwencja.	baran2.avi			
kl/s	12.5	22.4	27	30.7

Tablica 2.1: Wstępne porównie wydajności dostępnych metod.

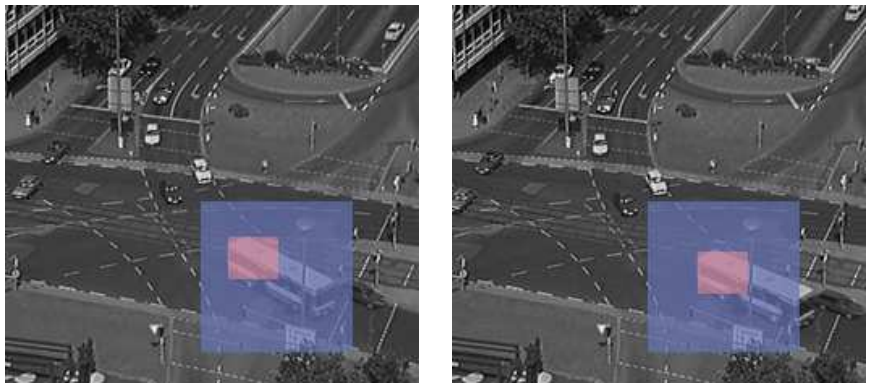
szą z metod jest dopasowywanie bloków, niezależnie od regulacji parametrów poszczególnych metod. Najsprawniej liczyła metoda korelacyjna dokonująca pełnego przeglądu z wczesnym kryterium kończącym. Jakość otrzymanych segmentów również w przypadku metod BM byłą najlepsza. W sekwencji *baran1.avi* tylko MHI dało spójny obszar, choć nie na całej długości filmu (Jest to efekt występowania problemu szczelinowego), z pozostałych, metoda BM spisała się najlepiej. W analizie filmu *ukazka2.avi* metoda BM oraz MHI dały spójny obszar. Ten krótki test wskazał na problem poruszony już wcześniej w rozważaniach teoretycznych, mianowicie podatność na zakłócenia. Pod tym względem najgorzej spisała się metoda lokalna LK, nawet dla dużych sąsiedztw szumy występowały i były rozpoznawane jako komponenty podobne do tych wykrywanych na obiekcie.

Mając na uwadze wspomniane wcześniej kryteria i rozważania oraz przeprowadzone wstępne testy zdecydowano się na wykorzystanie metody korelacyjnej dopasowywania bloków (BMA). Istotne cechy, na które zwrócono uwagę to dosyć duża odporność na szumy, a przede wszystkim duża ilość ramek. Prostota implementacji, szybkość i elastyczność dostępna dzięki możliwości zmian części parametrów, takich jak rozmiar obszaru poszukiwań, makrobloku. Intensywne prace nad zwiększeniem wydajności tych metod, głównie w dziedzinie szybkich algorytmów centralnie ukierunkowanych, czynią je tym bardziej atrakcyjnym do omówienia, poznania i wykorzystania na polu robotyki. W pracach [3, 13] prowadzono badania nad wykorzystaniem BMA w śledzeniu obiektów, wskazując celowość dalszych badań w tej dziedzinie. Poza tym można się spodziewać nowych szybszych algorytmów, a przede wszystkim, ich sprzętowych implementacji.

2.5 Teoria estymacji pola ruchu metodą dopasowywania bloków (Block Matching)

Wykrywanie ruchu obiektów w sekwencji obrazów jest zagadnieniem, do którego można podejść z wielu stron. W niniejszej pracy zdecydowano się na wykorzystanie metod powszechnie stosowanych w kompresji wideo MPEG, mianowicie, dopasowywanie bloków BM(*ang. Block Matching*). Podstawowymi cechami jest szybkość działania oraz prostota implementacji. Zasada działania jest intuicyjna i opiera się na poszukiwaniu fragmentu obrazu, z klatki aktualnej,

na obrazie referencyjnym, który wystąpił wcześniej, patrz rys. 2.6.

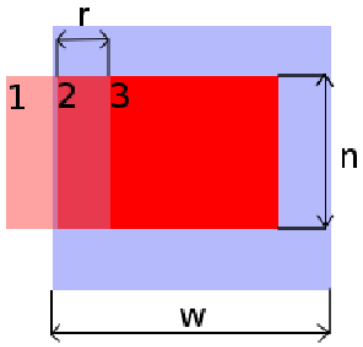


a) b)

Rysunek 2.6: Idea działania metody dopasowania bloków. Zaznaczono obszar poszukiwań i szukany fragment, a) klatka referencyjna b) klatka aktualna [1]

Błąd dopasowania makrobloków tworzy powierzchnię błędów na oknie poszukiwań i globalne minimum odpowiada najlepszemu dopasowaniu makrobloku z obrazu aktualnego i referencyjnego, co jednocześnie stanowi znaleziony wektor ruchu. Ponieważ powierzchnia błędu jest zazwyczaj niemonotoniczna, więc możliwe jest występowanie wielu lokalnych minimów w szczególności dla tych sekwencji obrazu z dużą intensywnością ruchu [20].

Podstawowymi parametrami charakteryzującymi metodę dopasowywania bloków są: wielkość obszaru poszukiwań fragmentu obrazu – w, wielkość makrobloku – n czyli wielkość poszukiwanego fragmentu oraz gęstość poszukiwań – r, odpowiadająca gęstości ułożenia wektorów ruchu w polu wektorowym. Wszystkie wielkości wyrażone są w pikselach. W kodowaniu MPEG przyjęto założenie, że makrobloki nie nachodzą na siebie, więc $r = n$, jednak dla celów badawczych i ten parametr jest istotny. Wymienione wyżej cechy obrazuje rysunek 2.5.



Rysunek 2.7: Podstawowe parametry metody dopasowywania bloków: w - okno poszukiwań, n - makroblok, r - gęstość. Numerami oznaczono kolejne makrobloki.

2.5.1 Miary błędu

Ponieważ dla rzeczywistej sekwencji, obarczonej zakłóceniami, prawdopodobieństwo znalezienia identycznego fragmentu w klatce referencyjnej równe jest zero , stosuje się miarę korelacji,

[1] Sekwencje pobrane ze strony http://i21www.ira.uka.de/image_sequences/

wspomnianą wcześniej w ogólnym wzorze (2.15).

Powszechnie w estymacji pola ruchu stosuje się różne miary błędu MAD (*ang. Mean Absolute Distortion*), SAD (*ang. Sum of Absolute Difference*) – interesującą w przypadku obliczeń numerycznych gdyż nie wymaga dzielenia. Dla przykładu, w równaniu (2.16) pokazano, zasadę wyliczania błędu MAD:

$$MAD(\Delta x, \Delta y) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |f_b(x+i, y+i) - f_r(x+\Delta x+i, y+\Delta y+j)|, \tag{2.16}$$

gdzie (x,y) to współrzędne makrobloku, N jego wielkość, f_b i f_r to odpowiednio obraz bieżący i referencyjny. We wzorze (2.16) przesunięcie Δ jest ograniczone oknem poszukiwań, musi ono zostać ograniczone do sensownej wielkości z powodu złożoności obliczeniowej. W kodowaniu MPEG za rozsądne uznaje się obszar poszukiwań wielkości $(x \pm 16, y \pm 16)$ pomiędzy dwoma kolejnym klatkami obrazu [17]. Jednak nawet przy tej wielkości okna przy zastosowania metody pełnego przeszukiwania (*ang. FS - Full Search*) złożoność obliczeniowa jest bardzo duża. Dla przykładu:

$$T\left(\frac{P}{N} \cdot \frac{Q}{N}\right) \cdot (2W+1)^2(2N^2-1) \cong 8TPQW^2 \cong 6.3 \cdot 10^9, \tag{2.17}$$

gdzie ilość klatek $T = 30kl/s$, wielkość okna $P = 288$ i $Q = 360$ pikseli oraz maksymalne przesunięcie $W = 16$ pikseli.

2.5.2 Szybkie metody estymacji

Pełny przegląd(FS) jest metodą najprostszą, globalną w zakresie okna poszukiwań ale siłową, co widać w równaniu (2.17). Dlatego zaczęto poszukiwać metod wydajnych i efektywnych obliczeniowo przy zachowaniu jakości przeglądu zupełnego. Opracowanie szybszych metod niż FS, nie byłoby możliwe, gdyby nie pewne założenia oparte na badaniach. Można przytoczyć kilka podstawowych postulatów i wniosków.

Podwaliny szybkich algorytmów

1. Większość rzeczywistych sekwencji wideo ma centralnie zgrupowaną dystrybucję wektorów ruchu. Badania wykazały, że nawet przy sekwencjach z meczu piłkarskiego, gdzie mamy do czynienia z dużą ilością poruszających się obiektów, więcej niż 80% bloków jest stacjonarna lub quasi-stacjonarna, co oznacza, że ruch występuje najczęściej w zakresie centralnego obszaru 3x3. Przy wolnej sekwencji 90-99% wektorów nie przekracza 3 pikseli [10]. Algorytmy takie jak TSS(*New Three Step Search*) [14] , 4SS (*Four Step Search*), DS (*Diamod Search*) [20] są centralnie ukierunkowane.

2. Ograniczenie ilości pikseli w makrobloku biorących udział w określaniu wartości błędu dopasowania przyspiesza działanie algorytmu. Wybierając powiedzmy co drugi, bezpośrednio zmniejszamy liczbę wykonywanych operacji. Również przerywanie obliczeń gdy wartość błędu przekroczy pewien zdefiniowany wcześniej na sztywno (metoda MVFAST (*ang. Motion Vector Field Adaptive Search Technique*) [8]) lub wyliczany adaptacyjnie próg [4], przynosi pożądane efekty.

3. Redukcja rozdzielczości w oczywisty sposób zmniejsza pole poszukiwań, jednak bardziej wyrafinowane metody stosują tę technikę do zgrubnego wyznaczenia obszaru poszukiwań, a następnie interpolują znaleziony wektor do rozmiaru oryginalnego. Wspomniano tę technikę w rozdziale 2.3.5.

4. Predykcja, na podstawie wektorów z poprzedniego wyliczonego pola ruchu lub z przystających makrobloków, pozwala na ukierunkowanie poszukiwań. Możemy tak robić przy założeniu, że każdy obiekt posiada pewną bezwładność poruszania się a tym samym jego położenie jest przewidywalne.

Wspomniany wyżej w części 2.5.2 wniosek o centralnie zgrupowanej dystrybucji wektorów ruchu był podstawowym założeniem algorytmów centralnie ukierunkowanych(zero ukierunkowanych) W tych metodach zakłada się niewielki ruch i występowanie globalnego minimum błędu estymacji w najbliższym otoczeniu punktu $(\Delta x, \Delta y) = (0,0)$. W najprostszym przypadku można sprawdzać wszystkie punkty kolejno od $(0,0)$ poruszając się makroblokiem na obrazie referencyjnym ruchem spiralnym, aż do krawędzi okna poszukiwań. W momencie napotkania wartości błędu SAD poniżej wartości progowej algorytm, kończyłby działanie a wskazany wektor byłby ostatecznym, (metoda stosowana w bibliotece OpenCV, w pracy oznaczona jako FS*).

Jednak, generalnie rzecz ujmując, wzór przeszukiwania i rozmiar okna poszukiwań wykorzystywane w szybkich algorytmach determinują nie tylko szybkość przeszukiwania ale i jakość, dlatego zaczęto poszukiwać bardziej wyrafinowanych metod, które analizując niewielką liczbę punktów wpadają w globalne minimum.

Poniżej zostanie przedstawiona zasada działania obu zaimplementowanych algorytmów: DS i zbudowanego na jego podstawie PMVFAST.

2.6 Przeszukiwanie karo - tajniki szybkiego algorytmu

Shan Zhu and Kai-Kuang jako pierwsi zaproponowali algorytm DS, w którym przeszukiwane miejsca tworzyły kształt karcianego karo – warstwice metryki Manhattan [20]. Badania pokazały [10, 20, 4, 8], że algorytmy bazujące na przeszukiwaniu karo są szybkie, jednak uzyskiwana jakość[1] nie jest zadowalająca w porównaniu do pełnego przeglądu. Dlatego wprowadzano kolejne modyfikacje co doprowadziło do powstania algorytmu PMVFAST [4].

2.6.1 Algorytm DS

Poniżej przedstawiono kilka pojęć definiujących przeszukiwane karo. Przyjmijmy, że obszar poszukiwań jest siatką miejsc zdefiniowaną równaniem (2.18).

$$S = \{s_1, s_2, \ldots, s_n\} \tag{2.18}$$

Natomiast (2.19) jest przyjętym systemem sąsiedztwa

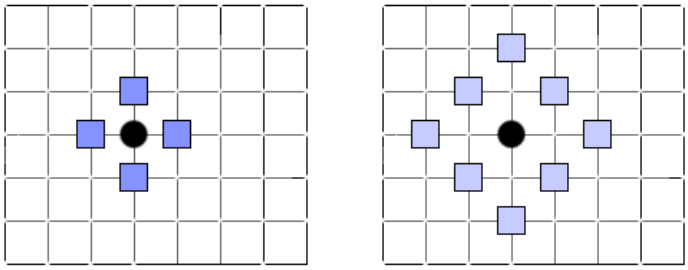
$$N^o = \{N_s^o, s = (i,j) \in S\}, \tag{2.19}$$

zdefiniowanym jako:

$$N_s^o = \{r = (k,l) \in S \mid |k-i| + |l-j| = o\}. \tag{2.20}$$

W algorytmie wykorzystuje się dwa rozmiary szablonów poszukiwań, które można zdefiniować jako sąsiedztwa piksela wraz z miejscem centralnym. Duże karo o rozpiętości 5 pikseli (rys. 2.8b) oraz małe wielkości 3 piksele (rys. 2.8a), które są odpowiednio sąsiedztwami rzędu pierwszego i drugiego miejsca s, zgodnie z definicją (2.20).

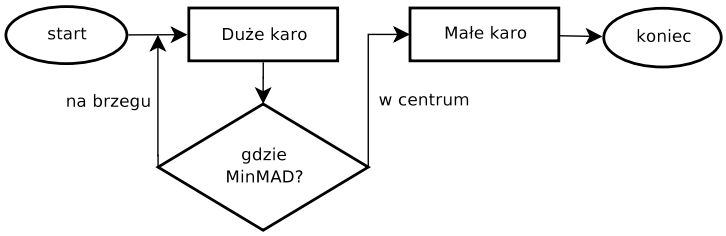
[1] Spotykane w literaturze tradycyjne miary jakości dla skompresowanego obrazu: PSNR(*ang. peak signal to noise ratio*) oraz MSE(*ang. mean square error*).



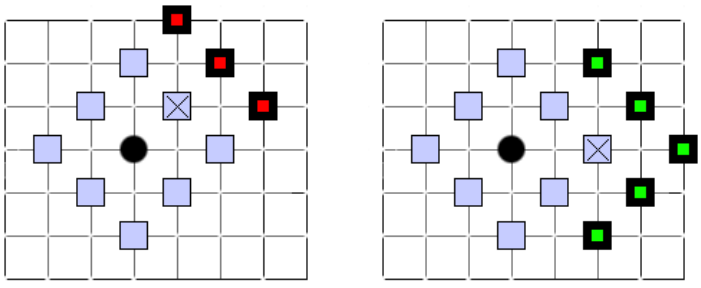
a) b)

Rysunek 2.8: Sąsiedztwo. • – miejsce od którego liczone jest sąsiedztwo, a) sąsiedztwo 1. rzędu(małe karo), b) sąsiedztwo 2. rzędu(duże karo).

Działanie algorytmu rozpoczynamy umieszczając duży romb na obrazie referencyjnym w miejscu odpowiadającym wektorowi przesunięcia $(0,0)$, więc w środku okna poszukiwania. Spośród dziewięciu punktów zostaje wybrany punkt z najmniejszą wartością MAD i on określa tymczasowy wektor przesunięcia$(\Delta x, \Delta y)$. W wypadku jeśli MAD znaleziono na brzegu karo, centrum dużego szablonu jest przesuwane w to miejsce i krok 1. jest powtarzany. Jeśli minimum znajduje się w środku karo, to w następnym kroku zmieniamy wielkość szablonu na mały (rysunek 2.8a), aby zbadać pod kątem istnienia MAD bezpośrednie sąsiedztwo 1. rzędu punktu wskazywanego przez wektor $(\Delta x, \Delta y)$. Znalezienie minimum w małym rombie kończy działanie algorytmu, miejsce jego wystąpienia określa wektor przemieszczenia. Przechodzimy do następnego makrobloku.



Rysunek 2.9: Schemat działania algorytmu przeszukiwania karo.



a) b)

Rysunek 2.10: Przesunięcie szablonu karo w zależności od miejsca wystąpienia punktu $(\times)$ – minMAD.

Tymczasowy wektor przemieszczenia przesuwa się po ścieżce nierosnących wartości błędu dopasowania. Ta właściwość w połączeniu z gwarancją natrafienia na globalne minimum, co

przy założeniu centralnie zgrupowanego ruchu jest spełnione, pozwala poprawnie wyznaczyć wartość przemieszczenia. O testach i badaniach tej metody można przeczytać w rozdziale 4.

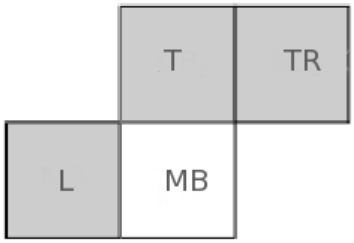
Należy zauważyć, że w zależności od miejsca wystąpienia minimum liczba nowo wyliczonych wartości MAD można ograniczyć do 3. lub 5., co znacznie przyspiesza działanie algorytmu. Obrazują to rysunki 2.10a, 2.10b.

2.6.2 Algorytm PMVFAST

Metoda PMVFAST zaproponowana w pracy [4] bazuje na algorytmie MVFAST [8] i modyfikuje przedstawiony wyżej algorytm dodając do niego elementy predykcji i adaptacji. Algorytm w porównaniu do DS jest skuteczniejszy, biorąc pod uwagę zarówno wydajność jak i jakość. Poniżej przedstawiono jego strukturę i cechy.

Do algorytmu DS wprowadzono wstępną analizę wektorów predykcji czasowej i przestrzennej, do których zaliczają sie:

- $\overrightarrow{00}$ - wektor (0,0),
- $\overrightarrow{T}$, $\overrightarrow{L}$, $\overrightarrow{TR}$ - najlepsze wektory z trzech przystających makrobloków: górnego, lewego i górnego z prawej (patrz rys. 2.11) – *region wsparcia*.
- $\overrightarrow{M}$ - mediana trzech poprzednich punktów, ponieważ rozkład wektorów ruchu skupia się właśnie wokół tej wartości [4] (w związku z tym algorytm PMVFAST nazywany jest medianowo ukierunkowanym),
- $\overrightarrow{-1}$ - wektor uzyskany dla tego makrobloku w poprzedniej estymacji.



Rysunek 2.11: Region wsparcia dla aktualnego makrobloku.

Na początku PMVFAST liczy SAD dla wektora $\overrightarrow{M}$ i zatrzymuje się gdy spełniony jest jeden z dwóch warunków stopu. Pierwszy zachodzi, jeśli $\overrightarrow{M} = \overrightarrow{-1}$ i SAD $\overrightarrow{M}$ jest mniejszy niż w $\overrightarrow{-1}$. Drugie kryterium jest spełnione, gdy SAD $\overrightarrow{M}$ jest mniejsze od wartości progowej.

Następnie PMVFAST liczy SAD dla kilku bardzo prawdopodobnych wektorów przemieszczenia ($\overrightarrow{L}$, $\overrightarrow{T}$, $\overrightarrow{TR}$, $\overrightarrow{-1}$, $\overrightarrow{(0,0)}$, $\overrightarrow{M}$) i zatrzymuje się w momencie spełnienia jednego z dwóch warunków. Pierwszy występuje, gdy najlepszy wektor ruchu równy jest wektorowi $\overrightarrow{-1}$ i dodatkowo minimalne SAD jest mniejsze niż dla $\overrightarrow{-1}$. Drugie kryterium zachodzi, gdy najmniejszy błąd dopasowania jest mniejszy od wartości progowej.

Trzecim punktem algorytmu po zakończeniu etapu wczesnej terminacji poszukiwań, jest wyznaczenie wektora ruchu zgodnie z minimum SAD i rozpoczęcie przeszukiwania karo. Wybór rozmiaru karo następuje po przeanalizowaniu warunków. Jeśli $\overrightarrow{M}$ jest równy $\overrightarrow{(0,0)}$ i wektory trzech przystających bloków są takie same, a wartość progu predykcyjnego powiązanego

z tymi wektorami jest duża, to minimum SAD poszukiwane jest dużym karo. W przeciwnym razie, gdy wektory z przystających bloków są równe wektorowi $\overrightarrow{-1}$, wykorzystywany jest małe karo i wykonywana jest tylko jedna iteracja. W innym wypadku, minimum jest poszukiwane z użyciem małego rombu.

Poniżej przedstawiono algorytm krok po kroku. Wartości progów wykorzystywane w MPEG, zgodne ze standardem ISO [2].

Działanie algorytmu krok po kroku

(Inicjacja)

K1 Ustaw parametry progowania T1a, T1b:
if (pierwsza kolumna, pierwszy wiersz) **then** $T1a = 512$, $T1b = 1024$;
else $T1a = min_{SAD}(T, L, TR)$, $T1b = T1a + 256$;
if $T1a < 512$ **then** $T1a = 512$.
if $T1a > 1024$ **then** $T1a = 1024$
If $T1b > 1792$ **then** $T1b = 1792$.
Ustaw $Found = 0$ and $PredEq = 0$
Wyznacz $\overrightarrow{M}$ jako $med(\overrightarrow{T}, \overrightarrow{TR}, \overrightarrow{L})$
Wyznacz predyktory: $\overrightarrow{L}$, $\overrightarrow{T}$, $\overrightarrow{TR}$, **if** makroblok jest na krawędzi:

- **if** blok w pierwszej kolumnie **then** $\overrightarrow{L} = (0,0)$.
- **if** blok w pierwszym wierszu **then** $\overrightarrow{T} = (0,0)$, $\overrightarrow{TR} = (0,0)$.
- **if** blok w ostatniej kolumnie przyjmij **then** $\overrightarrow{TR} = (0,0)$.

if $\overrightarrow{T}$ =$\overrightarrow{TR}$ = $\overrightarrow{L}$ **then**
$PredEq = 1$, $\overrightarrow{Pred} = \overrightarrow{T}$

(Wstępne liczenie predyktora)

K2 Policz $||\overrightarrow{M}||$, gdzie $\overrightarrow{M}$ jest wektorem mediany.
if $PredEq = 1$ **and** $\overrightarrow{Pred} = \overrightarrow{-1}$, ustaw Found=2

K3 **if** $||\overrightarrow{M}|| > 0$ **or** $T1b < 1536$ **or** $PredEq = 1$
wybierz małe karo **else** wybierz duże karo

K4 Policz SAD dla wektora $\overrightarrow{M}$, $MinSAD = SAD$
if $\overrightarrow{M} = \overrightarrow{-1}$ **and** $MinSAD < PrevFrmSAD$ **goto** K10.
if $SAD \leq 256$ **goto** K10.

K5 Policz SAD dla $\overrightarrow{L}$, $\overrightarrow{T}$, $\overrightarrow{TR}$, $\overrightarrow{-1}$, $\overrightarrow{(0,0)}$.
$MinSAD = min_{SAD}(\overrightarrow{L}, \overrightarrow{T}, \overrightarrow{TR}, \overrightarrow{-1}, \overrightarrow{(0,0)}, \overrightarrow{M})$
$\overrightarrow{MV}$ oznacza najlepszy wektor z $\overrightarrow{L}$, $\overrightarrow{T}$, $\overrightarrow{TR}$, $\overrightarrow{-1}$, $\overrightarrow{(0,0)}$

K6 **if** $MinSAD \leq T1a$ **goto** K10.
if $\overrightarrow{MV} = \overrightarrow{-1}$ **and** $MinSAD < PrevFrmSAD$ **goto** K10.

(Przeszukiwanie karo)

K7 Wykonuj przeszukiwanie w karo dużym (K9) lub małym (K8).
if $Found = 2$ przeszukaj karo jeden raz **goto** K10

K8 **if** przeszukiwanie małego karo, wykonuj kolejne iteracje, aż wektor w środku karo. **goto** K10

K9 **if** przeszukiwanie duże karo, wykonuj kolejne iteracje, aż wektor w środku karo, popraw małym rąbem. **goto** K10.

(Ostatni krok. Wykorzystaj wyliczony najlepszy wektor)

K10 Wektor ruchu wybrany jest zgodnie z najmniejszym SAD - MinSAD.

Rozdział 3

System wizyjny - implementacja

W tym rozdziale omówiono programową, jak i sprzętowa realizacja skonstruowanego systemu wizyjnego. W kolejnych punktach omówiono poszczególne etapy napisanego programu przetwarzania obrazu, począwszy od wstępnej analizy i filtracji, poprzez znalezienie interesującego pola ruchu, aż po wykorzystanie uzyskanych parametrów do określania przewidywanej trajektorii poruszania się obiektu. Na koniec przedstawiono sprzętową część projektu i wykorzystane narzędzia.

Zadaniem stworzonego systemu wizyjnego było wykrywanie obiektów z możliwością predykcji zachowań. Generalną zasadę działania programu można sprowadzić do trzech punktów:

1. obliczenie pola ruchu,

2. znalezienie obiektu,

3. predykcja i określenie pozycji.

3.1 Obliczenie pola ruchu

Podstawową funkcją jest znalezienie pola ruchu. W tym celu wykorzystane zostały trzy metody omówione w rozdziale 2.5. Wspomniane metody to: pełny przegląd z wczesnym warunkiem końca(FS*), zaimplementowany w bibliotece OpenCV [1], oraz dwa szybkie algorytmy zaimplementowane przez autora: przeszukiwanie karo (DS) oraz PMVFAST.

Funkcje zostały napisane zgodne ze standardem OpenCV, tak, by stanowiły rozszerzenie bibiloteki. Co więcej, umożliwiają one dobór parametrów r, n i w, które omówiono w rozdziale 2.5. Ponieważ funkcja FS z bibioteki OpenCV nie była w pełni konfigurowalna, stworzono również metodę FS dającą szersze pole zmian parametrów.

Efektem działania wszystkich wspomnianych metod są macierze wektorów przesunięć makrobloków w osi x i y (*velx*, *vely*). Następnie niezbędne jest przetransformowanie tych wektorów ze współrzędnych kartezjańskich do biegunowych, a otrzymane w ten sposób macierze orientacji (*orient*) i wartości wektorów (*mag*) są podstawą do dalszego etapu pracy.

Do wyliczania pola wektorowego wykorzystywane są dwie ramki(*img1*, *img2*) obrazu następujące po sobie w czasie . Obrazy są transformowane do skali szarości i wstępnie progowane filtrem lokalna mediana wielkości 3x3 w celu wyeliminowania szumów wysokiej częstotliwości. Należy wspomnieć, że funkcje estymacji przepływu optycznego zwracają obrazy w rozdzielczości mniejszej niż oryginalny obraz *img1*, *img2*. Zakładając, że makroblok jest kwadratem o rozmiarze $n \times n$, a odległość pomiędzy kolejnymi makroblokami w pionie i poziomie to r, wielkość obrazu wynikowego można wyliczyć wzorem $S_{velx,vely} = (S_{img1,img2} - n/r)$. Można

powiedzieć, że jest to istotna informacja, bo nasze wyliczanie pola wektorowego została wykonana z ziarnem r i wprawdzie tracimy cześć informacji, ale dzięki temu dalsze obliczenia prowadzone są szybciej, bo na mniejszym obrazie.

3.2 Znalezienie obiektu

Macierz wartości wektorów (*mag*) jest następnie progowana w celu uzyskania sylwetki poruszającego się obiektu (*silh*). Obcinane są w ten sposób wszystkie wartości pikseli równe zero. Sylwetka jest następnie poddawana transformacji lokalne minimum o rozmiarze 3x3 piksele w celu odfiltrowania jednostkowych wektorów ruchu nie tworzących większych pól.

Tak otrzymany binarny obraz sylwetki, który ma wartość 1 dla miejsc ruchomych obrazu oraz 0 dla stacjonarnych, jest poddawany segmentacji. Na obrazie znajdowane są wszystkie komponenty, jednak wektor kierunku dla obiektu liczony jest jedynie dla pól wektorowych odpowiadających wielkością szukanemu obiektowi. Jeśli więc suma wymiaru x i y szukanego obiektu mieści się w zadeklarowanym przedziale algorytm decyduje, że jest to szukany obiekt.

Należy również zaznaczyć, że jeśli ilość znalezionych komponent na obrazie będzie większa od wartości granicznej dana klatka sekwencji jest pomijana, celem wyeliminowania z obróbki obrazów o bardzo dużym zaszumieniu.

3.3 Wyznaczanie wektora przemieszczenia dla obiektu

Kolejne operacje na obrazach *orient* oraz *mag* zawierających odpowiednio orientacje wektorów i ich wartości, wykonywane są jedynie na punktach znajdujących się pod maską silh i należących do znalezionego obiektu. Z wszystkich wektorów orientacji wyznaczonych wcześniej, wyliczany jest ogólny kierunek charakteryzujący cały obiekt. Otrzymany w ten sposób wektor dla całego obiektu umieszczany jest w punkcie środka masy znalezionego obiektu. W tym celu liczone są momenty rzędu zerowego (3.1) i pierwszego (3.2):

$$m_{0,0} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \chi_U(x,y) \tag{3.1}$$

$$m_{1,0} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \chi_U(x,y)x;\ m_{0,1} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \chi_U(x,y)y, \tag{3.2}$$

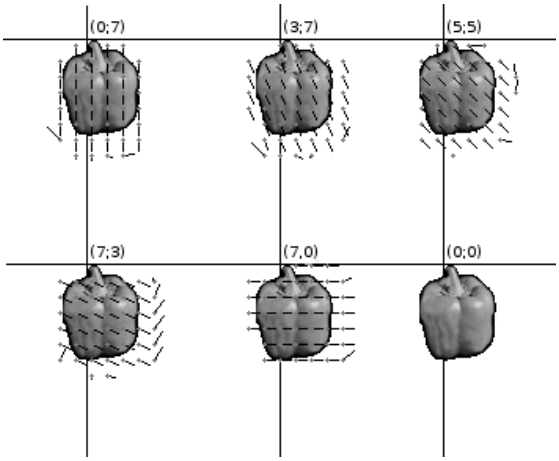
gdzie χ_U to funkcja charakterystyczna sylwetki U (maski). Ostateczne środek sylwetki wyznaczany jest jako $u_s = (x_s, y_s)$, gdzie $x_s = \frac{m_{1,0}}{m_{0,0}}$, $y_s = \frac{m_{0,1}}{m_{0,0}}$.

Przyjęto dwie proste metody określania kierunku: liczenie maksimum histogramu i liczenie średniej wartości. Poniżej przedstawiono dwie badane metody określania wektora dla obiektu.

3.3.1 Maksimum histogramu

W pracy [13] zaproponowano metodę poprawiania kierunku wektorów w komponencie do tego najbardziej reprezentowanego. Jak pokazały badania pola ruchu na obrazach syntetycznych oraz analiza histogramów orientacji wektorów z rzeczywistych sekwencji, wektory należące do jednego obiektu generalnie mają tendencję do wskazywania jednego kierunku. Jedynie na krawędziach występują zaburzenia i błędne wskazania, co widać na rysunku 3.3. Dlatego postanowiono, jako główny kierunek dla obiektu wybierać wartość kąta otrzymaną jako maksimum z histogramu wartości orientacji, znajdujących się pod sylwetką danej komponenty, bez poprawiania reszty jak w [13]. Ogólną wartość wektora obliczano jako średnią z tych wartości,

których odpowiadające wektory kierunku były w maksimum histogramu. Uzyskany w ten sposób kierunek nie zmieniał się płynnie. Jak zaobserwowano, czasami histogram kierunku mający zdecydowane maksimum osiągał kolejne zdecydowane maksimum po kilku klatkach, w ciągu których nie było jednej dominującej wartości. Doprowadziło to do podejścia uśredniającego wartość kierunku.



Rysunek 3.1: Przykład estymacji ruchu. Metoda FS. Widoczne błędne dopasowania na krawędziach pola wektorowego.

3.3.2 Średnia wartość

Drugim badanym podejściem jest wyliczanie średniej z wszystkich wektorów znajdujących się pod sylwetką komponenty zarówno dla wartości kierunku jak i długości wektora. Ta metoda zakłada wpływ błędów estymacji na otrzymywany kierunek jak i wartość, ponieważ nie eliminujemy wektorów wskazujących błędny kierunek. Przyniosło to jednak korzyść ponieważ zmian kierunku w sekwencji była bardziej płynna. Średnia była liczona z uwzględnieniem skoku wartości kąta z 360^o na 0^o w początku układu współrzędnych biegunowych (cykliczność składowej kątowej).

3.4 Predykcja trajektorii

Aby pożytecznie wykorzystać uzyskane parametry, system wizyjny został zaprojektowany, tak, aby mógł przewidywać trajektorię obserwowanego obiektu. Przewidywanie zachowania obiektu jest możliwe dzięki pamięci zmian orientacji obiektu i ostatnich wartości translacji. Program zapamiętuje wskazaną liczbę m ostatnich zmian orientacji w postaci wektora zmian:

$$A = [\delta_n, \delta_{n-1}, \ldots, \delta_{n-m}], \tag{3.3}$$

gdzie n to numer aktualnej ramki obrazu. Podobny wektor:

$$T = [t_n, t_{n-1}, \ldots, t_{n-m}], \tag{3.4}$$

stworzono z m ostatnich wartości przemieszczeń.

W badaniach porównano dwie metody wyznaczania trajektorii:

- opartą na uśrednionych wartościach wektorów A i T,

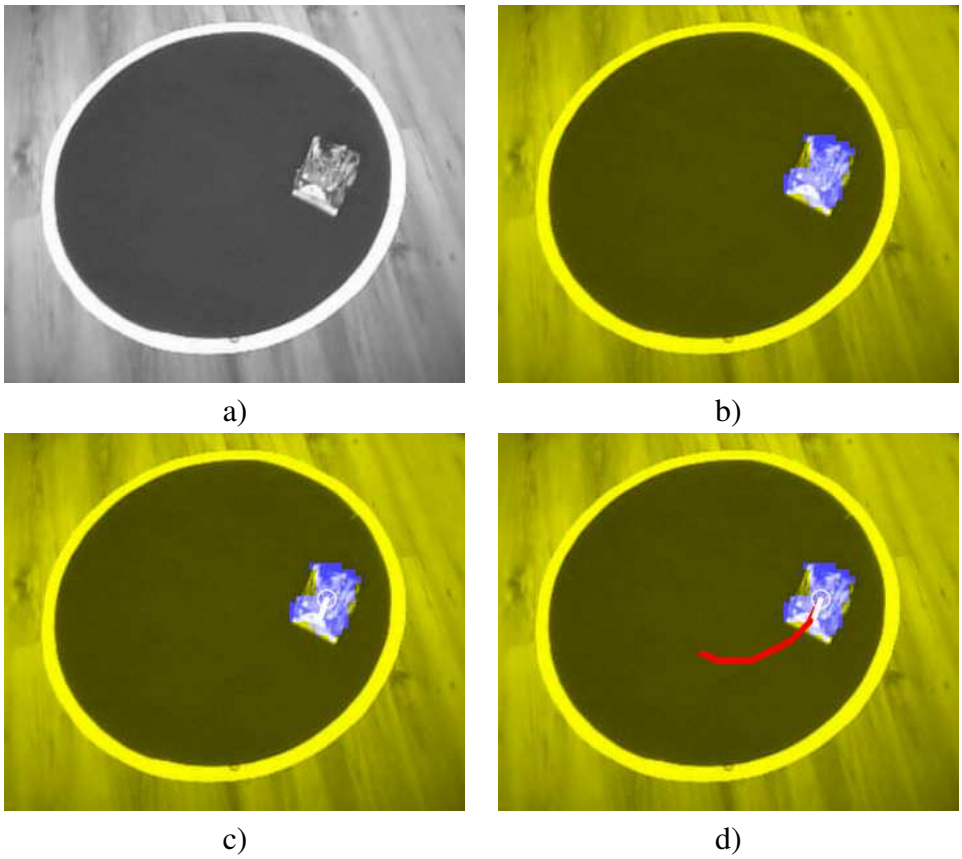
- bazującą na ważonej średniej wartości wektorów A i T.

W obu przypadkach otrzymujemy wartości δ_r i t_r. Krzywa predykcji zbudowana jest z rekurencyjnie wyliczanych punktów predykcji. $P = (x_{pp}, y_{pp})$ o współrzędnych ekranowych wyliczonych na podstawie wzoru (3.5), których każdy kolejny odpowiada przypuszczalnemu położeniu obiektu w następnej klatce obrazu i wyliczany jest względem wcześniej wyliczonego punku predykcji $P^{-1} = (x_{pp}^{-1}, y_{pp}^{-1})$.

$$x_{pp} = Round\left(x_{pp}^{-1} + t_r cos\left(\frac{\delta_r * \pi}{180}\right)\right),$$
$$y_{pp} = Round\left(y_{pp}^{-1} - t_r sin\left(\frac{\delta_r * \pi}{180}\right)\right), \tag{3.5}$$

Parametrami wpływającymi na działanie predykcji są: rozmiar bufora historii - m i długość czasu predykcji - p. Zarówno jedna jak i druga podawana jest jako liczba klatek. Ponadto sposób liczenia średniej wartości δ_r i t_r ma również znaczenie. Badania tych problemów przeprowadzono w rozdziale 4.

3.5 Działanie programu - ilustracje

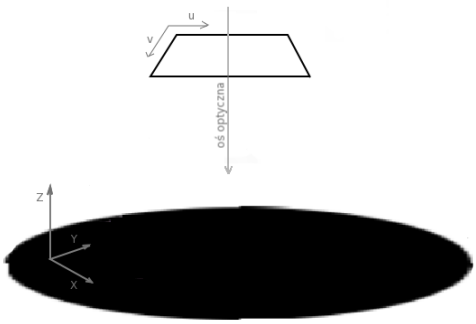


a) b) c) d)

Rysunek 3.2: Kolejne kroki programu. a) Obraz wejściowy, b)wyznaczone pole ruchu, c)określony wektor dla obiektu, d) predykcja.

3.6 Transformacja układu współrzędnych

Zaproponowany system wizyjny składa się z kamery internetowej umieszczonej nad sceną opisanej szerzej w rozdziale 3.7, traktującym o stanowisku pomiarowym. Aby można było wskazać położenie robota w układzie współrzędnych sceny niezbędna jest transformacja do układu kamery. Przyjęto założenie, że kamera jest umieszczona nad sceną tak, aby wskazanie pozycji trójwymiarowego obiektu widzianego na dwuwymiarowym obrazie było jak najbardziej dokładne. Pozbyto się w ten sposób jednego wymiaru, a więc, rozwiązano problem współpłaszczyzności wektorów prędkości obiektu i przepływu optycznego.



Rysunek 3.3: Idealne rzutowanie sceny na płaszczyznę obrazu.

W idealnym przypadku, oś optyczna kamery jest prostopadła do powierzchni określonej osiami X i Y stanowiącej płaszczyznę sceny. Przy takim założeniu wystarczająca jest transformacja dwuwymiarowa opisana macierzą (3.6), zawierającą translację w płaszczyźnie XY oraz rotację wokół osi Z. Ponieważ, osie Z układu kamery i sceny mają przeciwne kierunki niezbędny jest jeszcze obrót wokół osi X. Ostatecznie macierz ma postać:

$$T = \begin{bmatrix} c_\alpha & s_\alpha & t_x \\ s_\alpha & -c_\alpha & t_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.6}$$

3.6.1 Metoda kalibracji

Standardowo w wizji cyfrowej kalibracja polega na wyznaczeniu dwóch macierzy: P - parametrów wewnętrznych oraz K – parametrów zewnętrznych kamery (3.7). Wewnętrzne opisują cechy właściwe kamerze t.j. rozmiar piksela, długość ogniskowej, optyczny środek obrazu cyfrowego, natomiast zewnętrzne to transformacja współrzędnych z układu otoczenia do układu kamery [15].

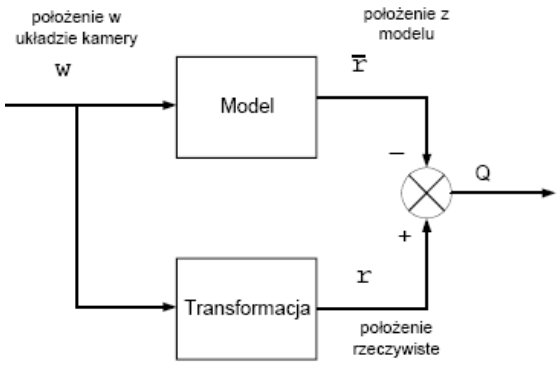
$$\tilde{w} = PK\tilde{r} \tag{3.7}$$

Ponieważ ostatecznie wszystkie te cechy zawarte są w jednej ostatecznej macierzy transformacji T (3.6), możliwe jest inne podejście do problemu.

Jeśli wektor (3.8) traktować jako współrzędne uogólnione punktu w układzie kamery (wizji), natomiast wektor (3.9) będzie reprezentować położenie punktu w układzie zewnętrznym to dysponując serią pomiarową współrzędnych odpowiednich punktów możliwa będzie identyfikacja macierzy transformacji.

$$w = [u, v, 1]^T \tag{3.8}$$

$$r = [x, y, 1]^T \tag{3.9}$$



Rysunek 3.4: Model transformacji

Model transformacji można przedstawić blokowo jak na diagramie 3.4.

$$\tilde{w} = T\tilde{r} \tag{3.10}$$

W celu uzyskania transformacji między układami (3.10) należy przeprowadzić serię p pomiarów, dzięki czemu otrzymamy wektory (3.11) oraz (3.12), które reprezentują, odpowiednio, serię pomiarów we współrzędnych układu wizyjnego oraz układu sceny, gdzie elementami są wektory (3.8) i (3.9).

$$W_p = [w_1, \ldots, w_p] \tag{3.11}$$

$$R_p = [r_1, \ldots, r_p] \tag{3.12}$$

$$Q = \sum_{i=1}^{p} (r_i - Tw_i)^T (r_i - Tw_i). \tag{3.13}$$

Kryterium jakości Q (3.13) jest błędem średnio-kwadratowym i dopasowanie modelu do transformacji jest tym lepsze im mniejszy jest błąd. Po zróżniczkowaniu otrzymujemy macierz transformacji (3.14). Z minimalizacji kryterium (3.13) otrzymujemy (3.14) [19].

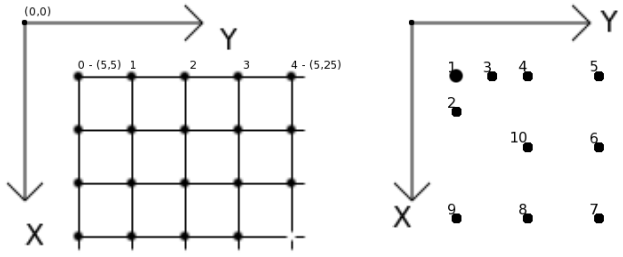
$$T_p = R_p W_p^T (W_p W_p^T)^{-1}. \tag{3.14}$$

W przeprowadzonych badaniach jakościowych jako kryterium przyjęto bezwzględny błąd.

3.6.2 Wzorce kalibracji

Na potrzeby zaprojektowanego systemu wizyjnego zaproponowano wzorce kalibracji, które posłużyły do identyfikowania przekształcenia układu współrzędnych. Do badań jakościowych wykorzystano wzorzec składający się z 19 punktów tworzących siatkę. Poszczególne punkty ponumerowane są od 0 do 18 i oddalone od siebie o 5cm. Siatka przedstawiona jest na rysunku 3.5a. Ułożenie siatki na powierzchni sceny determinuje orientację układu współrzędnych. Warunkiem koniecznym jest niewspółliniowość punktów wchodzących do próby tak aby macierz $(W_p W_p^T)$ była odwracalna. Badania dokładności kalibracji znajdują się w rozdziale 4.2.

Druga siatka (rys. 3.5b) została stworzona w celu zautomatyzowania identyfikacji przebiegającej praktycznie bez udziału użytkownika, składa się z dziesięciu punków tworzących wzór rozpoznawalny przez program auto detekcji. omówiony poniżej. Ponieważ punkty w założeniu są wieksze niż 1 piksel, więc współrzędne ekranowe określane są jako środek masy punktu.



a) b)

Rysunek 3.5: Szablony kalibracji, a) szablon testowy, b) szablon auto-identyfikacji.

3.6.3 Automatyczna detekcja wzorca

Kalibracja kamery przy pomocy wzorca złożonego z punktów wymaga podania współrzędnych punktów przy każdej zmianie orientacji układu współrzędnych lub zapamiętania ich i pracy w jednym ustawieniu. Aby zautomatyzować działanie programu i odciążyć użytkownika od mozolnego podawania współrzędnych punktów wskazywanych przez komputer, program został wyposażony w możliwość automatycznej detekcji wzorca.

Aby automatyczne wykrycie punktów było możliwe, należy przeprowadzić wstępną obróbkę obrazu. Punkty powinny być białe, mieć wzór zgodny z rysukiem 3.5b i znajdować się na największym czarnym obszarze widzianym obiektywem kamery.

Program rozpoczyna działanie od wstępnego progowania obrazu w skali szarości wartością podaną przez użytkownika(na tym kończy się jego udział). Akceptowane są te piksele, których wartości są mniejsze od zadeklarowanego progu W ten sposób otrzymujemy maskę negatywową(*mask*), docelowo tło/scena, do której tymczasowo nie wchodzą kontrastowe punkty. Taka sama operacja, tylko z odwrotnym progiem, daje sylwetkę z punktami *silh*. Obydwa obrazy mogą zaklasyfikować niepotrzebne elementu otoczenia, które należy usunąć z obrazów. Obraz *mask* poddawany jest transformacji lokalna mediana, w otoczeniu 9×9, aby włączyć piksele punktów do sylwetki, a następnie znajdowana jest największa komponenta na obrazie, w celu wydzielenia obszaru zainteresowania poszukiwania punktów. Dlatego istotne jest, żeby punkty znajdowały się na stosunkowo największym czarnym tle. Jedyne co pozostaje to filtrem lokalne minimum($\times 4$) zmniejszyć maskę *mask* i na obu obrazach *mask* i *silh* wykonać operację logicznej sumy, wydzielone zostaną wtedy te obszary, które odpowiadają za punkty z szablonu i znajdują się pod maską *mask*.

Zaproponowany szablon z punktami (rys. 3.5b) składa się z 10 punktów i jest wykrywany automatycznie po wstępnym progowaniu obrazu przy udziale użytkownika. Rzeczywiste położenie punktów z szablonu są zapisane w programie, po odczycie współrzędnych ekranowych zidentyfikowanych punktów program używa metody kalibracji kamery przedstawionej w rozdziale 3.6.1 Detekcja przebiega według następującego algorytmu:

Algorytm krok po kroku

K1 Znajdź punkty na obrazie.
Umieść punkty na liście punktów [1 ... 10].
Umieść największy punkt na pierwszej pozycji listy

K2 Szukaj punktu 2,3 - najbliższe punktowi 1 umieść na liście.

K3 Znajdź punkt 4’ - najbliższy do 2 spośród [4 ... 10]

Znajdź punkt 4” najbliższy do 3 pośród [4 ... 10]

K4 **if** $||\overrightarrow{(34'')}|| < ||\overrightarrow{(24')}||$ **then** 4” staje się punktem 4. na liście.
else zamień punkty 2. i 3. miejscami na liście, 4’ staje się 4.

K4 znajdź $X \in [5 \ldots 10]$spełniające warunek:
kąt$(\overrightarrow{(13)}, \overrightarrow{(1X)}) = 0$; dostanę 5.

K5 znajdź $X \in [6,7,8,9,10]$ spełniające warunek:
kąt$(\overrightarrow{(12)}, \overrightarrow{(1X)}) = 0$; dostanę 9.

K6 znajdź $X, Y \in [6,7,8,10]$ spełniające warunek:
kąt$(\overrightarrow{(19)}, \overrightarrow{(5X)} = 0$, kąt$(\overrightarrow{(19)}, \overrightarrow{(5Y)} = 0$; dostanę 6.,7. (Rozpoznane po odległości od 5).

K7 znajdź $X \in [8,10]$X spełniające warunek:
kąt$(\overrightarrow{(15)}, \overrightarrow{(9X)} = 0$, dostanę 8.

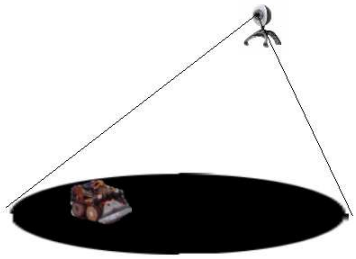
K8 ostatnią wartością jest punkt 10.

3.7 Stanowisko pomiarowe

Badania przeprowadzono na wcześniej zaprojektowanym stanowisku pomiarowym w którego skład wchodzi:

- kamera internetowa,
- komputer z biblioteka OpenCV,
- robot Ukázka klasy mini-sumo,
- dohyo - ring.

Wygląd stanowiska przedstawiono na rysunku 3.7



Rysunek 3.6: Poglądowy rysunek stanowiska pomiarowego.

3.7.1 Kamera internetowa

System wizyjny oparto na powszechnie dostępnej kamerze internetowej Creative Live Cam Vista IM o oznaczeniu 041e:4052 z matrycą CMOS VGA, z maksymalną wydajnością 30kl/s przy rozdzielczości 352x288 pikseli. Zarówno zasilanie jak i komunikacja z komputerem odbywa się przez łącze USB.

Ponieważ jednym z założeń była praca w otwartym środowisku np. Linux, niezbędne było znalezienie odpowiednich sterowników. Baza sterowników do urządzeń wideo pracujących pod otwartym oprogramowaniem nie jest tak tak szeroka jak pod systemem Windows, gdzie przede wszystkim producenci troszczą się o oprogramowanie dla swoich urządzeń. Jednak, sytuacja darmowego oprogramowania wciąż się poprawia i bazy obsługiwanych urządzeń sterowników t.j.: Gspca/Spca5xx – http://mxhaard.free.fr/spca5xx.html , OmniVision – http://www.wifi.com.ar/doc/webcam/ov511cameras.html , stają się coraz większe.

Trzeba zaznaczyć, że firma Creative wyprodukowała wspomnianą kamerę z różnymi chipsetami ale pod tą samą nazwą, dlatego należy zwrócić szczególną uwagę przy wyborze odpowiednich sterowników. Zastosowano zewnętrzne sterowniki OmniVision ov51x [1], pozwalające na pracę kamery w rozdzielczości VGA. przy 25kl/s. Pomimo, że sterowniki umożliwiały pobieranie obrazu z prędkością 25kl/s to biblioteka OpenCV w wersji 1.0.0 nakłada ograniczenie do 15kl/s, a funkcja cvSetCaptureProperty , szerzej opisana w [1], nie daje możliwości zmiany ilości kl/s. Podczas badań wykorzystywano wcześniej nagrane sekwencje w rozdzielczości 352x288 i 30kl/s, natomiast przy pracy w czasie rzeczywistym strumień ma 15kl/s.

3.7.2 Biblioteka OpenCV

Otwarte oprogramowanie OpenCV jest biblioteką funkcji C/C++ stworzoną przez firmę Intel na potrzeby obróbki obrazu w czasie rzeczywistym. Przykładowe zastosowania to HCI (*Human-Computer Interaction*), Identyfikacja obiektów, segmentacja, rozpoznawanie, rozpoznawanie twarzy, gestów, śledzenie ruchu. Więcej szczegółów można znaleźć pod adresami http://opencvlibrary.sourceforge.net/ oraz http://www.intel.com/technology/computing/opencv/ .

Biblioteka jest wciąż rozwijana i modyfikowana dlatego pomiędzy poszczególnymi wersjami mogą istnieć istotne różnice działania poszczególnych funkcji. Co więcej, dokumentacja [1], choć obszerna nie jest pozbawiona błędów, więc sugerowane jest poleganie na obszernych źródłach internetowych.

Wykorzystano komputer klasy PC z procesorem Athlon2k XP, wyposażony w 515MB RAM, system operacyjny Linux Gentoo(2.6.19).

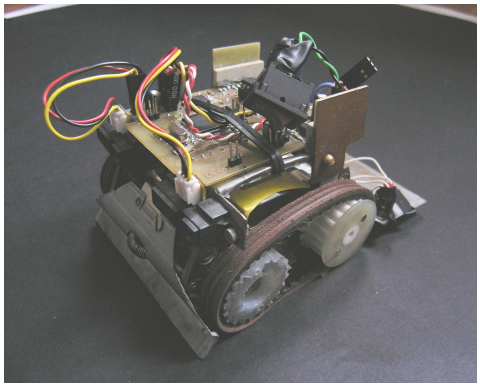
3.7.3 Robot Ukázka

Początkowo badania prowadzone były na obrazach sztucznych i sekwencjach dostępnych w internecie[2].

W ostatecznej formie obiektem zainteresowania systemu wizyjnego stał się robot Ukázka. Jest to autonomiczny robot mini-sumo stworzony przez grupę studentów: Krzysztofa Charustę, Pawła Kunę, Andrzeja Uramka, który odnosił sukcesy w ogólnopolskich zawodach robotów mini-sumo, we Wrocławiu 3., w Gdańsku 1.. Robot porusza się autonomiczne po ringu, orientując się w otoczeniu za pomocą czujników zewnętrznych. Więcej o robocie klasy mini-sumo można przeczytać w pracy [12].

[1] http://www.qbik.ch/usb/devices/showdr.php?id=121

[2] Pokaźny zbiór sekwencji i zdjęć do testowania systemów wizyjnych: http://www.cs.cmu.edu/~cil/v-images.html



Rysunek 3.7: Ukázka – autonomiczny robot minisumo.

Baran

Interesującym obiektem obserwacji okazała się maskotka, z uwaga na swój jednolity kolor stała się wymagającym obiektem obserwacji.

Rysunek 3.8: Baran testowy.

Rozdział 4

Wyniki eksperymentów

Przedstawiony w rozdziale 3 system wizyjny w założeniu miał posłużyć badaniom możliwości wykrywania ruchu i jego parametryzacji. W niniejszej pracy, z powodów wymienionych w rozdziale 2.4 nacisk położono na wykorzystanie metod korelacyjnych, a w szczególności metod bazujących na dopasowywaniu bloków. Przeprowadzono serię testów i badań pozwalających wychwycić istotne właściwości i cechy zaproponowanych metod. Poniżej przestawione zostały scharakteryzowane zdjęcia i sekwencje testowe oraz, w dalszej części przeprowadzone na nich badania i eksperymenty.

W tym rozdziale przyjęto oznaczenia skrótowe metod: FS - pełny przegląd, FS* - pełny przegląd z wczesnym progiem końca, DS - algorytm przeszukiwania karo, PMVFAST - przeszukiwania karo z predykcją i adaptacją.

4.1 Obrazy testowe

Badania prowadzono na specjalnie przygotowanych statycznych obrazach testowych oraz sekwencjach stworzonych przez autora. Poniżej zostaną przedstawione ich charakterystyki bez zdjęć, gdyż w całej pracy można znaleźć przykłady ich wykorzystania wraz ze stosownym opisem. Interesujący i bardzo obszerny zbiór filmów do testowania systemów wizyjnych można znaleźć na stronie `http://www.cs.cmu.edu/~cil/v-images.html` .

4.1.1 Obrazy statyczne

- *ettling* Dwie klatki z sekwencji ettling(skrzyżowanie w mieście Ettling), `http://i21www.ira.uka.de/image_sequences/` . Cechy: różnorodność rozmiarów obiektów od 10 do 70 pikseli, występują naturalne szumy, autobus wykonuje ruch większy niż 7 pikseli, 256x233, skala szarości.

- *onera* Dwa zdjęcia satelitarne huraganu, `http://www.onera.fr/` . Cechy: ruch rotacyjny, 256x256, skala szarości.

- *papryki* Dwie pary sztucznie wygenerowanych obrazków papryk przesuniętych o określone wektory. Cechy: znane translacje, obraz bez szumów. 352x288, skala szarości.

- *punkty* 19 punktów testowych. Dwie pary klatek: z kamery umieszczone nad linią ringu i w rzucie izometrycznym. Cechy: 352x288, kolor.

4.1.2 Sekwencje filmowe

- *ukazka2* Robot Ukázka poruszający się po kołowej trajektorii. Cechy: liczba ramek: 222, 30kl/s, 352x288, skala szarości.

- *ukazka5* Robot Ukázka poruszający się po trajektorii łamanej (zwrot w okolicach 70 ramki). Cechy: liczba ramek: 133; 30kl/s, 352x288, kolor, znacznie większe zaszumienie, niż *ukazka2.avi*.

- *baran1* Biały baranek poruszający się po prostej. Cechy: liczba ramek: 208, 30kl/s, 352x288, skala szarości, obiekt właściwie jednolitego koloru.

- *baran2* Biały baranek poruszający się po prostej z zatrzymaniem. Cechy: liczba ramek: 331, 30kl/s, 352x288, skala szarości, obiekt właściwie jednolitego koloru.

4.2 Badanie kalibracji układu kamery

Przeprowadzono badania skuteczności działania wybranej metody identyfikacji układu kamery ze względu na liczbę punktów pomiarowych oraz umiejscowienie kamery nad sceną. Stworzono szablon testowy widoczny na rysunku 3.5a tworzący siatkę punktów o znanych współrzędnych.

a) b)

Rysunek 4.1: a) Obraz z kamery nad linią, b) wyselekcjonowane punkty.

Testy prowadzono dla próby różnej wielkości z kamerą umieszczoną w dwóch pozycjach: nad linią ringu oraz w rzucie izometrycznym. Miarą jakości transformacji był bezwzględny błąd.

Przeprowadzono 5 serii pomiarowych dla każdego z dwóch badanych ustawień kamery, z inną orientacją punktów w każdej z nich. W każdej serii pomiarowej zmieniano wielkość próby od 3 do 19 punktów. Jako serię testową wykorzystano inne punkty o znanych współrzędnych.

Przedstawione poniżej tablice 4.1, 4.2, 4.3, 4.4 przedstawiają błąd bezwzględny wyliczony z różnicy wskazań położenia punktów serii testowej względem rzeczywistego położenia dla zsumowanych wyników z wszystkich pięciu serii. Jak można było przewidzieć, dokładność transformacji rośnie wraz ze zwiększeniem wielkości próby. Co ciekawe, nawet dla niewielkiej próby (n = 3) dokładność dopasowania dla kamery umieszczonej nad krawędzią ringu jest bliska wielkości piksela. Odchylenie standardowe również się zmniejsza jednak jego wartość nie przekracza wielkości piksela. Trzeba pamiętać, że system wizyjny pracuje w rozdzielczości

352x288 i rzeczywisty rozmiar piksela to 0.24*cm* dlatego odchylenie tej wielkości jest uzasadnione. Jak widać w tabeli 4.2 przy 19 próbach system wyznacza długość piksela z bardzo dobrą dokładnością.

	średnia [cm]		odchylenie [cm]	
n	x	y	x	y
3	0.2370	0.2950	0.3043	0.3793
5	0.2142	0.2403	0.2870	0.3299
7	0.1749	0.2016	0.2465	0.2926
10	0.1874	0.1898	0.2599	0.2712
19	0.1634	0.1740	0.2354	0.2498

Tablica 4.1: Średni błąd i odchylenie standardowe dla wszystkich prób. Kamera nad linią końcową.

n	$wyliczona[cm]$	$rzeczywista[cm]$
3	0.2377	0.2356
5	0.2390	0.2356
7	0.2361	0.2356
10	0.2403	0.2356
19	0.2353	0.2356

Tablica 4.2: Wielkość piksela w zależności od wielkości próby. Kamera nad linią.

Badania przeprowadzone dla kamery umieszczonej poza obrysem ringu, pokazały, że uzyskana dokładność jest mniejsza, ale mieści się w rozsądnych granicach jeśli uwzględnić niewielką rozdzielczość kamery. Ten wniosek jest istotny o tyle, że dopuszcza możliwości prowadzenia badań, gdy kamera nie jest umieszczona centralnie nad sceną, co czasami jest trudne do spełnienia.

	średnia [cm]		odchylenie [cm]	
n	x	y	x	y
3	0.1892	0.5023	0.1856	0.4518
5	0.2032	0.3659	0.2450	0.4610
7	0.1877	0.2931	0.2325	0.4221
10	0.2213	0.2677	0.2793	0.3866
19	0.1960	0.2381	0.2547	0.3456

Tablica 4.3: Średni błąd i odchylenie standardowe dla wszystkich prób. Kamera w rzucie izometrycznym.

4.3 Badania podstawowych cechy metody BM

Na początek przystąpiono do sprawdzenie ogólnych własności metody BM. Jak można przeczytać w rozdziale 2.5 tej pracy, trzema parametrami, którymi można wpływać na działanie algorytmu są: wielkość obszaru poszukiwań fragmentu obrazu – w, wielkość makrobloku –

n	$wyliczona[cm]$	$rzeczywista[cm]$
1	0.2845	0.2690
2	0.2784	0.2690
3	0.2734	0.2690
4	0.2754	0.2690
5	0.2695	0.2690

Tablica 4.4: Wielkość piksela w zależności od wielkości próby. Kamera w rzucie izometrycznym.

n (wielkość poszukiwanego fragmentu) oraz gęstość poszukiwań – r, odpowiadająca gęstości ułożenia wektorów ruchu w polu ruchu. Wyniki badań tych parametrów można znaleźć poniżej.

Badania początkowo prowadzone były na statycznych obrazach testowych zarówno sztucznych jak i rzeczywistych. Aby zapewnić najbardziej wiarygodne wyniki wykorzystano metodę pełnego przeglądu(FS) z wczesnym progiem zakończenia.

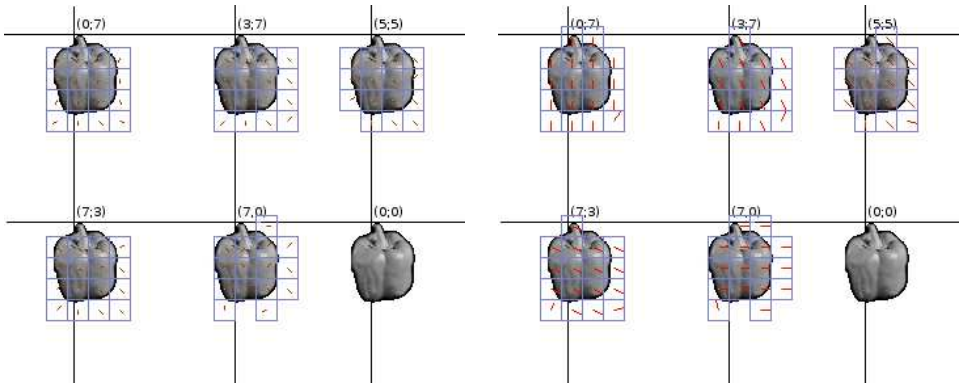
4.3.1 Wielkość obszaru poszukiwań

Z tym parametrem powiązane są dwie wielkości: czas obliczeń oraz prędkość poruszającego się obiektu. Zasadniczo, obszar poszukiwań wpływa bezpośrednio na szybkość algorytmu. Pamiętając, że wielkość okna określana jest wzorem $(2w+1)^2$, można przypuszczać, że nawet niewielkie zmniejszenie okna przyspiesza działanie. W eksperymentach zmieniano parametr w w zakresie od $(2\ldots15)$ i mierzono uzyskiwaną prędkość działania. Pozostałe parametry $r = n = 5$ nie były zmieniane. Tablica 4.5 pokazuje uzyskane wyniki.

Metoda	FS*					
w[pix]	2	3	4	5	7	15
sekwencja	ukazka2.avi					
kl/s	36	44.4	36.6	31	27.4	13.8
sekwencja.	baran2.avi					
kl/s	41.5	41	34.1	30.3	29.5	17

Tablica 4.5: Ilość uzyskiwanych klatek na sekundę w zależności od obszaru poszukiwań.

W ramach tego eksperymentu przeprowadzono również pomiary wielkości średniej i maksymalnej wektora kierunku dla obiektu. Wynosiły one odpowiednio 1.88 oraz 2.44 piksela dla sekwencji *ukazka2.avi* oraz 1.93 i 2.88 piksela dla sekwencji *baran1.avi.* Widać więc, że wektory ruchu są centralnie zgrupowane i w zasadzie niewielkie okno poszukiwań jest wystarczające. Problem pojawia się przy zmniejszonej liczbie klatek. Przeprowadzono eksperyment polegający na eliminacji ramek z sekwencji. Eliminując co drugą ramkę obrazu zmniejszano częstość próbkowania sygnału w dziedzinie czasu, co jest jednoznaczne ze zwiększeniem odległości pomiędzy odpowiadającymi sobie fragmentami obrazu. Nie prowadziło to jednak do błędnego wskazania kierunku, co może być związane z wielkością makrobloku, który choć przesuwany w zakresie okna poszukiwań, jednak obejmował szukany fragment obrazu referencyjnego. Dopiero pozostawienie co trzeciej klatki, prowadziło do częściowo błędnej estymacji kierunku ruchu, jest to szczególnie widoczne, kiedy chcemy dokonać predykcji.



a) b)

Rysunek 4.2: Problem niewielkiej liczby klatek na sekundę lub zbyt małego okna poszukiwań. Metoda FS, n = 16, r = 16; a) w = 7, b) w = 15.

4.3.2 Wielkość makrobloku

Wielkość makrobloku była zmieniana w zakresie od 5 do 16 pikseli. Znamienne jest, że wielkość makrobloku pociągała za sobą zwiększenie czasu działania. Pomimo większej dokładności dopasowania i pozbycia się efektu szczelinowego, co widać na sekwencji *baran1.avi*, relatywnie duże makrobloki nie są właściwe, gdyż tracimy kontur obiektu. Wielkość bloków należy przede wszystkim dopasować do wielkości obiektu, tak aby ich liczba przypadająca na obiekt była dostatecznie duża, a dopasowanie dokładne. Eksperymentalnie sprawdzono, że na sekwencjach *ukazka*, gdzie robot był wielkości 40x40 pikseli, zadowalająca wielkość makrobloku to 5 pikseli. Komponenta, złożona z takich bloków, tworzyła na dobrze teksturowanym obiekcie spójny obszar. Wydaje się słuszna teza, że małe bloki są bardziej podatne na zakłócenia obrazu.

Należy zaznaczyć, że w przypadku metod centralnie ukierunkowanych wielkość makrobloku ma znaczenie z uwagi na możliwość wykrycia jedynie lokalnego minimum zwłaszcza w zwykłej metodzie DS.

a) b)

Rysunek 4.3: Wykrywany obiekt w zależności od rozmiaru makrobloku a) n = 5 (istotnie duży efekt szczelinowy), b) n = 20.

4.3.3 Gęstość wektorów

Ten parametr w zasadzie można uznać za pomocniczy i nadmiarowy. Gdy przyjmiemy, że $r = n$ to cały obraz będzie pokryty blokami. Testy pokazały, że nie ma on większego znaczenia, jest istotny jedynie, gdy bloki są relatywnie duże w stosunku do obiektu. W powyższym wypadku można bloki zagęścić aby uzyskać dokładniejsze kontury, prowadzi to jednak do powstawania sylwetek obiektów większych niż rzeczywiste, a wektory na krawędziach mogą wskazywać błędny kierunek co może mieć znaczenie gdy tło nie jest jednolite.

4.3.4 Próg nieczułości w metodzie FS

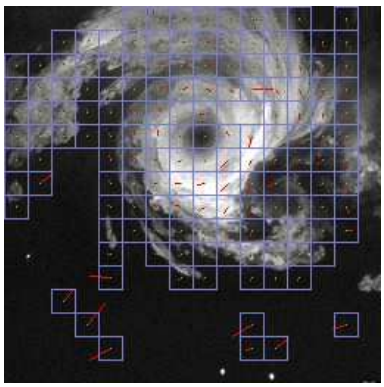
Kolejnym parametrem, który okazał się niezbędny w rzeczywistych zastosowaniach metody dopasowywania bloków, jest próg nieczułości. Jest to wartość określająca o ile mniejsze musi być znalezione minimum od wartości SAD występującej w punkcie $(0,0)$. W momencie kiedy wartość k dana wzorem (4.1) będzie mniejsza od zadanego progu należy przyjąć, że wykryte minimum nie jest dostatecznie „głębokie“ i umieścić wektor w punkcie $(0,0)$.

$$k = |SAD_{(0,0)} - SAD_{min(x,y)}| \tag{4.1}$$

Próby na sztucznym obrazie *ettling* pokazały, że jest to niezbędne w celu pozbycie się szumów. Dobranie poziomy progu jest istotne, a używane sztywno zadeklarowanej wartości mija się z celem. Zarówno wczesne kryterium terminacji, oraz progi adaptacyjne w metodzie PMVFAST są odpowiednikami przedstawionego progu nieczułości.

4.3.5 Ruch rotacyjny

Kolejną cechą wymagającą testów była możliwość wykrywania ruchu rotacyjnego. Jako obraz testowy posłużył huragan z obrazu *onera*. Otrzymane wyniki sugerują, że o jakości wykrytego ruch decyduje wielkość makrobloku i należy go dostosować do wielkości obserwowanego obiektu. Na fotografii 4.4 metoda nie wykrywa ruchu na granicy oka cyklonu z powodu zbyt dużych bloków, natomiast używanie małych makrobloków $n = 5$ niesie za sobą ryzyko złego dopasowania, więc warunkiem koniecznym jest duża rozdzielczość obrazu, aby użycie dużych (>10pikseli) bloków było możliwe. Przepływ optyczny sprowadza wszystko do translacji, więc dalsza obróbka pola wektorowego jest konieczna aby zapobiec wzajemnemu znoszeniu wektorów.



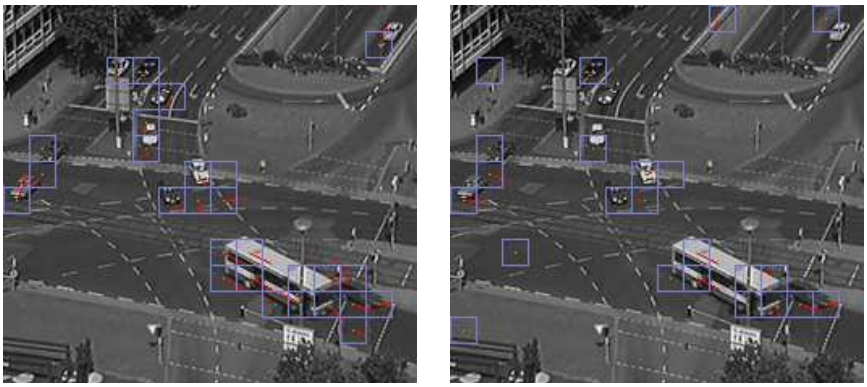
Rysunek 4.4: Ruch rotacyjny. Metoda FS, $n = 16$, $r = 16$, $w = 14$.

4.4 Badania szybkich algorytmów

4.4.1 DS

Zaimplementowano dwa szybkie algorytmy: DS i PMVFAST. Testy na obrazach syntetycznych *papryki* pokazały, że metoda DS działa całkiem poprawnie, a wyniki są porównywalne z pełnym przeglądem. Jednak już testy na obrazach rzeczywistych *ettling* i sekwencjach *ukazka* zweryfikowały powyższą tezę. Podstawową wadą DS jest brak jakiegokolwiek kryterium preferującego punkt $(0,0)$. Zaszumiony obraz rzeczywisty generuje niewielkie ruchy pozorne równoznaczne z lokalnym minimum, osiąganym przez algorytm, gdy brak jasno określonego kierunku wektora. Z uwagi na powyższe wady wypróbowano bardziej wyrafinowanej metody.

4.4.2 PMVFAST



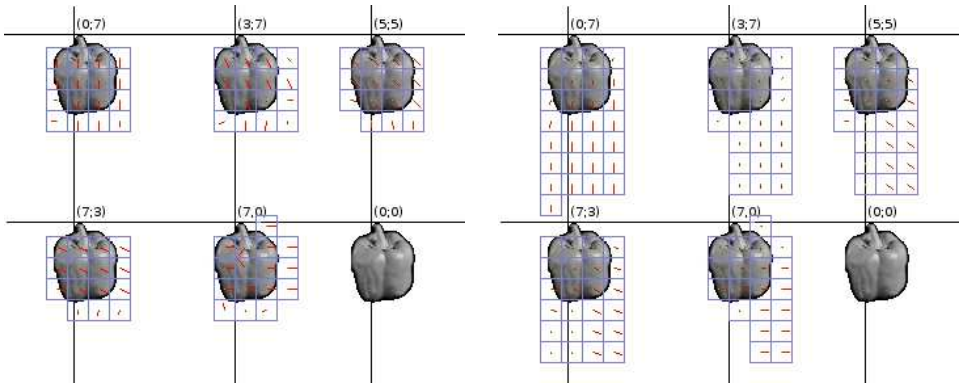
a) b)

Rysunek 4.5: Wykryty ruch w parze obrazów z sekwencji ettling, w = 29, n = 16, r =16. a) Metoda FS b) Metoda PMVFAST

Metoda PMVFAST dawała z kolei gorsze rezultaty na obrazach sztucznych, gdyż wektory z przystających bloków generowały niepożądany ruch dla aktualnie liczonego wektora (rys. 4.6). Jednak z powodzeniem stosowano ją w sekwencjach rzeczywistych. Jak wspomniano, pożądaną cechą jest aby wykrywany obiekt był pokryty w miarę dobrze dopasowaną sylwetką. Próby zmiany wielkości makrobloku z początkowej 16×16 na mniejsze przynosiły marne rezultaty, w postaci niespójnego obszaru i wykrywana błędnego kierunku. Problemem były źle dobrane wartości progów. Ponieważ, zaproponowane przez autorów w [4] i wprowadzone w normie ISO [2], wartości progowe adekwatne są jedynie dla makrobloków wielkości 16×16, zdecydowano się na skalowanie tych wielkości proporcjonalnie do powierzchni makrobloku, co przyniosło pożądany efekt w postaci większej czułości na ruch, a algorytm nie kończył działania przedwcześnie w początkowych fazach.

4.5 Badania szybkości działania algorytmów

Wspomniane algorytmy FS, FS*, DS oraz PMVFAST poddano badaniom efektywności. Ponieważ metody BM miała być narzędziem do zastosowań robotycznych, autorowi nie zależało na powielaniu testów przeprowadzonych w pracach [10, 20, 4, 9, 8, 14]. Postanowiono sprawdzić wydajność metod w systemie wizyjnym opisanym w rozdziale 3.7. Jak już zostało wspomniane, celem było wykrywanie ruchu obiektów i na tej podstawie predykcja zachowań. Postanowiono



a) b)

Rysunek 4.6: Działanie DS i PMVFAST na obrazie syntetycznym. Na rzeczywistych sekwencjach dają całkiem odmienne rezultaty

więc mierzyć czas potrzebny na obliczenia, gdy parametry danej metody będą dawały satysfakcjonujące rezultaty. Przyjęto, za wystarczające, gdy metoda zwracała spójny obszar w sekwencji oraz liczba bloków tworzących była większa niż 10. Wyniki przedstawiono w tabeli 4.6. W metodzie pełnego przeglądu (FS) wykorzystywano makrobloki rozmiaru 5×5 podobnie jak w FS*. DS nie zwracał poprawnych wyników, natomiast w PMVFAST wykorzystano bloki wielkości 16×16 nachodzące na siebie z $r = 15$.

sekwencja	ukazka2.avi			
Metoda	FS	FS*	DS	PMVFAST
kl/s	9.5	27.4	x	36

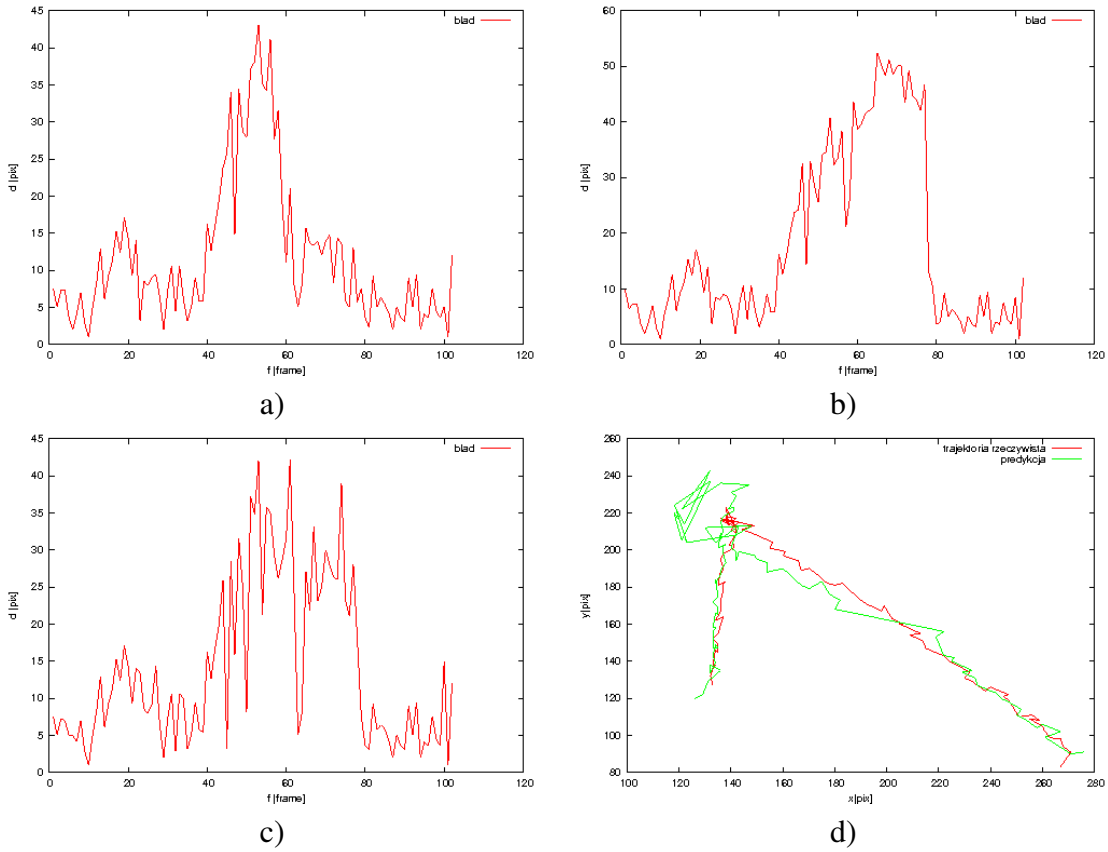
Tablica 4.6: Ilość uzyskiwanych klatek na sekundę w zależności od wykorzystanego algorytmu. Sekwencja *ukazka2.avi*.

4.6 Badanie predykcji

Ponieważ, jedną z ważniejszych cech zaproponowanego systemu wizyjnego jest przewidywanie trajektorii poruszającego się obiektu w zadanym horyzoncie czasowym, przeprowadzono badania możliwości jakie daje stworzony system wizyjny. Do badania wykorzystano dwie sekwencje *ukazka2.avi*, gdzie robot porusza się po okręgu oraz *ukazka5.avi*, z nagłym zwrotem.

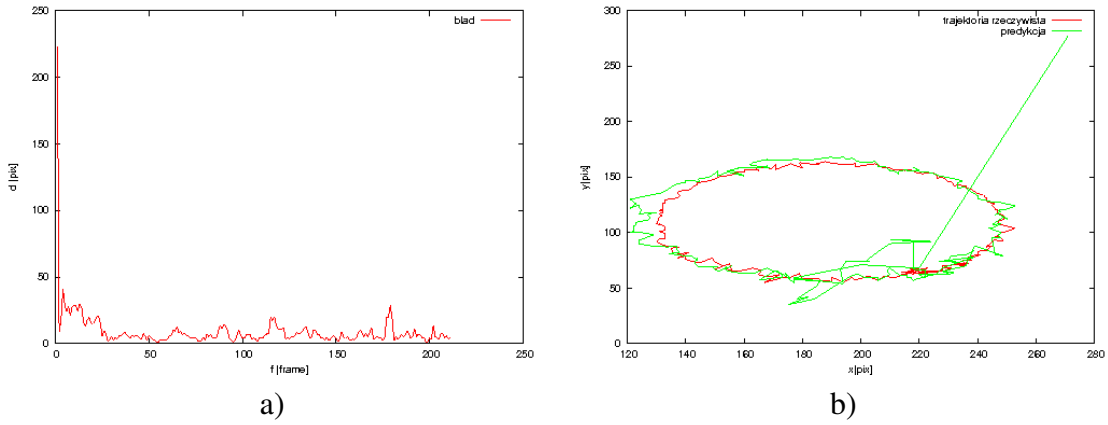
Przeprowadzono badanie poprawności predykcji poprzez pomiar bezwzględnego błędu wskazania pozycji. Wartości pozycji na końcu przewidywanej trajektorii były zapamiętywane i porównywane z rzeczywiście uzyskanymi przez system wizyjny, w ten sposób powstały wykresy 4.7. Na wykresach widać bezwzględny błąd predykcji, który może być bezpośrednio postrzegany jako odległość od przewidywanej pozycji do rzeczywistej. Widać, że w okolicach 60. ramki (odcięto początkowe 10 klatek) obrazu robot zmienia kierunek poruszania się, co natychmiast wpływa na zwiększenie błędu. Widać również, że uśrednianie zapamiętanych zmian kątów daje najbardziej stabilną krzywą błędu, również średni błąd jest najmniejszy i wynosi 11.4 piksela.

Wyliczanie średniej ważonej z historii zmian i z aktualnej wartości dawało porównywalne bądź gorsze rezultaty. Na wykresach 4.7b i 4.7c przedstawiono błędy dla algorytmów bazujących na średniej ważonej, promującej odpowiednio ostatni pomiar oraz dane z wektora historii.



Rysunek 4.7: Błąd predykcji w czasie, dla sekwencji *ukazka5.avi*, horyzont predykcji 10 kl., pamięć zmian 20 kl., metoda liczenia średniej: a)średnia arytmetyczna, b)średnia ważona 2:8, c) średnia ważona 8:2. d) Trajektorie rzeczywista i przewidywana dla powyższego przypadku i średniej arytmetycznej.

Podobne rezultaty otrzymano dla sekwencji *ukazka2*. Należy zaznaczyć, że uzyskane prognozy



Rysunek 4.8: Błąd predykcji dla sekwencji *ukazka2.avi*, horyzont predykcji 10 kl., pamięć zmian 10kl., a)Metoda liczenia średniej średnia ważonej z wagą 0.1 dla ostatniego pomiaru i 0.9 dla historii.

trajektorii mają charakter jedynie zgrubny, jak widać na wykresach, błąd jest dosyć duży, niezależnie od tego czy robot porusza się po stałej trajektorii czy zmienia szybko kierunek. Jest to spowodowane kilkoma czynnikami takimi jak: wysoki poziom szumów utrudniający esty-

mację, niska rozdzielczość kamery, a przede wszystkim, niska rozdzielczość metody. Możliwe również, że bardziej zaawansowane, adaptacyjne metody predykcji dałyby lepsze rezultaty.

Mimo to, predykcja zgrubna może być bardzo przydatna w wielu zastosowaniach, gdzie duża dokładność nie jest konieczna. Pamiętajmy, że obserwowany robot w obiektywie kamery jest wielkości 40 na 40 pikseli co przy średnim bezwzględnym błędzie na poziomie 7.5 piksela dla sekwencji *ukazka2*, przy dobrze dobranych parametrach, jest całkiem dobrym wynikiem. Nawet w przypadku sekwencji *ukazka5* maksymalne odchylenie sięga jedynie 45 pikseli.

W ramach testów sprawdzano również jaki wpływ ma zmniejszona liczba klatek na sekundę na predykcję. W przypadku gdy robot poruszał się po okręgu, zaobserwowano jedynie niewielki wzrost błędu predykcji. Znaczny spadek dokładności odnotowano natomiast na drugiej sekwencji, szczególnie w okolicach zwrotu, najlepszy uzyskany rezultat maksymalnego błędu to 60 pikseli dla dwukrotnie mniejszej częstotliwości próbkowania.

Rozdział 5

Podsumowanie

W niniejszej pracy, w części teoretycznej przedstawiono szerzej metod wykrywania przepływu optycznego wraz z ich charakterystyką. Podstawowe grupy to: metody gradientowe, częstotliwościowe, korelacyjne i oparte na historii ruchu. Przedstawione zostały ich najważniejsze cechy, podstawy teoretyczne i obszary zastosowania. Przegląd literatury miał na celu zapoznanie się z wiedzą w tym zakresie, a następnie wybranie metody, która pozwoliłaby zrealizować założone cele. Po analizie teoretycznej jak i praktycznym porównaniu sposobów estymacji przepływu optycznego, wybrano metodę dopasowywania bloków jako godną uwagi, szczególnie ze względu na: szybkość działania, prostotę implementacji, efektywność i elastyczność.

Wynikiem praktycznej części pracy jest system wizyjny wykonany na potrzeby badań i testów, bazujący na popularnej kamerze internetowej i bibliotece OpenCV. Część badawcza stworzonego systemu pozwala na testowanie wybranych i zaimplementowanych szybkich algorytmów z grupy metod korelacyjnych. Oprogramowanie daje możliwość badania wpływu wszystkich istotnych parametrów zaimplementowanych metod i testowanie ich, zarówno na pojedynczych klatkach obrazu, jak i na sekwencjach. Część użytkowa umożliwia śledzenie obiektów poruszających się po scenie, np. robota minisumo znajdującego się na ringu oraz predykcję jego trajektorii.

Skonstruowany system cechuje:

1. możliwość wykrywania ruchu obiektów,
2. praca w czasie rzeczywistym,
3. możliwość pracy na zaszumionym obrazie w niskiej rozdzielczości,
4. możliwość określania tendencji poruszania się obiektu.

Główną inspiracją do stworzenia wspomnianego systemu była chęć pogłębienia wiedzy z zakresu wizji cyfrowej oraz wyszukanie szybkich metod wykrywania i parametryzacji ruchu, mając na uwadze możliwości późniejszej sprzętowej implementacji. W pracy pokazano, że wykorzystanie metod korelacyjnych na polu robotyki jest opłacalne i dalsze prace w tym kierunku powinny być prowadzone.

Zachęcający jest fakt działania metody dopasowywania bloków w czasie rzeczywistym, co stwarza możliwości zastosowania w szerokiej gamie projektów robotycznych opierających się na wizji. Generalnie, rozsądne jest prowadzenie dalszych ulepszeń systemu wizyjnego w celu poprawienia szybkości działania. Ciekawa wydaje się perspektywa wykorzystania podejścia hierarchicznego wspomnianego w rozdziale 2.3.5. Pociągnęłoby to za sobą dalsze zwiększenie prędkości działania. Wspomniane obserwacje, w perspektywie sprzętowej implementacji systemu wizyjnego na mikrokontrolerach, są bardzo obiecujące.

Niniejsza praca daje szerokie możliwości dalszego badania problemu wykrywania ruchu w sekwencjach obrazu. Jedną ze ścieżek dalszych badań może być zwiększenie poprawności predykcji oraz rozszerzenie jej horyzontu czasowego. Może to prowadzić do bardziej wyrafinowanych zadań, takich jak przewidywanie schematów zachowań.

Poza tym, znając nawet niewielkie tendencje ruchu obiektów, łatwo sobie wyobrazić, że system wizyjny sam zawęża obszar zainteresowania na obrazie i analizuje jedynie fragmenty klatki, na których znalazły się śledzone obiekty. Przyniesie to oszczędności obliczeniowe ale również redukcję ilości transmitowanych danych, co będzie szczególnie istotne jeśli odczyt danych z kamery będzie odbywał się wąskim kanałem transmisyjnym. Należy również pamiętać, że coraz częściej spotykanym standardem w kamerach jest możliwość sprzętowego wyznaczenia regionu zainteresowania, więc rozsądne może być zaprojektowanie algorytmu predykcji właśnie pod taką kamerę, o bardzo dużej rozdzielczości, ale z wybieralnym regionem zainteresowania. Jeśli nawet kamera nie dawałaby takiej możliwości, to przewidywanie położenia obiektu pozwala na optyczne powiększenie danego fragmentu sceny i dogłębną analizę śledzonego celu. Do tego pomocna będzie transformacja układu współrzędnych sceny do układu kamery. Podobnych przykładów zastosowań i rozwinięć tematu można mnożyć wiele.

Podsumowując, wykryty ruch w sekwencji obrazów, jako informacja obrazowa jest bardzo istotny, z szerokimi możliwościami zastosowań w robotyce. Proponuje się dalsze badania metod dopasowywania bloków, w szczególności ich sprzętowe implementacje.

Dodatek A

Oprogramowanie i materiały dodatkowe

A.1 Oprogramowanie

opticalf2_BM

nazwa
`opticalf2_BM` - system wizyjny
synopsis
`$ opticalf2_BM [opcje] <plik_we>`
opis
Program pobiera obraz z kamery lub wskazanej sekwencji i zaczyna wykrywać ruch zgodnie z zadanymi parametrami (możlowość wykonania wcześniejszej kalibracji). Efekty pracy można obserwowac na ekranie, zostaną też zapisane do pliku out.avi.
opcje
`-c` – przechwytywanie z kamery jeśli taka jest dostępna
`-m <metoda>` – wybór metody, dostepne to: FS, FS*, DS, PMVFAST
`-p <r> <w> <n> <pr>` – parametry metody jeśli brak to ładowane standardowe
`-h` – pomoc
`-k` – pierwszym etapem programu będzie auto-kalibracja kamery

opticalf2s

nazwa
`opticalf2s` - testowanie metody na statycznych obrazach
synopsis
`$ opticalf2s <r> <w> <n> <pr> <obraz1> <obraz2>`
opis
Program do testowania metod na obrazach statycznych, wynik zapisywany do pliku out.png, dostępne metody to: FS, DS, FS*, PMVFAST.
opcje
`<r>` – gęstość wektorów
`<w>` – obszar poszukiwań
`<n>` – wielkość makrobloku
`<pr>` – wartość progu wczesnego końca

opticalf_CV

nazwa
opticalf_CV - prezentacja metod dostępnych w bibliotece OpenCV
synopsis
$ opticalf_CVf [opcje] <komenda>
opis
Program do testowania metod z OpenCV(FS*, LK, HS) na sekwencjach wideo, wynik zapisywany do pliku out.avi.
opcje
–

kalibracja

nazwa
kalibracja - kalibracja układu kamery systemu wizyjnego
synopsis
$ kalibracja [opcje] <plik_we> [plik_wy]
opis
Funkcja do testowania kalibracji. Na wejściu oczekuje <plik_we> pliku .png z obrazem, na którym znajdują sie punkty pomiarowe, [plik_wy] to nazwa pliku do którego beda zapisane dane z kalibracji.
opcje
-h – wyświetlenie pomocy
-a – zezwolenie na pracę automatyczną (niezbędny specjalny wzorzec)

histogram

nazwa
histogram - rysuje histogram obrazka
synopsis
$ histogram [opcje] <pd> <pg> <pojemniki> <obraz>
opis
Rysuje histogram jasności obrazka o zadanej nazwie i zapisuje wynik w pliku hist.png. Należy podać niezbędne parametry tj, zakres wartości i ilość pojemminków. Przed obróbką obraz będzie tranformowany do skali szarości.
opcje
-h – wyświetlenie pomocy
<pd> – dolna granica wartośći
<pg> – górna granica wartości
<pojemniki> – liczba pojemników
<obraz> – obraz źródłwy histogramu

histogram_avi

nazwa
histogram_avi - rysuje histogram kolejnych klatek
synopsis
$ histogram_avi [opcje] <pd> <pg> <pojemniki> <sekwencja>
opis

Wczytuje kolejne klatki pliku wideo i wyświetla histogram jasności.
opcje
-h – wyświetlenie pomocy
<pd> – dolna granica wartości
<pg> – górna granica wartości
<pojemniki> – liczba pojemników
<sekwencja> – film z którego będą pobierane klatki i liczony histogram

A.2 Pliki

Makefile histogram histogram_avi kalibracja
opticalf_CV
opticalf2_BM opticalf2s
src/ - pliki źródłowe
inc/ - pliki nagłówkowe
seq/ - pliki sekwencji testowych
ukazka1.avi ukazka2.avi ukazka5.avi baran1.avi baran2.avi
img/ - obrazy testowe
ettling1.png ettling2.png onera1.png onera2.png
papryki_01.png papryki_02.png p19_0.png
p19_1.png p19_2.png p19_3.png
doc/ - literatura w formacie pdf

Bibliografia

[1] *Open Source Computer Vision Library Reference Manual*. Intel Corporation, 2000.

[2] Information technology – coding of audio-visual objects –part 7: Optimised software for mpeg-4 visual tools. *ISO/IEC JTC1/SC29/WG11*, 2001.

[3] S.-C. Cheung A. Gyaourova, C. Kamath. Block matching for object tracking. *Lawrence Livermore National Laboratory*, 2003.

[4] Ming L. Liou Alexis M. Tourapis, Oscar C. Au. Predictive motion vector field adaptive search technique (pmvfast) - - enhancing block based motion estimation.

[5] B.G. Schunc B.K.P. Horn. Determining optical flow. *Artificial Intelligence*, (17):185–203, 1981.

[6] T. Kanade D.B. Lucas. An iterative image registration technique with an application to stereo vision. pages 121–130, 1981.

[7] James W. Davis Gary R. Bradski. Motion segmentation and pose recognition with motion history gradients. *Machine Vision and Applications*, (13):174–184, 2002.

[8] P.I. Hosur and K.K. Ma. Motion vector field adaptive fast motion estimation. In *Second International Conference on Information, Communications and Signal Processing (ICICS '99)*, 1999.

[9] Kai-Ting Cheng Shang-Yu Yeh Wei-Nien Chen Chia-Yang Tsai Tian-Sheuan Chang Hsueh-Ming Hang Hung-Chih Lin, Yu-Jen Wang. Algorithms and dsp implementation of h.264/avc. *IEEE*, (7), 2006.

[10] M. Ranganath A. A. Kassim J. Y. Tham, S. Ranganath. Novel unrestricted center-biased diamond search algorithm for block motion estimation. *IEEE Transaction On Circuits And System for Video Technology*, (8), 1998.

[11] S.S. Beauchemin J.L. Barron, D.J. Fleet. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[12] A. Uramek K. Charusta. Structure and software of autonomous minisumo robot. In *In. Proc. IV Student Scientific Conference(KNS)*, volume 2, 2006.

[13] E. Viarani L. Di Stefano. Vehicle detection and tracking using the block matching algorithm. *Department of Electronics, Computer Science and Systems (DEIS)*.

[14] Yeong-Kang Lai. A memory efficient motion estimator for three step search block-matching algorithm. *IEEE Transaction on Consumer Electronics*, 2001.

[15] Robyn Owens. *Computer Vision IT412, Department of Computer Science*. http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT9/lect9.html , 1997.

[16] Fabian Ernst Harm Peters Patrick Meuwissen, Ramanathan Sethuraman and Rafael Peset Llopis. Segment-based motion estimation using a block-based engin.

[17] Władysław Skarbek(red.). *Multimedia Algorytmy i standardy kompresji*. Akademicka oficyna wydawnicza PLJ, Warszawa, 1998.

[18] J.L. Barron S.S. Beauchemin. The computation of optical flow. *ACM Computing Surveys*, 27(3):433 – 467, 1996.

[19] Marek Wnuk. Notatki do wykładu systemy wizyjne. 2006.

[20] Shan Zhu and Kai-Kuang Ma. A new diamond search algorithm for fast block-matching motion estimation. *IEEE Transaction On Image Processing*, 9(2):287–290, 2000.

Spis treści

Spis rysunków

Spis tablic