POLITECHNIKA WROCŁAWSKA

WYDZIAŁ MECHANICZNY

KIERUNEK: Automatyka i Robotyka

SPECJALNOŚĆ: Systemy Produkcyjne

PRACA DYPLOMOWA

MAGISTERSKA

Opracowanie koncepcji systemu intuicyjnego programowania robotów

TEMAT\_PO\_ANG

AUTOR:

Kamil Kozdrowiecki

PROMOTOR:

Dr inż. Krzysztof Chrapek

W10/I24

OCENA PRACY:

WROCŁAW 2016

Spis treści

[1. Cel i zakres pracy 4](#_Toc451715310)

[2. Założenia projektowe 4](#_Toc451715311)

[3. Teoria stereowizji 5](#_Toc451715312)

[3.1. Obrazowanie punktu z przestrzeni trójwymiarowej 5](#_Toc451715313)

[3.2. Geometria epipolarna 6](#_Toc451715314)

[3.3. Triangulacja w stereowizji 7](#_Toc451715315)

[4. Dobór osprzętu 9](#_Toc451715316)

[4.1. Jednostka obliczeniowa 9](#_Toc451715317)

[4.2. Kamery 9](#_Toc451715318)

[4.3. Biblioteki programowe 10](#_Toc451715319)

[5. Komunikacja jednostka obliczeniowa - robot 10](#_Toc451715320)

[5.1. Stawiane wymagania 10](#_Toc451715321)

[5.2. Protokół TCP/IP 11](#_Toc451715322)

[6. Oprogramowanie jednostki obliczeniowej 11](#_Toc451715323)

[6.1. Struktura 11](#_Toc451715324)

[6.2. Kalibracja kamer 14](#_Toc451715325)

[6.3. Ustawienie parametrów filtrowania 15](#_Toc451715326)

[6.4. Program wykonawczy 16](#_Toc451715327)

[6.5. Klasa CStereoCalibration 16](#_Toc451715328)

[6.6. Klasa CFilterCalibration 16](#_Toc451715329)

[6.7. Klasa CStereoVision 17](#_Toc451715330)

[6.8. Klasa CTCPConnection 17](#_Toc451715331)

[7. Oprogramowanie robota 18](#_Toc451715332)

[7.1. Struktura 18](#_Toc451715333)

[8. Testowanie oprogramowania 19](#_Toc451715334)

[8.1. Kalibracja kamer 19](#_Toc451715335)

[8.2. Filtrowanie 20](#_Toc451715336)

[8.3. Analiza dokładności 24](#_Toc451715337)

[9. Podsumowanie 24](#_Toc451715338)

[10. Bibliografia 24](#_Toc451715339)

[11. Spis ilustracji 24](#_Toc451715340)

[12. Spis tabel 24](#_Toc451715341)

[13. Spis załączników 24](#_Toc451715342)

# Cel i zakres pracy

Coraz większego znaczenia w przemyśle nabierają układy, które w łatwy sposób można zaadaptować do zmian parametrów procesowych wymuszonych przez modyfikacje wytwarzanego przedmiotu, czy też usprawnień w technologii procesu, montażu. Większość takich elastycznych maszyn wymaga od operatora wiedzy i doświadczenia, nabytych przez kosztowne kursy i szkolenia, aby dostosować nastawy lub programy urządzeń nawet do pomniejszych zmian w procesie. Biorąc pod uwagę powyższe tendencje rozwoju technologii, za cel niniejszej pracy magisterskiej postawiono opracowanie intuicyjnego systemu programowania punktów dla robotów przemysłowych wykorzystując stereowizyjny układ kamer oraz wskaźnik laserowy. Wprowadzenie takiego rozwiązania przyspieszy procedurę przystosowania elastycznego systemu wytwórczego do nowych produktów, szczególnie   
w przypadku jednostkowych produkcji. Zakres pracy obejmuje:

* stworzenie oprogramowania, zdolnego do wyznaczenia położenia punktu wskazywanego przez promień lasera na podstawie stereowizji,
* umożliwienie łatwej kalibracji, adaptacji systemu do warunków otoczenia,
* sprzężenie komunikacyjne aplikacji wizyjnej z robotem przemysłowym,
* zaprogramowanie robota do interpretacji odebranych danych i wykonania odpowiedniej czynności na ich podstawie.

# Założenia projektowe

Wprowadzenie robotów do przemysłu znacznie zwiększyło elastyczność automatyzacji produkcji pozwalając na względnie szybkie dostosowanie procesu do elementów technologicznie podobnych. Początkowo roboty wykorzystywano w dużych zakładach globalnych potentatów rynkowych, głównie motoryzacyjnych, do powtarzalnych procesów takich jak spawanie, zgrzewanie, paletyzacja, gdzie wymagana jest zdolność zręcznej manipulacji. Wraz z rozwojem robotyki i technologiami kryjącymi się za nią, powstaje coraz więcej modeli dostosowanych do konkretnych grup zadań i zakresów obciążeń. Jednocześnie presja konkurencji wymusza ciągłe obniżanie cen robotów. Pomimo tych czynników sprzyjających robotyzacji, w mniejszych zakładach niechętnie stosuje się te urządzenia, przeważnie ze względu na wysoki koszt kursów obsługi, konserwacji i programowania robota lub usługi przeprogramowania, co przy częstych modyfikacjach procesu jest nieopłacalne. Systemy ułatwiające programowanie robotów dają możliwość szybkiego, intuicyjnego określania punktów, trajektorii i akcji przez operatora, który nie musi posiadać specjalistycznej wiedzy.

Przegląd artykułów, powiązanych z ułatwieniem programowania robotów przemysłowych, pozwolił wyłonić kilka interesujących rozwiązań dotyczących tej kwestii. Zespół   
z uniwersytetu Tarapaca w Chile opracował metodę zadawania trasy robota spawającego wykorzystującą urządzenie *Kinect*[13], które składa się z dwóch kamer oraz oprogramowania mapującego głębię obrazu oraz wykrywania i śledzenia ludzkiej sylwetki. Punkty w przestrzeni zadawane są poprzez wskazanie dłonią danego miejsca, do którego ma przemieścić się efektor i wykonanie drugą dłonią gestu interpretowanego jako potwierdzenie akcji. Niestety taki układ nie pozwala określić orientacji, dlatego została ona z góry zdefiniowana jako odchylenie 45° od pionowego kierunku, głównie dlatego że jest to optymalny kąt spawania MIG, do którego ten system jest wykorzystywany. Podobną metodę, wykorzystującą gesty dłoni do określania punktu i akcji, opatentowała firma ABB[14].

|  |  |
| --- | --- |
| Zalety:   * możliwość rozwinięcia systemu  o dodatkowe gesty – akcje, * duża dostępność i mały koszt urządzeń, * brak fizycznego kontraktu z robotem podczas uczenia. | Wady:   * brak możliwości, utrudnione określenie orientacji, * przebywanie w strefie roboczej podczas uczenia robota, * mały kąt widzenia urządzenia zmniejsza strefę roboczą robota, * duży wpływ oświetlenia otoczenia na pracę kamer. |

Kolejne rozwiązanie przedstawione zostało w artykule „ Optical Tracking System”[10], gdzie autorzy również wykorzystali stereowizję do określania położenia punktów podczas uczenia robota. Elementem wskazującym jest dioda LED, emitująca światło z zakresu podczerwieni, umieszczona w poręcznym markerze. Kamery wyposażono w optyczny filtr pasmowo-przepustowy odpowiedni do danej długości światła IR-LED. Zredukowano w ten sposób znaczy wpływ światła słonecznego i otoczenia na uzyskane wyniki. Pierwotnie, jednostką przetwarzającą i analizującą obrazy z kamer był komputer jednopłytkowy wyposażony w dwurdzeniowy procesor ARM-A9 o częstotliwości taktowania 1 GHz, jednak jego moc obliczeniowa okazała się niewystarczająca, dlatego w późniejszym etapie wykorzystano komputer klasy PC z czterordzeniowym procesorem Intel i7-4702MQ   
o częstotliwości taktowania 2.2 GHz. Program analizujący obrazy działał w systemie operacyjnym Ubuntu 12.04 i opierał się o bibliotekę programową OpenCV. Badanie dokładności systemu przeprowadzono umieszczając marker w uchwycie robota i porównując położenia uzyskane ze sterownika robota z wyznaczonymi przez układ stereowizyjny.   
W badanym zakresie -212,6 ÷ 278,1 mm największy zarejestrowany błąd ( różnica   
w położeniu) wyniósł 3 mm.

|  |  |
| --- | --- |
| Zalety:   * tanie, łatwo dostępne urządzenia, * wykorzystanie opensource’owej biblioteki programowej OpenCV, * brak fizycznego kontaktu z robotem podczas wskazywania punktów. | Wady:   * przebywanie w strefie roboczej robota podczas wskazywania punktów, * niemożliwe określenie orientacji, * wymagana duża moc obliczeniowa. |

W przeglądzie metod programowania robotów przemysłowych [15] opublikowanym   
w 2012 r. omówiono sposób programowania przez ręczne naprowadzanie robota. Technika ta polega na chwyceniu końcówki robota i poprowadzeniu jej do odpowiedniej pozycji. Zapamiętana może zostać zarówno pozycja końcowa, jak i tor ruchu prowadzący do niej. Metoda wymaga wyposażenia robota w wewnętrzne czujniki momentu siły, bądź skorzystania z zewnętrznych sensorów siły sprzężonych ze sterowaniem napędami robota. Operator prowadząc efektor robota obiema rękami ma utrudnione zadanie wskazywanie akcji do wykonania, np. rozpoczęcie, zatrzymanie rejestrowania toru. Jednym z rozwiązań tego problemu jest wprowadzenie terminalu głosowego przetwarzającego wydane polecenia na zaprogramowane akcje systemu [16].

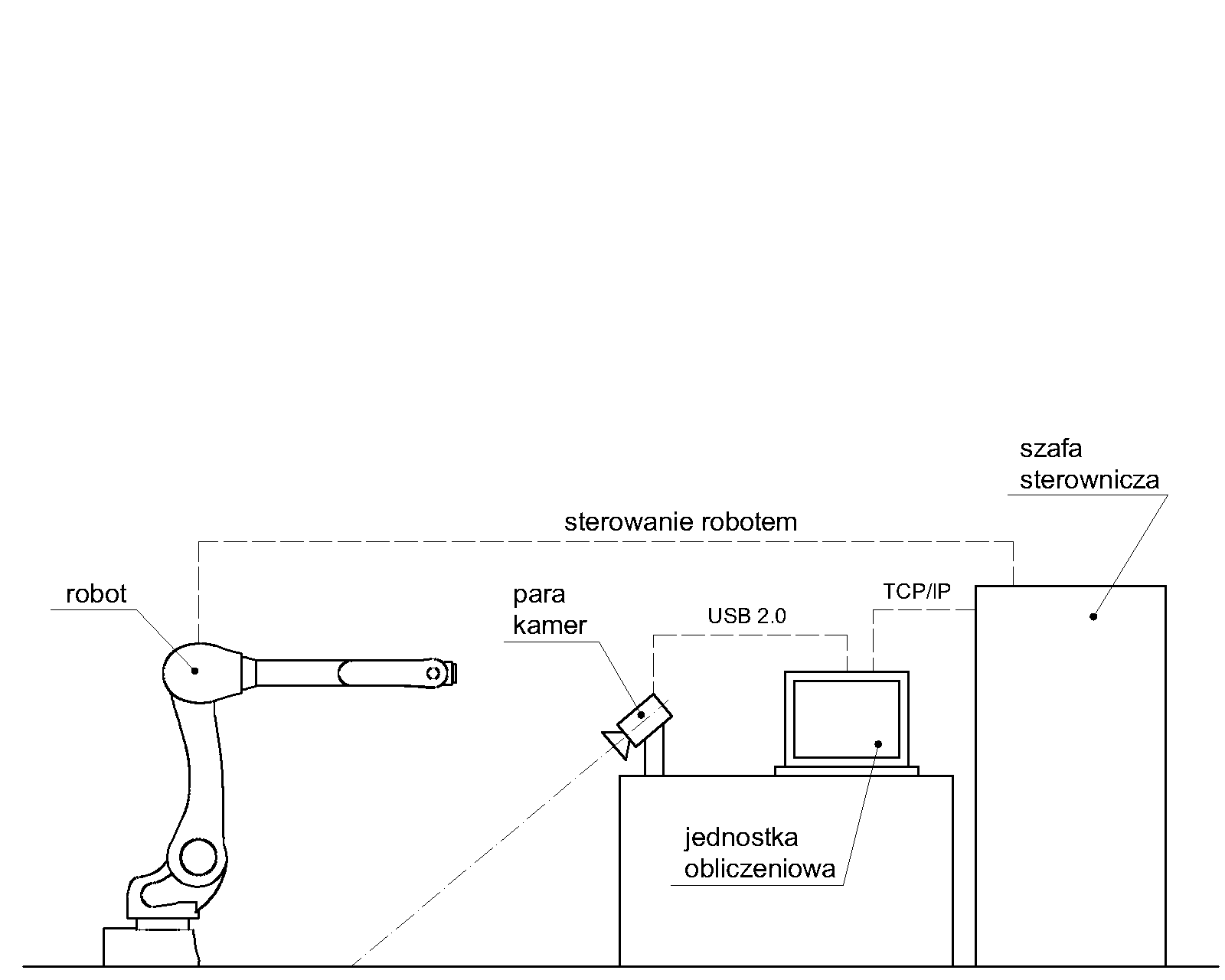
|  |  |
| --- | --- |
| Zalety:   * określenie zarówno położenia jak  i orientacji, * rejestracja toru ruchu do zadanej pozycji. | Wady:   * fizyczny kontakt z robotem. |

Do rozpoznania i analizy położenia zadanego punktu w przestrzeni wykorzystano dwie jednakowe kamery umieszczone w tej samej płaszczyźnie. Wykorzystując technikę stereowizji opisaną w rozdziale 3 możliwe jest wyznaczenie głębi punktu z dwóch obrazów, a co za tym idzie, określenie przestrzennej pozycji danego piksela. Korzyści wynikające z zastosowania tej metody to m. in.:

* względnie ekonomiczne rozwiązanie w porównaniu do innych technik, np. skaner laserowy,
* dostępność wymaganych elementów,
* niewielki rozmiar urządzenia,
* możliwość dostosowania wykrywanych punktów do urządzenia wskazującego przez duży zakres metod filtrowania,
* możliwość wykorzystania pobranych obrazów do innych operacji np. identyfikacja obiektu, wyznaczenie rozmiaru obiektu.

Przyjęto, że urządzeniem wskazującym zadane położenia będzie wskaźnik laserowy   
emitujący czerwone światło. Decyzja ta została podjęta na podstawie kilku czynników:

* ułatwienie procedury filtrowania i wydobycia odpowiednich pikseli, ze względu na dużą jasność punktu i charakterystyczny względem tła kolor,
* możliwość wskazania punktu z daleka, nie będąc bezpośrednio w strefie roboczej robota,
* łatwo dostępne, tanie, kieszonkowe urządzenie wskazujące, nie wymagające dużej mocy zasilania.



Rys. 2.1 Schemat stanowiska

# Teoria stereowizji

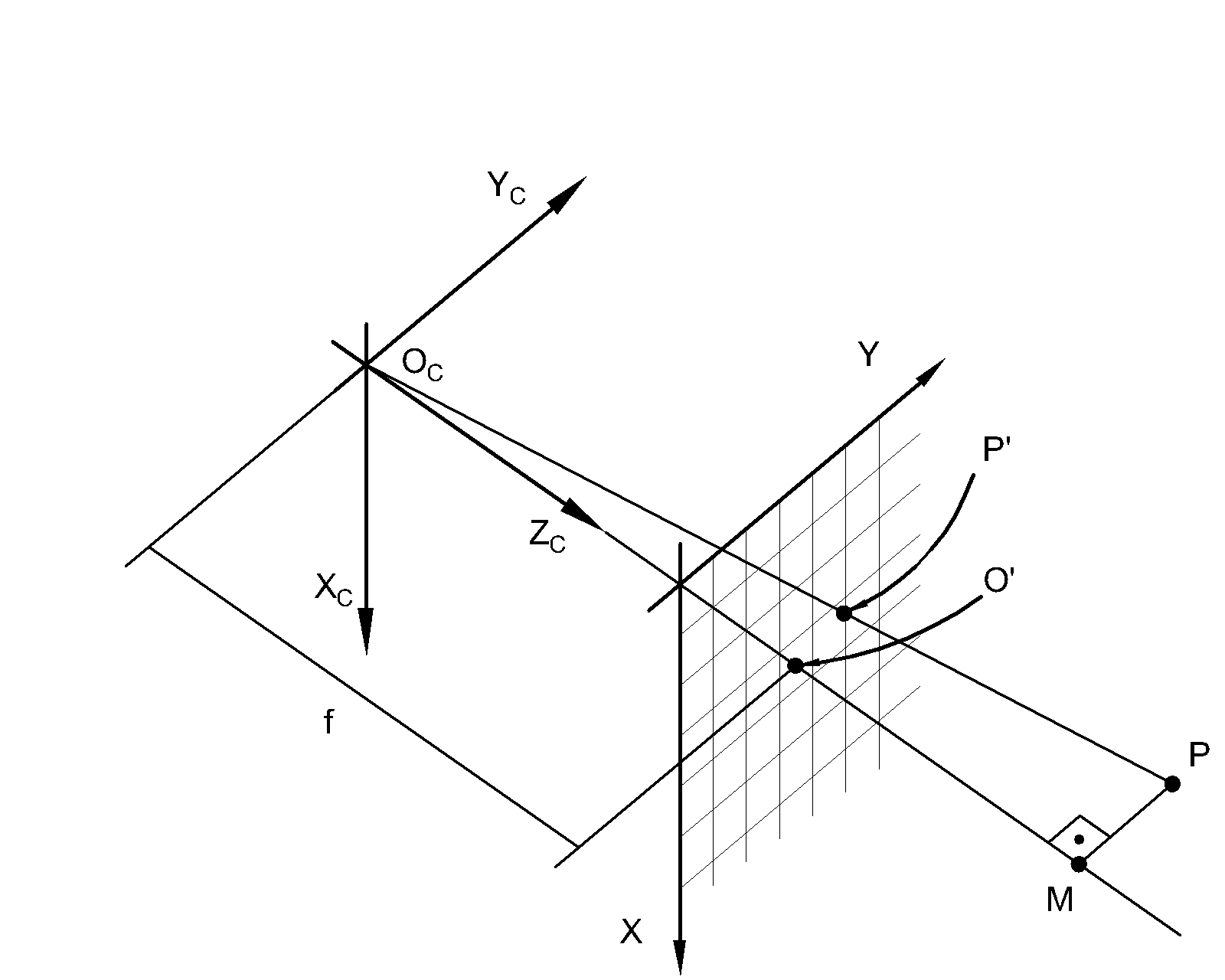
Już jako niemowlę człowiek przyswaja sobie praktyczną umiejętność wykorzystania stereowizji do funkcjonowania w naturalnym środowisku. W biologicznym aspekcie znana jest jako „ widzenie binokularne”. Technika, czerpiąc pomysły garściami z przyrody, wykorzystała również i tę metodę do określania głębi obrazu.

Stereowizja znajduje zastosowanie najczęściej w robotyce mobilnej, gdzie autonomiczne pojazdy wykonują mapę otoczenia w czasie rzeczywistym, aby na bieżąco planować trasę do zadanego celu i omijać ewentualne przeszkody. Szeroko pojęte systemy wizyjne są coraz częściej doceniane, ze względu na ogromną elastyczność wykorzystania w zależności od potrzeb technologicznych. Kamery używane są w takich aplikacjach jak: analiza zużycia ostrza narzędzia, wykrywanie i kategoryzacja defektów, rozpoznawanie pisma ręcznego, śledzenie obiektów, identyfikacja twarzy i w szybko rozwijającej się technologii wirtualnej rzeczywistości.

## Obrazowanie punktu z przestrzeni trójwymiarowej

Układ współrzędnych kamery wyznaczają osie , , oraz punkt centralny . O odległość ogniskową od płaszczyzny przesunięta jest płaszczyzna obrazowa z lokalnym układem współrzędnych i siatką pikselową, której rozmiar elementu podstawowego wynosi . Punkt na płaszczyźnie obrazowej jest rzutem punktu wzdłuż osi . Prosta wyznaczona przez punkt i jest osią optyczną. Zobrazowanie punktu położonego   
w przestrzeni trójwymiarowej polega na perspektywicznym zrzutowaniu tego punktu wzdłuż prostej przechodzącej przez punkt centralny układu kamery i punkt na płaszczyznę obrazową, wyznaczając w ten sposób punkt . Wykorzystując podobieństwo trójkątów i można utworzyć następujące zależności [1]:

|  |  |  |
| --- | --- | --- |
|  |  | (3.1) |



Rys. 3.1 Projekcja punktu w przestrzeni na płaszczyznę obrazową kamery

Biorąc pod uwagę fakt, że rzeczywista kamera posiada niedokładności w postaci błędów wykonania elementów oraz sposobu ich montażu, do równania (3.1) wprowadza się parametry korekcyjne.

|  |  |  |
| --- | --- | --- |
|  |  | (3.2) |
|  |  | (3.3) |

Przekształcając powyższe równania do postaci macierzowej otrzymuje się:

|  |  |  |
| --- | --- | --- |
|  |  | (3.4) |

Macierz określana jest jako macierz kamery i jest dla niej niezmienna. Wartości określają przesunięcie środka płaszczyzny obrazowania względem osi optycznej, wynikające   
z nieidealnego montażu matrycy światłoczułej. Kształt matrycy wymusza wprowadzenie osobnych ogniskowych dla odpowiednich kierunków.

Inną nieprawidłowością powodującą radialne zniekształcenie obrazu nasilające się wraz   
z odległością od osi optycznej jest dystorsja sferyczna. Aby zredukować jej wpływ wprowadza się kolejną korektę w postaci [2]:

|  |  |  |
| --- | --- | --- |
|  |  | (3.5) |
|  |  | (3.6) |

, gdzie:

|  |  |  |
| --- | --- | --- |
|  |  | (3.7) |

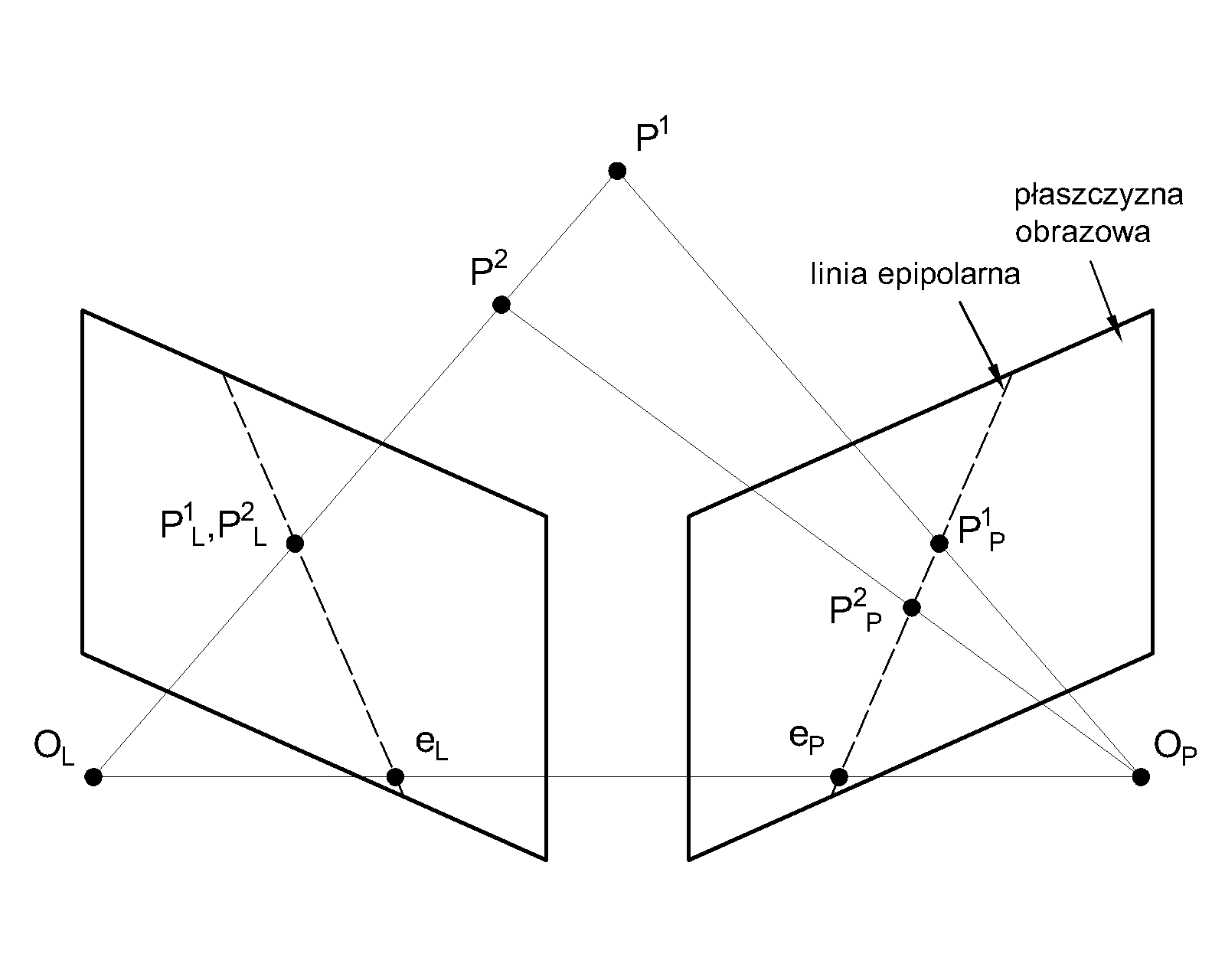
Dystorsja tangencjalna jest kolejnym efektem deformującym obraz, choć w mniejszym stopniu niż dystorsja sferyczna. Wprowadzenie poniższego równania   
minimalizuje skutki zniekształcenia [2]:

|  |  |  |
| --- | --- | --- |
|  |  | (3.8) |

## Geometria epipolarna

Podstawową koncepcją związaną ze stereowizją jest geometria epipolarna. Na rys. 3.2 zaznaczono dwa punkty położone w przestrzeni trójwymiarowej. Rzuty tych punktów na płaszczyzny obrazowań kamery lewej i prawej odpowiadają kolejno punktom . Rzut prostej przechodzącej przez punkt centralny danej kamery i punkt w przestrzeni na płaszczyznę obrazową drugiej kamery nazywamy linią epipolarną. Punkty epipolarne są punktami przebicia płaszczyzn obrazowych kamer przez odcinek łączący punkty centralne kamer. W przypadku kanonicznego układu kamer, gdy płaszczyzny obrazowań obu kamer leżą w tej samej płaszczyźnie, punkty epipolarne znajdują się w nieskończoności.

Używając wyłącznie lewej kamery w sytuacji przedstawionej na rys. 3.2 wiadome jest, że punkty leżą na tej samej prostej przechodzącej przez środek projekcji , jednak określenie odległości każdego punktu od płaszczyzny obrazowej jest niemożliwe.. Aby mieć podstawę do odróżnienia obu punktów i wyznaczenia ich pozycji wymagany jest punkt odniesienia. Odnajdując rzut punktu płaszczyźnie obrazowej prawej kamery można wyznaczyć linię epipolarną, na której musi leżeć również rzut punktu , na bazie którego możemy ocenić różnice w położeniach obu punktów w przestrzeni.



Rys. 3.2 Geometria epipolarna

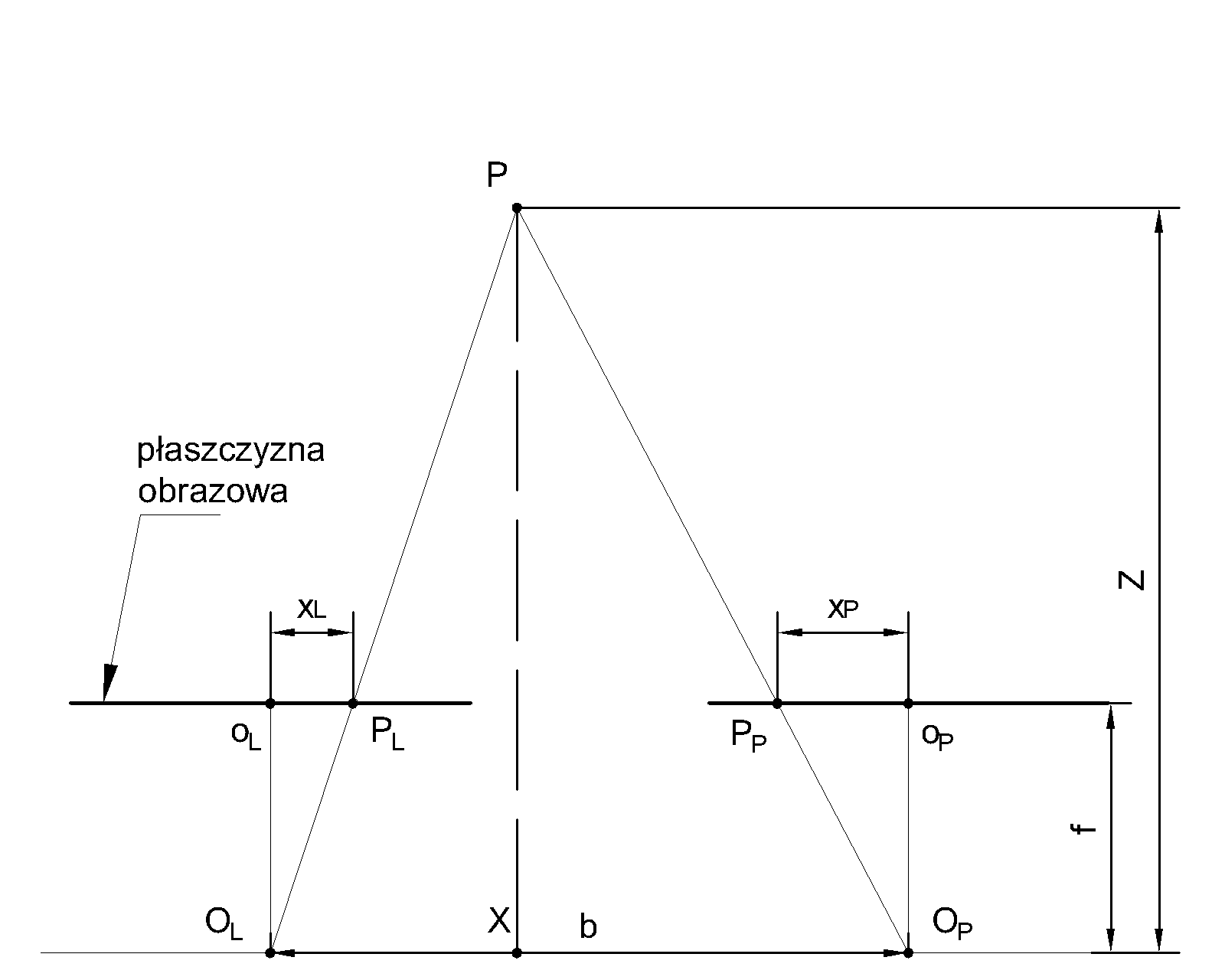
Na podstawie powyższego rys. 3.2 można zauważyć, że każdy punkt obrazowy punktu z przestrzeni trójwymiarowej znajduje się na płaszczyźnie obrazowej wyłącznie na odpowiednich liniach epipolarnych[1]. Jest to jedno z ważniejszych założeń geometrii epipolarnej, dzięki któremu można zredukować przeszukiwanie odpowiadających punktów na obu obrazach z dwóch wymiarów do jednego, znacząco zmniejszając potrzebny czas na obliczenia.

## Triangulacja w stereowizji

Analiza głębi w stereowizji opiera się na wyznaczeniu różnicy w horyzontalnym położeniu danego punktu na obrazach z obu kamer. Wyznaczona głębia danego punktu jest odwrotnie proporcjonalna do uzyskanej różnicy w położeniu horyzontalnym zrzutowanego punktu na płaszczyzny obrazowe. Zależność ta wynika z następującego rozumowania:

Na rys. 3.3 przedstawiono schematycznie projekcję punktu na płaszczyzny obrazowe dwóch kamer w układzie kanonicznym, który zakłada, że:

* płaszczyzny obrazowania kamer są do siebie równoległe,
* osie optyczne kamer są do siebie równoległe,
* odległości ogniskowych kamer są sobie równe.



Rys. 3.3 Triangulacja w układzie kanonicznym

Różnica w położeniu zrzutowanego punktu na płaszczyznach obrazowych kamer wynosi [1]:

|  |  |  |
| --- | --- | --- |
|  |  | (3.9) |
|  |  | (3.10) |

Ze względu na podobieństwo par trójkątów i oraz i :

|  |  |  |
| --- | --- | --- |
|  |  | (3.11) |

Po przekształceniu otrzymuje się:

|  |  |  |
| --- | --- | --- |
|  |  | (3.12) |

, gdzie:

– różnica w horyzontalnym położeniu od środka matryc zobrazowanego punktu

– odległość ogniskowa kamer

– odległość między kamerami

– odległość punktu od kamer

Powyższe równanie jest prawdziwe dla kamer o idealnie równoległych płaszczyznach obrazowania. Oczywiście w praktyce jest to prawie niemożliwe do osiągnięcia, dlatego dodatkowo przekształca się obrazy w procesie rektyfikacji. Polega ona na wirtualnym przesunięci i obrocie obrazów, tak aby płaszczyzny obrazowe obu kamer były na wspólnej płaszczyźnie. W wyniku poprawnej rektyfikacji uzyskuje się równoległe, horyzontalne linie epipolarne przechodzące przez odpowiadające sobie punkty na dwóch obrazach.

# Dobór osprzętu

## Jednostka obliczeniowa

Wstępny projekt uwzględniał wykorzystanie popularnych w ostatnich latach komputerów jednopłytkowych takich jak: Raspberry Pi, Banana Pi, czy BeagleBone. Pomimo niewielkich rozmiarów, posiadają niezbędne do projektu interfejsy 2xUSB i Ethernet. Przegląd podobnych rozwiązań [10], w których również wykorzystano takie jednostki wykazał, że cechują się one za niską mocą obliczeniową w stosunku do złożoności algorytmów przetwarzania obrazów.   
Z tego powodu porzucono te założenia na korzyść komputera PC. Ostatecznie wszystkie testy zostały przeprowadzane na laptopie HP Pavilion dv6, którego główną specyfikacje przedstawiono w tab. 4.1.

Tab. 4.1 Specyfikacja jednostki obliczeniowej

|  |  |
| --- | --- |
| System operacyjny | Windows 7 Home x64 |
| Procesor | 8 x Intel Core i7 2.20 GHz |
| Pamięć | 8 GB |
| Karta graficzna | AMD Radeon HD 6700M |

## Kamery

Dobór kamer był kluczowym elementem przy wstępnych założeniach. Głównymi parametrami, które miały wpływ na wybór urządzenia były:

* rozmiar matrycy,
* maksymalna rozdzielczość,
* liczba klatek na sekundę,
* możliwość konfiguracji parametrów ( ekspozycja, wzmocnienie),
* typ przewodu.

Rozdzielczość obrazu wpływa na dokładność wyznaczanej głębokości punktu ale jednocześnie zwiększa wymaganą moc obliczeniową do przetworzenia większej ilości danych. Z tych względów przyjęto, że kamera będzie pobierać obraz w rozdzielczości 640x480. Istotną i często pomijaną cechą jest możliwość kontrolowania ustawień kamery. Zazwyczaj producenci kamer internetowych implementują do swoich produktów automatyczne dopasowanie wzmocnienia lub ekspozycji, których użytkownik nie może kontrolować i przy pewnym oświetleniu często skutkuje to pojawianiem się szumów. Możliwość sterowania tymi parametrami daje kolejne stopnie swobody np. przy filtrowaniu obrazu. Liczba klatek na sekundę nie jest już tak istotną cechą, ze względu na długi okres przetwarzania pojedynczej klatki, przez co trudno jest wykorzystać całkowity potencjał szybkości pobierania obrazu przez kamerę.

Podczas projektowania i testowania systemu korzystano z kamery *Logitech C905*. Charakteryzuje się możliwą maksymalną rozdzielczością większą od zakładanej, przez co nie wykorzysta się w pełni jej potencjału mając na uwadze ograniczoną moc obliczeniową. Układ optyczny wykonała firma *Carl Zeiss* znana z wytwarzania wielorakiego sprzętu optycznego. Dodatkowo producent udostępnia oprogramowanie umożliwiające ręczny dobór parametrów takich jak: ekspozycja, wzmocnienie, nasycenie Domyślnie kamera pracuje na ustawieniach automatycznych, które zawsze można przywrócić po zmianach.

Tab. 4.2 Specyfikacja kamery

|  |  |
| --- | --- |
| Ilość pikseli | 2 Mpix |
| Maks. rozdzielczość | 1600 x 1200 |
| Liczba klatek na sekundę | 30 |
| Typ przewodu | USB 2.0 |

## Robot przemysłowy

Stereowizyjny system testowany był na robocie przemysłowym firmy Kawasaki model RS010L. Zaliczany jest do grupy robotów szybkich, charakteryzujących się dużą dynamiką, precyzją ruchu oraz smukłą sylwetką. Maksymalny zasięg efektora na wysokości 465 mm wynosi 1925 mm [13]. Ogólne parametry robota przedstawiono w tab. 4.3. Za sterowanie serwomechanizmami w przegubach robota odpowiedzialny jest kontroler E40. Trwała pamięć o pojemności 8 MB pozwala na zapisanie dużej ilości programów i punktów, które mogą być wprowadzane przez kontroler ręczny, bądź z wykorzystaniem jednego z dostępnych interfejsów: USB, RS232C, Ethernet. Dwa ostatnie pozwalają na włączenie robota do sieci komunikacyjnej i tym samym współpracę z innymi urządzeniami.

Tab. 4.3 Specyfikacja techniczna robota Kawasaki RS010L [13]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Waga robota [kg] | 230 |  | Zakres ruchu[°] | |  | Prędkości [°/s] | |
| Maksymalny udźwig [kg] | 10 | JT1 | +/- 180 | JT1 | 190 |
| Maksymalny zasięg [mm] | 1925 | JT2 | +150 ~ -105 | JT2 | 205 |
| Ilość stopni swobody | 6 | JT3 | +150 ~ -163 | JT3 | 10 |
| Powtarzalność [mm] | +/- 0,06 | JT4 | +/- 270 | JT4 | 400 |
|  | | JT5 | +/- 145 | JT5 | 360 |
| JT6 | +/- 360 | JT6 | 610 |

## Biblioteki programowe

Projektowanie i pisanie programów analizujących obrazy na jednostkę centralną opierało się o bibliotekę funkcji OpenCV. Jest to jedna z najpopularniejszych bibliotek programowych służąca do szeroko pojętego przetwarzania obrazów: filtrowania, transformacji, kalibracji, analizy ruchu, wykrywania i śledzenia obiektów. Wiele przykładowych programów, kilka napisanych poradników[5][6], jak i dobrze rozwinięta dokumentacja[7] poszczególnych modułów i funkcji ułatwiają proces zapoznawania się z oprogramowaniem i jego możliwościami. Dodatkowo OpenCV wydane jest na bardzo liberalnej licencji BSD, według której można modyfikować i korzystać z biblioteki zarówno w zakresie edukacyjnym, jak   
i komercyjnym zezwalając na rozprowadzanie napisanych programów w postaci zbudowanej   
i kodu źródłowego, a nawet włączać je do zamkniętego oprogramowania[3].

Główne funkcje biblioteki, wykorzystywane w kalibracji i wyznaczaniu trójwymiarowej reprezentacji punktów zawarte się w module *calib3d*. W czasie projektowania i pisania niniejszej pracy magisterskiej korzystano z wersji OpenCV 3.1.

# Komunikacja jednostka obliczeniowa - robot

## Stawiane wymagania

Wymogi odnoszące się do komunikacji jednostki obliczeniowej z robotem uwzględniają następujące cechy:

* Dostępność portów, złączy – takie same porty lub złącza w sterowniku robota   
  i jednostce obliczeniowej eliminują potrzebę stosowania konwerterów
* Rodzaje obsługiwanych protokołów – sterownik robota nie jest tak elastyczny jak jednostka obliczeniowa i ogranicza zbiór możliwych protokołów komunikacyjnych
* Funkcje programowe – łatwość i przejrzystość funkcji inicjujących komunikację, odbierania i wysyłania danych
* Obsługa błędów – protokół z zaimplementowaną kontrolą danych i ew. retransmisją pozwala uniknąć dodatkowej obsługi błędów na poziomie aplikacji
* Szybkość transmisji danych – mniej istotna cecha, ze względu na okresową, niewielką ilość przesyłanych informacji

Wykorzystywany sterownik robota E40 w wersji podstawowej posiada trzy interfejsy: USB, Ethernet ( 100BASE-TX) i RS232C, z czego USB przeznaczone jest do zapisywania   
i odczytywania danych z nośników zewnętrznych. RS232C jest szeregowym standardem komunikacyjnym, cechującym się prędkością transmisji danych do 20 kb/s na maksymalną odległość 15 m. Pomimo wystarczających parametrów interfejs ten jest zazwyczaj przeznaczony do programowania robota przez terminal komputera PC. Ostatnim możliwym wyborem, co nie znaczy że najgorszym, jest standard Etherner. Technologia 100BASE-TX wykorzystuje jako medium nieekranowaną skrętkę miedzianą zakończoną złączami 8P8C, dla których odpowiadające porty znajdują się obecnie w każdym komputerze. Możliwa prędkość przesyłania danych wynosi 100 Mb/s, przy czym na pewno nie zostanie wykorzystany w pełni ten potencjał.

## Protokół TCP/IP

Uwzględniając stawiane wymagania co do sposobu komunikacji jednostki obliczeniowej   
z robotem zdecydowano się na wybór standardu Ethernet 100BASE-TX. Jest to pierwsza, fizyczna warstwa modelu sieci OSI będąca częścią całej techniki sieciowej Ethernet   
i określająca medium oraz interfejsy sieciowe. W przypadku trzeciej warstwy, sieciowej, zarządzającej połączeniem na poziomie jego ustanawiania, utrzymywania i rozłączania oraz dobierającej optymalną trasę dla danych, nie ma większego wyboru protokołu poza najpopularniejszym IPv4 identyfikującym adresatów po adresie IP składającym się z czterech, ośmiobitowych liczb oddzielanych kropkami. Do kolejnej, wyższej warstwy transportowej zalicza się protokoły TCP lub UDP. Wysyłane paczki danych przekazywane są przez pośrednie węzły sieci, często o różnych przepustowościach, opóźnieniach, czy topologiach. Te różnice mogą wpływać na opóźnienie w dostarczeniu danych, jak również powodować złą kolejność otrzymywanych pakietów danych, bądź ich utratę. Protokół TCP został tak opracowany, aby zapewnić dostarczenie danych w niezmienionej formie do adresata. Korzysta przy tym z takich procedur jak: kontrola przepływu, potwierdzenie odbioru i retransmisja. Gwarancja odebrania identycznej informacji jak przy nadawaniu może wiązać się z kosztem zwiększonego całkowitego czasu przesyłania informacji. Drugi protokół warstwy transportowej skupia się na dzieleniu całej wiadomości na mniejsze, podstawowe fragmenty określane jako datagramy   
i strumieniowym wysyłaniu do adresata. Ponadto sam adresat nie musi wcześniej zaakceptować połączenia, tak jak w przypadku TCP, by odbierać informacje. Umożliwia to nadawanie wiadomości do wielu odbiorców równocześnie, tzw. broadcasting. Istotną cechą protokołu UDP odróżniającą go od TCP jest brak mechanizmów potwierdzenia i retransmisji datagramów w przypadku wadliwego odbioru, co oznacza, że jeśli chcemy mieć pewność odbioru oryginalnej, niezmienionej informacji należy w wyższej warstwie zaimplementować algorytm kontrolny odbieranych datagramów. Protokół UDP znajduje zastosowanie tam, gdzie wymagana jest duża szybkość transmisji danych np. wideokonferencje, gry sieciowe, strumieniowanie dźwięku[9].

W wykonywanym systemie, informacje o przypuszczalnej, niewielkiej wadze kilkudziesięciu bajtów będą wysyłane okresowo co ~1 s, co jest w stanie zapewnić praktycznie każda sieć, więc protokoły o szybszej transmisji danych nie są wymagane. Z tego względu jako warstwę transportową przyjęto protokół TCP, dzięki któremu można uniknąć implementowania dodatkowych procedur kontrolnych, aby mieć pewność dostarczenia prawidłowych informacji.

# Oprogramowanie jednostki obliczeniowej

## Struktura

W jednostce obliczeniowej zaimplementowano dwa programy:

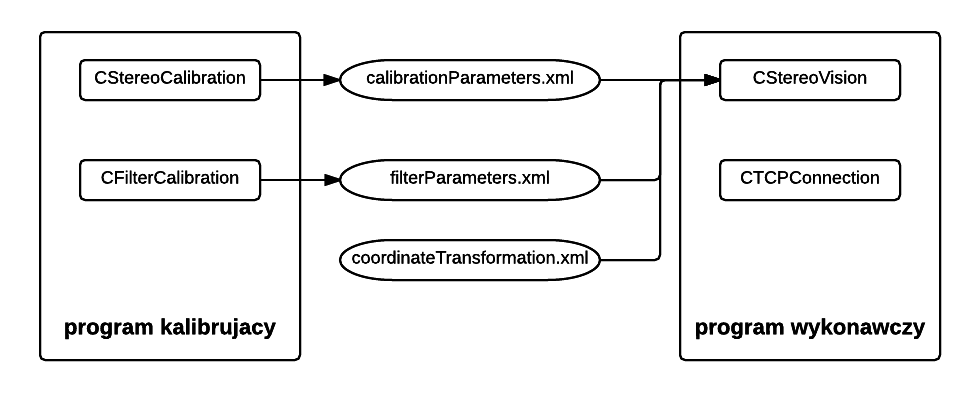
* kalibrujący - z dwiema opcjami:
  + kalibracja kamer – przeprowadzenie procesu kalibracji z wykorzystaniem szachownicy w celu wyznaczenia i zapisania do zewnętrznego pliku współczynników i macierzy niezbędnych do przeprowadzenia poprawnej analizy obrazów w programie wykonawczym,
  + ustawienie parametrów filtrowania – empiryczne wyznaczenie progów filtrowania obrazów w jednej z dwóch metod: RGB lub HSV
* wykonawczy – główny program realizujący ogólną sekwencję: pobranie obrazów, analiza obrazów, wyznaczenie położenia punktu oraz przesłania informacji   
  o punkcie do robota.

Program kalibracyjny opiera się na dwóch klasach: *CFilterCalibration* oraz *CStereoCalibration*, których zadaniami są odpowiednio: przeprowadzenie procedury ustawień filtrowania i procedury kalibracji kamer. Obiekty tych klas tworzone są w zależności od wyboru przez użytkownika początkowej opcji po uruchomieniu programu. Ostatnim i istotnym efektem programów kalibrujących jest wygenerowanie wynikowych plików z zapisanymi parametrami kamer i ustawieniami filtrowania. Służą one jako dane wejściowe do programu wykonawczego. Na diagramie zależności klas od plików ( rys. 6.1) zaznaczono, która klasa tworzy dany plik konfiguracyjny. Jedynym przypadkiem, gdzie użytkownik musi ingerować w dane konfiguracyjne jest plik *coordinateTransformation.xml*, który odpowiada za przekształcenie układu współrzędnych kamery na układ robota.

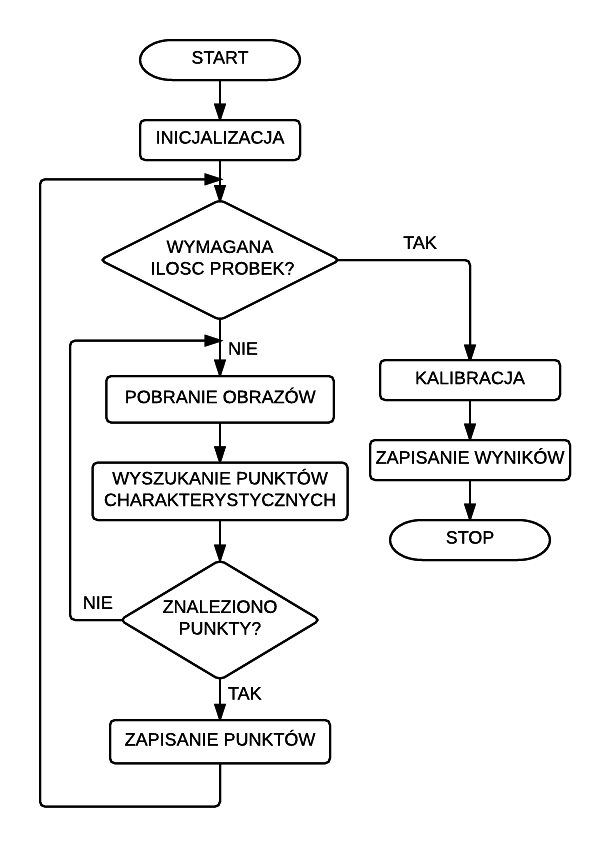
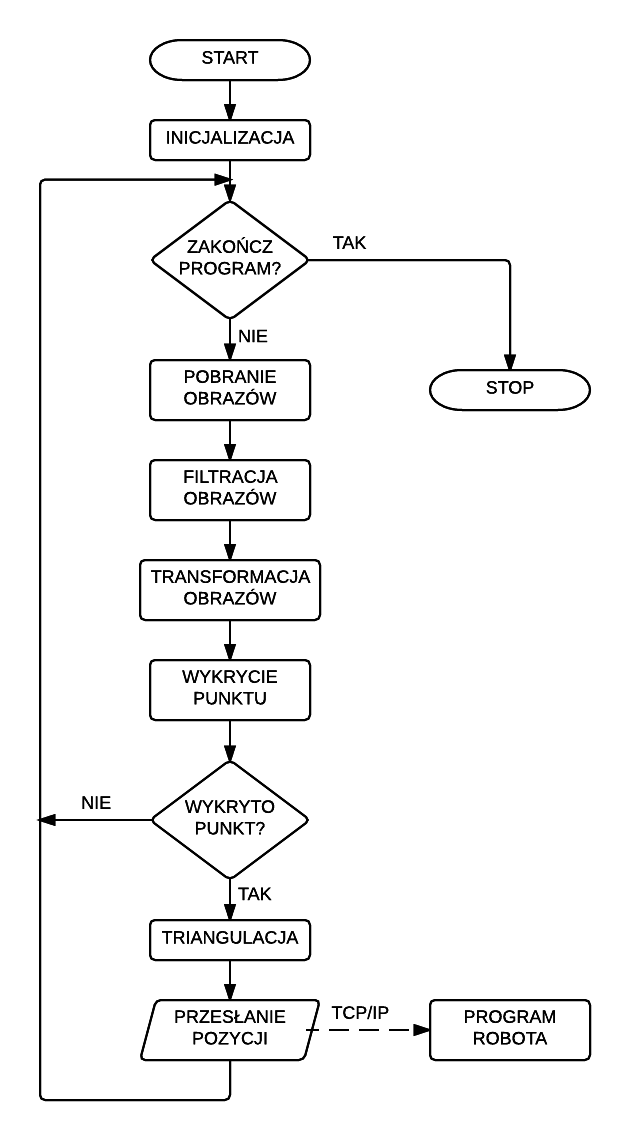
Zasadę działania głównego programu wykonawczego, analizującego położenie punktu wskaźnika opisuje schemat blokowy na rys. 6.2 Podobnie jak w przypadku kalibracji, pierwszym krokiem jest inicjalizacja, podczas której sprawdzany jest dostęp do pliku wynikowego programu kalibrującego, odczytywane są dane i weryfikuje się połączenie   
z kamerami. Następnie pobierana jest para obrazów, która poddawana jest szeregowi operacji:

* filtracji - wyodrębnienie punktu świetlnego wskaźnika od tła,
* transformacji – kompensacja dystorsji obrazów, rektyfikacja,
* wykrycia punktu – uzyskanie położenia punktu wskaźnika z przekształconych wcześniej obrazów.

Jeśli wykryto punkty na obu obrazach to na podstawie ich położeń rekonstruuje się przestrzenne położenie punktu wskaźnika. Uzyskana pozycja przesyłana jest do robota, który podejmuje odpowiednio zaprogramowaną akcję. Szczegółowy opis działania programu wykonawczego opisano w rozdz. 6.4.



Rys. 6.1 Diagram zależności klas i plików konfiguracyjnych



Rys. 6.2 Ogólny schemat blokowy działania programu kalibracji kamer

Rys. 6.3 Ogólny schemat blokowy działania programu wykonawczego

## Kalibracja kamer

Przed przystąpieniem do kalibracji należy zaopatrzyć się we wzorzec, w postaci czarno – białej szachownicy o znanej ilości pól wszerz i wzdłuż oraz długości boku pojedynczego pola. Należy pamiętać, aby wzorzec znajdował się na płaskiej powierzchni. Wszelkie zafalowania, wybrzuszenia powierzchni mogą skutkować nieprawidłowo wyznaczonymi położeniami punktów specyficznych, a co za tym idzie dużym błędem kalibracji.

Ogólna procedura kalibracji kamer przedstawiona została na rys. 6.3. Po uruchomieniu programu kalibrującego prowadzony jest w konsoli dialog z użytkownikiem, w celu wprowadzenia niezbędnych informacji do wykonania kalibracji:

* szerokość i wysokość szachownicy – wartości liczone w ilościach pół szachownicy używanej do znajdowania specyficznych punktów,
* długość boku pola szachownicy – długość w mm pojedynczego pola szachownicy,
* liczba wymaganych próbek – określa minimalną ilość obrazów z wykrytymi punktami charakterystycznymi która jest wymagana przy procedurze kalibracji. Większa liczba próbek podwyższa jakość kalibracji, jednak znacząco wydłuża czas jej wykonywania. Przeważnie wystarczająca wartość to 20.
* numery identyfikacyjne obu kamer – zazwyczaj są to liczby 1 i 2. Jeśli nie można otworzyć którejkolwiek kamery, program wyświetli stosowną informację i zapyta ponownie o numer ID.

Jeśli wszystkie wartości zostaną poprawnie wprowadzone, pojawiają się dwa okna wyświetlające obrazy z obu kamer. W tym momencie należy tak manipulować wzorcem, aby znalazł się on w kadrach obu kamer. Poprawne wykrycie szachownicy sygnalizowane jest zaznaczeniem specyficznych punktów w postaci kolorowych okręgów na obrazach kamer. Jeżeli wykryto punkty na obu obrazach, zapisywane są do bufora położenia tych punktów   
i oznaczane jako kolejna próbka. Przed dodaniem kolejnej próbki występuje dwusekundowa przerwa pozwalająca uniknąć zduplikowania próbki i daje czas użytkownikowi na ustawienie wzorca pod inną orientacją lub położeniem.

|  |  |
| --- | --- |
| Rys. 6.4 Wzorzec w postaci szachownicy | Rys. 6.5 Wykryte punkty charakterystyczne |

Po tym, jak zdobyta będzie wymagana ilość próbek, określona przy inicjalizacji programu, następuje proces analizy otrzymanych danych i obliczenie na tej podstawie parametrów kamer. Procedura trwa kilka minut i trwa tym dłużej im więcej jest próbek. Zakończenie pracy oznajmiane jest stosownym komunikatem w konsoli informującym również o czasie obliczeń   
i orientacyjnej wartości błędu kalibracji. Na zakończenie użytkownik ma możliwość zapisu parametrów do pliku, jeśli jest usatysfakcjonowany małym błędem, lub odrzucenia wyników kalibracji w przeciwnym wypadku.

## Ustawienie parametrów filtrowania

Pobierane obrazy są konwertowane do postaci cyfrowej według różnych modeli. Jednym   
z nich, często wykorzystywanym w informatyce jest trójwymiarowy układ RGB, gdzie punkt reprezentowany jest przez trzy współrzędne barw przyjmujące wartości od 0 do 255: R – czerwonej, G – zielonej oraz B – niebieskiej. Kombinacja współrzędnych w odpowiednich proporcjach daje dowolny kolor. Innym modelem jest układ HSV, który dzieli się na współrzędne: H – barwa, S – nasycenie oraz V – jasność i jego reprezentacja geometryczna jest przedstawiona w postaci walca. Oba wyżej wymienione sposoby reprezentacji koloru zostały wykorzystane do progowanie obrazu. Jeśli współrzędne punktu w danej przestrzeni barw znajdowały się w przedziale określonych wartości ( progów) to punkt ten przyjmował wartość 255 w jednowymiarowym obrazie:

|  |  |
| --- | --- |
| Rys. 6.6 Układ przestrzeni kolorów RGB | Rys. 6.7 Układ przestrzeni kolorów HSV |

W programie kalibracyjnym dostępna jest druga opcja umożliwiająca ręczny dobór parametrów filtrowania. Po wybraniu odpowiedniej opcji, program pyta o kolejne wymagane informacje:

* numer identyfikacyjny kamery – wymagana jest tylko jedna kamera, która zazwyczaj posiada ID o wartości 1. Jeśli nie można uruchomić kamery, program ponownie pyta o ID.
* metoda filtrowania – określa przestrzeń kolorów, dla której w kolejnym etapie określane są granice progowania obrazu wejściowego. Podaje się liczbę przypisaną do danej przestrzeni: RGB lub HSV.

Wprowadzenie poprawnych danych wejściowych pozwala przejść do etapu ustalania wartości progowania obrazu. Pojawiają się dwa okna:

* *cam* – wyświetla w czasie rzeczywistym niezmieniony obraz pobrany z kamery,
* *filtered* – wyświetla binarny obraz po filtrowaniu; piksele mogą przyjmować jedną z dwóch wartości 0 lub 255.

W oknie *filtered* znajdują się suwaki służące do zmiany wartości progów. Obok każdego z nich wyświetlana jest nazwa progu w zależności od wybranej przestrzeni barw wraz z aktualną wartością. Zmiany wartości przez przesunięcie suwaków są na bieżąco realizowane   
i uwidocznione w oknie wynikowym filtrowania. Rys. 6.8 przedstawia okno *cam* z obrazem wejściowym pobranym przez kamerę, natomiast okno *filtered* na rys. 6.9 uwidacznia efekt filtrowania obrazu wejściowego w przestrzeni HSV na podstawie parametrów dobranych w taki sposób, aby uwidocznić czerwone obiekty.

Jeżeli użytkownik jest usatysfakcjonowany z wybranych parametrów, wciśnięcie dowolnego klawisza przy aktywnym jednym z obu okien powoduje zatrzymanie aktualizacji obrazów i wyświetlenie komunikatu w konsoli o możliwości zapisu parametrów i metody filtrowania do pliku.

|  |  |
| --- | --- |
| Rys. 6.8 Podgląd na obraz z kamery | Rys. 6.9 Okno z nastawionymi progami HSV  i wynikowym obrazem filtrowania |

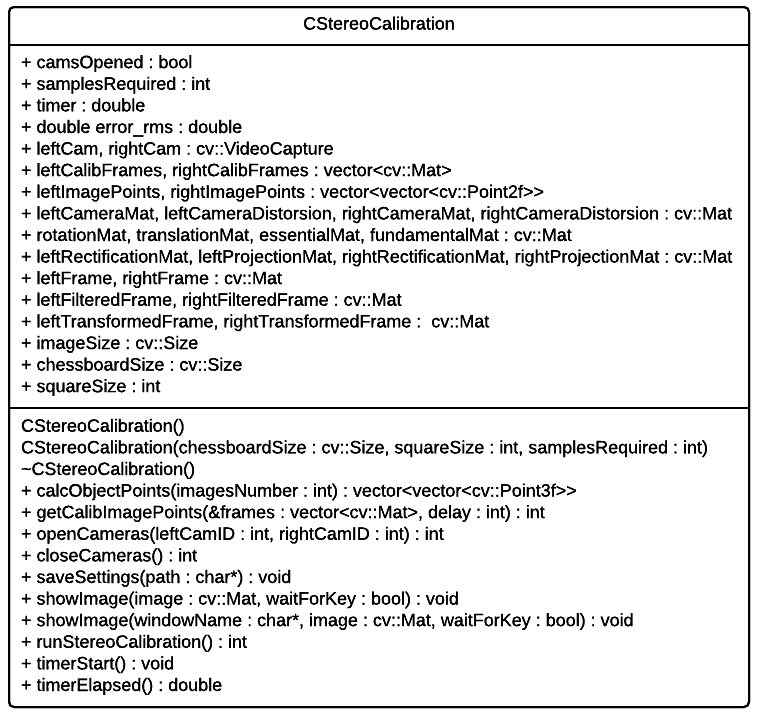
## Program wykonawczy

Zgodnie z ogólnym schematem działania głównego programu przedstawionym na rys. 6.2 pierwszym etapem po uruchomieniu programu jest inicjalizacja. W tym kroku podobnie jak   
w programie kalibrującym, należy podać numery ID kamer. Oprócz tego wczytywane są także dane z plików zewnętrznych, wymagane przy późniejszych procesach. W przypadku braku któregokolwiek z pliku program informuje w konsoli o braku danego pliku i kończy swoje działanie. Drugim etapem jest nawiązanie połączenia z robotem. Prowadzony jest dialog   
z użytkownikiem w konsoli, w celu uzyskania skonfigurowanego adresu IP robota i numeru portu zdefiniowanego w programie robota. Po poprawnym wprowadzeniu wszystkich danych   
i wczytaniu plików konfiguracyjnych program przechodzi do głównej pętli programu.

Pierwszym krokiem jest pobranie z obu kamer obrazów, które następnie zostają skorygowane w procesie transformacji uwzględniającej parametry wczytane z wynikowego pliku kalibracji kamer. W kolejnym kroku skorygowane obrazy zostają poddane filtracji, która wykonuje progowanie na podstawie metody i wartości uzyskanych z pliku ustawień filtrowania. Jeżeli na obu wynikowych, dwustanowych obrazach zostanie wykryty punkt charakterystyczny, funkcja triangulacji wyznacza jego położenie przestrzenne względem układu kamer. Pierwsze dwie próbki są ignorowane, aby dać możliwość operatorowi wskazania laserem odpowiedniego miejsca. Kolejne próbki włączane są do bufora, a po przekroczeniu wymaganej ilości próbek liczona jest średnia wartość położenia, która następnie jest transformowana do układu współrzędnych robota. Położenie punktu dla każdej z trzech osi zamieniane jest na ciąg znaków, który jest przesyłany do robota zgodnie z protokołem TCP/IP, pod warunkiem poprawnie ustanowionego połączenia.

## Klasa CStereoCalibration

Klasa *CStereoCalibration* zajmuje się przeprowadzeniem procesu pobierania odpowiednich próbek obrazów, w celu wyznaczenia współczynników korekcyjnych i macierzy kamer



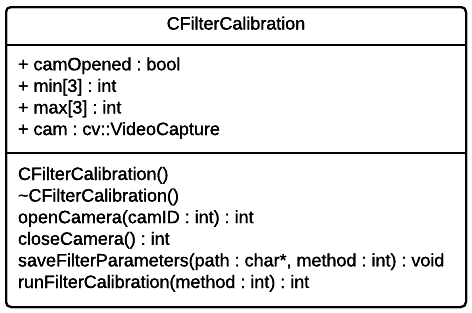
Rys. 6.10 Diagram UML klasy CStereoCalibration

Opis metod zawartych w klasie:

* *calcObjectPoints()* – generuje wektory punktów charakterystycznych szachownicy położonych na płaszczyźnie ( wartości w osi Z równe są 0)   
  z przesunięciami zgodnymi z danymi wprowadzonymi przy opisywaniu szachownicy. Punkty te są odniesieniem do zebranych próbek przy procesie porównywania   
  i wyznaczania współczynników kamer. Jako argument podaje się ilość zebranych próbek dla pojedynczej kamery,
* *getCalibImagePoints()* – funkcja wyszukuje we wprowadzonej parze obrazów punktów charakterystycznych wzorca, wyświetla je w oknach i wpisuje je do bufora próbek. Poza parą obrazów, argumentem funkcji jest czas w sekundach, który musi minąć przed pobraniem nowej próbki ( domyślnie 4 sekundy),
* *openCameras()* – uruchamia kamery o podanych numerach ID, a w przypadku niepowodzenia informuje, której kamery nie można otworzyć,
* *closeCameras()* – zamyka, zwalnia uruchomione wcześniej kamery,
* *saveSettings()* – zapisuje atrybuty obiektu związane z kalibracją kamer do pliku   
  o podanej ścieżce,
* *showImage()* – przeciążona metoda służąca do wyświetlania obrazów, głównie do celów diagnostycznych,
* *runStereoCalibration()* – główna procedura przeprowadzająca proces kalibracji kamer,
* *timerStart()* – uruchamia licznik czasu,
* *timerElapsed()* – procedura zwracająca ilość czasu w sekundach jaki upłynął od uruchomienia licznika czasu – *timerStart().*

## Klasa CFilterCalibration

Klasa *CFilterCalibration* przeprowadza użytkownika przez proces ręcznego doboru parametrów progowania dla wybranej przestrzeni kolorów. Wyznaczone wartości przechowywane są w atrybutach tablicowych *min*, *max* ale równie dobrze mogą zostać zapisane do pliku.



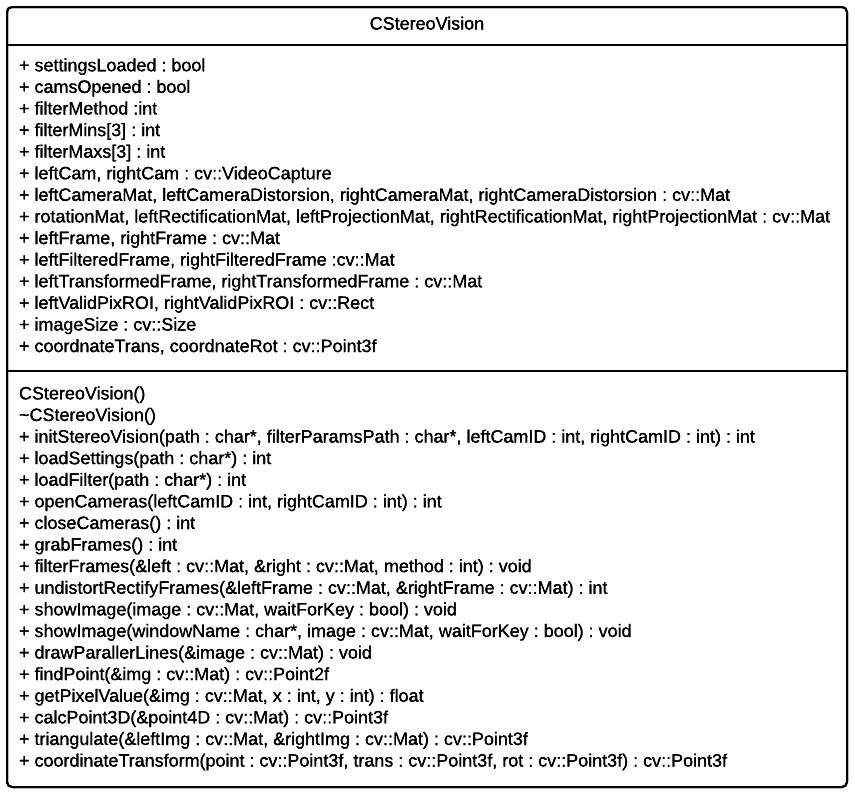
Rys. 6.11 Diagram UML klasy CFilterCalibration

Opis metod zawartych w klasie:

* *openCamera()* – uruchamia kamerę o podanym ID i informuje o ewentualnym niepowodzeniu,
* *closeCamera()* – zamyka, zwalnia wcześniej uruchomioną kamerę
* *saveFilterParameters()* – zapisuje do pliku o podanej ścieżce wartości atrybutów min[], max[] oraz wprowadzoną metodę filtrowania ( dwie możliwości: RGB lub HSV)
* *runFilterCalibration()* – główna metoda przeprowadzająca użytkownika przez proces dobierania wartości progów dla wybranej przestrzeni kolorów ( RGB lub HSV).

## Klasa CStereoVision

Klasa *CStereoVision* odpowiedzialna jest za korekcję i filtrowanie obrazów na podstawie wczytanych parametrów, które przechowywane są w większości atrybutów o typie *cv::Mat*. Zawarte atrybuty o nazwie *frame* przechowują aktualny obraz, który został przekształcony przez procedurę odpowiednią do nazwy tego atrybutu, np. obraz przechowywany w zmiennej *leftFilteredFrame* jest wynikiem filtrowania obrazu pobranego z lewej kamery.



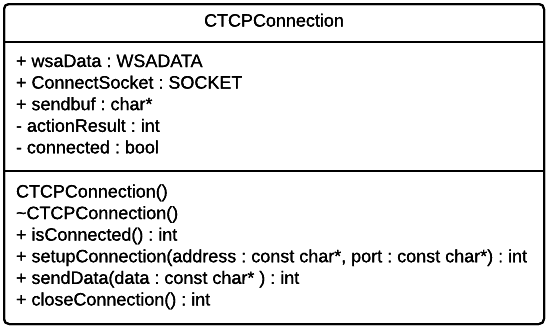
Rys. 6.12 Diagram UML klasy CStereoVision

Opis metod zawartych w klasie:

* *initStereoVision()* – procedura inicjująca, wywołuje funkcje wczytujące poszczególne dane niezbędne dla pozostałych metod klasy. Wymagane jest podanie ścieżek dostępu do plików z parametrami kalibracji i filtracji oraz numerów ID kamer,
* *loadSettings()* – wczytuje dane związane z parametrami kamer z wynikowego pliku kalibracji kamer o podanej ścieżce dostępu,
* *loadFilter()* – wczytuje dane z pliku o podanej ścieżce, wykorzystywane w procesie filtrowania, wygenerowane przez program dobierający progi filtrowania,
* *openCameras()* – uruchamia kamery o podanych numerach ID i informuje   
  o ewentualnym problemie z ich otworzeniem,
* *closeCameras()* – zamyka uruchomione kamery,
* *grabFrames()* – pobiera pojedyncze klatki z obu kamer i przechowuje je w atrybutach *leftFrame, rightFrame*,
* *filterFram*es*() –* proguje wprowadzone obrazy na podstawie wartości atrybutów *filterMins[]*   
  i *filterMaxs[]*  w wybranej przestrzeni kolorów (RGB lub HSV),
* *undistortRectifyFrames()* – transformuje obraz przeprowadzając rektyfikacje   
  i korekcję dystorsji wprowadzonych obrazów na podstawie wartości atrybutów wczytanych przez procedurę *loadSettings(),*
* *showImage()* – przeciążona metoda służąca do wyświetlenia obrazów, głównie do celów diagnostycznych,
* *drawParallelLines()* – rysuje na podanym obrazie horyzontalne, równoodległe od siebie linie. Wykorzystywana jest do oceny rektyfikacji obrazów,
* *findPoint()* – funkcja znajduje geometryczny środek wyznaczonego obszaru na wprowadzonym, binarnym, odfiltrowanym obrazie. Wykorzystywana w metodzie *triangulate()*,
* *getPixelValue() –* funkcja zwracająca wartość piksela o współrzędnych określonych w argumentach na danym obrazie,
* *calcPoint3D() –* funkcja pomocnicza, transformująca dany punkt z układu homogenicznego do układu kartezjańskiego,
* *triangulate() –* funkcja określa położenie względem układu kamer wykrytego punktu   
  w przestrzeni na podstawie wprowadzonych dwóch binarnych obrazów, wykorzystując metodę triangulacji,
* *coordinateTransform() –* przenosi dany punkt z układu współrzędnych kamer na układ robota o wprowadzonej wartości translacji i rotacji.

## Klasa CTCPConnection

Klasa *CTCPConnect* odpowiedzialna jest za obsługę połączenia TCP/IP.



Rys. 6.13 Diagram UML klasy CTCPConnection

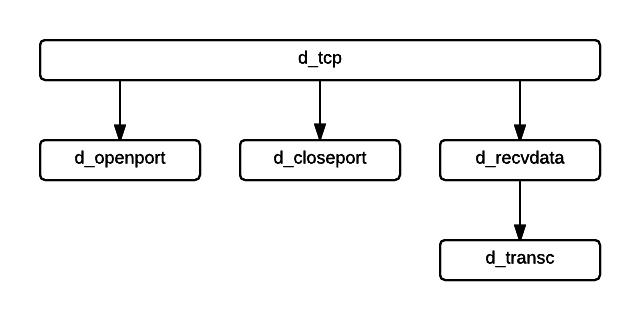
* *isConnected()* – zwraca wartość *true*, jeśli jest aktywne połączenie, w przeciwnym wypadku wartość wynosi *false,*
* *setupConnection() –* nawiązuje połączenie na podstawie wprowadzonego adresu IP   
  i numeru portu. Informuje użytkownika o występowaniu ewentualnych błędów   
  i problemów z ustanowieniem połączenia,
* *sendData() –* wysyła podany ciąg znaków do aktywnego odbiorcy. W przypadku wystąpienia błędów, w konsoli pojawia się stosowna adnotacja,
* *closeConnection() –* zamyka aktywne połączenie.

# Oprogramowanie robota

## Struktura

Oprogramowanie robota składa się z głównego programu *d\_tcp* i 4 podprogramów odpowiedzialnych za:

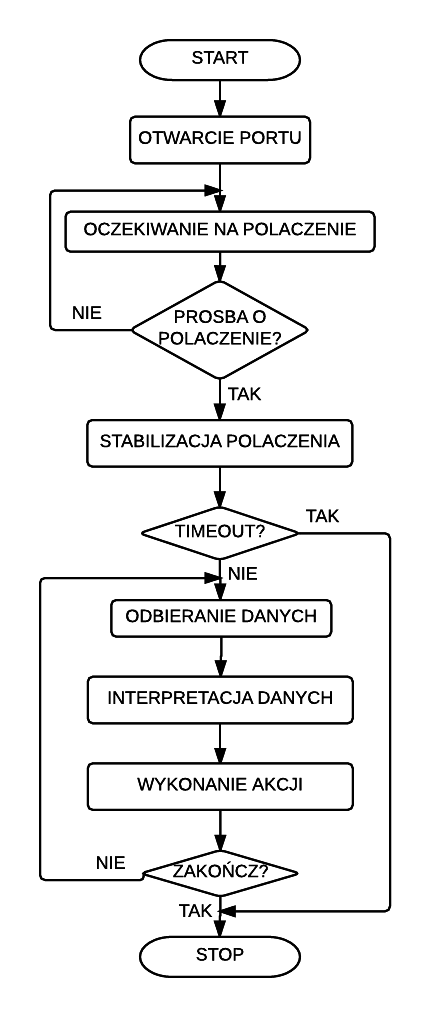
* d\_openport – poprawne otworzenie portu TCP i nasłuchiwania na prośbę połączenia
* d\_closeport – zatrzymanie nasłuchiwania i zamknięcie portu
* d\_recvdata – odbieranie otrzymywanych pakietów danych
* d\_transc – interpretacje otrzymanych danych i wykonanie odpowiedniej akcji



Rys. 7.1 Układ podprogramów programu robota

Po uruchomieniu głównego programu *d\_tcp* od razu następuje przejście do podprogramu *d\_openport.* W nim aktywowana jest funkcja *TCP\_LISTEN* otwierająca wcześniej zdefiniowany port. Następnie w pętli wywoływana jest funkcja *TCP\_ACCEPT* monitorująca przychodzące prośby o połączenie. W przypadku akceptacji połączenia, jest ono stabilizowane z czasowym ograniczeniem, tzw. timeout, po przekroczeniu którego dane połączenie jest odrzucane.

Kolejnym etapem jest odbieranie danych. Odpowiada za to podprogram *d\_recvdata*, który odbiera przychodzące dane i zapisuje je do bufora o pojemności 255 znaków. Jeśli zostanie odebrana cała nadana paczka danych następuje przejście do podprogramu *d\_transc*. Jego zadaniem jest interpretacja informacji zawartych w buforze, na którą składa się odseparowanie odpowiednich ciągów znaków i zamiana ich na wartości liczbowe. Kolejnym krokiem jest wykonanie zaprogramowanych akcji z danymi uzyskanymi z połączenia, które również zawarte jest w *d\_transc*. W czasie wykonywanego działania aktywowany jest sygnał 2002 uniemożliwiający odebranie kolejnych informacji i nadpisanie aktualnych, co skutkowałoby zmianą wykonywanej czynności. Po zakończeniu akcji, w tym przypadku ruchu, sygnał 2002 jest wyłączany, ponownie pozwalając odbierać dane, dopóki nie zostanie aktywowany sygnał 2003 kończący proces nasłuchiwania i zamykający port w podprogramie *d\_closeport*.



Rys. 7.2 Ogólny schemat blokowy działania programu robota

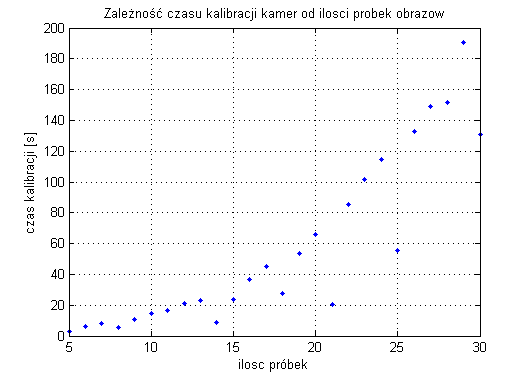
# Testowanie oprogramowania

## Kalibracja kamer

Pozyskiwany z kamer obraz nie jest idealnym dwuwymiarowym rzutem monitorowanej przestrzeni. Głównymi przyczynami zaburzeń obrazu są niedokładności w wykonaniu oraz pozycjonowaniu względem siebie elementów urządzenia. O ile centralny obszar obrazu nie jest zdeformowany, o tyle zewnętrzna część jest tym bardziej zniekształcona, im dalej od centrum obrazu. Skutkami tego może być zaniżona dokładność w wyznaczaniu przestrzennego położenia punktu przez triangulacje, dlatego przed uruchomieniem głównego programu stereowizyjnego należy skrupulatnie przeprowadzić proces kalibracji obrazu.

Kalibracja kamer w układzie stereowizyjnym polega na wyznaczeniu współczynników korekcyjnych omówionych w rozdziale 3. W tym celu wymagane jest pobranie próbek obrazów, na których będzie możliwe wyznaczenie specyficznych punktów, znając jednocześnie rzeczywiste powiązania wymiarowe pomiędzy tymi punktami. Idealnym wzorcem, którego punkty można łatwo wykryć i zmierzyć jest szachownica. Ilość pól wzdłuż i wszerz oraz długość boku pola w mm są niezbędnymi parametrami, które należy wprowadzić jako argumenty do programu kalibrującego. Podczas pobierania próbek obrazu, wyświetlane są obrazy z obu kamer, aby ułatwić poprawną manipulacje szachownicy w kadry kamer. Gdy program wykrywa specyficzne punkty – wierzchołki kwadratów – na obrazie wejściowym,   
w odpowiednich miejscach rysowane są kolorowe okręgi. Pomiędzy pobraniem próbek obrazów występuje odstęp czasowy ( domyślnie dwusekundowy), aby zapobiec zduplikowaniu próbek, co może prowadzić do błędnych wyników kalibracji.

Ilość próbek, na podstawie których przeprowadzane są obliczenia, jest jednym z elementów decydującym o dokładności wyników i długości przetwarzania danych. Na wykresie 8.1 przedstawiono wyniki badanej zależności czasu obliczeń od ilości próbek. Do kalibracji wykorzystana została szachownica o rozmiarze 9x6 i długości boku pola równym 25 mm, co oznacza że każda próbka ( obrazy z dwóch kamer) zawierała 108 wykrytych punktów. Należy mieć na uwadze, że system operacyjny przeznaczał część czasu procesora na inne procesy, przez co wyniki mają zaniżoną dokładność, jednak widoczna jest nieliniowa tendencja wzrostowa.



Rys. 8.1 Wykres zależności czasu kalibracji kamer od ilości próbek obrazu

Wielokrotnie przeprowadzone kalibracje z wykorzystaniem szachownicy 9x6 mieszczącej się na kartce formatu A4 wykazały, że przy rozdzielczości obrazów 640x480 zasięg wykrywania poprawnych punktów wzorca jest ograniczony do około 2 m. Brak próbek z takich odległości może skutkować zaniżoną jakością określania położenia dalekich punktów.

## Filtrowanie

Wyznaczenie efektywnej metody filtrowania i jej parametrów jest jednym   
z najtrudniejszych procesów przetwarzania obrazów. Wydobycie z obrazu obiektów zainteresowania utrudnia wiele czynników, m.in.:

* niejednorodność tła,
* zmienne oświetlenie,
* zmienne parametry kamery,
* szumy.

Niektóre z powyższych problemów można rozwiązać stosując dodatkowe źródło stałego oświetlenia, zapewniające jednakową jasność monitorowanego obszaru lub wykorzystać jednokolorowe maty, podłoża ułatwiające wykrycie i odrzucenie tła. Dopracowany algorytm filtrowania pozwala często uniknąć stosowania dodatkowych procedur pochłaniających wiele zasobów i czasu, mających na celu sprecyzowanie położenia szukanego obiektu. Z tego powodu istotne jest przeprowadzenie poprawnej i dokładnej filtracji na obrazie wejściowym.

Testowana filtracja przebiega na dwóch poziomach. Pierwszym poziomem są wewnętrzne parametry kamery *Logitech C905*, które można zmieniać dzięki oprogramowaniu producenta. Niestety nie można wprowadzać, ani odczytywać wartości liczbowych konkretnych parametrów, a jedynie ustalać ukryte wartości przez pasek przesuwny. Istotne jest, aby ustawienia na obu kamerach były jednakowe, w przeciwnym razie kolejna warstwa filtrowania będzie operować na różnych obrazach wejściowych, co może doprowadzić do wykrycia błędnych punktów, bądź też wyeliminowania poprawnych. Wyróżniono dwie grupy parametrów: ustalone automatycznie przez oprogramowanie kamery oraz ręcznie.

|  |  |
| --- | --- |
| Nastawy automatyczne:  Ekspozycja, wzmocnienie i równowaga bieli dobierane są na podstawie analizy pobranego, aktualnego obrazu. |  |
| Nastawy ręczne:  Wszystkie parametry są dobierane ręcznie i pozostają stałe. |  |

Druga warstwa odnosi się do sposobu zapisu przestrzeni kolorów do postaci cyfrowej. Dostępne są dwie możliwości wyboru układu kolorów, według których obraz wejściowy zostanie przetworzony: RGB lub HSV. Na podstawie koniunkcji trzech parametrów danego modelu kolorów i ich wartości granicznych, tzw. progów, wyznaczany jest binarny obraz wynikowy.

Celem testu filtrowania było stworzenie obrazu z wyodrębnionym obszarem plamki światła lasera o długości 690 nm i mocy 35 mW oraz odrzucenie pozostałych, nieistotnych informacji tła. Przetworzony dwuwymiarowy obraz zawiera piksele, które mogą przyjąć jeden z dwóch stanów: 0 – informacja nieistotna lub 255 – szukana informacja. Po ustawieniu wewnętrznych parametrów kamery, uruchomiono program, który w czasie rzeczywistym pobierał i progował obraz z kamer w wybranej przestrzeni barw. Doświadczalnie dobierano parametry progowania, śledząc na bieżąco wyświetlany obraz wynikowy, tak aby uzyskać wyłącznie obszar wskazany przez światło lasera. Wyniki testowania procedury filtrowania przy parametrach opisanych w tab. 8.1 przedstawiono w tab. 8.2, gdzie czerwonym okręgiem zaznaczono szukany, czerwony punkt.

Dla automatycznych nastaw, pomimo różnych kombinacji zakresu progowania, na żadnym obrazie z obu przestrzeni barw nie udało się całkowicie odrzucić tła. W przypadku obrazu RGB niektóre z białych, jaśniejszych pikseli pozostało na wynikowym obrazie. Progowanie przestrzeni HSV wyeliminowało zdecydowaną większość tła, pozostawiając jedynie kilka pikseli, kosztem utraty dużej części obszaru światła lasera, przez co trudno odróżnić szukany punkt od kilku nieodfiltrowanych pikseli tła.

Druga seria obrazów wejściowych, została pobrana przez kamerę o ręcznie ustawionych parametrach. Zdecydowanie zmniejszono ekspozycję i wzmocnienie, aby zmniejszyć ilość nadmiernie jasnych pikseli. Jednocześnie zwiększono intensywność kolorów, by zintensyfikować widoczność czerwonej plamki światła. Progowanie obrazu wejściowego skutecznie wyeliminowało tło, pozostawiając jedynie piksele odpowiadające laserowi. W przestrzeni HSV utracono nieco więcej informacji, niż w układzie RGB.

Najlepszy rezultat filtrowania został uzyskany stosując ręczne nastawy kamery dla danych warunków otoczenia oraz progując wartości pikseli w przestrzeni RGB, gdzie udało się odfiltrować tło, bez odrzucania pikseli przypadających na światło lasera. Spostrzeżono, że zwiększenie intensywności kolorów w parametrach kamery powoduje okresową destabilizacje wynikowego obrazu progowania. Automatyczne nastawy kamery poprawiają widoczność na obrazie dla ludzkiego oka, jednak nie zawsze wywołują pożądany efekt. Dla otoczenia, na którym testowano filtracje, programowy dobór wzmocnienia i ekspozycji często skutkował uzyskaniem za jasnych pikseli tła, przez co główny czynnik wyróżniający światło lasera od tła – jasność – tracił swój priorytet, a odróżnienie wzmocnionych pikseli od właściwej plamki światła lasera stawało się zdecydowanie utrudnione.

Tab. 8.1 Wartości progów filtrowania

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | B | G | R |  | H | S | V |
| Automatyczne  nastawy kamery | MIN | 0 | 0 | 255 | 7 | 0 | 255 |
| MAX | 210 | 224 | 255 | 26 | 133 | 255 |
| Ręczne  nastawy kamery | MIN | 0 | 0 | 96 | 100 | 0 | 114 |
| MAX | 161 | 162 | 255 | 200 | 255 | 255 |

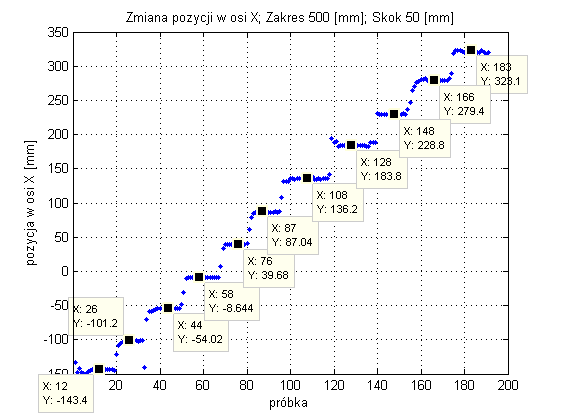
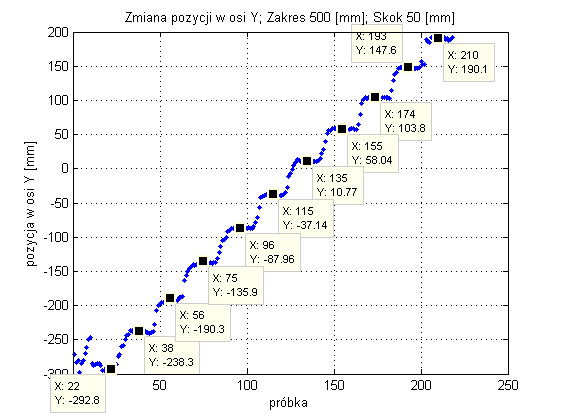
Tab. .2 Wynikowe obrazy testowanych metod filtrowania

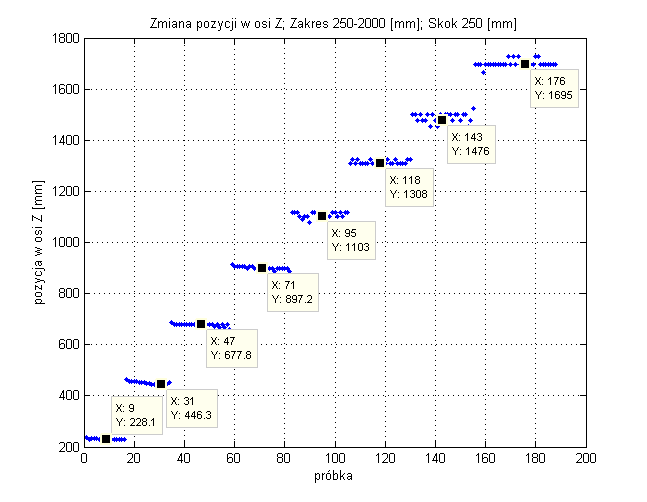
|  |  |  |
| --- | --- | --- |
|  | Automatyczne nastawy kamery | Ręczne nastawy kamery |
| Obraz wejściowy |  |  |
| Progowanie w przestrzeni RGB |  |  |
| Progowanie w przestrzeni HSV |  |  |

Powyższe metody filtrowania zakładają, że plamka światła wskaźnika będzie najintensywniejszym obszarem na monitorowanej scenie. Podczas wielokrotnego testowania oprogramowania filtrującego w różnych warunkach oświetleniowych zauważono, że silne refleksy świetlne np. na gładkich metalowych powierzchniach skutecznie zakłócają procedurę triangulacji, z powodu wielu niewłaściwie wykrytych punktów. Metody progowania w obu przestrzeniach kolorów nie pozwoliły wyeliminować tego efektu.

## Analiza dokładności

Dokładność wyznaczania pozycji punktu w przestrzeni zbadano w trzech testach, osobno dla każdej osi. Pomiar dla osi X i Y przeprowadzono wykorzystując wzorzec, na którym zaznaczono linię o długości 500 mm z punktami co 50 mm. Światło lasera kierowano na zaznaczone punkty i odczekiwano pewien okres czasu na pobranie próbek. Program w sposób ciągły pobierał obraz i wyznaczał położenie punktu, chyba że nie został wykryty. W przypadku badania położenia wzdłuż osi Z nieruchomy promień lasera skierowany był możliwie równolegle do osi Z. Punkty pomiarowe umieszczono na zakresie odległości 250 – 2000 mm od kamer z odstępami co 250 mm. W każdym punkcie pomiarowym stawiano płaski obiekt tak, aby padała na niego wiązka światła, umożliwiając pomiar głębi i zapisanie wyników do pliku.





[KOMENTARZ DO WYNIKÓW]

# Podsumowanie i wnioski

# Bibliografia

1. Bogusław Cyganek, „Komputerowe przetwarzanie obrazów”
2. <http://docs.opencv.org/3.0-beta/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html>
3. <https://pl.wikipedia.org/wiki/Licencja_BSD>
4. D. Rzeszotarski, P. Strumiłło, P. Pełczyński, B. Więcej, A. Lorenc, „System obrazowania stereoskopowego sekwencji scen trójwymiarowych”
5. G. Bradsky, A. Kaehler, „Learning OpenCV”
6. J. Howse, S. Puttemans, Q. Hua, U. Sinha, “OpenCV 3 Blueprints”
7. <http://docs.opencv.org/3.0-beta/modules/refman.html>
8. <https://pl.wikipedia.org/wiki/RS-232>
9. <https://pl.wikipedia.org/wiki/User_Datagram_Protocol>
10. Gernot Korak, Gernot Kucera, art. “ Optical tracking system” w “ INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS”, 2015, VOL. 61
11. Stefano Mattocia, seminarium “ Stereo Visions: Algorithms and Applications”, <http://vision.deis.unibo.it/~smatt/Seminars/StereoVision.pdf>
12. <https://www.astor.com.pl/produkty/robotyzacja/roboty-kawasaki/seria-r-szybkie-roboty.html#jkefel-1-3>
13. “ Kinect-based trajectory teaching for industrial robots”
14. “ Bare hand robot path teaching”, ABB, patent
15. “ Recent progress on programming methods for industrial robots”
16. “ The need for an intuitive teaching method for small and medium enterprises”

# Spis ilustracji

# Spis tabel

# Spis załączników