

POLITECHNIKA WROCŁAWSKA  
WYDZIAŁ MECHANICZNY

---

KIERUNEK: Automatyka i Robotyka

SPECJALNOŚĆ: Systemy Produkcyjne

PRACA DYPLOMOWA  
MAGISTERSKA

Opracowanie koncepcji systemu  
intuicyjnego programowania robotów

Developing the concept of the system  
intuitive robot programming

AUTOR:

Kamil Kozdrowiecki

PROMOTOR:

Dr inż. Krzysztof Chrapek  
W10/K3

OCENA PRACY:

---

WROCŁAW 2016



## SPIS TREŚCI

1.	Cel i zakres pracy .....	4
2.	Przegląd podobnych rozwiązań .....	5
3.	Teoria stereowizji .....	7
3.1.	Obrazowanie punktu z przestrzeni trójwymiarowej.....	7
3.2.	Geometria epipolarna .....	9
3.3.	Triangulacja w stereowizji .....	10
3.4.	Podsumowanie teorii stereowizji .....	12
4.	Konfiguracja stanowiska .....	13
4.1.	Jednostka obliczeniowa.....	13
4.2.	Kamery .....	13
4.3.	Robot przemysłowy.....	14
4.4.	Biblioteki programowe.....	14
4.5.	Układ stanowiska .....	15
5.	Komunikacja jednostka obliczeniowa - robot .....	16
5.1.	Stawiane wymagania.....	16
5.2.	Protokół TCP/IP .....	16
6.	Oprogramowanie jednostki obliczeniowej .....	18
6.1.	Struktura .....	18
6.2.	Kalibracja kamer .....	21
6.3.	Ustawienie parametrów filtrowania .....	23
6.4.	Program wykonawczy .....	25
6.5.	Klasa CStereoCalibration .....	26
6.6.	Klasa CFilterCalibration .....	27
6.7.	Klasa CStereoVision .....	28
6.8.	Klasa CTCPCConnection .....	30
7.	Oprogramowanie robota .....	31
8.	Testowanie oprogramowania.....	33
8.1.	Kalibracja kamer .....	33
8.2.	Filtrowanie .....	34
8.3.	Analiza dokładności .....	38
9.	Możliwości rozwinięcia systemu.....	42
10.	Podsumowanie i wnioski .....	43
11.	Bibliografia.....	45
12.	Spis ilustracji .....	45
13.	Spis tabel.....	46
14.	Spis załączników .....	46

## **1. 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 wysłanych z poziomu aplikacji wizyjnej i wykonania odpowiedniej czynności na ich podstawie.

## 2. PRZEGLĄD PODOBNYCH ROZWIĄZAŃ

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 ciągle 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 wykorzystując urządzenie *Kinect*[1], 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  $45^\circ$  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[2].

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”[3], 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 znaczny 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  $-212,6 \div 278,1$  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 [4] 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 dźwiękowe na zaprogramowane akcje systemu [5].

#### 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 wydobywania odpowiednich pikseli, ze względu na dużą jasność punktu i charakterystyczny względem tła kolor,
- możliwość wskazania punktu z daleka, nie będąc bezpośrednio w strefie działania robota,
- łatwo dostępne, tanie, kieszonkowe urządzenie wskazujące, nie wymagające dużej mocy zasilania.

### 3. TEORIA STEREOWIZJI

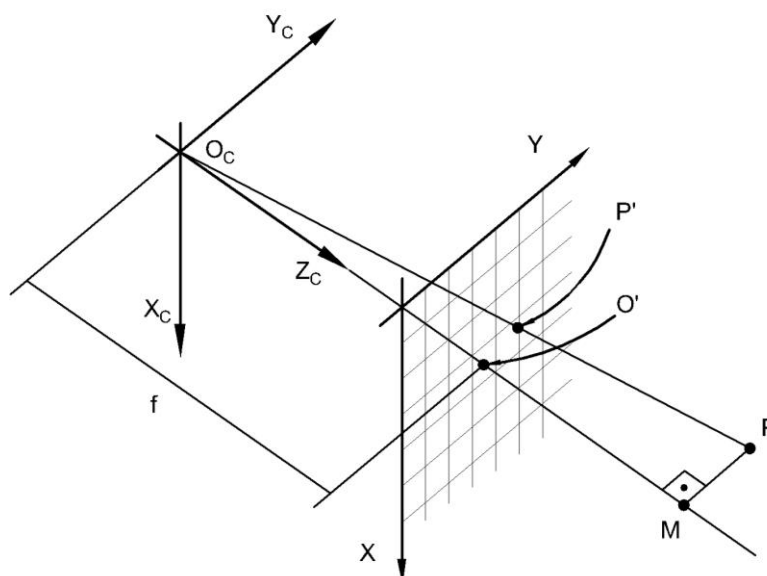
Już jako niemowlę człowiek przyswaja sobie praktyczną umiejętność wykorzystania stereowizji do funkcjonowania w naturalnym środowisku. W biologicznym aspekcie znana jest jako „widzenie binokularne”. Technika, czerpiąc pomysły garściami z przyrody, wykorzystwała również i tę metodę do określania głębi 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[6]. Szeroko pojęte systemy wizyjne są coraz częściej doceniane, ze względu na ogromną elastyczność wykorzystania w zależności od potrzeb operacyjnych. 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.

#### 3.1. OBRAZOWANIE PUNKTU Z PRZESTRZENI TRÓJWYMIAROWEJ

Układ współrzędnych kamery wyznaczają osie  $X_C$ ,  $Y_C$ ,  $Z_C$  oraz punkt centralny  $O_C$ . O odległość ogniskową  $f$  od płaszczyzny  $X_C Y_C$  przesunięta jest płaszczyzna obrazowa z lokalnym układem współrzędnych  $XY$  i siatką pikselową, której rozmiar elementu podstawowego wynosi  $s_x \times s_y$ . Punkt  $O'$  na płaszczyźnie obrazowej jest rzutem punktu  $O_C$  wzdłuż osi  $Z_C$ . Prosta wyznaczona przez punkt  $O_C$  i  $O'$  jest osią optyczną. Zobrazowanie punktu  $P$  położonego w przestrzeni trójwymiarowej polega na perspektywicznym zrzutowaniu tego punktu wzdłuż prostej przechodzącej przez punkt centralny układu kamery  $O_C$  i punkt  $P$  na płaszczyznę obrazową, wyznaczając w ten sposób punkt  $P'$ . Wykorzystując podobieństwo trójkątów  $\Delta O_C O' P'$  i  $\Delta O_C M P$  można utworzyć następujące zależności [7]:

$$\frac{P'_X}{f} = \frac{P_X}{P_Z} \Leftrightarrow P'_X = f \frac{P_X}{P_Z}, \quad P'_Y = f \frac{P_Y}{P_Z} \quad (3.1)$$



Rys. 3.1 Projektacja punktu w przestrzeni na płaszczyznę obrazową kamery – na podstawie[7]

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:

$$P'_X = f_X \frac{P_X}{P_Z} + c_X \quad (3.2)$$

$$P'_Y = f_Y \frac{P_Y}{P_Z} + c_Y \quad (3.3)$$

Przekształcając powyższe równania do postaci macierzowej otrzymuje się:

$$\begin{bmatrix} P'_X \\ P'_Y \\ 1 \end{bmatrix} = \begin{bmatrix} f_X & 0 & c_X \\ 0 & f_Y & c_Y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_X \\ P_Y \\ P_Z \end{bmatrix} = A \begin{bmatrix} P_X \\ P_Y \\ P_Z \end{bmatrix} \quad (3.4)$$

Macierz  $A$  określana jest jako macierz kamery i jest dla niej niezmienna. Wartości  $c_X, c_Y$  określają przesunięcie środka płaszczyzny obrazowania względem osi optycznej, wynikające z nieidealnego montażu matrycy światłoczułej. Kształt matrycy wymusza wprowadzenie osobnych ogniskowych  $f_X, f_Y$  dla odpowiednich kierunków. Powyższe parametry nazywane są parametrami wewnętrznymi kamery.

Niedoskonałości układu optycznego wprowadzają również nieprawidłowości określane jako dystorsja sferyczna objawiająca się radialnymi zniekształceniami obrazu nasilającymi się wraz z odległością od osi optycznej oraz dystorsja tangencjalna. Aby zredukować ich wpływ wprowadza się kolejną korektę w postaci [8]:

$$P''_X = P'_X(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_1 P'_X P'_Y + p_2 (r^2 + 2P'^2_X) \quad (3.5)$$

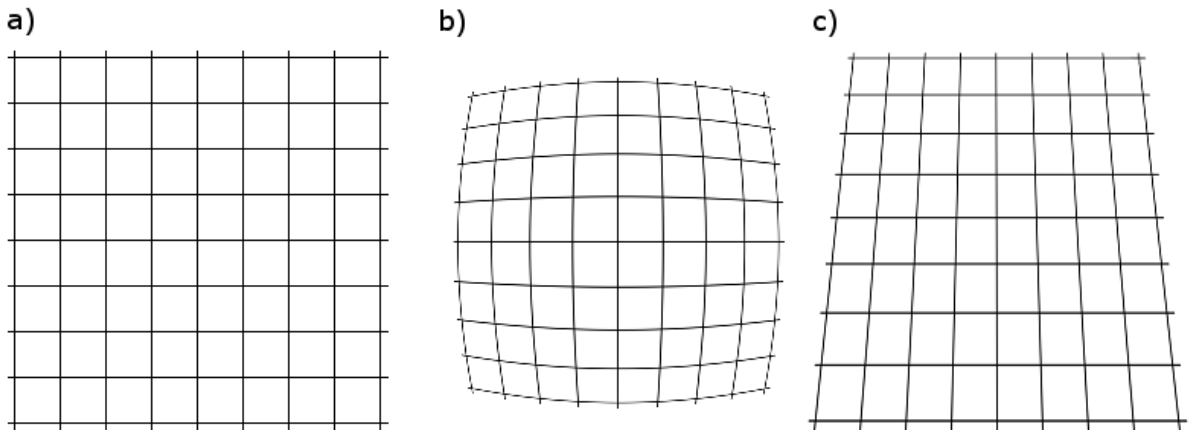
$$P''_Y = P'_Y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2p_2 P'_X P'_Y + p_1 (r^2 + 2P'^2_Y) \quad (3.6)$$

, gdzie:

$$r^2 = P'^2_X + P'^2_Y$$

$k_1, k_2, k_3$  – współczynniki korekcji dystorsji sferycznej

$p_1, p_2$  – współczynniki korekcji dystorsji tangencjalnej



Rys. 3.2 Przedstawienie wpływu zniekształceń:  
a) obraz idealny, b) dystorsja sferyczna, c) dystorsja tangencjalna



Wartość położenia wyznaczonych punktów zależna jest od wybranego układu odniesienia. Zazwyczaj przyjmuje się układ kamery, względem którego opisywane są punkty, jednak w przypadku wielu kamer pozostaje problem określenia ich usytuowania względem siebie. Przekształcenie położenia punktu względem kamery do zewnętrznego układu odniesienia opisują zewnętrzne parametry kamery, na które składają się wektor translacji i macierz rotacji[7]:

$$R = \begin{bmatrix} r_{11} & r_{21} & r_{31} \\ r_{21} & r_{22} & r_{32} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad T = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \quad (3.7)$$

Przesunięcie układu kamery do układu zewnętrznego po odpowiednich osiach wyraża wektor translacji T, natomiast macierz rotacji określa orientację odpowiadających sobie osi.

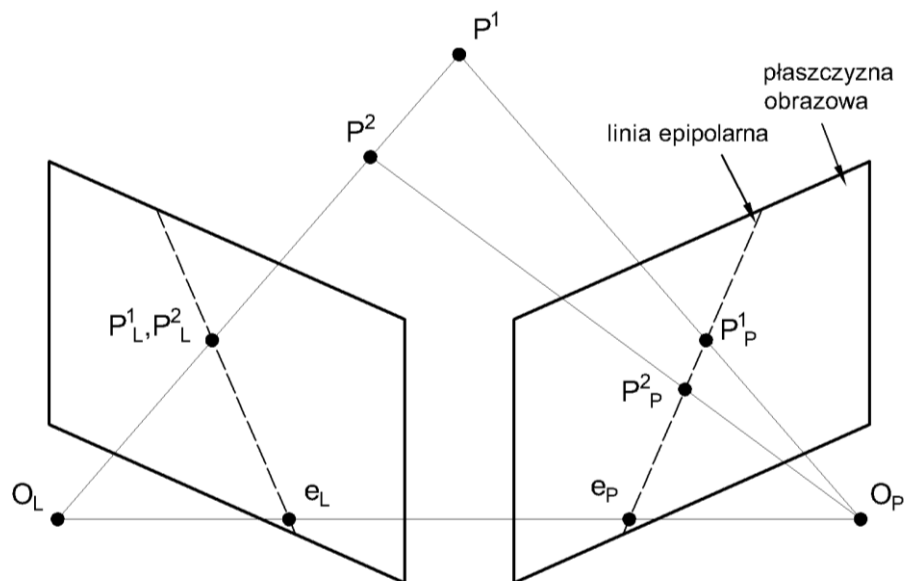
Uwzględniając parametry wewnętrzne i zewnętrzne kamer otrzymuje się równanie projekcji przestrzennego punktu na płaszczyznę obrazową[8]:

$$\begin{bmatrix} P'_x \\ P'_y \\ 1 \end{bmatrix} = A \times [R|T] \times \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{21} & r_{31} & t_x \\ r_{21} & r_{22} & r_{32} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} \quad (3.8)$$

### 3.2. GEOMETRIA EPIPOLARNA

Podstawową koncepcją związaną ze stereowizją jest geometria epipolarna. Na rys. 3.3 zaznaczono dwa punkty  $P^1, P^2$  położone w przestrzeni trójwymiarowej. Rzuty tych punktów na płaszczyzny obrazowań kamery lewej i prawej odpowiadają kolejno punktom  $P_L^1, P_L^2, P_P^1, P_P^2$ . Rzut prostej przechodzącej przez punkt centralny danej kamery i punkt w przestrzeni na płaszczyznę obrazową drugiej kamery nazywamy linią epipolarną. Punkty epipolarne  $e_P, e_L$  są punktami przebiecia 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. 3.3 wiadome jest, że punkty  $P^1, P^2$  leżą na tej samej prostej przechodzącej przez środek projekcji  $O_L$ , 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  $P^2$  płaszczyźnie obrazowej prawej kamery można wyznaczyć linię epipolarną, na której musi leżeć również rzut punktu  $P^1$ , na bazie którego możemy ocenić różnice w położeniach obu punktów w przestrzeni.



Rys. 3.3 Geometria epipolarna – na podstawie [9]

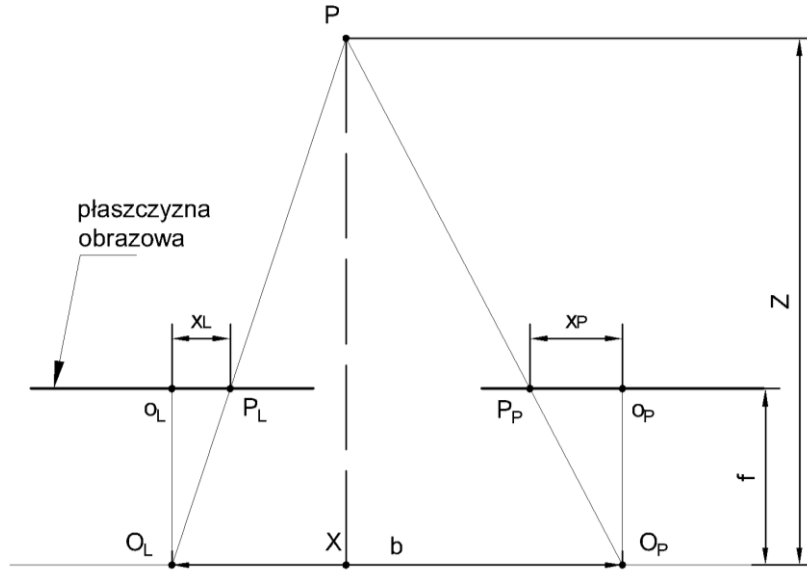
Na podstawie powyższego rys. 3.3 można zauważyć, że każdy punkt obrazowy punktu  $P$  z przestrzeni trójwymiarowej znajduje się na płaszczyźnie obrazowej wyłącznie na odpowiednich liniach epipolarnych[7]. Jest to jedno z ważniejszych założeń geometrii epipolarnej, dzięki któremu można zredukować przeszukiwanie odpowiadających punktów na obu obrazach z dwóch wymiarów do jednego, znacząco zmniejszając w ten sposób potrzebny czas na obliczenia.

### 3.3. TRIANGULACJA W STEREOWIZJI

Analiza głębi w stereowizji opiera się na wyznaczeniu dysparycji, czyli różnicy w horyzontalnym położeniu danego punktu na obrazach z obu kamer. Wyznaczona głębokość danego punktu jest odwrotnie proporcjonalna do uzyskanej różnicy w położeniu horyzontalnym zrzutowanego punktu na płaszczyzny obrazowe. Zależność ta wynika z opisanego poniżej rozumowania.

Na rys. 3.4 przedstawiono schematycznie projekcję punktu  $P$  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. 3.4 Triangulacja w układzie kanonicznym – na podstawie [7]

Różnica w położeniu rzutowanego punktu  $P$  na płaszczyznach obrazowych kamer wynosi[7]:

$$d = |DE| + |FG| = x_L + x_P \quad (3.9)$$

$$d = f \left( \frac{x_L}{f} + \frac{x_P}{f} \right) \quad (3.10)$$

Ze względu na podobieństwo par trójkątów  $\Delta PO_L X$  i  $\Delta O_L O_L P_L$  oraz  $\Delta PXO_P$  i  $\Delta O_P O_P P_P$ :

$$d = f \left( \frac{|O_L X|}{Z} + \frac{|O_P X|}{Z} \right) = f \frac{b}{Z} \quad (3.11)$$

Po przekształceniu otrzymuje się:

$$Z = f \frac{b}{d} \quad (3.12)$$

, gdzie:

$d$  – dysparycja

$f$  – odległość ogniskowa kamer

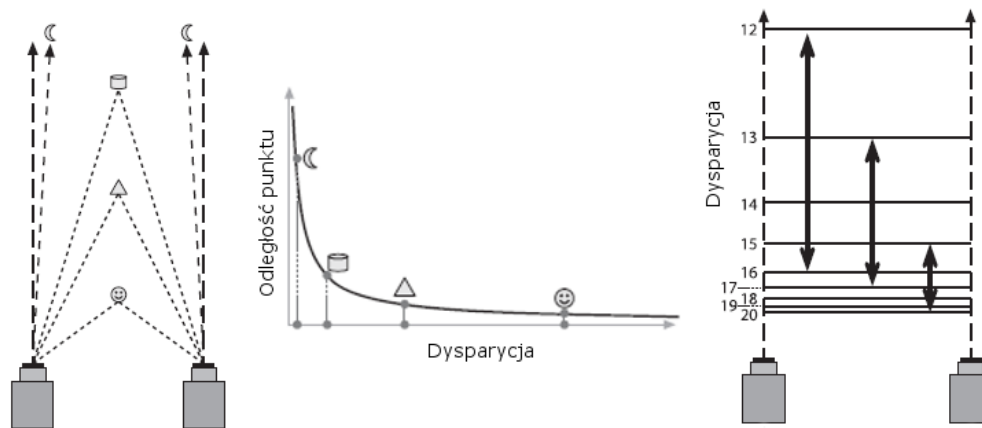
$b$  – odległość między kamerami

$Z$  – 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 przesunięciu i obrocie obrazów, tak aby płaszczyzny obrazowe obu kamer były na wspólnej płaszczyźnie. W wyniku poprawnej rektyfikacji do układu kanonicznego uzyskuje się równoległe, horyzontalne linie epipolarne przechodzące przez odpowiadające sobie punkty na dwóch obrazach.

Analizując wynikowe równanie (3.12) można dostrzec, że odległość punktu ( $Z$ ) od dysparycji ( $d$ ) jest funkcją homograficzną. Oznacza to, że im większa odległość punktu od

kamer, tym mniejsza jest dysparycja – różnica pikseli pomiędzy obrazami, a więc jednocześnie zaniżona jest rozdzielczość pomiaru głębi. Aby poprawić rozdzielczości pomiaru można zwiększyć odległość między kamerami (b), jednak wiąże się to ze zmniejszeniem wspólnej, pokrywającej się strefy obrazowań obu kamer, a co za tym idzie, ograniczeniem obszaru wykrywania punktu.



*Rys. 3.5 Schemat wyjaśniający zaniżoną rozdzielczość przy większym oddaleniu od kamer[9]*

### 3.4. PODSUMOWANIE TEORII STEREOWIZJI

Zebranie i zapoznanie się z dostępną wiedzą na temat zasad działania stereowizji pozwoliło na ustalenie kilku czynników, na które warto zwrócić uwagę podczas projektowania systemu wizyjnego. Każdy układ optyczny kamery posiada niedoskonałości objawiające się zniekształceniem obrazu, które w znaczący sposób może obniżyć jakość triangulacji. Aby zredukować niechciany wpływ deformacji należy wyznaczyć niezbędne współczynniki korekcyjne omówione w rozdziale 3.2.

Na podstawie analizy równania (3.12) można spodziewać się zmniejszonej dokładności wyznaczenia położenia punktu wzdłuż osi optycznej kamer wraz ze wzrostem odległości od kamery. Jednocześnie, oddalanie kamer od siebie pozwala zwiększyć dysparycję, a tym samym dokładność pomiaru. Biorąc te informacje pod uwagę, podczas instalowania i kalibracji systemu na stanowisku należy tak dobrać odległość pomiędzy kamerami, żeby pokrywające się pola widzenia obejmowały strefę działania robota, jednocześnie nie przekraczając jej zanadto, aby zmaksymalizować rozdzielczości pomiaru.

## 4. KONFIGURACJA STANOWISKA

### 4.1. 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 [3], 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 specyfikację 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

### 4.2. 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

#### 4.3. 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 [10]. 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 [10]

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

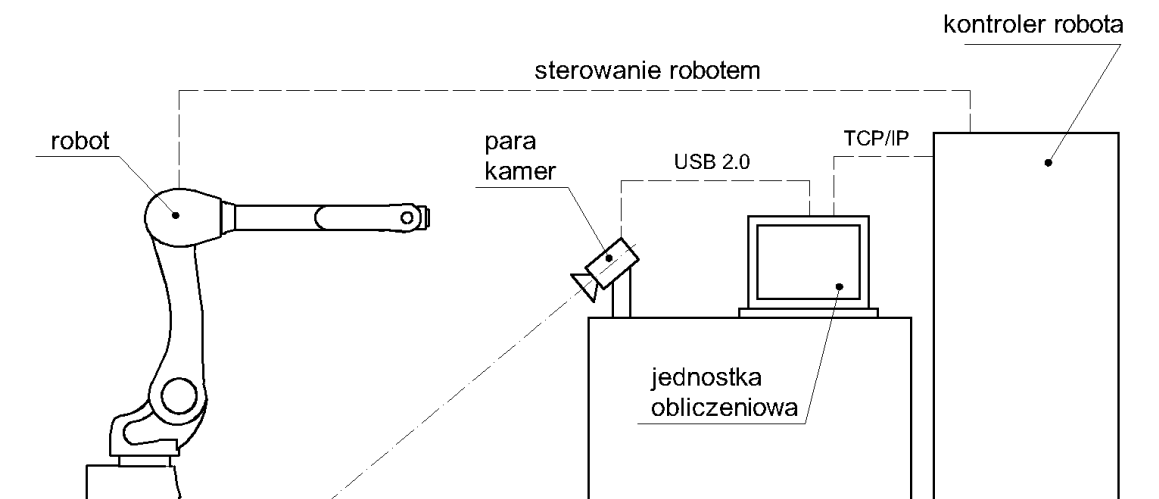
#### 4.4. BIBLIOTEKI PROGRAMOWE

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[9][11], jak i dobrze rozwinięta dokumentacja[12] 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[13].

W czasie projektowania i pisania niniejszej pracy magisterskiej korzystano z wersji OpenCV 3.1 zbudowanej pod 64-bitowy system operacyjny Windows. Główne funkcje biblioteki, wykorzystywane w kalibracji i wyznaczaniu trójwymiarowej reprezentacji punktów zawarte są w module *calib3d*.

#### 4.5. UKŁAD STANOWISKA

Poza doбором kamer o wystarczających parametrach, należy również określić ich wzajemnie położenie względem siebie i robota. Wyróżnić można dwa układy stereowizyjne: kanoniczny – przedstawiony w rozdziale 3.3 oraz o nierównoległym umiejscowieniu kamer. W większości zastosowań systemów stereowizyjnych stosuje się układ kanoniczny, choć rezultaty pomiaru położenia punktu przez układ kamer o prostopadle skierowanych osiach wskazują na większą rozdzielczość dla osi Z[3]. Ostatecznie zdecydowano się na kanoniczny układ kamer, ze względu na zwartą, kompaktową budowę, przez co nie trzeba stosować przedłużeń przewodów USB. Dodatkowo łatwiej jest kontrolować jednakową pozycję kamer montując je na jednym statywie z możliwością zmiany kąta nachylenia. Układ stereowizyjny umieszczono na stole oddalonym od robota o 1,5 m i skierowano obiektywy w stronę jego strefy roboczej. Kamery podłączono do jednostki obliczeniowej wykorzystując dwa porty USB 2.0. Sprzężenie komunikacyjne jednostki obliczeniowej z szafą sterowniczą robota zrealizowano w sieci lokalnej wykorzystując protokół TCP/IP. Architektura sieci działa na zasadzie klient – serwer, gdzie jednostka centralna – klient wysyła żądanie wykonania usługi przez robota – serwer. Komunikacja występuje w jedną stronę, od jednostki obliczeniowej do robota.



Rys. 4.1 Schemat stanowiska

## **5. KOMUNIKACJA JEDNOSTKA OBLICZENIOWA - ROBOT**

### **5.1. 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 Ethernet. 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.

### **5.2. 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, niezmięnionej 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[14].

W wykonywanym systemie, informacje o przypuszczalnej, niewielkiej wadze kilkudziesięciu bajtów będą wysyłane okresowo co  $\sim 1$  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.

## 6. OPROGRAMOWANIE JEDNOSTKI OBLICZENIOWEJ

### 6.1. STRUKTURA

W jednostce obliczeniowej zaimplementowano dwa programy:

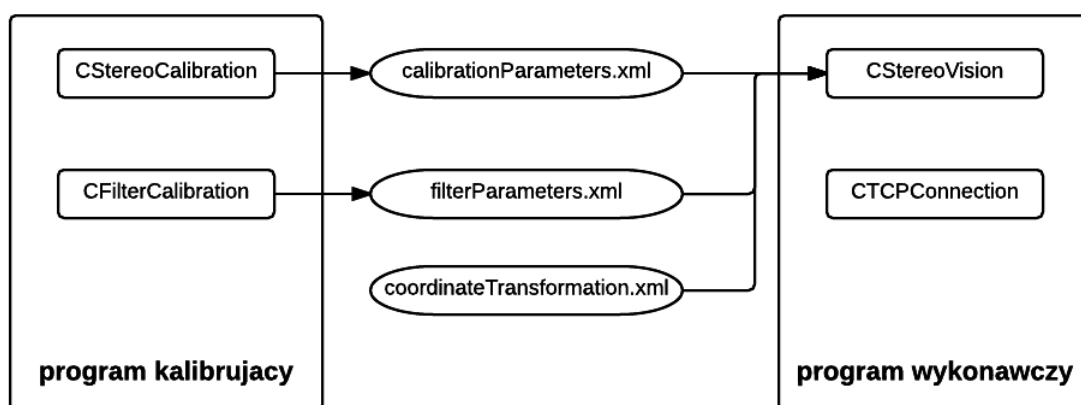
- kalibrujący, z dwiema opcjami:
  - kalibracja kamer – przeprowadzenie procesu kalibracji z wykorzystaniem szachownicy w celu wyznaczenia i zapisania do zewnętrznego pliku współczynników i macierzy niezbędnych do przeprowadzenia poprawnej analizy obrazów w programie wykonawczym,
  - ustawienie parametrów 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 położenia 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

Tab. 6.1 Opis parametrów zawartych w pliku calibrationParameters.xml

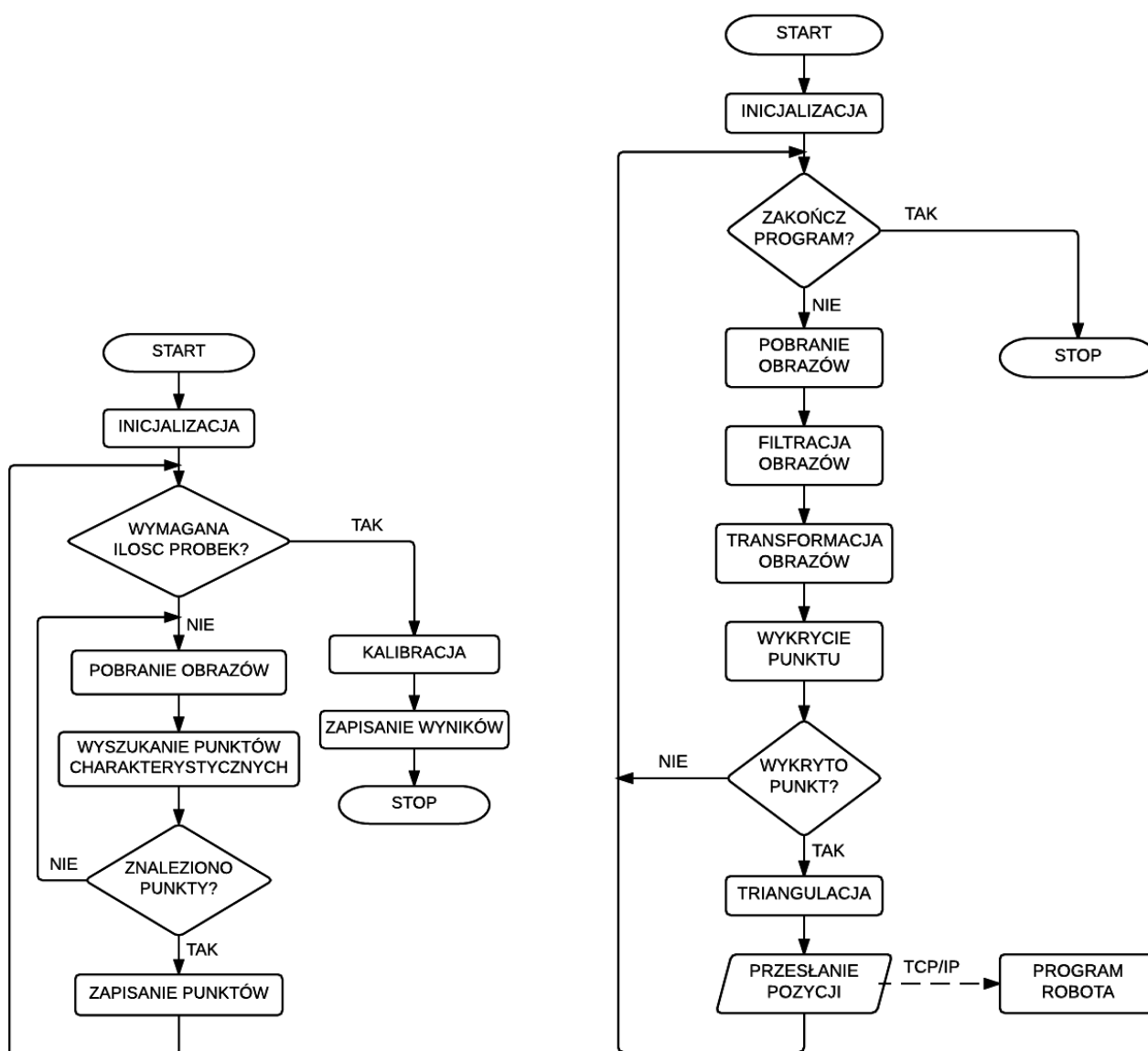
<i>calibrationParameters</i>		
Nazwa parametru	Typ	Opis
<i>Date</i>	<i>time_t</i>	Data i godzina przeprowadzenia kalibracji
<i>leftCameraMat</i> , <i>rightCameraMat</i>	<i>cv::Mat</i>	Parametry wewnętrzne kamery, patrz roz. 3.1
<i>leftCameraDistorsion</i> , <i>rightCameraDistorsion</i>	<i>cv::Mat</i>	Współczynniki korekcji dystorsji, patrz roz. 3.1
<i>rotationMat</i>	<i>cv::Mat</i>	Macierz rotacji pomiędzy układem współrzędnych pierwszej kamery i drugiej, patrz roz. 3.1
<i>translationMat</i>	<i>cv::Mat</i>	Wektor translacji pomiędzy układami współrzędnych kamer, patrz roz. 3.1
<i>leftRectificationMat</i> , <i>rightRectificationMat</i>	<i>cv::Mat</i>	Macierz 3x3 rektyfikacji (rotacji) obrazu
<i>leftProjectionMat</i> , <i>rightProjectionMat</i>	<i>cv::Mat</i>	Macierz 3x4 projekcji układu współrzędnych kamery po rektyfikacji
<i>imageSize</i>	<i>cv::Size</i>	Rozdzielczość obrazu, np. 640x480
<i>errorRMS</i>	<i>double</i>	Wartość błędu kalibracji w pikselach

Tab. 6.2 Opis parametrów zawartych w pliku filterParameters.xml

<i>filterParameters</i>		
Nazwa parametru	Typ	Opis
<i>method</i>	<i>int</i>	Wartość odpowiada metodzie kalibracji: 1 - RGB, 2 - HSV
<i>min1</i>	<i>int</i>	Dolna wartość progu filtrowania dla pierwszej składowej piksela
<i>min2</i>	<i>int</i>	Dolna wartość progu filtrowania dla drugiej składowej piksela
<i>min3</i>	<i>int</i>	Dolna wartość progu filtrowania dla trzeciej składowej piksela
<i>max1</i>	<i>int</i>	Górna wartość progu filtrowania dla pierwszej składowej piksela
<i>max2</i>	<i>int</i>	Górna wartość progu filtrowania dla drugiej składowej piksela
<i>max3</i>	<i>int</i>	Górna wartość progu filtrowania dla trzeciej składowej piksela

Tab. 6.3 Opis parametrów zawartych w pliku *coordinateTransformation.xml*

<i>coordinateTransformation</i>		
Nazwa parametru	Typ	Opis
<i>translationX</i>	<i>float</i>	Wartość przesunięcia w mm wzdłuż osi X
<i>translationY</i>	<i>float</i>	Wartość przesunięcia w mm wzdłuż osi Y
<i>translationZ</i>	<i>float</i>	Wartość przesunięcia w mm wzdłuż osi Z
<i>rotationX</i>	<i>float</i>	Kąt obrotu w stopniach układu względem osi X
<i>rotationY</i>	<i>float</i>	Kąt obrotu w stopniach układu względem osi Y
<i>rotationZ</i>	<i>float</i>	Kąt obrotu w stopniach układu względem osi Z



Rys. 6.3 Ogólny schemat blokowy działania programu kalibracji kamer

Rys. 6.2 Ogólny schemat blokowy działania programu wykonawczego

## 6.2. 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. Przykładowy wzorzec widnieje na rys. 6.4. 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. 6.2. 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 pól 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 utworzyć którejkolwiek kamery, program wyświetli stosowną informację i zapyta ponownie o numer ID.

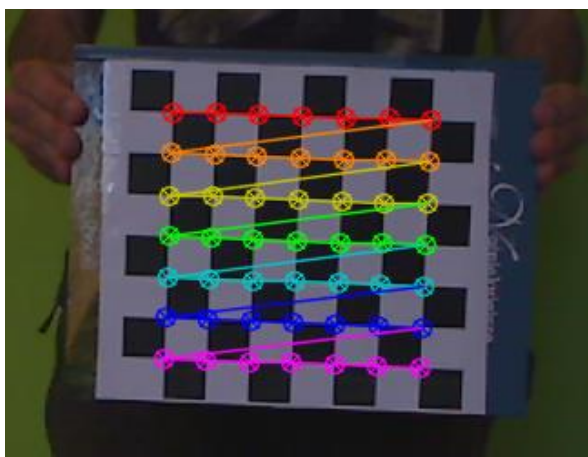
Jeśli wszystkie wartości zostaną poprawnie wprowadzone, pojawiają się dwa okna wyświetlające obrazy z obu kamer. W tym momencie należy tak manipulować wzorcem, aby znalazł się on w kadrach obu kamer. Poprawne wykrycie szachownicy sygnalizowane jest zaznaczeniem specyficznych punktów w postaci kolorowych 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 domyślnie pięciosekundowa przerwa pozwalająca uniknąć zduplikowania próbki i daje czas użytkownikowi na ustawienie wzorca pod inną orientacją lub położeniem.

Istotnym czynnikiem, który wpływa na niski błąd wyznaczania współczynników kalibracyjnych kamer, co przyczynia się do późniejszej dokładności określania położenia punktu, jest sposób manipulowania wzorcem podczas kalibracji. Wzorzec powinno ustawiać się w pod różnymi kątami i położeniami tak, aby zebrać próbki w kilku strefach obrazu. Proponowana procedura manipulacji wzorcem przedstawia się następująco:

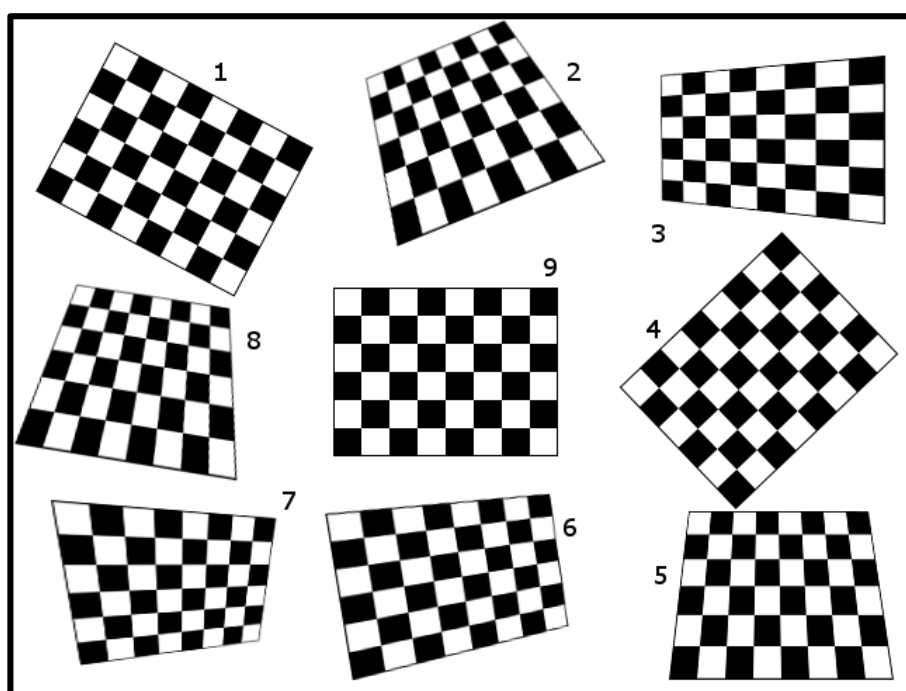
- 1) ustawić wzorzec w odległości około 1,2 m od kamer,
- 2) zebrać po jednej próbce z każdego wierzchołka obrazu i z połowy odległości pomiędzy sąsiednimi wierzchołkami,
- 3) zebrać kilka próbek z centralnych okolic obrazu,
- 4) ustawić wzorzec w odległości około 0,6 m od kamer,
- 5) zebrać próbki w podobny sposób jak dla poprzedniej odległości,
- 6) ustawić wzorzec w takiej odległości, dla której pokrywa około 60% obrazu,
- 7) zebrać kilka próbek.



Rys. 6.4 Wzorzec w postaci szachownicy



Rys. 6.5 Wykryte punkty charakterystyczne



Rys. 6.6 Przykładowa manipulacja wzorcem w kadrze kamery

Podczas kalibracji może wystąpić sytuacja, w której program nie będzie w stanie wykryć charakterystycznych punktów szachownicy. Taki skutek zazwyczaj spowodowany jest następującymi przyczynami:

- błędnie podany rozmiar szachownicy,
- niewystarczająca rozdzielczość kamer,
- za mała wielkość pola szachownicy,
- za duża odległość wzorca od kamer,
- niejednorodne, niewystarczające lub zbyt intensywne oświetlenie wzorca,
- różne ustawienia parametrów kamer.

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 i punktów charakterystycznych na wzorcu. Zakończenie pracy oznajmiane jest stosownym komunikatem w konsoli, informującym również o czasie obliczeń i orientacyjnej wartości błędu kalibracji. Kalibrację można uznać za udaną, jeśli wartość błędu będzie mniejsza od 0,5. Na zakończenie użytkownik ma możliwość zapisu parametrów do pliku *calibrationParameters.xml*, jeśli jest usatysfakcjonowany małym błędem, lub odrzucenia wyników kalibracji w przeciwnym wypadku.

### 6.3. USTAWIENIE PARAMETRÓW FILTROWANIA

Pobierane obrazy przez kamery są konwertowane do postaci cyfrowej według różnych modeli. Jednym z nich, często wykorzystywanym w informatyce jest trójwymiarowy układ RGB (rys. 6.7), gdzie pojedynczy piksel 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 (rys. 6.8), 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 progowania obrazu w celu stworzenia dwustanowego obrazu wynikowego z wykrytym punktem wskaźnika. Jeśli współrzędne punktu w danej przestrzeni barw znajdowały się w przedziale określonych wartości (progów) to punkt ten przyjmował wartość 255 w jednowymiarowym obrazie, a pozostały zbiór pikseli wartość 0:

$$P' = 255 \Leftrightarrow P_R \in \langle R^{Min}; R^{Max} \rangle \wedge P_G \in \langle G^{Min}; G^{Max} \rangle \wedge P_B \in \langle B^{Min}; B^{Max} \rangle \quad (6.1)$$

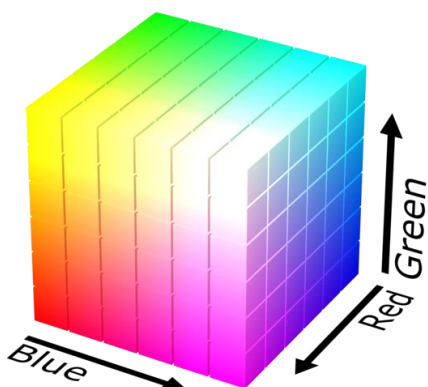
, gdzie:

$P'$  - wartość piksela po filtrowaniu,

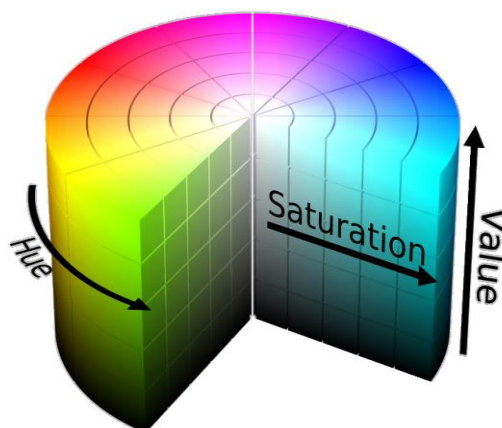
$P_R, P_G, P_B$  – składowe czerwieni, zieleni i barwy niebieskiej piksela,

$R^{Min}, G^{Min}, B^{Min}$  - wartości dolnych progów dla poszczególnych składowych,

$R^{Max}, G^{Max}, B^{Max}$  – wartości górnych progów dla poszczególnych składowych.



Rys. 6.7 Układ przestrzeni kolorów RGB [15]



Rys. 6.8 Układ przestrzeni kolorów HSV[15]

W programie kalibracyjnym, prócz wyznaczania wartości korekcyjnych dla kamer dostępna jest również druga opcja umożliwiająca ręczny dobór parametrów filtrowania do wykrycia punktu światła wskaźnika. Po wybraniu tej 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,
- metodę filtrowania – określa przestrzeń kolorów, dla której w kolejnym etapie określone są granice progowania obrazu wejściowego. Podaje się liczbę przypisaną do danej przestrzeni: RGB lub HSV.

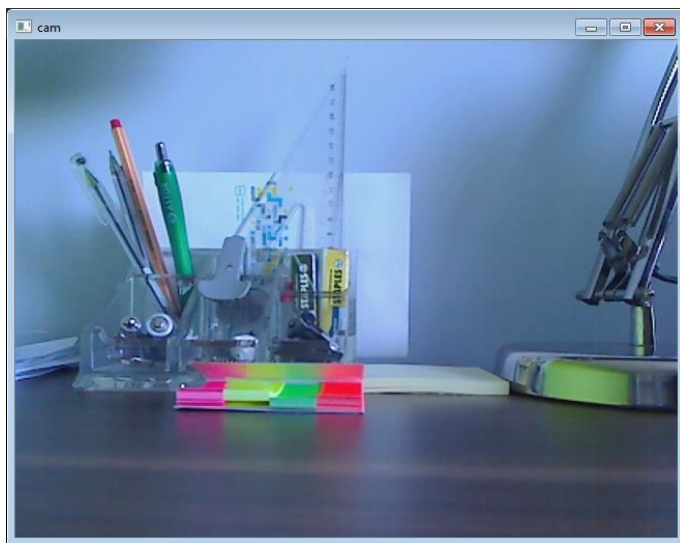
Wprowadzenie poprawnych danych wejściowych pozwala przejść do etapu ustalania wartości progowania obrazu. Pojawiają się dwa okna:

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

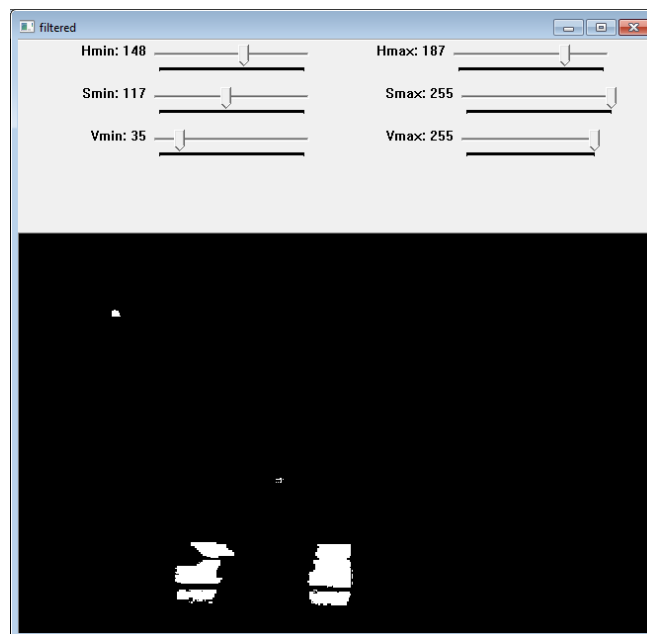
W oknie *filtered* znajdują się suwaki służące do zmiany wartości progów. Obok każdego z nich wyświetlana jest nazwa progu w zależności od wybranej przestrzeni barw wraz z aktualną wartością. Zmiany wartości przez przesunięcie suwaków są na bieżąco realizowane i uwidocznione w oknie wynikowym filtrowania. Rys. 6.9 przedstawia okno *cam* z obrazem wejściowym pobranym przez kamerę, natomiast okno *filtered* na rys. 6.10 uwidacznia efekt filtrowania obrazu wejściowego w przestrzeni HSV na podstawie parametrów dobranych w taki sposób, aby wyróżnić czerwone obiekty.

Jeżeli użytkownik jest zadowolony 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. 6.9 Podgląd na obraz z kamery



Rys. 6.10 Okno z nastawionymi progami HSV i wynikowym obrazem filtrowania

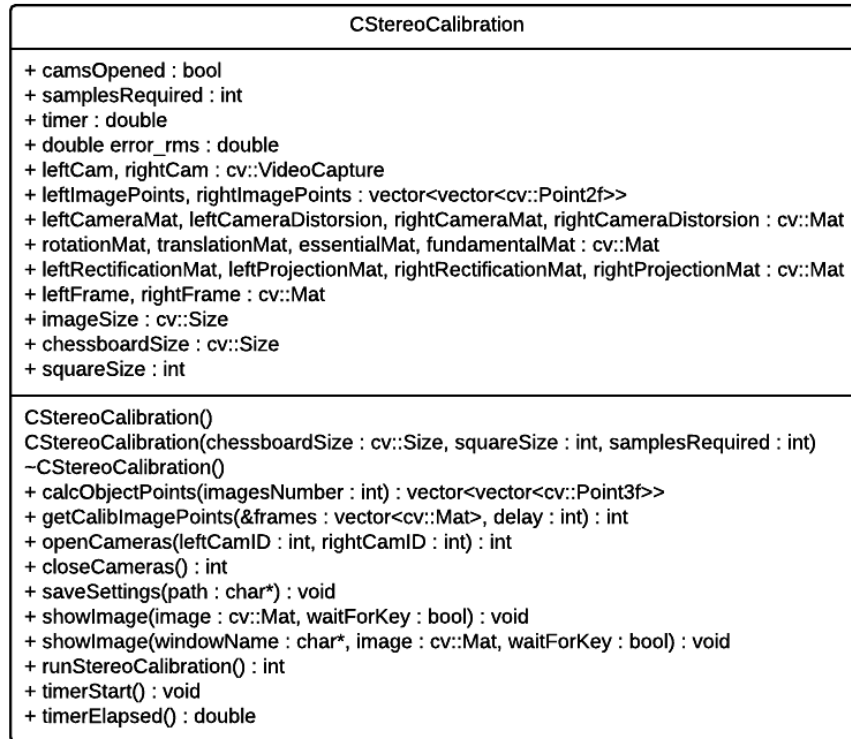
#### 6.4. PROGRAM WYKONAWCZY

Zgodnie z ogólnym schematem działania głównego programu przedstawionym na rys. 6.3 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 opisane w rozdziale 6.1, wymagane przy późniejszych procesach. W przypadku braku któregośkolwiek pliku program informuje w konsoli o braku dostępu do niego 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 wychodzi z etapu inicjalizacji. Przed przejściem do właściwej pętli zadaniowej, wyświetlane są skorygowane obrazy z obu kamer wraz z naniesioną siatką poziomych, równoodległych linii, które pozwalają na ostatnią ocenę rektyfikacji obrazów. Punkty charakterystyczne z obu obrazów powinny leżeć na tej samej linii, jeśli kalibracja kamer została wykonana poprawnie. Wcisnąc dowolny klawisz, przechodzi się 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 punkty 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 kontrolera robota zgodnie z protokołem TCP/IP, pod warunkiem poprawnie ustanowionego połączenia. W konsoli drukowane są informacje o wartościach współrzędnych uśrednionego punktu względem kamer i robota oraz wyświetlany jest łańcuch znaków wysłany do kontrolera robota.

## 6.5. KLASA CSTEREOCALIBRATION

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



Rys. 6.11 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 liczbę 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 5 sekund),
- *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,

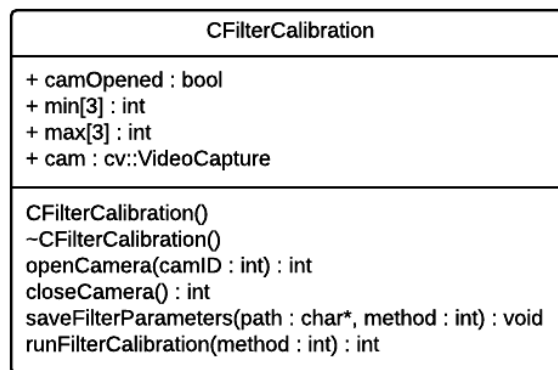
- *timerStart()* – uruchamia licznik czasu, którego wartość w sekundach można uzyskać odwołując się do funkcji *timerElapsed()*,
- *timerElapsed()* – procedura zwracająca ilość czasu w sekundach jaki upłynął od uruchomienia licznika czasu – *timerStart()*.

Opis atrybutów klasowych:

- *camsOpened* – flaga oznajmująca, czy kamery są poprawnie uruchomione,
- *samplesRequired* – liczba całkowita, większa od 4, wymaganych próbek do kalibracji kamer, wartość wprowadza użytkownik,
- *timer* – liczba taktów procesora, służy do wyznaczania czasu przez metody *timerStart()* oraz *timerElapsed()*,
- *leftCam*, *rightCam* – obiekty obsługujące kamery,
- *leftImagePoints*, *rightImagePoints* – wektory przechowujące znalezione punkty charakterystyczne wzorca podczas kalibracji,
- *chessboardSize* – wielkość wzorca – szachownicy, ilość pól w szerz i wzdłuż,
- *squareSize* – długość boku pojedynczego pola szachownicy w mm,
- pozostałe atrybuty związane są z macierzami parametrów układu stereowizyjnego, które są wyznaczane i zostały opisane są w rozdziale 6.1.

#### 6.6. 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. 6.12 Diagram UML klasy *CFilterCalibration*

Opis metod zawartych w klasie:

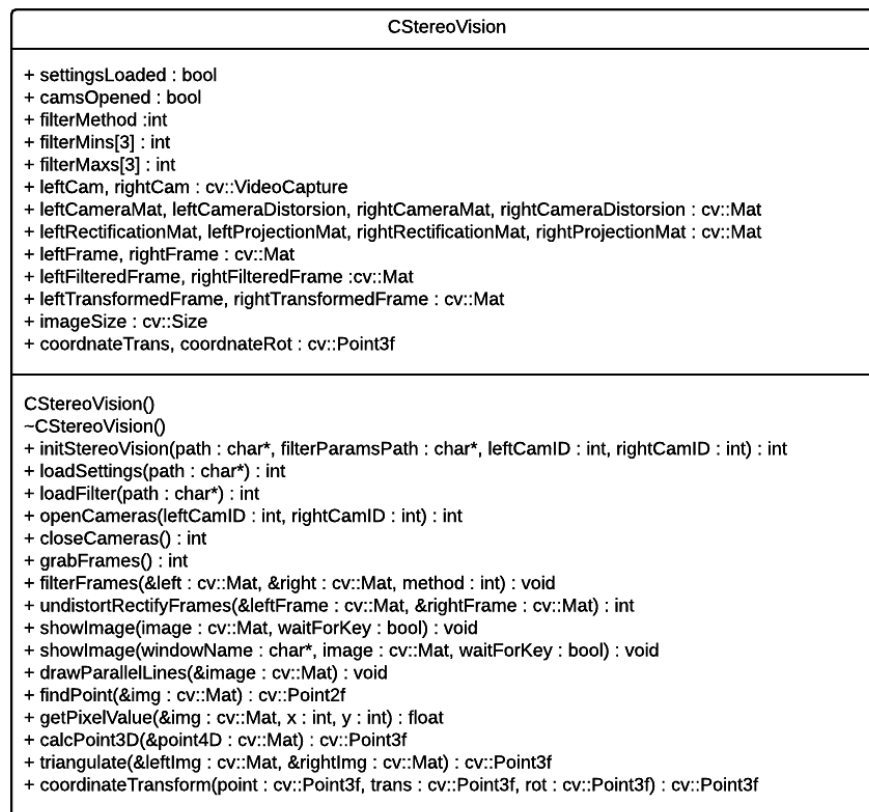
- *openCamera()* – uruchamia kamerę o podanym ID i informuje o ewentualnym niepowodzeniu,
- *closeCamera()* – zamyka, zwalnia wcześniej uruchomioną kamerę
- *saveFilterParameters()* – zapisuje do pliku o podanej ścieżce wartości atrybutów *min[]*, *max[]* oraz wprowadzoną metodę filtrowania ( dwie możliwości: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

Opis atrybutów klasowych:

- *camOpened* – flaga przyjmuje wartość *true*, jeśli kamera jest poprawnie otwarta, w przeciwnym wypadku wynosi *false*,
- *min[3]* – tablica trzech wartości dolnych progów filtrowania ustawionych przez użytkownika,
- *max[3]* – tablica trzech wartości górnych progów filtrowania ustawionych przez użytkownika,
- *cam* – obiekt obsługujący kamerę.

## 6.7. 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. 6.13 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,

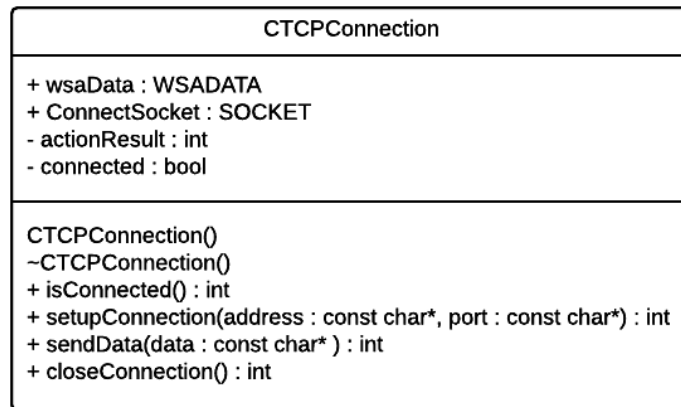
- *openCameras()* – uruchamia kamery o podanych numerach ID i informuje o ewentualnym problemie z ich otwarciem,
- *closeCameras()* – zamyka uruchomione kamery,
- *grabFrames()* – pobiera pojedyncze klatki z obu kamer i przechowuje je w atrybutach *leftFrame*, *rightFrame*,
- *filterFrames()* – proguje wprowadzone obrazy na podstawie wartości atrybutów *filterMins[]* i *filterMaxs[]* w wybranej przestrzeni kolorów (RGB lub HSV),
- *undistortRectifyFrames()* – transformuje obraz przeprowadzając rektyfikację 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.

Opis atrybutów klasowych:

- *settingsLoaded* – flaga określa, czy wszystkie wymagane parametry zostały poprawnie wczytane z plików,
- *camsOpened* – flaga informująca o poprawnie uruchomionych kamerach,
- *filterMethod* – wartość odpowiadająca wybranemu rodzajowi filtrowania RGB lub HSV, wczytana z pliku *filterParameters.xml*,
- *filterMins[3]* – tablica trzech wartości dolnych progów filtrowania wczytanych z pliku *filterParameters.xml*,
- *filterMas[3]* – tablica trzech wartości górnych progów filtrowania wczytanych z pliku *filterParameters.xml*,
- *leftCam*, *rightCam* – obiekty obsługujące kamery,
- *leftFrame*, *rightFrame* – aktualne obrazy pobrane z kamer,
- *leftFilteredFrame*, *rightFilteredFrame* – wynikowe obrazy filtrowania,
- *leftTransformedFrame*, *rightTransformedFrame* – wynikowe obrazy rektyfikacji i korygowania obrazów wejściowych,
- *coordinateTrans* – wektor zawierający wartości translacji w mm układu współrzędnych kamer na układ robota wzdłuż odpowiednich osi,
- *coordinateRot* – wektor zawierający kąty obrotu w stopniach układu współrzędnych kamer na układ robota w odpowiednich osiach,
- pozostałe atrybuty związane są z macierzami parametrów układu stereowizyjnego, które są wyznaczane i zostały opisane są w rozdziale 6.1.

## 6.8. KLASA CTCPConnection

Klasa *CTCPConnect* odpowiedzialna jest za obsługę połączenia TCP/IP. Wykorzystuje ona biblioteki systemowe *Windows*, więc działa wyłącznie pod tą rodziną systemów operacyjnych.



Rys. 6.14 Diagram UML klasy *CTCPConnection*

Opis metod zawartych w klasie:

- *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.

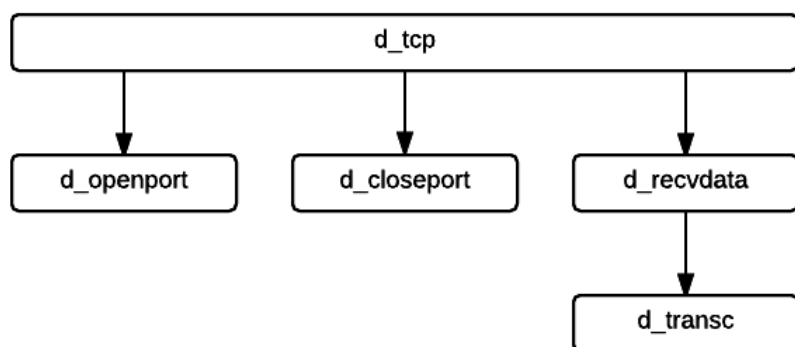
Opis atrybutów klasowych:

- *wsaData* – struktura przechowująca informacje o wersji socketu, biblioteki,
- *ConnectSocket* – obiekt wykorzystywany do komunikacji,
- *actionResult* – zmienna, do której wprowadzane są zwracane wartości funkcji obsługujących połączenie,
- *connected* – flaga oznajmiająca aktywne połączenie.

## 7. OPROGRAMOWANIE ROBOTA

Oprogramowanie robota opracowano na podstawie instrukcji [16] i składa się z głównego programu *d\_tcp* oraz czterech podprogramów odpowiedzialnych za:

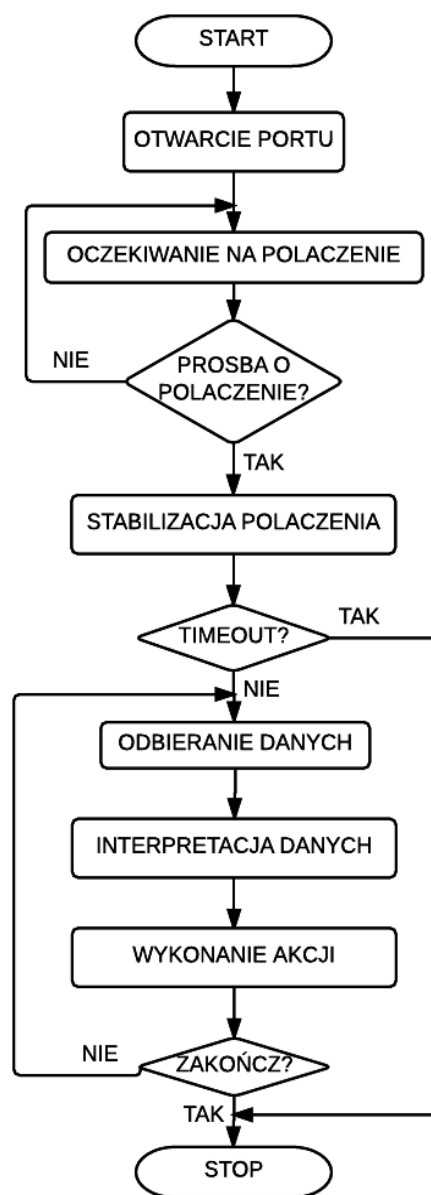
- *d\_openport* – poprawne otwarcie portu TCP o danym numerze i nasłuchiwanie próśb o połączenie,
- *d\_closeport* – zatrzymanie nasłuchiwania i zamknięcie portu,
- *d\_recvdata* – odbieranie otrzymywanych pakietów danych,
- *d\_transc* – interpretacje otrzymanych danych i wykonanie na ich podstawie odpowiedniej akcji,
- *d\_repairport* – osobny, niewielki program resetujący port, którego używa się po niepoprawnym zatrzymaniu programu, aby zamknąć otwarty port.



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 numer portu nasłuchiwania w zmiennej *port*. Należy mieć na uwadze, że w przypadku dobranego robota *Kawasaki RS010L* numer portu musi zawierać się w zakresie 8192 – 65535. Następnie w pętli wywoływana jest funkcja *TCP\_ACCEPT* monitorująca przychodzące prośby o połączenie. W przypadku akceptacji połączenia, jest ono stabilizowane z czasowym ograniczeniem, określonym w sekundach w zmiennej *timeout*, po przekroczeniu którego dane połączenie jest odrzucane. Po udanym ustabilizowaniu połączenia program wyczeka przychodzących pakietów informacji.

Kolejnym etapem jest odbieranie danych. Odpowiada za to podprogram *d\_recvdata*, który odbiera przychodzące dane i zapisuje je do bufora *recv[0]* 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. Następnym krokiem jest wykonanie zaprogramowanych akcji z danymi uzyskanymi z połączenia. Podczas testów wykorzystywano prostą, jednoliniową



Rys. 7.2 Ogólny schemat blokowy działania programu robota

instrukcję *JMOVE TRANS( )* zawartą w podprogramie *d\_transc*, która przemieszcza efektor z interpolacją przegubową do położenia określonego przez odebrane wartości współrzędnych. W przypadku bardziej złożonych akcji zaleca się stworzenie odpowiednich podprogramów uruchamianych na podstawie specyficznych, odebranych danych. W czasie wykonywanego działania aktywowany jest sygnał 2002 uniemożliwiający odebranie kolejnych informacji i nadpisanie aktualnych, co skutkowałoby zmianą wykonywanej czynności. Po zakończeniu akcji, w przypadku testowania jest to ruch do zadanej pozycji, sygnał 2002 jest wyłączany, ponownie pozwalając odbierać dane, dopóki nie zostanie aktywowany sygnał 2003 kończący proces nasłuchiwania i zamykający port w podprogramie *d\_closeport*.

Rozwinięcie systemu działań robota można zrealizować dodając podrzędne programy do *d\_transc*, uruchamiane odpowiednio do otrzymanych poleceń i argumentów przesłanych przez jednostkę obliczeniową.



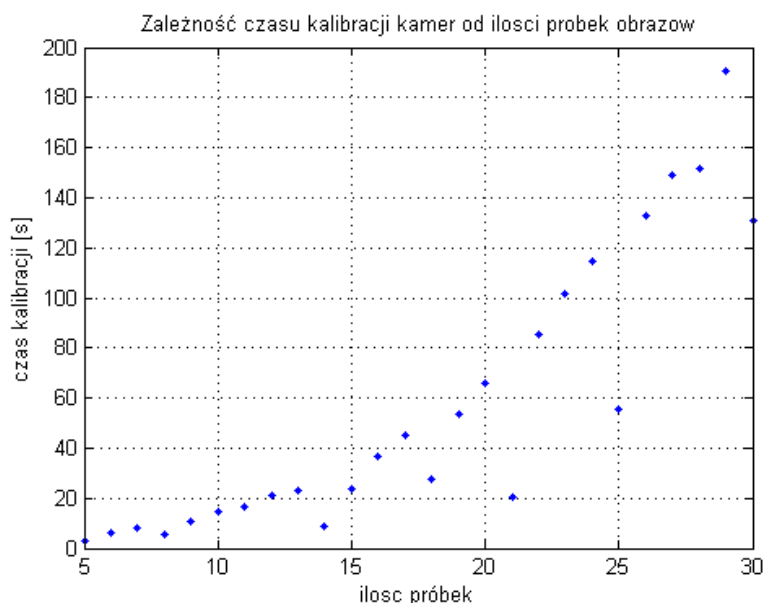
## 8. TESTOWANIE OPROGRAMOWANIA

### 8.1. 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 triangulację, 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 manipulację szachownicy w kadry kamer. Gdy program wykrywa specyficzne punkty – wierzchołki kwadratów – na obrazie wejściowym, w odpowiednich miejscach rysowane są kolorowe okręgi. Pomiedzy pobraniem próbek obrazów występuje odstęp czasowy ( domyślnie pięciosekundowy), 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 poprawnego wykrywania punktów wzorca jest ograniczony do około 2 m. Brak próbek z takich odległości może skutkować zaniżoną jakością określania położenia dalekich punktów.

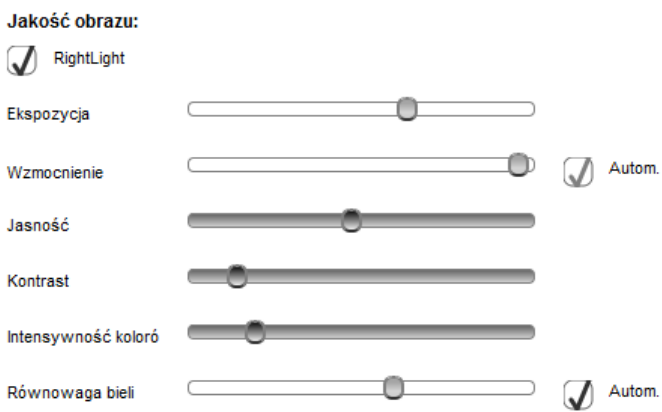
## 8.2. FILTROWANIE







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,
- szумы.

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.

<p>Nastawy automatyczne:  Ekspozycja, wzmocnienie i równowaga bieli dobierane są na podstawie analizy pobranego, aktualnego obrazu.</p>	 <p>The screenshot shows a camera settings menu with the following elements:</p> <ul style="list-style-type: none"> <li><b>Jakość obrazu:</b> A checked checkbox labeled 'RightLight'.</li> <li><b>Ekspozycja:</b> A horizontal slider.</li> <li><b>Wzmocnienie:</b> A horizontal slider with a checked checkbox labeled 'Autom.' to its right.</li> <li><b>Jasność:</b> A horizontal slider.</li> <li><b>Kontrast:</b> A horizontal slider.</li> <li><b>Intensywność koloró</b>: A horizontal slider.</li> <li><b>Równowaga bieli:</b> A horizontal slider with a checked checkbox labeled 'Autom.' to its right.</li> </ul>
---	---

<p>Nastawy ręczne: Wszystkie parametry są dobierane ręcznie i pozostają stałe.</p>	<p><b>Jakość obrazu:</b></p> <p><input type="checkbox"/> RightLight</p> <p>Ekspozycja </p> <p>Wzmocnienie  <input type="checkbox"/> Autom.</p> <p>Jasność </p> <p>Kontrast </p> <p>Intensywność koloró </p> <p>Równowaga bieli  <input type="checkbox"/> Autom.</p>
--	---

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 pozostały 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.

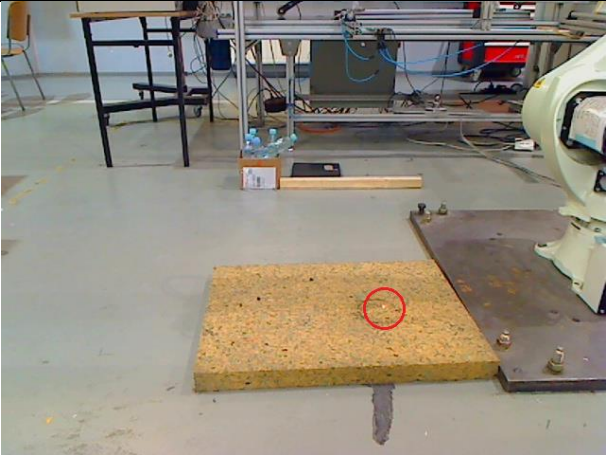

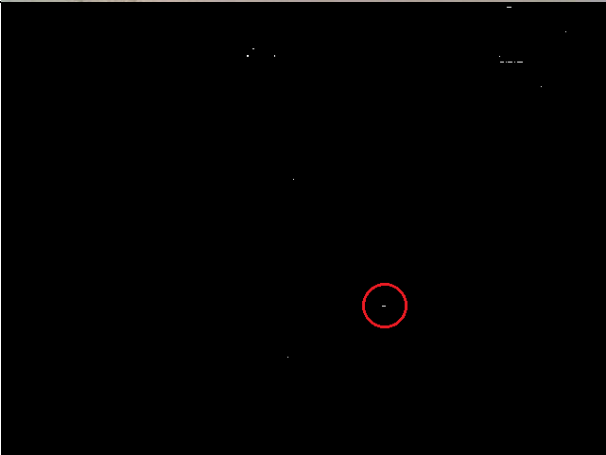
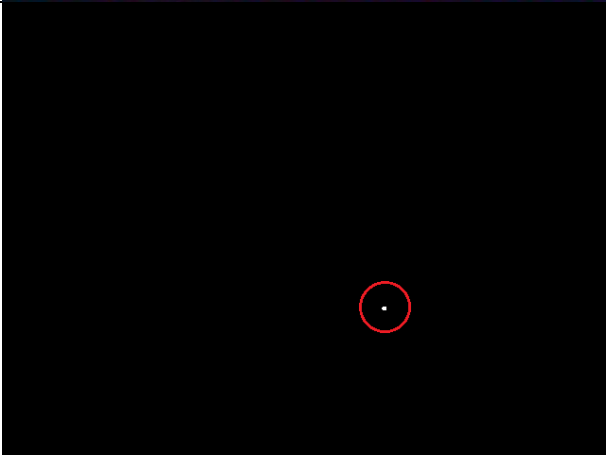
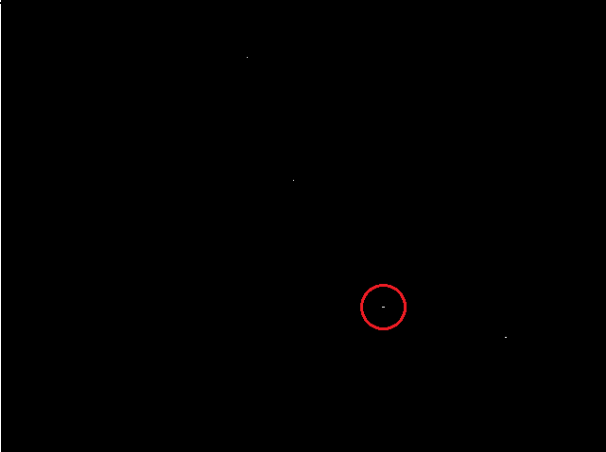
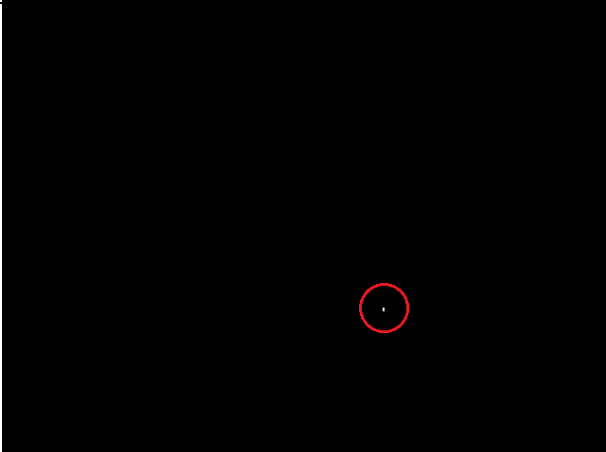
Najlepszy rezultat filtrowania został uzyskany stosując ręczne nastawy kamery dla danych warunków otoczenia oraz progując wartości pikseli w przestrzeni RGB, gdzie udało się odfiltrować tło, bez odrzucania pikseli przypadających na światło lasera. Spostrzeżono, że zwiększenie intensywności kolorów w parametrach kamery powoduje okresową destabilizację 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 skutkowało 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

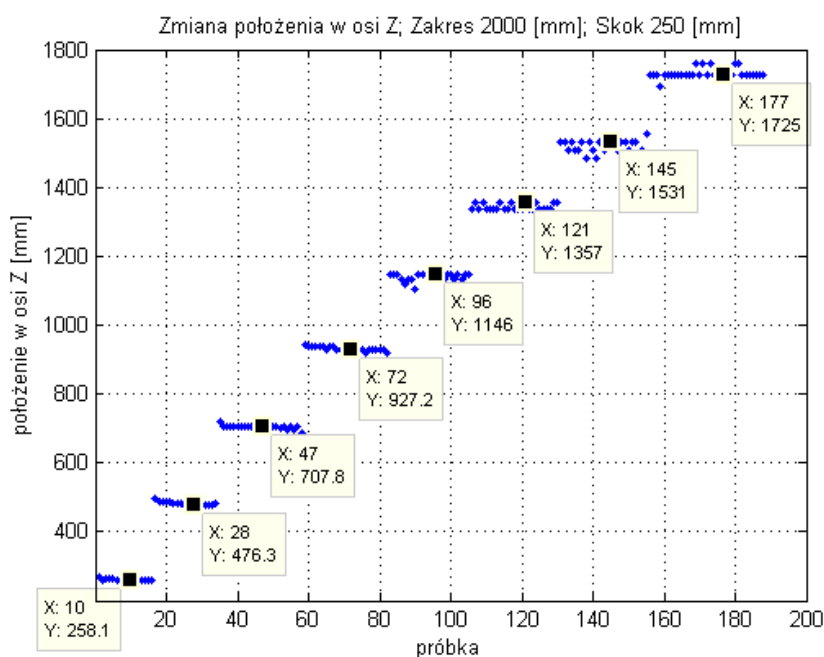
Tab. 8.2 Wynikowe obrazy testowanych metod filtrowania

	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.

### 8.3. ANALIZA DOKŁADNOŚCI

Pomiar dokładności wyznaczania położenia wzdłuż osi Z przez układ stereowizyjny zorganizowano przez umiejscowienie znaczników na horyzontalnej powierzchni w jednej linii, w odstępach 250 mm do uzyskania zakresu 0 – 2000 mm. W punkcie zerowego znacznika umieszczono kamery w układzie kanonicznym, kierując obiektywy w stronę znaczników pod takimi kątami, żeby projekcja linii znaczników położona była w pobliżu pionowej linii przechodzącej przez środek obrazu. Możliwie równolegle do osi Z układu kamer skierowano promień lasera. Poprzez umieszczenie przeszkody na drodze światła nad danym znacznikiem odległości, odbitą część światła w postaci plamki rejestrował system stereowizyjny określając jej odległość od kamer. Program pomiarowy w trybie ciągłym rejestrował, wyznaczał i zapisywał do pliku położenie wykrytych punktów. Przeszkodę umieszczano nad kolejnymi znacznikami, zaczynając od oddalonego o 250 mm od układu. Wyniki uzyskanych położeń przedstawiono na rys. 8.2.



Rys. 8.2 Wykres skokowej zmiany położenia w osi Z wykrytego punktu

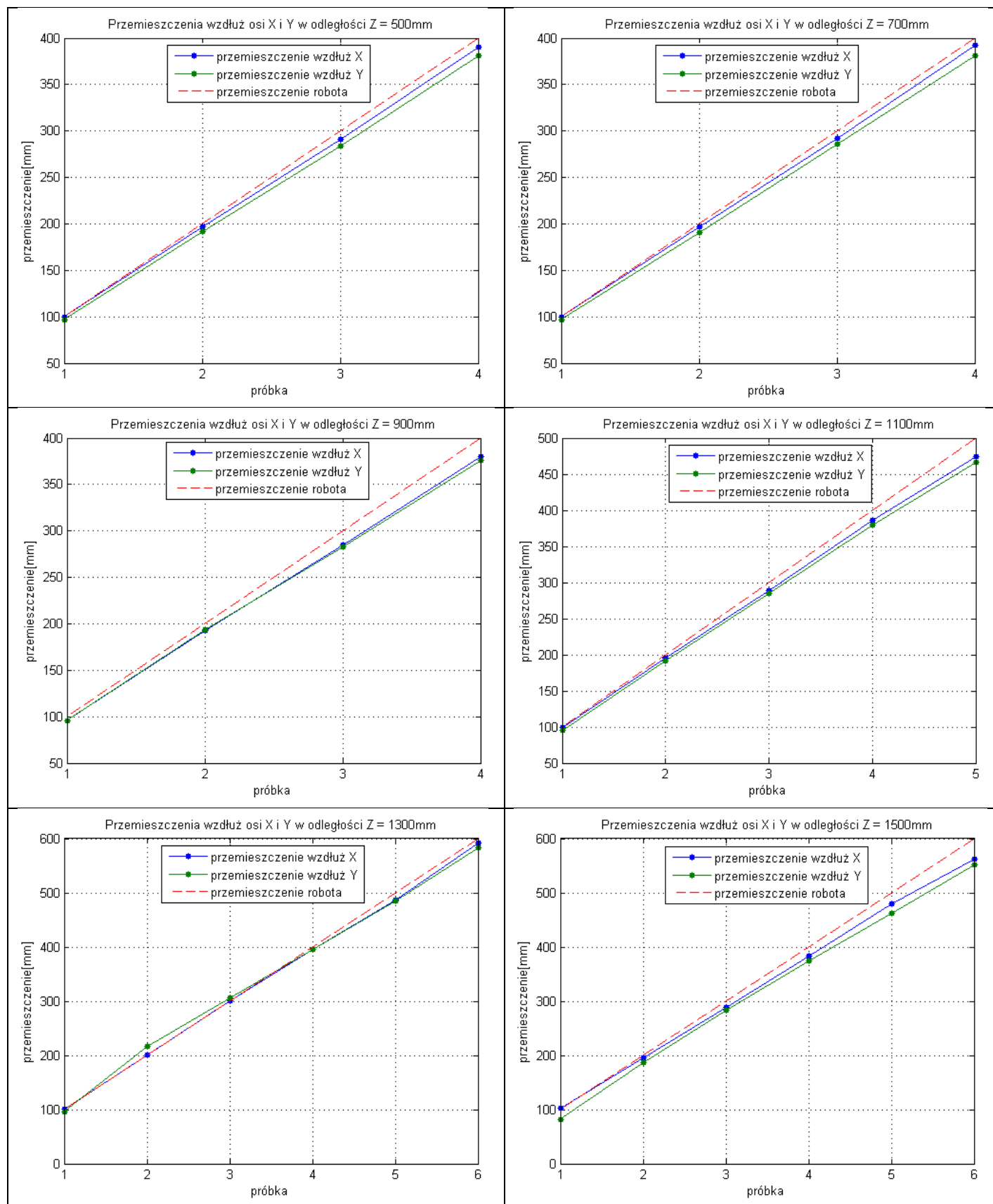
Tab. 8.3 Błąd względny wyznaczonego położenia punktu w osi Z

Położenie znacznika [mm]	Wyznaczone położenie [mm]	Błąd względny [%]
250	258,1	3,2
500	478,5	4,3
750	707,8	5,6
1000	927,2	7,3
1250	1146,0	8,3
1500	1357,0	9,5
1750	1531,0	12,5
2000	1725,0	13,8

Analizując tab. 8.3 zauważalna jest tendencja wzrostowa wartości błędu względnego wraz z rosnącą odległością punktu od kamer. Największą zmianę odnotowano dla przejścia 1500 – 1750 mm. Powyżej odległości 1500 mm błąd względny wynosi więcej niż 10%, co w zdecydowanej większości zastosowań jest nieakceptowalne. Wyniki potwierdzają wskazania zawarte we wnioskach odnoszących się do teorii stereowizji, gdzie zwrócono uwagę na możliwe obniżenie rozdzielczości pomiaru wraz ze wzrostem odległości od kamer.

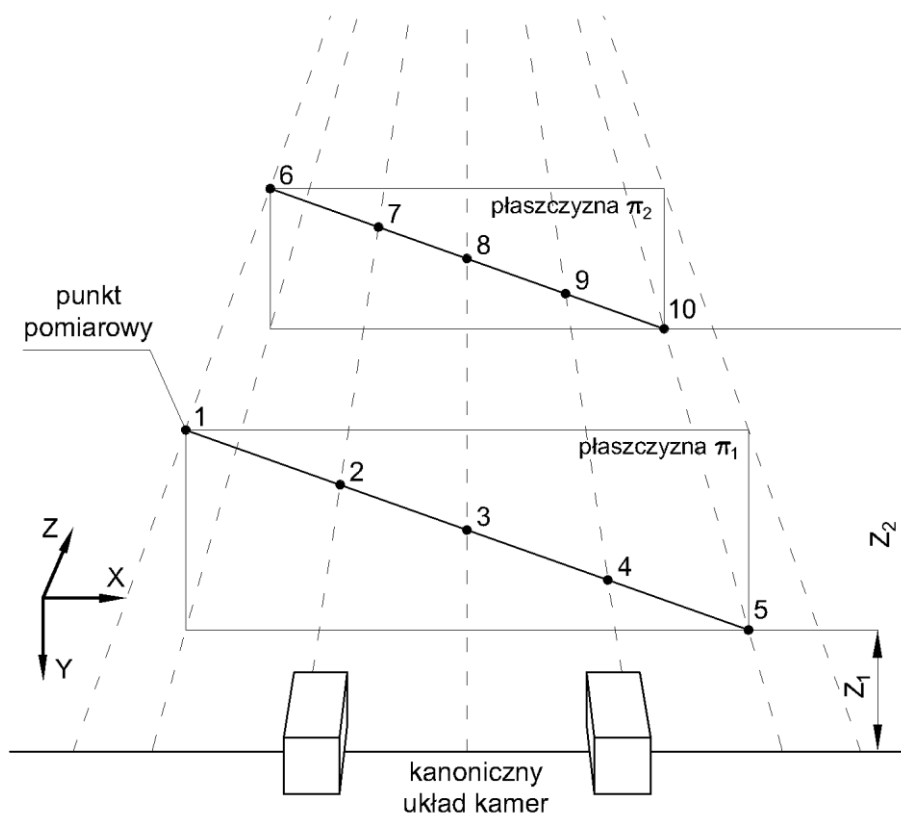
Badanie dokładności wyznaczenia przemieszczenia punktu w płaszczyźnie XY w zależności od jego odległości od kamer zrealizowano wykorzystując stanowisko robota przemysłowego. Na efektorze robota umieszczono znacznik o jaskrawym, wyróżniającym się kolorze. Przeprowadzono kalibrację parametrów filtrowania, aby dostosować je do wykrywania znacznika i eliminowania tła. Kamery umieszczono w taki sposób, aby ich płaszczyzna obrazowania była możliwie równoległa do pionowej płaszczyzny XZ robota. W danych odległościach od kamer, przemieszczano efektor skokowo o stałą wartość równą 100 mm w osiach robota X i Z tak, że rejestrowany przez kamery tor ruchu w przybliżeniu pokrywał się z przekątną obrazu. Co każdy skok zapisywano z dokładnością do 1 mm położenie znacznika wyznaczone przez system stereowizyjny. Po osiągnięciu granicy pola widzenia kamer oddalano efektor ze znacznikiem o kolejne 200 mm od układu wizyjnego i ponownie rejestrowano położenia znacznika. Początkowa odległość wynosiła 500 mm. Ideowy schemat pomiaru z zaznaczonymi, punktami pomiarowymi przedstawia rys. 8.3.

Z otrzymanych wyników wyliczono przemieszczenie dla obydwu osi X i Y odejmując wartość aktualnego położenia od położenia w próbie pierwszej dla danej odległości Z. Przebiegi dla obu osi X i Y prezentują wykresy na rys. 8.4. Zaznaczono również czerwoną linią kreskową referencyjne, wzorcowe przemieszczenie robota, które dla obu osi ma takie same wartości. Znając stałą wartość skoku – przemieszczenia efektora i uzyskane przemieszczenia na podstawie triangulacji stereowizyjnej, można porównać zestawy danych i ocenić dokładność systemu obliczając średni błąd względny przemieszczenia dla danych odległości.



Rys. 8.3 Wykresy wyznaczonych przemieszczeń wzdłuż osi X i Y dla danych odległości Z od kamery





Rys. 8.4 Schemat pomiaru dokładności wyznaczenia przemieszczenia  $X$ ,  $Y$  w zależności od odległości  $Z$

Tab. 8.4 Średnie wartości błędów względnego dla danych osi  $X$  i  $Y$  w zależności od odległości  $Z$  od kamer

Odległość $Z$ [mm]	Średni błąd bezwzględny		Średni błąd względny	
	$X$ [mm]	$Y$ [mm]	$X$ [%]	$Y$ [%]
500	5,50	11,50	1,75	4,27
700	4,75	11,25	1,54	4,23
900	11,50	12,75	4,38	4,67
1100	11,40	16,60	3,17	5,27
1300	4,50	10,33	1,07	3,53
1500	15,67	26,33	3,80	8,61

Wykresy przemieszczeń dla odległości  $Z$  równych 500 i 700 mm są do siebie podobne, co potwierdzają również zbliżone wartości średnich błędów. Zauważalna jest różnica pomiędzy wartościami średnich błędów obu osi, choć spodziewano się ich na tym samym poziomie. Jedynie dla odległości  $Z = 900$  mm, linie przemieszczeń osi  $X$  i  $Y$  pokrywają się i wartości błędów są porównywalne. Znaczący wzrost wartości błędów w osi  $X$  wystąpił dla próbek z odległości 900 i 1100 mm. Największe różnice pomiędzy przemieszczeniami wyznaczonymi przez układ stereowizyjny i kontroler robota odnotowano dla najdalszego położenia 1500 mm.

## 9. MOŻLIWOŚCI ROZWINIĘCIA SYSTEMU

W czasie wykonywania i testowania systemu intuicyjnego programowania punktów zrodziło się kilka pomysłów, które mogłyby usprawnić i rozwinąć działanie opracowanego systemu.

Podczas testowania systemu zauważono, że w niektórych przypadkach ręczne określanie wartości translacji i rotacji układu współrzędnych kamer na robota jest dosyć kłopotliwe. Ułatwieniem tego zadania była by dodatkowa opcja dostępna z poziomu programu kalibracji systemu, wyznaczająca wartości transformacji układu na podstawie wyznaczonego punktu położenia wskaźnika względem kamer oraz ręcznie wprowadzonych współrzędnych efektora robota, znajdującym się w tym samym punkcie.

Stworzony program wykonawczy, określający położenie punktu wskaźnika jest podstawą do opracowania algorytmów zbierających dane i przetwarzające je na określoną funkcjonalność. Jedną z wartościowych funkcji była by możliwość określania zamkniętego obszaru powierzchni, na przykład przez wskazanie wierzchołków, w celu:

- zadania obszaru roboczego, w którym robot miałby wykonywać swoje działania,
- zastrzeżenia obszaru, którego robot powinien unikać,
- określania wielkości obiektu manipulowanego i jego orientacji.

Przez jedynie wskazanie danego punktu robot nie jest w stanie określić, jakie zadanie ma wykonać, chyba że jest na stałe zaprogramowana jedna akcja. Niezbędnym rozwiązaniem tego problemu jest zaprojektowanie ręcznego kontrolera, za pomocą którego operator miałby możliwość wskazywania punktów i powiązania z nim konkretnych akcji. Kilka sugerowanych przycisków funkcyjnych zaznaczono w tabeli 9.1.

*Tab. 9.1 Wykaz proponowanych przycisków kontrolera*

L.p.	Typ przycisku	Funkcja
1	monostabilny	włączenie światła wskaźnika
2	bistabilny	w zależności od pozycji: <ul style="list-style-type: none"><li>• wskazywanie pojedynczych punktów</li><li>• wskazywanie obszarów</li></ul>
2	monostabilny	zapamiętanie wskazanego punktu
3	monostabilny	zapamiętanie orientacji kontrolera
4	bistabilny	aktywacja akcji 1; np. podnieś przedmiot
5	bistabilny	aktywacja akcji 2; np. opuść przedmiot
6	bistabilny	przycisk bezpieczeństwa, zatrzymanie ruchu

Istotną wadą tego systemu jest brak możliwości określania orientacji efektora w danym punkcie. Tą niedogodność można zniwelować dołączając do kontrolera żyroskop, którego odczyty przekazywane były by do jednostki obliczeniowej.

## 10. PODSUMOWANIE I WNIOSKI

W niniejszej pracy magisterskiej udało się opracować niezbędne komponenty do intuicyjnego systemu programowania robotów. Napisano program przeprowadzający użytkownika przez proces kalibracji kamer z wykorzystaniem wzorca w postaci szachownicy oraz ułatwiający dobór odpowiedniej metody i parametrów filtrowania obrazu do danego otoczenia i rodzaju wykrywanego punktu. Wykonany główny program wykonawczy bazując na plikach konfiguracyjnych, będących wynikiem kalibracji, koryguje zniekształcenia obrazu i poddaje go filtrowaniu, w celu wykrycia punktu wskazanego wskaźnikiem laserowym. Funkcja triangulacji, na podstawie położenia punktu na obu obrazach wyznacza jego przestrzenne położenie. Dobrany protokół komunikacyjny jednostki obliczeniowej z robotem TCP/IP, wraz z napisaną klasą *TCPConnection* obsługującą połączenie, gwarantują dostarczenie wysłanych danych, np. wyznaczonego położenia punktu, do robota w niezmienionej postaci. Zaimplementowany program robota poprawnie odczytuje uzyskane dane i interpretuje je, wykonując na ich podstawie odpowiednią akcję – przemieszczenie efektora do zadanego położenia.

Proces kalibracji ma ogromny wpływ na poprawne wyznaczanie punktu przez program wykonawczy. Należy przeprowadzać go skrupulatnie i weryfikować efekty, aż do uzyskania zadowalających wyników. Kalibracja szachownicą o rozmiarze 9x6 oraz długością pojedynczego pola równą 25 mm, może okazać się niewystarczająca przy odległości od kamer wynoszącej 1,5 m, dlatego warto skorzystać z większego wzorca.

Zrzuty obrazów wykonywane przez dobrane kamery nie są ze sobą zsynchronizowane co oznacza, że występuje niewielkie opóźnienie czasowe pomiędzy wykonaniem zdjęcia przez jedną kamerę i drugą. Ten efekt może prowadzić do błędnie ustalonych punktów charakterystycznych wzorca podczas kalibracji, szczególnie jeśli obrazy są pobierane w trakcie ruchu, manipulacji wzorcem. Jednym z rozwiązań tego problemu jest zastosowanie urządzeń dedykowanych do stereowizji np. *Kinect*, które równolegle wykonują zdjęcia i często mają możliwość przeprowadzenia korekcji zniekształceń obrazu, co odciążałoby jednostkę obliczeniową w przeprowadzanej analizie.

Oświetlenie otoczenia ma znaczący wpływ na jakość filtrowania obrazu. Zmienne warunki oświetleniowe strefy działania robota wymuszają ciągle dostosowywanie wartości progów filtrowania. Znaczne i trudne do zniwelowania zakłócenia wprowadzają refleksy świetlne odbite pojawiające się np. na gładkich, metalowych powierzchniach. Automatyczne nastawy kamer mogą w niektórych przypadkach potęgować efekt refleksu przez za duże wzmocnienie jasności obrazu. Ręczny dobór ustawień kamer redukuje wpływ zakłóceń ale również pozwala na wstępne wyróżnienie jasnego światła wskaźnika od tła przez zmniejszenie ekspozycji. Dobieranie parametrów filtrowania jest przeprowadzane w sposób empiryczny. Sporządzone oprogramowanie w znaczący sposób usprawnia ten etap. Podstawowy interfejs graficzny w postaci suwaków z aktualnymi wartościami progów filtrowania oraz wyświetlany na bieżąco obraz wynikowy ułatwiają użytkownikowi dostosowanie ustawień do warunków oświetleniowych. Warto rozważyć zastosowanie fizycznych filtrów optycznych o wąskim paśmie przepustowym.

Zaprojektowane oprogramowanie w powiązaniu z dobranym sprzętem mogą potencjalnie zostać wykorzystane do procesów technologicznie podobnych do lutowania, spawania, jednak tylko w niewielkiej odległości od układu stereowizyjnego. Dalekie położenie skutkuje za niską dokładnością pozycjonowania efektora, aby konkurować z klasycznymi metodami zadawania trasy, bądź z systemem bezpośredniego prowadzenia robota [4][5]. Względny błąd przemieszczenia rośnie wraz z odległością od kamer, co ogranicza zasięg działania systemu do około 1,5 m, jednak duża część robotów przemysłowych nie przekracza takiego promienia działania. Korzystanie z względnie małego wzorca mogło mieć wpływ na obniżenie jakości

kalibracji, a co za tym idzie zmniejszenie dokładności wyznaczania położenia punktu. Aby poprawić jakość pomiaru, poza wykorzystaniem większego wzorca, można rozważyć zwiększenie rozdzielczości pobieranego obrazu, jednak należy liczyć się z wydłużeniem czasu przetwarzania pojedynczej klatki.

## 11. BIBLIOGRAFIA

- [1] „Kinect-based trajectory teaching for industrial robots”, Universidad de Tarapaca, 2014
- [2] ABB, patent WO2014093822 A1, „Bare hand robot path teaching”, 2014
- [3] G. Korak, G. Kucera, „Optical tracking system”, 2015
- [4] Z. Pan, J. Polden, N. Larkin, S. van Duin, J. Norrish, „Recent progress on programming methods for industrial robots”, University of Wollongong, 2012
- [5] R. D. Schraft, C. Meyer, „The need for an intuitive teaching method for small and medium enterprises”, Fraunhofer IPA
- [6] D. Murray, J. Little, „Using real-time stereo vision for mobile robot navigation”, University of British Columbia, Computer Science Dept.
- [7] Bogusław Cyganek, „Komputerowe przetwarzanie obrazów”
- [8] OpenCV 3.0, dokumentacja - API, moduł *calib3d*, [http://docs.opencv.org/3.0-beta/modules/calib3d/doc/camera\\_calibration\\_and\\_3d\\_reconstruction.html](http://docs.opencv.org/3.0-beta/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html) [dostęp 20.05.2016]
- [9] G. Bradsky, A. Kaehler, „Learning OpenCV”
- [10] Astor, <https://www.astor.com.pl/produkty/robotyzacja/roboty-kawasaki/seria-r-szybkie-roboty.html#jkefel-1-3> [dostęp 07.05.2016]
- [11] J. Howse, S. Puttemans, Q. Hua, U. Sinha, „OpenCV 3 Blueprints”
- [12] OpenCV 3.0, dokumentacja - API <http://docs.opencv.org/3.0-beta/modules/refman.html> [dostęp 20.05.2016]
- [13] Wikipedia, Licencja BSD, [https://pl.wikipedia.org/wiki/Licencja\\_BSD](https://pl.wikipedia.org/wiki/Licencja_BSD) [dostęp 07.05.2016]
- [14] Wikipedia, UDP, [https://pl.wikipedia.org/wiki/User\\_Datagram\\_Protocol](https://pl.wikipedia.org/wiki/User_Datagram_Protocol) [dostęp 08.05.2016]
- [15] Wikipedia, HSV, [https://en.wikipedia.org/wiki/HSL\\_and\\_HSV](https://en.wikipedia.org/wiki/HSL_and_HSV) [dostęp 10.05.2016]
- [16] „Instrukcja komunikacji przy użyciu TCP/IP”

## 12. SPIS ILUSTRACJI

- Rys. 3.1 Projekcja punktu w przestrzeni na płaszczyznę obrazową kamery – na podstawie [7]
- Rys. 3.2 Przedstawienie wpływu zniekształceń
- Rys. 3.3 Geometria epipolarna – na podstawie [9]
- Rys. 3.4 Triangulacja w układzie kanonicznym – na podstawie [7]
- Rys. 3.5 Schemat wyjaśniający zanizoną rozdzielczość przy większym oddaleniu od kamer [9]
- Rys. 4.1 Schemat stanowiska
- 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
- Rys. 6.4 Wzorzec w postaci szachownicy
- Rys. 6.5 Wykryte punkty charakterystyczne
- Rys. 12.1 Przykładowa manipulacja wzorcem
- Rys. 6.7 Układ przestrzeni kolorów RGB [15]
- Rys. 6.8 Układ przestrzeni kolorów HSV [15]
- Rys. 6.9 Podgląd na obraz z kamery
- Rys. 6.10 Okno z nastawionymi progami HSV i wynikowym obrazem filtrowania
- Rys. 6.11 Diagram UML klasy *CStereoCalibration*
- Rys. 6.12 Diagram UML klasy *CFilterCalibration*
- Rys. 6.13 Diagram UML klasy *CStereoVision*
- Rys. 6.14 Diagram UML klasy *CTCPConnection*
- Rys. 7.1 Układ podprogramów programu robota

Rys. 7.2 Ogólny schemat blokowy działania programu robota  
 Rys. 8.1 Wykres zależności czasu kalibracji kamer od ilości próbek obrazu  
 Rys. 8.2 Wykres skokowej zmiany położenia w osi Z wykrytego punktu  
 Rys. 12.2 Schemat pomiaru dokładności wyznaczenia przemieszczenia X, Y w zależności od odległości Z  
 Rys. 12.3 Wykresy wyznaczonych przemieszczeń wzdłuż osi X i Y dla danych odległości Z od kamer

### 13. SPIS TABEL

Tab. 4.1 Specyfikacja jednostki obliczeniowej  
 Tab. 4.2 Specyfikacja kamery  
 Tab. 4.3 Specyfikacja techniczna robota Kawasaki RS010L [10]  
 Tab. 6.1 Opis parametrów zawartych w pliku *calibrationParameters.xml*  
 Tab. 6.2 Opis parametrów zawartych w pliku *filterParameters.xml*  
 Tab. 6.3 Opis parametrów zawartych w pliku *coordinateTransformation.xml*  
 Tab. 8.1 Wartości progów filtrowania  
 Tab. 8.2 Wynikowe obrazy testowanych metod filtrowania  
 Tab. 8.3 Błąd względny wyznaczonego położenia punktu w osi Z  
 Tab. 8.4 Średnie wartości błędu względnego dla danych osi X i Y w zależności od odległości Z od kamer  
 Tab. 9.1 Wykaz proponowanych przycisków kontrolera

### 14. SPIS ZAŁĄCZNIKÓW

Listingi programów:

- {1} *main.cpp* ( kalibracja)
- {2} *StereoCalibration.h*
- {3} *StereoCalibration.cpp*
- {4} *FilterCalibration.h*
- {5} *FilterCalibration.cpp*
- {6} *main.cpp* ( wykonawczy)
- {7} *StereoVision.h*
- {8} *StereoVision.cpp*
- {9} *TCPConnection.h*
- {10} *TCPConnection.cpp*
- {11} Program robota