POLITECHNIKA WARSZAWSKA
WYDZIAŁ ELEKTRYCZNY
INSTYTUT STEROWANIA I ELEKTRONIKI PRZEMYSŁOWEJ

PRACA DYPLOMOWA MAGISTERSKA
na kierunku Automatyka i Robotyka

Daniel Gozdera
Nr albumu: 202619

Paweł Golba
Nr albumu: 202617

Rok akad.: 2009/2010
Warszawa, 23.11.2009

Zastosowanie aktywnej wizji do manipulacji i rekonstrukcji nieznanych struktur trójwymiarowych

Zakres pracy:

1. *Wprowadzenie i sformułowanie celu pracy*
2. *Kalibracja układu oko-ręka*
3. *Identyfikacja , separacja i lokalizacja obiektów na podstawie obrazu*
4. *Utworzenie hierarchicznego modelu struktury w celu określenia kolejności dekompozycji*
5. *Zaplanowanie i wykonanie ruchów manipulatora*
6. *Podsumowanie i wnioski*

Kierujący pracą: dr inż. Witold Czajewski

Termin złożenia pracy: *15.09.2010*

Praca wykonana i obroniona pozostaje własnością Instytutu i nie będzie zwrócona wykonawcy.

Oświadczenie autorów pracy dyplomowej magisterskiej.

Świadomi odpowiedzialności prawnej oświadczamy, że niniejsza praca dyplomowa została napisana przez nas samodzielnie i nie zawiera treści uzyskanych w sposób niezgodny zobowiązującymi przepisami.

Oświadczamy również, że przedstawiona praca dyplomowa nie była wcześniej przedmiotem postępowania związanego z uzyskaniem tytułu zawodowego w uczelni wyższej. Oświadczamy, że badania i wyniki zamieszczone w niniejszej pracy dyplomowej zostały sfinansowane i wykonane z wykorzystaniem aparatury, sprzętu i oprogramowania będących własnością Wydziału Elektrycznego Politechniki Warszawskiej. Niniejsza praca jest utworem zbiorowym i stanowi własność intelektualną osoby kierującej pracą oraz naszą. Oświadczamy ponadto, że niniejsza wersja pracy jest identyczna z załączoną wersją

elektroniczną.

Zobowiązujemy się, że nie wykorzystamy ani nie opublikujemy wyników pracy bez zgody osoby kierującej pracą jak i współautora oraz kierownika Jednostki, w której pracę wykonano.

.....	.....
Gozdera Daniel	Golba Paweł
.....	.....
PESEL	PESEL

Zastosowanie aktywnej wizji do manipulacji i rekonstrukcji nieznanych struktur trójwymiarowych

Streszczenie

Przedstawiona praca dyplomowa miała na celu połączenie manipulatora z kamerą, tak aby na podstawie obrazu z kamery możliwe było dokonanie lokalizacji pewnych obiektów w odniesieniu do układu bazowego robota. Następnie wykorzystano informacje o położeniach tych obiektów tak, aby umożliwić robotowi manipulacje nimi. Poczyniono odpowiednie założenia dotyczące zarówno wyglądu jak i kształtu obiektów. Wszystkie wykorzystane obiekty są sześcianami o identycznych wymiarach, posiadają natomiast unikalne powierzchnie ścian w celu możliwości ich rozróżnienia i określenia orientacji.

Wykonanie projektu wiązało się z koniecznością odpowiedniej kalibracji i montażu kamery na ramieniu manipulatora w celu sprzężenia obrazu w niej widzianego z obszarem zasięgu robota. W pierwszej części pracy przedstawiono proces kalibrowania kamery i jego wpływ na obraz wykorzystywany do lokalizacji.

W następnej części pracy został przybliżony proces koordynacji kamery z manipulatorem. Przedstawiono występujące w obrębie stanowiska układy współrzędnych oraz relacje między nimi. Zagadnienie to wymagało wykonania wielu przekształceń i obliczeń macierzowych oraz ich implementacji w języku C++. Napisano blok programu umożliwiający sprowadzenie widzianych w kamerze obiektów do bazowego układu współrzędnych manipulatora. Uwzględniono także funkcje autokorekty koordynacji przy każdorazowym uruchomieniu stanowiska.

Kolejny etap prac przedstawia zaprojektowanie i implementacje głównego algorytmu działania stanowiska. Wykorzystano sekwencyjną strukturę algorytmu i wydzielono w niej bloki odpowiadające za poszczególne operacje. Zostały także przybliżone: warunki w jakich będzie pracować manipulator, wygląd struktury obiektów oraz metodologie jej dekompozycji.

W następnej części dotyczącej przetwarzania obrazu opisano wykorzystane funkcje i metody rozpoznawania obiektów oraz ich działanie. Zobrazowane zostały poszczególne etapy wydobywania z obrazu informacji o znajdujących się na nich obiektach. Zaproponowano także metodę klasyfikacji poprawności wykrycia obiektu, przetestowano oraz zoptymalizowano jej parametry. Optymalizacja została wykonana pod kątem szybkości działania algorytmu oraz jego dokładności. Dane uzyskane z obrazu wykorzystano do

określenia położenia i orientacji obiektów oraz zaimplementowano ich automatyczne obliczanie.

Jako kolejny został opisany problem zaplanowania manipulacji znanych już obiektów. Zaproponowano i zaimplementowano analityczną metodę określania kolejności dekompozycji struktury. Zrealizowano jej funkcje poprzez skonstruowanie odpowiedniego algorytmu.

Ostatni etap pracy przedstawia proces sekwencji ruchów i czynności wykonywanych przez manipulator podczas dekompozycji obiektów. Opisano także warunki mające wpływ na rodzaj wykonywanych ruchów oraz obliczenia wykonywane w celu unifikacji orientacji obiektów. Jako etap końcowy procesu manipulacji wykonywana jest sekwencja odwzorowywania wcześniej zdefiniowanej struktury obiektów aby zaprezentować zrealizowanie założeń pracy.

Do znaczących sukcesów pracy można zaliczyć: zrealizowanie stanowiska zgodnie z założeniami pracy, prawidłowo wykonaną koordynację układu kamery i układu manipulatora oraz realizację wykrywania zdefiniowanych obiektów. Dodatkowymi wartościami wyniesionymi z pracy są: optymalizacja klasyfikacji rozpoznawania obiektów oraz konstrukcja i implementacja algorytmów w języku programowania wysokiego poziomu. Do głównych wniosków z pracy należały: złożoność problematyki rozpoznawania obrazu i jego optymalizacja, analiza i konstrukcja algorytmów oraz planowanie manipulacji i obsługa robota.

Application of active vision for manipulation and reconstruction of unknown, three-dimensional structures

Summary

The purpose of this master thesis was to connect the manipulator with a camera in such way that object localization could be possible on the basis of image processing. Object localization means recognizing the matrix of the transformation between the base coordinate system of the robot and the coordinate system of the object. The next step was manipulation of this objects by the robot. Selected objects are cubes of the same size – puzzles with distinctive images on their sides. These images made the objects easier to recognize.

In order to complete the project, it was necessary to calibrate the camera and mount the camera on the robot's gripper in a proper way. In the first chapter of the thesis the camera calibration process is described and it is also explained why calibration of the camera is so important.

The most important issue of the next chapter is the subject of hand - eye coordination. Coordinate systems present in the area of the robotic station and relations between them are described. In order to perform the coordination, it was necessary to find many transformations between the coordinate systems, calculate appropriate matrix equations and implement it in C++ language. Part of code realizing object localization, in refer to the base coordinate system of the robot, was written. Automatic correction of hand - eye coordination was also prepared in every start of operating.

The following stage of the thesis shows the project and implementation of the basic algorithm of operation. Special, separate functions that realize particular stages of object localization are presented. Environmental conditions are specified in which manipulator will be working. It was very important to define the shape and the form of structures that could be made from the objects . Methodology of decomposition of these structures was also prepared.

The next chapter is related to image processing. Methods of objects recognizing and used functions are described here. Particular stages of getting information about objects on the basis of the image from the camera are presented in this chapter. A method how to verify the correctness of recognizing objects is proposed. The code was written in such a way that speed of working could be the highest, simultaneously keeping sufficient accuracy of object

recognition and localization. Data from the image were used to determine position and orientation of the objects.

The most important issue of the following part is the subject of manipulation of the objects by the robot. Algorithm of the analytic method for object decomposition is described in this chapter.

The last chapter of the thesis is about programming of the robot. In order to decompose the structure, it was necessary to plan moves of the manipulator. Conditions which could have influence on the kind of movement are described and also it is shown which calculations are necessary to unify the orientation of the objects. The final result of the manipulation was creation of structure that was planned and defined before.

Main contributions of the master thesis include: preparation of the station in accordance with predefined conditions, correct hand - eye coordination and realization of robust object recognition. Numerous algorithms were implemented in the C++ language. It must be stressed that the code was optimized in terms of speed during object recognition phase.

Spis treści

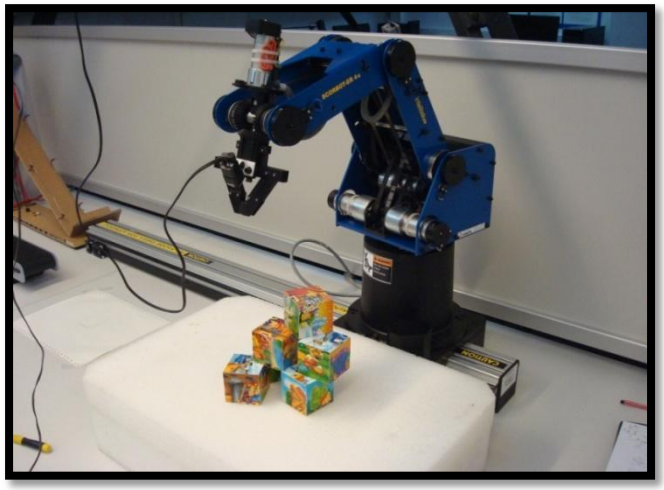
1. Wprowadzenie i sformułowanie celu pracy *(Paweł Golba)*

Zasadniczym celem niniejszej pracy dyplomowej było połączenie manipulatora z kamerą, tak aby na podstawie obrazu z kamery możliwe było dokonanie lokalizacji pewnych obiektów w odniesieniu do układu bazowego robota. Dane obiekty po odpowiednim ułożeniu tworzą pewną strukturę trójwymiarową. Założono, działanie polegające na rozpoznaniu tej struktury, następnie sekwencyjnym zdekomponowaniu i ostatecznie ułożeniu według pewnego porządku.

Wykonanie projektu wiązało się z realizacją celów pośrednich takich jak: kalibracja kamery, koordynacja układu oko-ręka, opracowanie algorytmu rozpoznawania i lokalizacji elementów, opracowanie algorytmu sekwencyjnej dekompozycji rozpoznanej struktury trójwymiarowej, opracowanie algorytmu uporządkowania obiektów, oprogramowanie manipulatora w języku C. Wymienione zagadnienia zostały opisane w kolejnych rozdziałach pracy.

Do wykonania projektu wykorzystano manipulator Scorboter 4u firmy Intelitek. Wybrana kamera została umieszczona bezpośrednio na chwytaku robota. Zaprojektowane algorytmy działania zostały zaimplementowane w języku C i C++ w środowisku Microsoft Visual Studio. Do przetwarzania obrazów posłużono się darmową biblioteką OpenCV (Open Source Computer Vision).

Na rys. 1.1. przedstawiony jest widok stanowiska, na którym został zrealizowany projekt. Do pracy magisterskiej dołączona została płyta CD, zawierająca kod źródłowy programu oraz film prezentujący pracę stanowiska.



Rys. 1.1: Widok stanowiska

2. Dobór i kalibracja kamery *(Paweł Golba)*

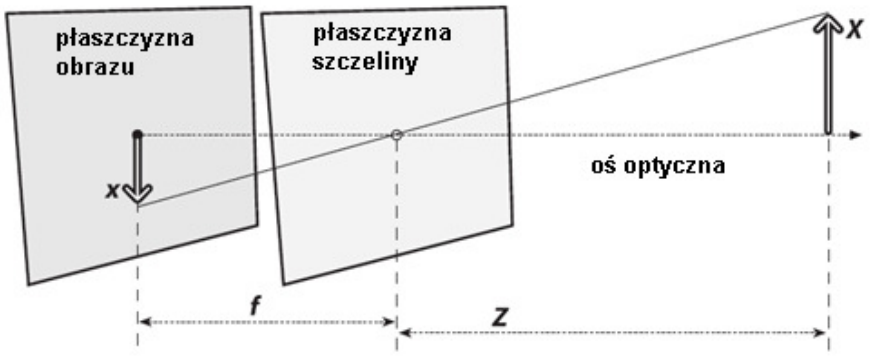
Pierwszym krokiem realizacji projektu był dobór kamery oraz jej kalibracja. Proces kalibracji kamery dostarcza model geometryczny kamery i model zniekształceń wprowadzany przez układ optyczny kamery. Te dwa modele określają tzw. parametry wewnętrzne kamery. Znajomość parametrów wewnętrznych kamery jest niezbędna do usunięcia zniekształceń obrazu oraz do obliczenia parametrów zewnętrznych kamery, czyli wektora translacji i macierzy rotacji kamery względem wybranego układu odniesienia (np. chwytaka manipulatora czy obserwowanego obiektu). Parametry zewnętrzne posłużą do koordynacji „oko ręka" oraz będą wykorzystywane do lokalizowania obiektów.

2.1. Matematyczny model kamery

Matematyczny model kamery wyjaśnia i opisuje co oznaczają parametry wewnętrzne kamery. Rysunki: 2.1.1, 2.1.2, 2.2.1, 2.2.2 oraz 2.3.1 zostały wykonane na podstawie rysunków z książki „Learning OpenCV".

Na rys. 2.1.1 przedstawiony jest model kamery z obiektywem otworkowym. W tym prostym modelu światło przechodząc przez otwór o bardzo małej średnicy, tworzy na płaszczyźnie rzutowania odwrócony obraz obserwowanego obiektu. Obraz powstały na płaszczyźnie rzutowania jest zawsze ostry, a jego rozmiar powiązany jest z rzutowanym obiektem poprzez pojedynczy parametr kamery: długość ogniskowej. Na rys 2.1.1 jest to odległość f pomiędzy płaszczyzną szczeliny a płaszczyzną obrazu. Zachodzi tu zależność opisana wzorem:

$$-x = f\ (X/Z) \quad (2.2.1)$$

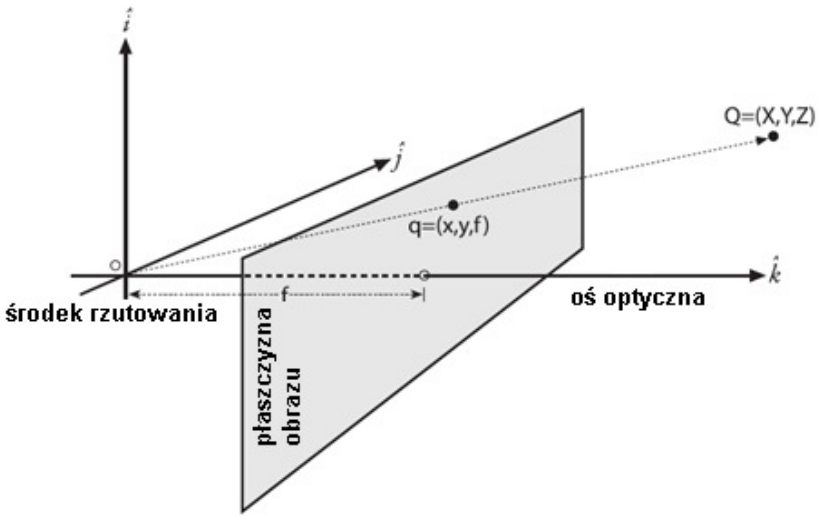


Rys. 2.1.1: Model kamery z obiektywem otworkowym

Równoważną do modelu kamery z obiektywem otworkowym jest forma pokazana na rys. 2.1.2. W modelu tym płaszczyzny szczeliny i obrazu zostały zamienione miejscami. W konsekwencji szczelina jest teraz reprezentowana przez środek rzutowania. Każdy promień biegnie od punktu obiektu do środka rzutowania. Obraz powstaje w wyniku przecięcia tych promieni z płaszczyzną obrazu w odległości f od środka rzutowania. Ponadto zachodzi zależność:

$$x/f = X/Z \quad (2.2.2)$$

Obraz nie jest odwrócony, dlatego w tym wzorze nie występuje znak „minus".



Rys. 2.1.2: Model kamery po zamianie płaszczyzn obrazu i szczeliny

Punkt na przecięciu płaszczyzny obrazu z osią optyczną to punkt środkowy obrazu. Niedokładne zamocowanie matrycy w kamerze powoduje, że punkt środkowy w rzeczywistości nie znajduje się idealnie na osi optycznej. Wprowadzono zatem dodatkowe dwa parametry c_x i c_y, które będą opisywać możliwe do wystąpienia przemieszczenie punktu środkowego.

Punkt q znajdujący się na ekranie, powstał po zrzutowaniu punktu Q i jest opisany przez współrzędne x_{ekranu} i y_{ekranu} wyrażone w pikselach według poniższych wzorów.

$$x_{ekranu} = f_x\left(\frac{X}{Z}\right) + c_x \quad (2.2.3)$$

$$y_{ekranu} = f_y\left(\frac{Y}{Z}\right) + c_y \quad (2.2.4)$$

Pojawienie się dwóch różnych ogniskowych f_x i f_y jest spowodowane tym, że poszczególne piksele czujnika kamery mogą nie być kwadratowe. Ogniskowe f_x i f_y wyrażone są w pikselach. Fizyczna długość ogniskowej F wyrażona jest w jednostkach długości np. milimetrach.

Powiązanie ogniskowych f_x, f_y i F możliwe jest dzięki wprowadzeniu odpowiednich współczynników s_x i s_y. (w jednostkach piksel/jednostka długości), według zależności: $f_x = Fs_x$ i $f_y = Fs_y$. W procesie kalibracji możliwe jest wyznaczenie jedynie ogniskowych f_x i f_y, a nie F, co jest wystarczające z punktu widzenia zastosowania kamery jako narzędzia pomiarowego.

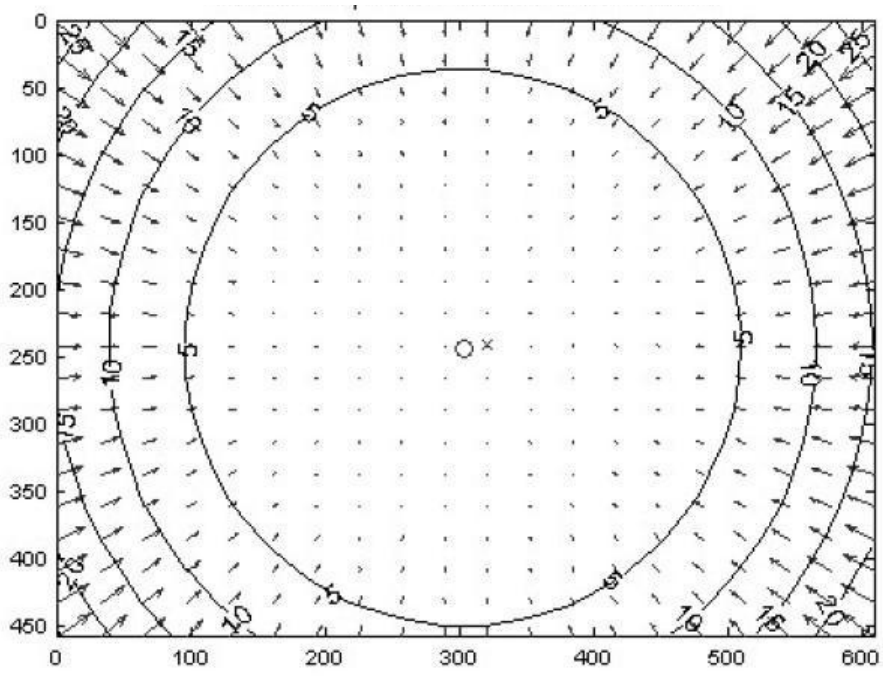
Przedstawiono cztery główne parametry kamery dotyczące modelu geometrycznego, które będą wykorzystywane w dalszych etapach projektu. Są to ogniskowa f_x i f_y i punkt środkowy c_x i c_y. Po wprowadzeniu do opisu przekształceń rzutowych współrzędnych homogenicznych, parametry rozmieszczono w macierzy M o rozmiarze 3 na 3, którą nazwano *macierzą parametrów wewnętrznych kamery*. Poniższa zależność opisuje przekształcenie rzutowe punktów Q, q we współrzędnych homogenicznych:

$$q = MQ \quad (2.1.5)$$

$$q = \begin{bmatrix} x \\ y \\ w \end{bmatrix} \qquad M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \qquad \mathrm{Q} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

2.2. Zniekształcenia wprowadzane przez soczewki

Efektem niepożądanym przy stosowaniu soczewek w kamerach jest występowanie zniekształceń obrazu. Istnieją dwie główne przyczyny zniekształceń obrazu. Pierwsza z nich związana jest z kształtem soczewek. Rezultatem wypukłości soczewek są zniekształcenia promieniowe, które powodują powstanie na obrazie efektu tzw. „beczki” lub „rybiego oka”. Silne zakłócenia położenia pikseli występują blisko brzegów obrazu tak jak to pokazano na rys. 2.2.1.



Rys. 2.2.1 Wykres zniekształceń promieniowych dla typowych soczewek kamery

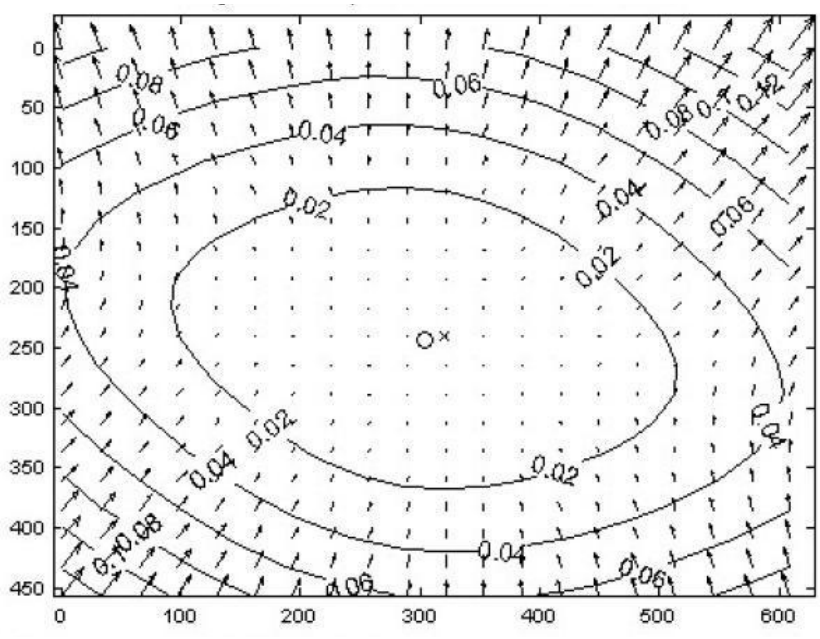
Zniekształcenia promieniowe mogą być opisane kilkoma pierwszymi niezerowymi wyrazami szeregu Taylora, według poniższych zależności:

$$x_{skorygowany} = x[1 + k_1 r^2 + k_2 r^4 + k_3 r^6] \quad (2.2.1)$$

$$y_{skorygowany} = y[1 + k_1 r^2 + k_2 r^4 + k_3 r^6] \quad (2.2.2)$$

x, y to współrzędne punktu na matrycy przed korekcją, $x_{skorygowany}$, $y_{skorygowany}$ to współrzędne tego punktu po korekcji. W przypadku typowych soczewek, dwa pierwsze współczynniki k_1 i k_2, są wystarczające do usunięcia zniekształceń promieniowych.

Druga istotna przyczyna zniekształceń obrazu wynika z niedokładnego złożenia poszczególnych elementów kamery, a w szczególności soczewek i matrycy kamery. Jeżeli soczewki nie są umieszczone idealnie równolegle w stosunku do matrycy, na obrazie pojawiają się zniekształcenia, które nazwano zniekształceniami stycznymi tak jak to pokazano na rys. 2.2.2



Rys. 2.2.2 Wykres zniekształceń stycznych dla typowych soczewek kamery

Do opisu tych zniekształceń wprowadzono dwa kolejne współczynniki:

$$x_{skorygowany} = x + [2p_1y + p_2(r^2 + 2x^2)] \quad (2.2.3)$$

$$y_{skorygowany} = y + [p_1(r^2 + 2y^2) + 2p_2x] \quad (2.2.4)$$

W rezultacie uzyskano pięć współczynników opisujących zniekształcenia, które będą wykorzystywane w dalszej części projektu. Macierz D o rozmiarze 1 na 5 zawierającą te współczynniki nazwano *wektorem zniekształceń.*

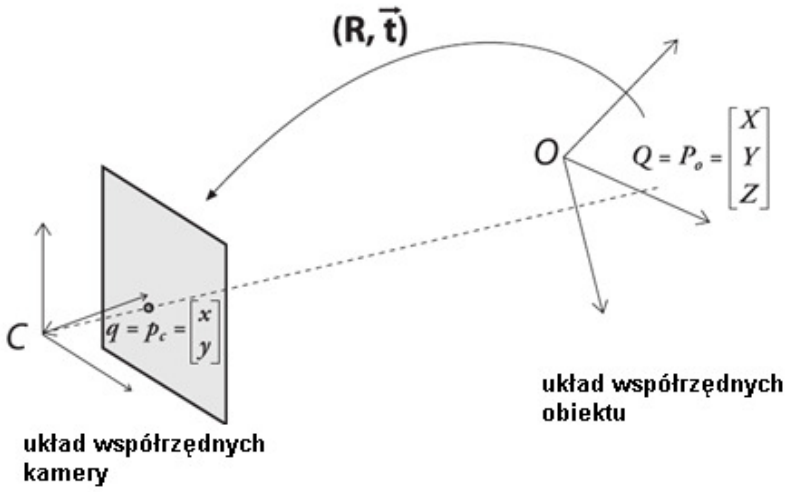
$$D = [k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3]$$

Należy podkreślić, że pozorna niekonsekwencja umieszczenia współczynników *k* i *p* w poniższym wektorze jest podyktowana po pierwsze koniecznością zgodności wstecznej biblioteki OpenCV, a także faktem, iż współczynnik k_3 jest estymowany niezwykle rzadko (tylko dla soczewek typu „rybie oko" o bardzo silnych zniekształceniach promienistych) i zazwyczaj przyjmuje wartość równą zeru.

2.3. Kalibracja

Parametry składowe *macierzy parametrów wewnętrznych M* oraz *wektora zniekształceń D* uzyskano w procesie kalibracji kamery przy pomocy pakietu narzędziowego *Camera Calibration Toolbox* w programie MATLAB.

W tym pakiecie kalibracja wykonywana jest na podstawie obiektu planarnego o regularnej strukturze. Jako obiekt kalibracyjny została wykorzystana plansza szachownicy. Na rys. 2.3.1 układ współrzędnych obiektu opisuje szachownicę.



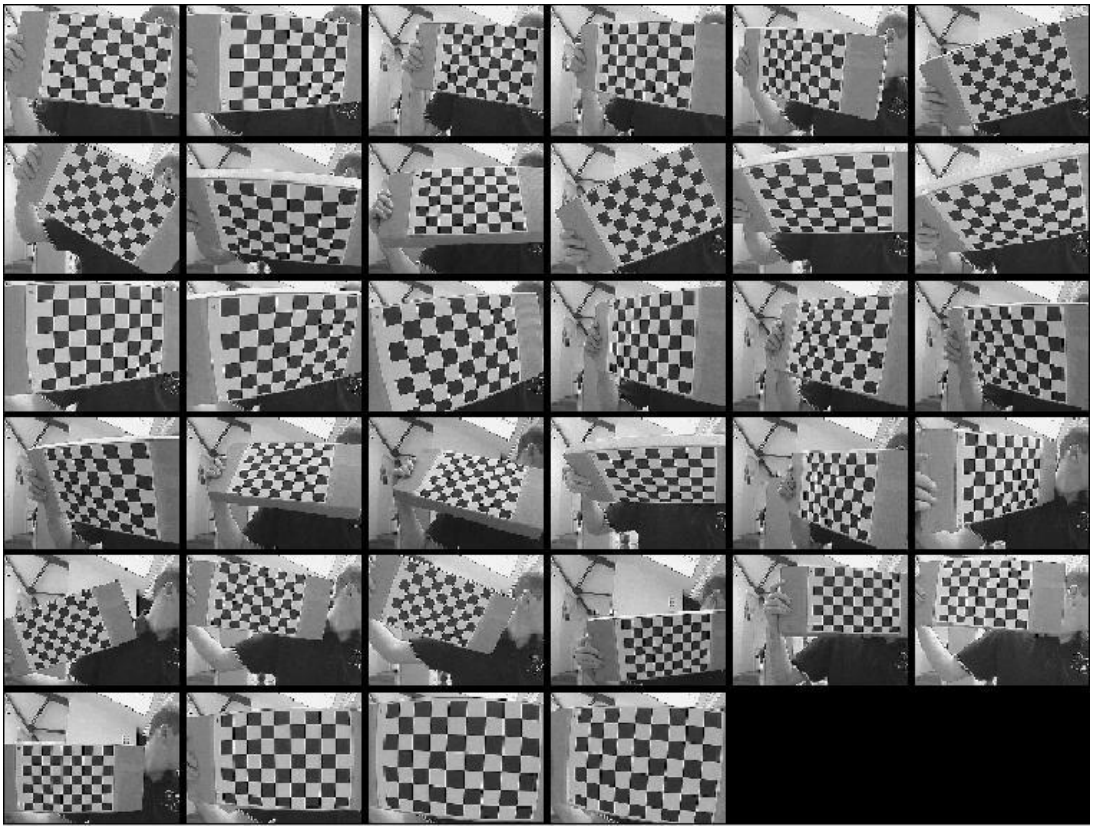
Rys. 2.3.1 Rotacja i translacja obiektu względem kamery

Transformacja układu szachownicy względem układu kamery opisana jest poprzez macierz rotacji R oraz wektor translacji T, przy czym rotację określają trzy kąty obrotu wokół trzech osi układu współrzędnych. Translacja zaś jest opisana trzema współrzędnymi przesunięcia wzdłuż trzech osi układu współrzędnych. Otrzymano zatem 6 parametrów określanych mianem *parametrów zewnętrznych kamery* względem jakiegoś obiektu. Należy zauważyć, że przy zmianie położenia i orientacji obiektu względem kamery, parametry wewnętrzne kamery pozostają niezmienione, zmianie ulegają zaś tylko parametry zewnętrzne. Rozwiązanie równania wiążącego dwa układy współrzędnych kamery i szachownicy jest celem kalibracji i pozwala na uzyskanie pożądanych parametrów wewnętrznych kamery.

Do obliczenia wszystkich geometrycznych parametrów potrzeba przynajmniej dwóch widoków obiektu planarnego (dwóch zdjęć szachownicy). Parametry opisujące zniekształcenia zaś mogą być obliczone przy pomocy nawet jednego zdjęcia regularnego wzorca – szachownicy. W praktyce, w celu uzyskania jak najdokładniejszych wyników do kalibracji wykonuje się większą liczbę zdjęć szachownicy – kilkanaście lub kilkadziesiąt.

2.4. Procedura kalibracji

Konieczne do przeprowadzenia procesu kalibracji kamery, zdjęcia obiektu planarnego, zostały wykonane przy użyciu specjalnie napisanej do tego celu aplikacji. Na rys 2.4.1 znajduje się 35 zdjęć przedstawiających szachownicę usytuowaną w przestrzeni w różnych pozycjach względem układu kamery.



Rys. 2.4.1 Zdjęcia wykonane do kalibracji

Pakiet narzędziowy do kalibracji *Camera Calibration Toolbox for MATLAB* został pobrany ze strony: http://www.vision.caltech.edu/bouguetj/calib_doc/ i zapisany w odpowiednim folderze programu MATLAB.

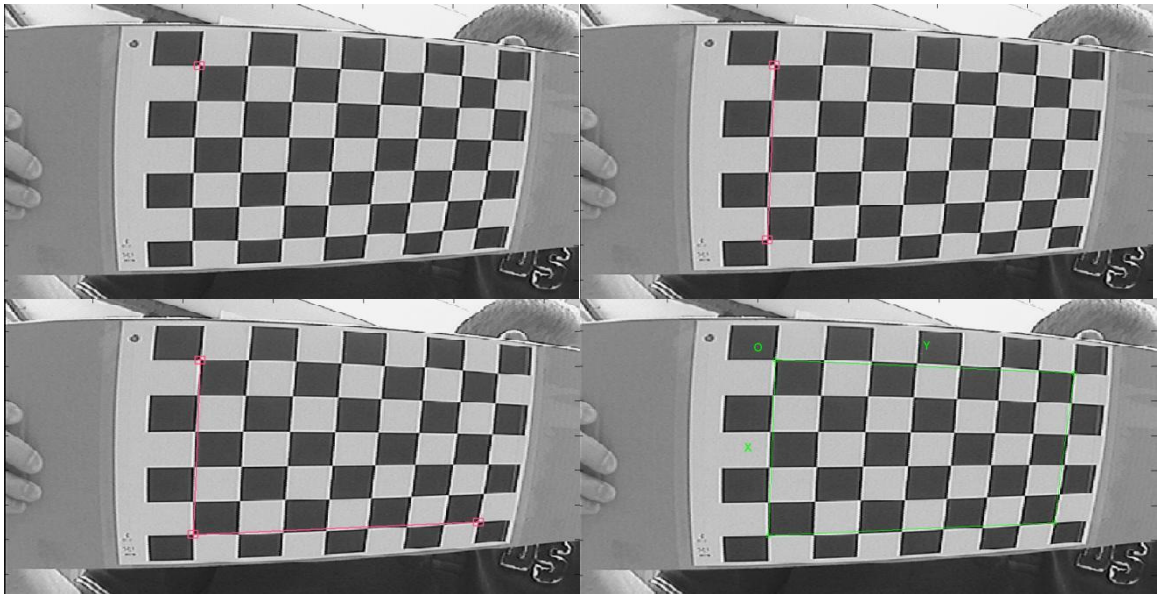
W celu uruchomienia pakietu narzędziowego należało użyć w programie MATLAB komendy *calib*. Pierwszym krokiem było wczytanie zdjęć kalibracyjnych do pamięci MATLAB. Pojawiło się okno dialogowe wyboru sposobu wczytania zdjęć szachownicy. Pierwsza opcja odpowiadała za wczytanie wszystkich zdjęć kalibracyjnych do pamięci jednorazowo. Druga zaś powodowała wczytanie każdego zdjęcia oddzielnie.

Rys. 2.4.2 Menu początkowe pakietu do kalibracji

W przypadku 35 zdjęć wystarczyło wczytanie jednorazowe, ze względu na niewielką ilość danych. Kolejne okno dialogowe dotyczyło menu dostępnych funkcji.

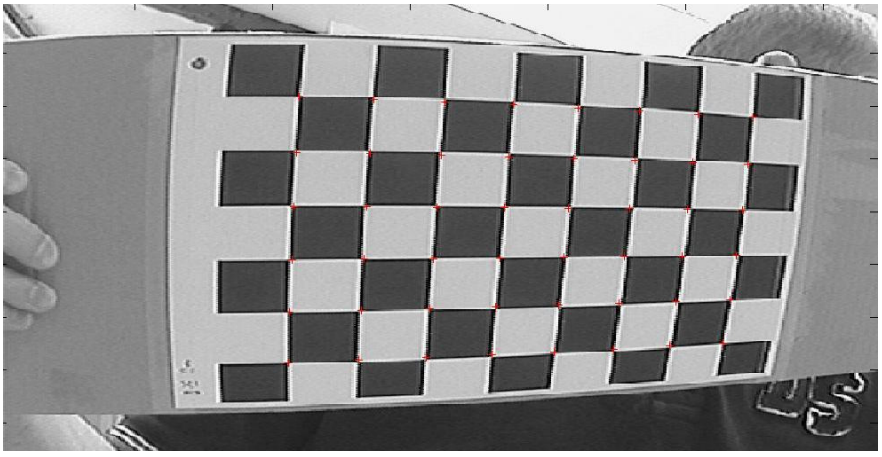
Rys. 2.4.3 Główne menu pakietu do kalibracji

W pierwszej kolejności należało wyznaczyć siatkę narożników szachownicy. Posłużyła do tego funkcja: *extract grid corners*. Na proces ten składało się kilka etapów. Pierwszy z nich polegał na zaznaczeniu czterech skrajnych narożników szachownicy tak jak to pokazano na rys. 2.4.4.



Rys. 2.4.4 Wyznaczanie czterech narożników szachownicy

Przy wyznaczaniu czterech skrajnych narożników szachownicy ważne było ustalenie kolejności zaznaczania i zachowywanie jej w przypadku pozostałych zdjęć. Kolejny etap wiązał się z podaniem liczby kwadratowych pól wzdłuż osi OX i OY powstałego układu współrzędnych szachownicy oraz podaniem długości boku kwadratu wzdłuż osi OX i OY.
Na podstawie zebranych danych algorytm programu MATLAB obliczył współrzędne wewnętrznych narożników szachownicy oraz zaznaczył te narożniki za pomocą czerwonych punktów, tak jak to pokazano na rys. 2.4.5.



Rys. 2.4.5 Pozycje narożników wewnętrznych szachownicy

W przypadku gdy położenia czerwonych znaczników odbiegały od właściwych lokalizacji narożników, konieczne było dokonanie korekcji, aż do uzyskania pełnej zbieżności.

Podobną procedurę przeprowadzono dla pozostałych 34 zdjęć szachownicy. Po wyznaczeniu siatki narożników wszystkich zdjęć szachownicy, możliwe było dokonanie kalibracji kamery. W menu głównym wybrano funkcję *calibration*. MATLAB wykonał odpowiednie obliczenia i zwrócił wartości pożądanych parametrów. W celu poprawienia dokładności wyników użyto funkcji *recomp. corners*, która ponownie przeliczyła położenia narożników na wszystkich zdjęciach kalibracyjnych w cyklu automatycznym. Następnie ponownie wybrano funkcję *calibration* i otrzymano wyniki w formie jak pokazano poniżej:

```
Focal Length:          fc = [ 773.82598   747.94159 ] ± [ 4.17718   3.84015 ]
Principal point:       cc = [ 362.21323   227.53996 ] ± [ 6.32992   4.93806 ]
Skew:             alpha_c = [ 0.00000 ] ± [ 0.00000  ]   => angle of pixel axes =
                                90.00000 ± 0.00000 degrees
Distortion:            kc = [ -0.55551   0.40102   0.00708   -0.00089  0.00000 ]
                          ± [  0.02219   0.13485   0.00193   0.00194  0.00000 ]
Pixel error:          err = [ 0.28520   0.50609 ]
```

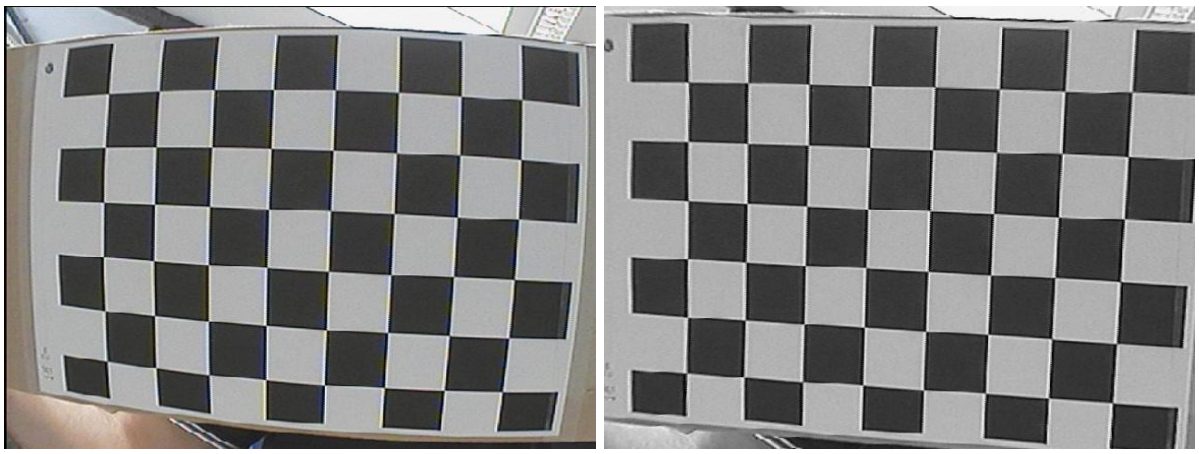
Są to kolejno: długość ogniskowej, punkt środkowy, przekoszenie, współczynniki zniekształceń oraz błąd pikselowy.

Po podstawieniu tych parametrów do postaci ogólnych *macierzy parametrów wewnętrznych M* oraz *wektora zniekształceń D,* otrzymano:

$$M = \begin{bmatrix} 773.82598 & 0 & 362.21323 \\ 0 & 747.94159 & 227.53996 \\ 0 & 0 & 1 \end{bmatrix},$$

$$D = [-0.55551 \quad 0.40102 \quad 0.00708 \quad -0.00089 \quad 0.00000]$$

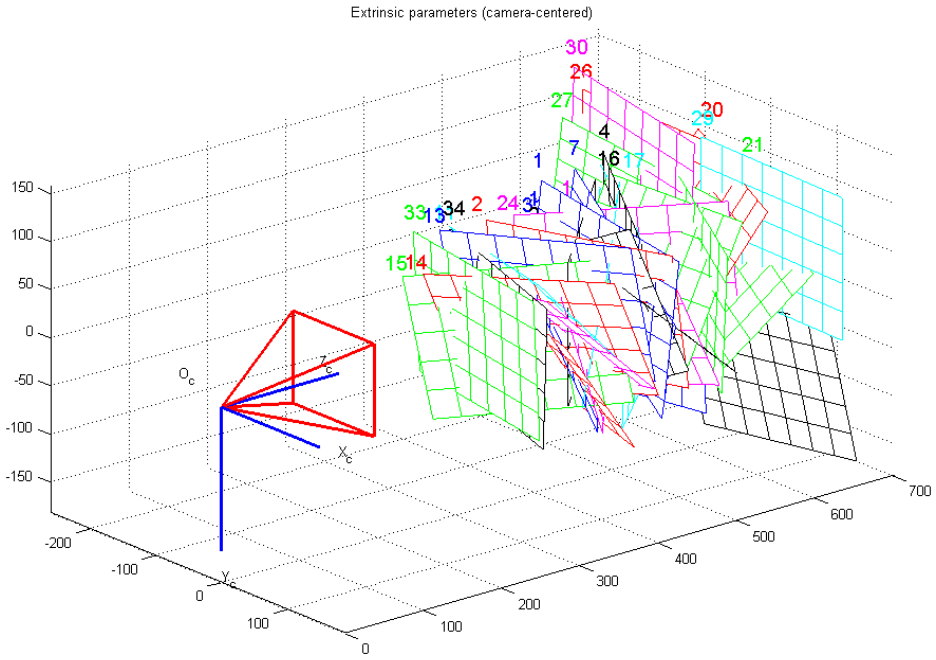
Na rys. 2.4.6 pokazano przykład wykorzystania współczynników z wektora D w celu usunięcia zniekształceń wprowadzanych przez układ optyczny.



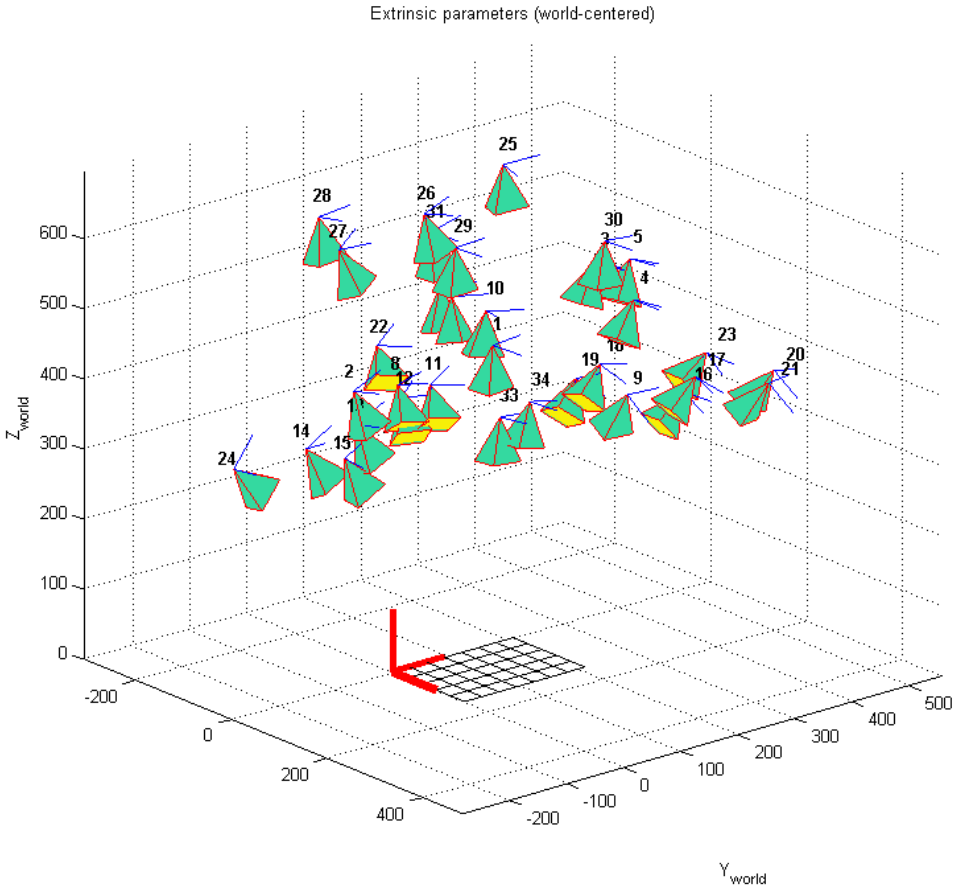
Rys. 2.5.6 Efekt usunięcia zniekształceń obrazu

Za pomocą funkcji *show extrinsic* z menu głównego pakietu kalibracyjnego pokazano przestrzeń 3D związaną z układem kamery i układami szachownic ze zdjęć kalibracyjnych (patrz rys. 2.4.7 oraz 2.4.8). Są to parametry zewnętrzne, określające położenie i orientację jednego obiektu względem drugiego.

Dalsze etapy realizacji projektu związane były z poznaniem położenia i orientacji danego obiektu względem układu kamery, a następnie względem układu bazowego robota. Jednym z celów pracy było chwytanie klocków przez manipulator, w kolejnym rozdziale opisano jak powiązano znany układ współrzędnych kamery z układem współrzędnych globalnych robota, czyli w jaki sposób skoordynowano oko z ręką.



Rys. 2.4.7 Parametry zewnętrzne – kamera w centrum



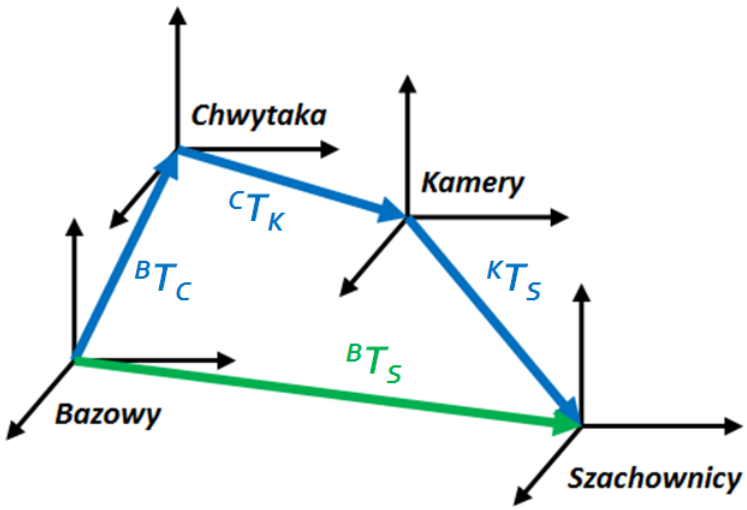
Rys. 2.4.8 Parametry zewnętrzne – szachownica w centrum

3. Koordynacja układu oko-ręka *(Daniel Gozdera)*

W robotyce do matematycznego opisu przestrzeni rzeczywistej powszechnie używa się trójwymiarowych, prawoskrętnych, kartezjańskich układów współrzędnych. Dla określenia położenia punktu w dowolnym układzie wykorzystuje się wektor przestrzenny, którego początek pokrywa się z początkiem układu współrzędnych, zaś koniec znajduje się w danym punkcie. W przyjętej wcześniej strukturze stanowiska wyróżnione zostały następujące układy:

- bazowy (związany z podstawą manipulatora)
- chwytaka (związany z punktem roboczym manipulatora)
- kamery (związany z punktem ogniskowej kamery)
- szachownicy (związany z narożnikiem szachownicy)
- obiektu (związany z narożnikiem klocka sześciennego)

Wszystkie te układy znajdują się w obszarze stanowiska. Układ bazowy jest układem globalnym (statyczny układ odniesienia), natomiast wszystkie pozostałe układy są związane z poszczególnymi obiektami i przemieszczają wraz z nimi.



Rys. 3.1. Struktura układów współrzędnych stanowiska

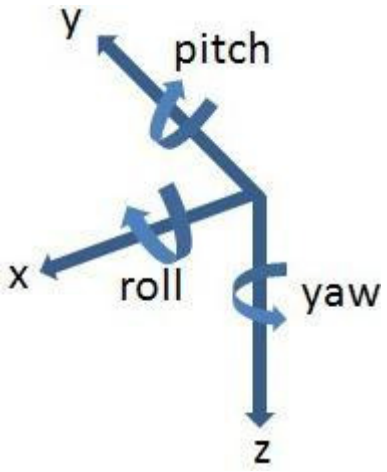
Do określenia relacji pomiędzy aktualnym położeniem dwóch układów współrzędnych stosuje się wektor translacji, natomiast do określenia relacji między orientacją układów stosuje się macierz rotacji.

a) $\begin{bmatrix} x \\ y \\ z \end{bmatrix}$ b) $\begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{bmatrix}$

Rys. 3.2. Ogólna postać: a) wektora translacji b) macierzy rotacji

Translacja jest liniowym przemieszczeniem, a wielkości wektora x, y, z, określają współrzędne początku jednego układu współrzędnych w drugim układzie współrzędnych. Macierz rotacji składa się z 9 elementów które jednoznacznie opisują orientacje osi jednego układu współrzędnych w drugim układzie współrzędnych. W kartezjańskim trójwymiarowym układzie współrzędnych orientacje jednoznacznie można określić za pomocą trzech kątów:

- Yaw (odchylenie kierunkowe) α
- Pitch (pochylenie wzdłużne) β
- Roll (przechył boczny) γ



Rys. 3.3. Orientacja w układzie współrzędnych

Macierz rotacji można uzyskać poprzez złożenie trzech rotacji wokół osi bazowego układu odniesienia:

- Obrotu wokół osi ${}^0OX$ o kąt α
- Obrotu wokół osi ${}^0OY$ o kąt β
- Obrotu wokół osi ${}^0OZ$ o kąt γ

co można zapisać:

$$\boldsymbol{R_{YPR}} = R({}^0OX, \gamma)\cdot R({}^0OY, \beta)\cdot R({}^0OZ, \alpha) = R({}^0OZ, \gamma)\cdot R({}^IOY, \beta)\cdot R({}^{II}OX, \alpha)$$

,gdzie:

0 – pierwotny układ odniesienia

I – układ powstały w wyniku rotacji układu 0 względem os OZ o kat γ

II – układ powstały w wyniku rotacji układu I względem os OY o kat β

YPR – układ powstały w wyniku rotacji układu III względem os OX o kat α.

W celu umożliwienia manipulatorowi lokalizacji obiektów znajdujących się w jego obszarze roboczym, konieczne było wykonanie koordynacji układu „oko-ręka", czyli powiązania ze sobą układów współrzędnych kamery i chwytaka. Układy współrzędnych są powiązane za pomocą transformacji homogenicznych (jednorodnych), opisujących translację i rotację jednego układu względem drugiego.

Rysunek 3.3 przedstawia układy współrzędnych wykorzystywane w procesie koordynacji oraz transformacje między nimi. Dzięki znajomości wszystkich powiązań pomiędzy układami współrzędnych możliwe jest wyznaczenie położenia i orientacji obiektów we wspólnym układzie bazowym. Uzyskana w ten sposób lokalizacja bezwzględna obiektów została wykorzystana w późniejszym etapie do manipulacji nimi.

$$\begin{bmatrix} a_x & b_x & c_x & x \\ a_y & b_y & c_y & y \\ a_z & b_z & c_z & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rys. 3.4 Ogólna postać homogenicznej macierzy transformacji

3.1. Wykonanie koordynacji „oko-ręka"

Koordynacja ma na celu wyznaczenie macierzy transformacji układu kamery względem układu chwytaka ${}^{B}\boldsymbol{T}_{C}$. Została ona obliczona przy użyciu pozostałych transformacji, które zobrazowano na rys. 3.1. Zależności między nimi można opisać równaniem macierzowym:

$${}^{B}\boldsymbol{T}_{S} = {}^{B}\boldsymbol{T}_{C} \cdot {}^{C}\boldsymbol{T}_{K} \cdot {}^{K}\boldsymbol{T}_{S} \qquad \text{, gdzie:}$$

${}^{B}\boldsymbol{T}_{S}$ – macierz transformacji układu szachownicy względem układu bazowego

${}^{B}\boldsymbol{T}_{C}$ – macierz transformacji układu chwytaka względem układu bazowego

${}^{C}\boldsymbol{T}_{K}$ – macierz transformacji układu chwytaka względem układu kamery

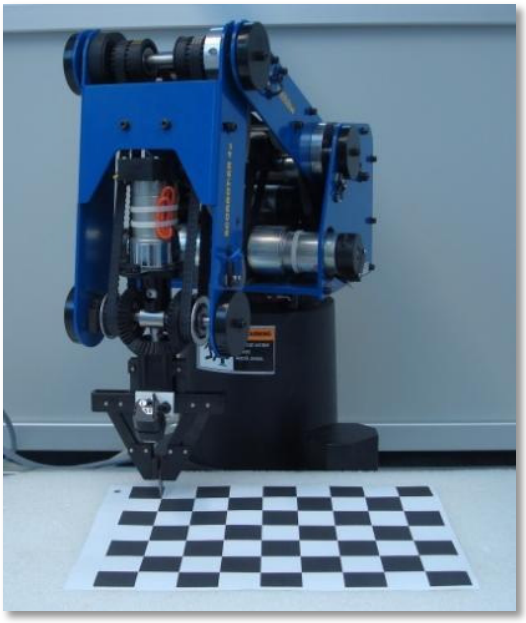
${}^{K}\boldsymbol{T}_{S}$ – macierz transformacji układu kamery względem układu szachownicy

Po przekształceniach otrzymano zależność pozwalającą na obliczenie macierzy koordynacji:

$$^{C}T_{K} = {}^{C}T_{B} \cdot {}^{B}T_{S} \cdot {}^{S}T_{K}$$

$$^{C}T_{K} = (^{B}T_{C})^{-1} \cdot {}^{B}T_{S} \cdot (^{K}T_{S})^{-1} .$$

Następnie wyznaczono macierz układu szachownicy względem układu kamery $^{B}T_{S}$ poprzez ustawienie szachownicy w obszarze roboczym manipulatora w taki sposób aby osie OX, OY, OZ układu szachownicy były równoległe do odpowiadających im osi układu bazowego. Ustawienie szachownicy zrealizowano poprzez wyrysowanie przy użyciu manipulatora z pewnego punktu u jego podstawy linii wzdłuż osi OX i OY, następnie płaszczyznę szachownicy ustawiono pod odpowiednim kątem do podstawy, tak aby pokrywała się z płaszczyzną OXY układu bazowego. Położenie i orientacja zostały sprawdzone przy użyciu narzędzia umieszczonego w chwytaku manipulatora poprzez kolejne zakreślenie na płaszczyźnie szachownicy konturu szachownicy. W ten sposób macierz rotacji $^{B}R_{S}$ upraszcza się do postaci macierzy jednostkowej, a wektor translacji odpowiada początkowi układu współrzędnych szachownicy w układzie bazowym.



Rys. 3.1.1. Pomiar położenia i orientacji układu szachownicy

Współrzędne x, y, z wektora zostały wyznaczone przy użyciu narzędzia umieszczonego w chwytaku manipulatora, a ich wartości odczytane z oprogramowania dostarczonego przez producenta. Za pomocą tego samego oprogramowania został wykonany

pomiar odpowiedniego ustawienia orientacji szachownicy. Pomiar położenia początku układu współrzędnych szachownicy zwrócił wynik:

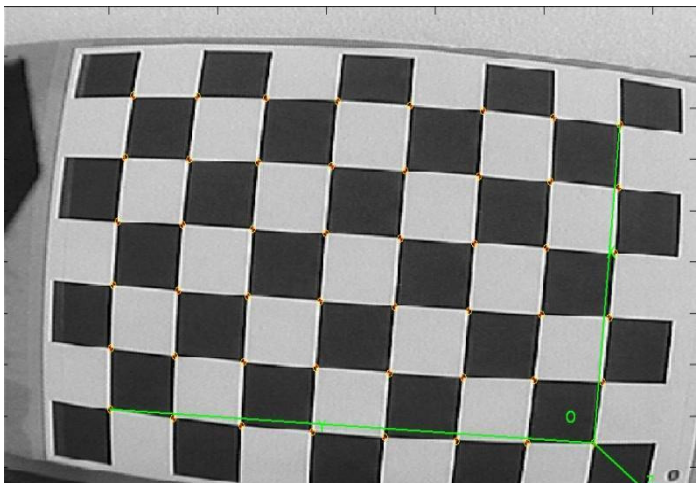
$$X = 232.94 \qquad Y = -73.95 \qquad Z = 98.90$$

$$^{B}\boldsymbol{T}_{S} = \begin{bmatrix} 1 & 0 & 0 & 232.94 \\ 0 & 1 & 0 & -73.95 \\ 0 & 0 & 1 & 98.90 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Kolejną macierzą, która została określona jest macierz transformacji układu szachownicy względem układu kamery $^{K}\boldsymbol{T}_{S}$. Jej wyznaczenie zostało zrealizowane na drodze obliczeń numerycznych przy użyciu oprogramowania dostępnego w środowisku MATLAB jako dodatkowa biblioteka narzędziowa o nazwie „*Camera Calibration Toolbox*" wspomniana w rozdziale 2.4.

Rys. 3.1.2. Wyznaczenie macierzy transformacji $^{K}\boldsymbol{T}_{S}$

Jako informacje wejściowe zostało podane zdjęcie szachownicy zrobione przez kamerę umieszczoną na chwytaku manipulatora oraz parametry uzyskane wcześniej jako wynik kalibracji kamery. Parametry kalibracji pozwalają skorygować wykonane przez kamerę zdjęcie.

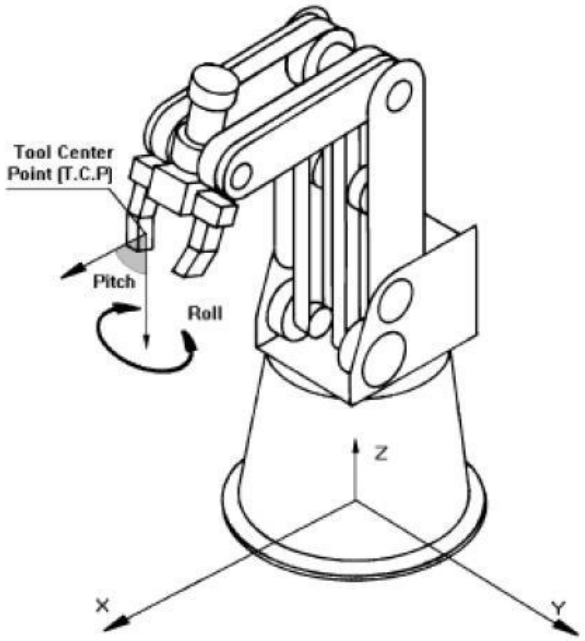


Rys. 3.1.3. Wyznaczania narożników szachownicy

Biblioteka narzędziowa MATLABa wymaga aby na zdjęciu wskazać z pewnym przybliżeniem w odpowiedniej kolejności cztery zewnętrzne narożniki szachownicy, po czym dokonuje dokładnego rozpoznania współrzędnych wszystkich narożników i wylicza przekształcenie pomiędzy ogniskową kamery a płaszczyzna szachownicy. Pierwszy wybrany narożnik oznacza początek układu współrzędnych. Bardzo istotnym parametrem jest podanie długości i szerokości prostokątów, z których złożona jest szachownica w celu określenia odległości od obiektu. Jako wynik obliczeń zwrócona została macierz transformacji o wartościach:

$$ {}^{K}\boldsymbol{T_S} = \begin{bmatrix} 0.070609 & -0.997410 & 0.013714 & 85.743057 \\ -0.997399 & -0.070794 & -0.013510 & 95.458726 \\ 0.014446 & -0.012724 & -0.999815 & 337.456889 \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

Ostatnią macierzą potrzebna do wykonania koordynacji była macierz transformacji układu chwytaka względem układu bazowego ${}^{B}\boldsymbol{T_C}$. Ponieważ oprogramowanie dostarczone przez producenta manipulatora nie umożliwia automatycznego zwrócenia orientacji chwytaka w postaci macierzy rotacji, a jedynie zwraca informacje o aktualnych wartościach kątów Pitch i Roll względem układu nadgarstka, konieczne było wykonanie dodatkowych obliczeń.



Rys. 3.1.4. Układ bazowy i chwytaka manipulatora

W celu rozwiązania problemu szybkiego sprowadzania położenia i orientacji chwytaka do układu bazowego w postaci macierzy transformacji, została stworzona funkcja o pięciu parametrach wejściowych (X, Y, Z, Pitch, Roll) zwracająca gotowy wynik w postaci macierzowej. Pierwszym etapem obliczeń jest zbudowanie macierzy rotacji. Jest ona złożeniem trzech macierzy rotacji: względem osi OZ układu bazowego o kąt γ, następnie względem osi OY nowo powstałego układu o kąt β, oraz względem osi OX kolejnego nowopowstałego układu o kąt α. Rotacje te można zapisać równaniem:

$$\boldsymbol{R_{YPR}} = R({}^{B}OZ, \gamma)\cdot R({}^{I}OY, \beta)\cdot R({}^{II}OX, \alpha) \quad \text{, gdzie:}$$

$$\gamma = \text{arctan2 (Y, X)}$$

$$\beta = \text{- Pitch}$$

$$\alpha = \text{Roll.}$$

$$\boldsymbol{R_{YPR}} = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} = \begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_z \\ a_z & b_z & c_z \end{bmatrix}.$$

Następnie do tak utworzonej macierzy rotacji dołączony zostaje wektor translacji oraz dolny wiersz odpowiedzialny za współczynniki skali.

$$ {}^{B}\boldsymbol{T}_{C} = \begin{bmatrix} \lceil a_x & b_x & c_x \rceil & X \\ \lfloor a_y & b_y & c_y \rfloor & Y \\ \lfloor a_z & b_z & c_z \rfloor & Z \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

Dla tak zdefiniowanej funkcji wprowadzano następujące parametry odpowiadające położeniu i orientacji chwytaka w momencie wykonywania zdjęcia szachownicy:

X = 326.28 Y = 21.80 Z = 372.07 Pitch = -48.31° Roll = 2.47°

Jako wynik otrzymano macierz transformacji o wartościach:

$$ {}^{K}\boldsymbol{T}_{S} = \begin{bmatrix} 0.663620 & -0.034492 & 0.747273 & 326.28 \\ 0.044338 & 0.998993 & 0.006735 & 21.80 \\ 0.746754 & 0.028663 & 0.664482 & 372.07 \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

W ten sposób uzyskane macierze transformacji zostały użyte do koordynacji układu oko-ręka, czyli wyznaczenia macierzy ${}^{C}\boldsymbol{T}_{K}$:

$$ {}^{C}\boldsymbol{T}_{K} = ({}^{B}\boldsymbol{T}_{C})^{-1} \cdot {}^{B}\boldsymbol{T}_{S} \cdot ({}^{K}\boldsymbol{T}_{S})^{-1} = \begin{bmatrix} -0.007607 & -0.654945 & 0.755638 & -54.02 \\ -0.998448 & -0.036707 & -0.041868 & 2.98 \\ 0.055158 & -0.754785 & -0.653649 & 35.99 \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

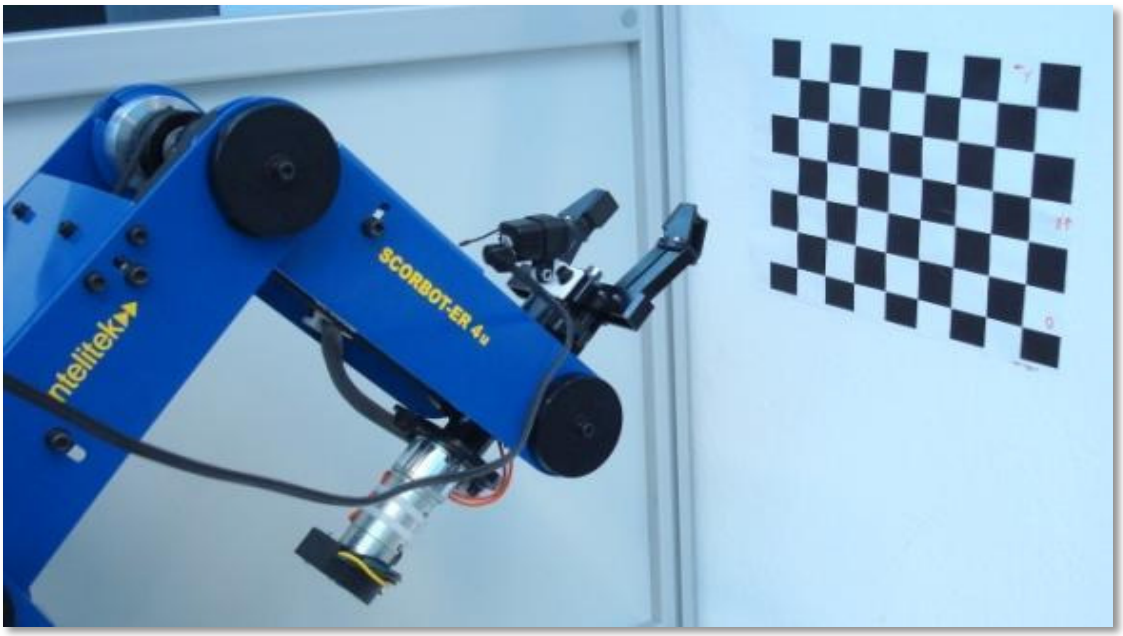
3.2. Automatyczna koordynacja

Kamera została zainstalowana na chwytaku robota za pomocą połączenia śrubowego umożliwiającego regulację. Regulacja pozwala na dopasowanie kierunku obserwacji. Ponosi to za sobą zmianę wartości macierzy układu „oko-ręka” ${}^{C}\boldsymbol{T}_{K}$. Ze względu na to, konieczna jest każdorazowa aktualizacja wartości tej macierzy po uruchomieniu stanowiska.

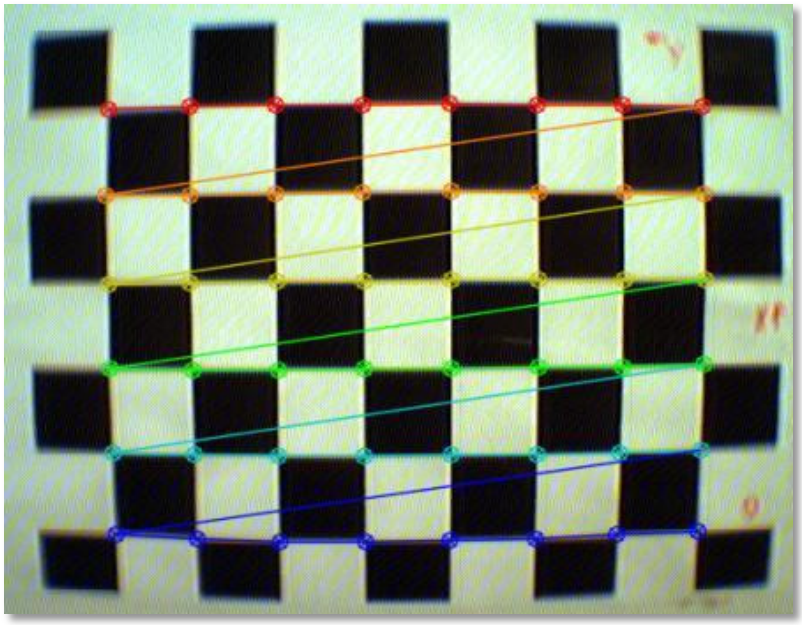
Proces aktualizacji macierzy koordynacji odbywa się tak samo jak opisany w rozdziale 3.1. Jedyną różnicą jest zmiana położenia i orientacji szachownicy wykorzystywanej w tym procesie. Szachownica została zamocowana stacjonarnie na ścianie w pobliżu stanowiska tak, aby znajdowała się w „zasięgu wzroku” kamery. W ten sposób została wyeliminowana

konieczność każdorazowego ustawiania jej w tym samym położeniu, tak aby osie układu szachownicy były równoległe do osi układu bazowego.

Ponieważ szachownice zainstalowano poza obszarem roboczym manipulatora, nie było możliwe wyznaczenie jej położenia na drodze pomiarów przy użyciu narzędzia. W celu uniknięcia niedokładności wynikających z pomiarów metodami technicznymi zdecydowano określić położenie szachownicy na drodze obliczeń macierzowych przy użyciu funkcji biblioteki OpenCV. Posłużyły temu dwie zasadnicze funkcje: „`cvFindChessboardCorners`" oraz „`cvFindExtrinsicCameraParams2`". Ich działanie zastępuje funkcje wykorzystaną w środowisku MATLABa do wyznaczenia macierzy transformacji układu ${}^{K}\boldsymbol{T}_{S}$ opisanej w rozdziale 3.1. Pierwsza funkcja wykryła szachownicę i zwróciła współrzędne czterech narożników. Wynikiem obliczeń drugiej funkcji były translacja i rotacja, czyli poszukiwana transformacja ${}^{K}\boldsymbol{T}_{S}$.



Rysunek 3.2.1. Wyznaczanie macierzy transformacji ${}^{K}\boldsymbol{T}_{S2}$



Rys. 3.2.2. Wyznaczania narożników szachownicy

Za pomocą kamery umieszczonej na manipulatorze wykonano zdjęcie szachownicy znajdującej się na ścianie stanowiska i użyto na nim ww. funkcji do rozpoznania narożników szachownicy i wyznaczenia macierzy transformacji ${}^{K}\boldsymbol{T}_{S2}$. Wynik działania funkcji przedstawiono poniżej.

$$ {}^{K}\boldsymbol{T}_{S2} = \begin{bmatrix} 0.081023 & -0.996464 & 0.022231 & 92.30 \\ -0.996676 & -0.080810 & 0.010325 & 101.13 \\ -0.008492 & -0.022994 & -0.999699 & 366.77 \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

Następnie w sposób omówiony w rozdziale 3.1 wyznaczono na podstawie aktualnego położenia chwytaka macierz transformacji ${}^{B}\boldsymbol{T}_{C2}$.

$$ {}^{B}\boldsymbol{T}_{C2} = \begin{bmatrix} -0.045445 & -0.998376 & 0.034340 & -24.39 \\ 0.764358 & -0.056885 & -0.642277 & 410.22 \\ 0.643188 & -0.002940 & 0.765702 & 427.38 \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

Do wyznaczenia położenia i orientacji szachownicy w układzie bazowym manipulatora wykorzystano zależność:

$$ {}^{B}\boldsymbol{T}_{S2} = {}^{B}\boldsymbol{T}_{C2} \cdot {}^{C}\boldsymbol{T}_{K} \cdot {}^{K}\boldsymbol{T}_{S2} $$

$$ {}^{B}\boldsymbol{T}_{S2} = \begin{bmatrix} 0.040717 & -0.998462 & 0.037611 & 67.14 \\ 0.006466 & -0.037378 & -0.999279 & 712.38 \\ 0.999150 & 0.040931 & 0.004934 & 317.58 \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

Obliczone w ten sposób umiejscowienie szachownicy koordynacyjnej posłużyło w kolejnym etapie pracy do aktualizacji wartości macierzy transformacji układu kamery względem układu chwytaka manipulatora. Aktualizacja wykonywana jest na drodze obliczeń macierzowych poprzez wykonanie zdjęcia szachownicy znajdującej się na ścianie, a następnie wyznaczenie zaktualizowanych wartości macierzy ${}^{C}\boldsymbol{T}_{K2}$ za pomocą zależności:

$$ {}^{C}\boldsymbol{T}_{K2} = ({}^{B}\boldsymbol{T}_{C3})^{-1} \cdot {}^{B}\boldsymbol{T}_{S2} \cdot ({}^{K}\boldsymbol{T}_{S3})^{-1} $$

$$ {}^{C}\boldsymbol{T}_{K2} = \begin{bmatrix} -0.007332 & -0.658327 & 0.752696 & -55.06 \\ -0.999829 & -0.007918 & -0.016666 & 2.68 \\ 0.016931 & -0.752690 & -0.658156 & 35.32 \\ 0 & 0 & 0 & 1 \end{bmatrix}. $$

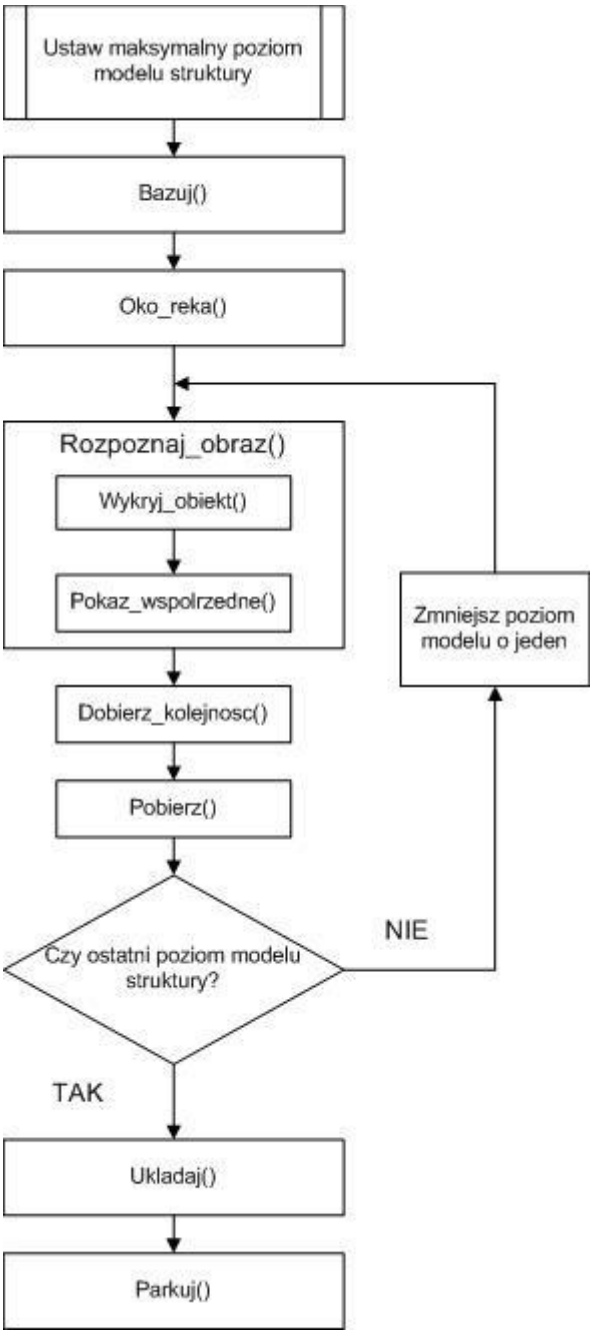
Widać iż macierze ${}^{C}\boldsymbol{T}_{K}$ oraz ${}^{C}\boldsymbol{T}_{K2}$ różnią się nieznacznie wartościami, co w dostępnych warunkach stanowiska jest do zaakceptowania i jednocześnie pozwala stwierdzić poprawność zastosowanej metody automatycznej koordynacji i wykonanych obliczeń.

4. Szkielet algorytmu operacji *(Daniel Gozdera)*

Zrealizowanie celu pracy którym jest współpraca manipulatora z kamerą, oraz manipulacja sześciennymi obiektami zostało przedstawione jako umiejętność ułożenia układanki z sześciennych klocków. Proces ten został podzielony na wykonywane kolejno etapy:

1. Bazowanie manipulatora
2. Koordynacja układu „oko-ręka”
3. Rozpoznawanie obiektów
4. Lokalizacja rozpoznanych obiektów
5. Dobór kolejności dekompozycji rozpoznanych obiektów
6. Dekompozycja struktury zlokalizowanych obiektów
7. Ułożenie obiektów według zdefiniowanego wzorca
8. Parkowanie manipulatora

Ze względu za założony model struktury sześciennych obiektów, trzeci, czwarty, piaty oraz szósty etap procesu jest wykonywany iteracyjnie i powtarzany dla każdego poziomu modelu struktury aż do osiągnięcia ostatniego z nich. Hierarchiczny model struktury obiektów jest podzielony na kolejne poziomy poczynając od najniższego i został on dokładnie przedstawiony w rozdziale numer 7. Szkielet algorytmu wykonywania procesu został zobrazowany na poniższym rysunku, a nazwy poszczególnych jego etapów korespondują z nazwami funkcji (wykorzystywanymi w kodzie programu), które je wykonują.



Rys. 4.1. Szkielet algorytmu operacji

4.1. Opis etapów

Etap „Bazuj()” jest zawsze wykonywany po uruchomieniu stanowiska i ma na celu skoordynowanie ruchów manipulatora. Manipulator kalibruje wszystkie osie przy wykorzystaniu umieszczonych na nich mikro-łączników. Proces kalibracji jest zakańczany ustawieniem manipulatora w pozycji bazowej zdefiniowanej przez producenta. Dzięki temu etapowi możliwe jest zadawanie i określanie pozycji punktu roboczego chwytaka w bazowym, kartezjańskim, trójwymiarowym układzie współrzędnych.

Etap „Oko_ręka()” służy wykonaniu automatycznej koordynacji, opisanej w rozdziale numer 3. Ma ona na celu powiązania ze sobą układów współrzędnych kamery i chwytaka. Jest ona wykonywana także każdorazowo po uruchomieniu stanowiska, aby uniknąć błędów określania lokalizacji obiektów będących w zasięgu pola widzenia kamery. Etap ten wykonywany jest poprzez wykonanie zdjęcia szachownicy znajdującej się na ścianie w pobliżu stanowiska i rozpoznanie jej narożników. Następnie wykonywane są obliczenia mające na celu obliczenie aktualnych wartości elementów macierzy transformacji ${}^{C}\boldsymbol{T_{K2}}$. Cały proces przebiega automatycznie a jego szczegóły zostały przedstawione w poprzednim rozdziale.

Kolejnym procesem jest rozpoznawanie obrazu widzianego w kamerze i wyznaczenie położenia i orientacji wykrytych obiektów. Obsługuje go funkcja o nazwie „Rozpoznaj_obraz()” i składa się z dwóch osobnych etapów, wykonywanych kolejno jeden po drugim.

Funkcja „Wykryj_obiekt()” ma na celu wykonanie zdjęcia obszaru roboczego manipulatora, a następnie poddanie go procesowi rozpoznawaniu obrazu, w taki sposób aby odnaleźć na nim obiekty, które zostały zdefiniowane w założeniach pracy. W przypadku wykrycia tychże obiektów, określane są współrzędne narożników ścian, sześciennych kostek ustawionych frontem do obiektywu kamery. Współrzędne te są wyznaczane jako punkty znajdujące się na zdjęciu w pikselach. Etap ten został dokładnie przedstawiony w rozdziale numer 5.

Kolejny etap wykonywany przez funkcję o nazwie „Pokaż_wspolrzedne()” realizuje drogą obliczeń macierzowych, wyznaczenie współrzędnych rozpoznanych wcześniej narożników obiektu w bazowym układzie współrzędnych manipulatora. Wykonywane jest za pomocą przekształcenia homogenicznego obliczenie współrzędnych narożników obiektu w trójwymiarowym kartezjańskim w układzie kamery. Następnie współrzędne tych punktów są transformowane do układu bazowego manipulatora. Końcową operacją jest wyprowadzenie na ekran monitora obliczonych współrzędnych narożników obiektu oraz jego orientacja w bazowym układzie manipulatora. Etap ten został dokładniej opisany w rozdziale o numerze 6.

Etap realizowany przez funkcję o nazwie „Dobierz_kolejnosc()” ma na celu określenie kolejności oraz sposobu chwytania i pobierania z obszaru roboczego manipulatora rozpoznanych i zlokalizowanych obiektów.

Czynność ta jest wykonywana po to, aby uniknąć podczas manipulacji kolizji robota z wykrytymi obiektami oraz zdekomponować strukturę obiektów możliwie jak najmniejszą liczbą ruchów. Etap ten został dokładnie opisany w rozdziale o numerze 8.

Następnym etapem jest proces wykonywany przez funkcję o nazwie „Pobierz()". W jego skład wchodzi wykonanie manipulacji robota mające na celu dekompozycję struktury rozpoznanych obiektów. Ruchy robota są realizowane zgodnie z wcześniej wyznaczoną kolejnością, a obiekty odkładane na wcześniej zdefiniowane pozycje o określonej orientacji. Algorytm tych operacji został dokładniej opisany w rozdziale o numerze 9.

Etap siódmy, realizowany poprzez funkcję o nazwie "Ukladaj()", składa się z wykonania przez manipulator ruchów zaplanowanych w celu ułożenia struktury obiektów zgodnie z wcześniej zaplanowanym wzorcem. Wykorzystany jest przy tym osobny obszar, w którym manipulator wykonuje operacje mające na celu ustawienie obiektu w odpowiedniej orientacji. Manipulator posiada zaledwie pięć stopni swobody, przez co ustawienie prawidłowej orientacji wymaga wykonania większej liczby operacji niż w przypadku manipulatora o większej liczbie stopni swobody. Cały etap został przedstawiony w rozdziale o numerze 9.

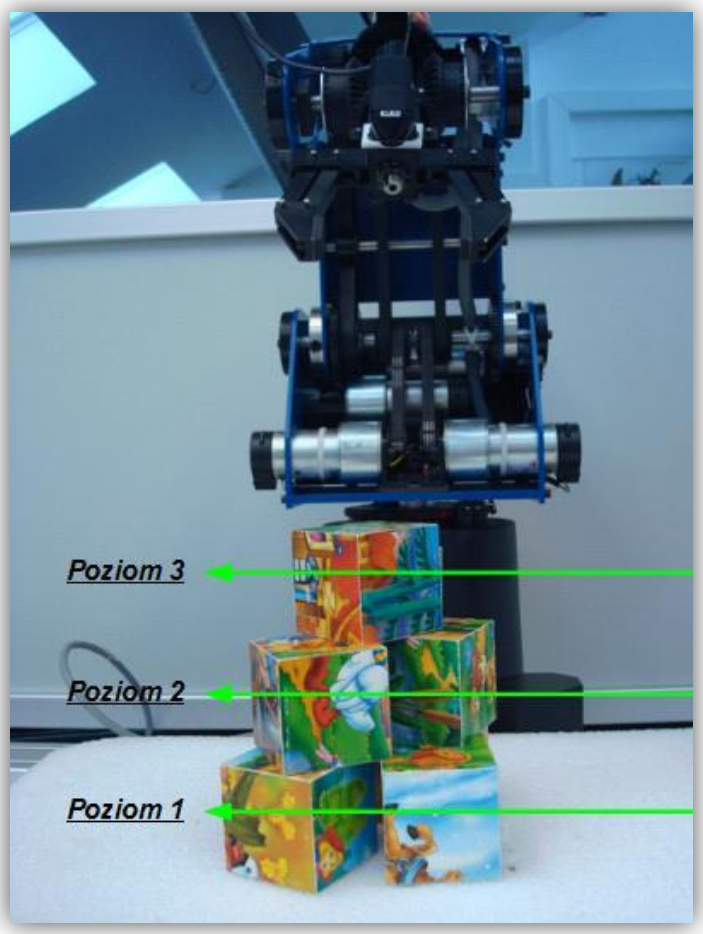
Ostatni etap procesu realizowany funkcją o nazwie „Parkuj" ma na celu ustawienie manipulatora w pozycji bazowej zdefiniowanej przez producenta, tak aby możliwe było bezpieczne wyłączenie stanowiska, a przy jego kolejnym uruchomieniu prawidłowe wykonanie procesu bazowania.

5. Hierarchiczny model struktury *(Daniel Gozdera)*

W celu zaplanowania odpowiedniej kolejności dekompozycji struktury złożonej z pojedynczych obiektów konieczne było założenie pewnego jej modelu. Założono iż obiekty widziane w kamerze będą ułożone jeden na drugim bez użycia dodatkowych elementów. Dzięki temu założeniu możliwe było wydzielenie w modelu poziomów. Została nadana im odpowiednia hierarchia, która decyduje o kolejności dekompozycji. Z przestrzeni roboczej manipulatora została wydzielona pewna przestrzeń mniejszych rozmiarów, w której wykonywane są operacje na strukturze obiektów. Ze względu na niewielką przestrzeń roboczą oraz niedużą liczbę dostępnych obiektów założony został trójpoziomowy model struktury, a obszar na którym będą znajdować się obiekty poddawane procesowi rozpoznawania i manipulacji ograniczony do odpowiednich wymiarów. W celu wykorzystania jak największej dostępnej przestrzeni roboczej zdecydowano, iż struktura będzie umieszczona na odpowiedniej podstawie. Założono także iż zdjęcia obserwowanej sceny będą wykonywane z pozycji o stałych współrzędnych X oraz Y układu bazowego, natomiast współrzędna Z będzie stała dla każdego poziomu struktury i oddalona od niego o 170 mm. Odległość między kamerą, a podstawą struktury została dobrana w taki sposób aby obserwowana scena była możliwie jak największa, przy zachowaniu parametrów obrazu pozwalających na prawidłowe rozpoznawanie na niej obiektów. Zdefiniowano następujące poziomy modelu struktury w układzie bazowym manipulatora:

- Poziom 1 Z = 78,5 mm
- Poziom 2 Z = (78,5 + 55) mm
- Poziom 3 Z = (78,5 +110) mm

gdzie 78,5 mm jest współrzędną Z środka ciężkości obiektu znajdującego się na poziomie 1 w bazowym układzie współrzędnych, natomiast 55 mm określa wysokość pojedynczego obiektu struktury.



Rys. 5.1. Hierarchiczny model struktury

Analogicznie do poziomów struktury zostały zdefiniowane pozycje z których wykonywane są zdjęcia obserwowanej sceny.

- Pozycja zdjęcia poziomu 1

  X = 350 mm , Y = 0 mm , Z = (170+78,5+) mm
- Pozycja zdjęcia poziomu 2

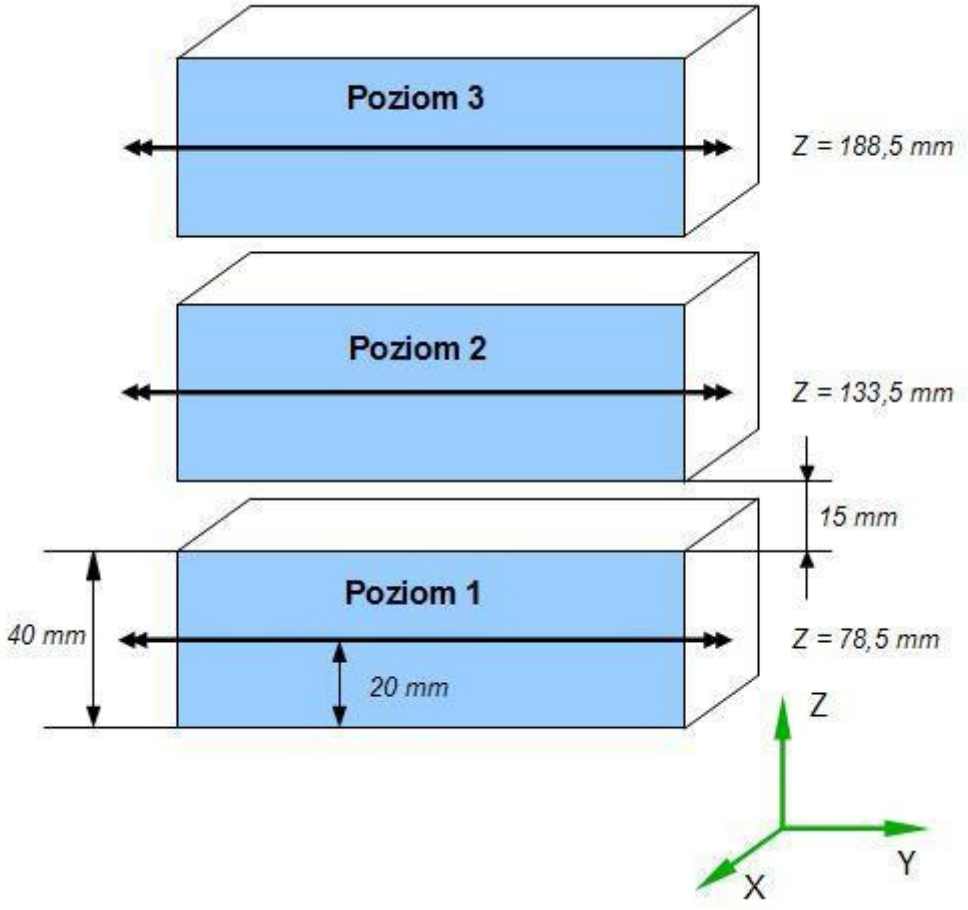
  X = 350 mm , Y = 0 mm , Z = (170+78,5+55) mm
- Pozycja zdjęcia poziomu 3

  X = 350 mm , Y = 0 mm , Z = (170+78,5+110) mm

Zostały one przedstawione, jako położenie punktu roboczego manipulatora w układzie bazowym, przy czym orientacja chwytaka została ustawiona tak aby kamera była skierowana pionowo w kierunku podstawy.

5.1. Klasyfikacja poziomu obiektu struktury

Zdefiniowane trzy poziomy modelu struktury określają na jakich wysokościach mogą znajdować się wykryte przez kamerę obiekty. Po wykryciu obiektu wyznaczane są współrzędne jego górnych narożników, a następnie jego środka ciężkości. Współrzędna na osi Z bazowego układu kartezjańskiego decyduje o tym na jakim poziomi znajduje się dany obiekt. Założone wysokości poziomów opisane w rozdziale 5.1 definiują ich płaszczyznę, natomiast ze względu na niedokładności określania pozycji obiektów w przestrzeni rzeczywistej oraz fakt, iż płaszczyzna podstawy stanowiska nie jest równoległa do płaszczyzny OXY układu bazowego, konieczne było wprowadzenie pewnych tolerancji poziomów. Każdy poziom zdefiniowano z tolerancją o takiej samej wartości. Ze względu na to, iż poziomy są oddalone od siebie o 55 mm w celu jednoznacznego określenia przynależności obiektu do danego poziomu konieczne było, aby tolerancja wynosiła maksymalnie 27,5 mm. Wartość tolerancji została ustawiona na 20 mm, w ten sposób powstał pewien zakres o wysokości 15 mm pomiędzy obszarami poziomów nie przynależący do żadnego z nich, służący do identyfikacji obiektów o nieprawidłowej wysokości.



Rys. 5.1.2. Reprezentacja modelu struktury

Oznaczone na rysunku 5.1.2 kolorem niebieskim obszary reprezentują przestrzeń w której musi się znaleźć punkt środka ciężkości danego obiektu, aby mógł on zostać przyporządkowany do konkretnego poziomu. Jeśli natomiast punkt środka ciężkości znajdzie się w przestrzeni pomiędzy tymi obszarami, obiekt taki zostanie uznany jako nieprawidłowy i nie zostanie mu przyporządkowany żaden z poziomów. Zatem, aby obiekt znalazł się na odpowiednim poziomie, wartość współrzędnej Z jego środka ciężkości musi należeć do jednego z przedziałów:

- Poziom 1 , $Z \in (58{,}5 ; 98{,}5)$ mm
- Poziom 2 , $Z \in (113{,}5 ; 153{,}5)$ mm
- Poziom 3 , $Z \in (168{,}5 ; 208{,}5)$ mm

Manipulacje obiektami odbywają się iteracyjnie, w hierarchii od najwyższego do najniższego poziomu. W jednaj iteracji dekomponowane są wszystkie rozpoznane obiekty należące do danego poziomu, przy czym kolejność dekompozycji obiektów znajdujących się na jednym poziomie jest określana przy użyciu funkcji „Dobierz_kolejnosc()”. Etap doboru kolejności dekomponowania struktury obiektów został opisany w rozdziale o numerze 8.

6. Identyfikacja obiektów na podstawie obrazu *(Paweł Golba)*

Po zakończeniu bazowania osi manipulatora oraz po dokonaniu automatycznej koordynacji „oko ręka” manipulator ustawiał się w zdefiniowanej pozycji. Pozycja ta została tak wybrana, aby kamera umieszczona na chwytaku robota obserwowała trójwymiarową strukturę z klocków z góry. Patrz rys 1.1.1 i rys 6. 1.

Rys.6. 1 Widok z góry struktury, utworzonej przez sześć klocków

Po wykonaniu zdjęcia kamerą umieszczoną na chwytaku manipulatora, kolejnym krokiem było rozpoznanie na tym zdjęciu obiektu, który określono, jako cel chwycenia. Danymi wyjściowymi takiego rozpoznania powinny być współrzędne w pikselach czterech skrajnych narożników obiektu. Dane te były niezbędne do obliczenia macierzy rotacji i wektora translacji danego obiektu względem kamery i późniejszego zlokalizowania obiektu, czyli poznania jego położenia i orientacji w układzie bazowym robota.

Klocki wybrane do projektu to sześcienne bryły, które łącznie prezentują 36 różnych obrazków na swoich ściankach. Wykrywanie klocków w projekcie zrealizowano w oparciu o ilustracje znajdujące się na ściankach klocków. Pliki przechowujące zdjęcia ścianek klocków zapisano w folderze projektu.

Funkcja odpowiadająca za wykrywanie obiektów, posługiwała się tymi zdjęciami, które w dalszej części tej pracy będą określane także, jako zdjęcia wzorcowe.

Zanim zostanie opisany przebieg algorytmu działania funkcji wykryj_objekt(), najpierw należy przybliżyć metodę, która jest częścią zasadniczą tej funkcji.

6.1. Metoda SURF

Ilustracje znajdujące się na ściankach klocków charakteryzują się dużą różnorodnością cech, posiadają mnóstwo punktów charakterystycznych. Właściwość ta pozwoliła na zaproponowanie użycia metody SURF (Speeded Up Robust Features) [5]. Jest to nowoczesna metoda, która pozwala na wykrycie takich cech jak również ich opis – jest zatem detektorem i deskryptorem. SURF, jak sama nazwa ma wskazywać odznacza się szybszym działaniem. To szybsze działanie odniesione jest w stosunku do poprzedniczki metody SURF, mianowicie do metody SIFT (Scale-invariant feature transform), [5, 6].

Zasadę działania tych metod – znajdowanie zależności pomiędzy obrazami, można podzielić na trzy zasadnicze etapy.

Pierwszy z nich wiąże się z wyborem punktów charakterystycznych obrazu w miejscach wyróżniających się, takich jak narożniki, bloby, T - połączenia. Najważniejszą właściwością działania detektora punktów charakterystycznych jest powtarzalność, tj. niezawodne znajdowanie tych samych punktów charakterystycznych w różnych warunkach obserwacyjnych (na różnych zdjęciach tej samej sceny).

Kolejny etap wiąże się z opisem punktów charakterystycznych. Sąsiedztwo każdego punktu charakterystycznego reprezentowane jest przez wektor cech. Deskryptor ten musi być wyróżniający się i jednocześnie odporny na szum, błędy detekcji oraz geometryczne i fotometryczne deformacje.

Ostatecznie, wektory opisujące punkty charakterystyczne powinny do siebie pasować w przypadku różnych zdjęć tej samej sceny. Dopasowanie realizowane jest często na podstawie odległości pomiędzy wektorami. Wymiar deskryptora ma bezpośredni wpływ na czas działania algorytmu. Mniejszy wymiar jest pożądany, gdyż przyspiesza pracę jednak kosztem pogorszenia się cechy, jaką jest wyróżnianie się.

6.2. Zastosowanie SURF do wykrywania klocków

W odniesieniu do obecnego projektu zastosowanie metody SURF polegało najpierw na wyznaczeniu punktów charakterystycznych pewnego obrazu, nazywanego wzorcowym i opisaniu tych punktów. Następnie na zdjęciu przedstawiającym ten sam obraz, jednak wykonanym z innego ujęcia, a zatem po zmianie skali, po zajściu rotacji, także po zmianie oświetlenia i po pojawieniu się innych cech, algorytm pozwalał na ponowne odnalezienie punktów charakterystycznych obrazu wzorcowego. Właściwość ta została wykorzystana do rozpoznawania klocków. Zdjęciem wzorcowym były obrazki ze ścianek klocków, zdjęciem

wykonanym z innego ujęcia – zdjęcie z kamery umieszczonej na chwytaku robota przedstawiającej strukturę z klocków np. rys. 6.1.

Rys.6.2.1 Obrazek odpowiadający jednej ze ścianek klocka zastosowanego w projekcie

W projekcie posłużono się metodą SURF także ze względu na dostępność implementacji tej metody w OpenCV. Na ilustracji 6.2.1 przedstawiona jest jedna ze ścianek jednego z klocków wykorzystanych w projekcie. Poniżej opisane zostanie szczegółowo wykrycie tej ścianki, a zatem i klocka na zdjęciu obserwowanej sceny z rys. 6.1. Zgodnie z metodą SURF najpierw należało wyznaczyć cechy charakterystyczne zdjęcia wzorcowego. Posłużyła do tego celu funkcja z biblioteki OpenCV:

void cvExtractSURF(const CvArr image, const CvArr* mask,*

*CvSeq** keypoints, CvSeq** descriptors,*

CvMemStorage storage, CvSURFParams params);*

Parametry tej funkcji oznaczają kolejno:

- *image* – jest to 8 bitowe zdjęcie wejściowe w skali szarości,
- *mask* – parametr opcjonalny, 8 bitowa maska; cechy będą znajdowane tylko w obszarach, które zawierają więcej niż 50% niezerowej maski pikselowej,
- *keypoints* - parametr wyjściowy, jest to podwójny wskaźnik do sekwencji punktów charakterystycznych. Sekwencja ta będzie składać się ze struktur CvSURFPoint przechowujących takie informacje jak opisano poniżej:
  - *typedef struct CvSURFPoint*

    {

    CvPoint2D32f pt; // położenie cechy w obszarze zdjęcia

    int laplacian; // -1, 0, lub +1 znak laplasjan w punkcie, wykorzystywany do porównania cech,

    int size; // rozmiar cechy

float dir; // orientacja cechy: 0 – 360 stopni

float hessian; // wartość hesjana, może być użyta do oszacowania w przybliżeniu stopnia nasilenia cechy

}

- descriptors – opcjonalny parametr wyjściowy, podwójny wskaźnik do sekwencji deskryptorów. W zależności od wyboru ustawień w strukturze CvSURFParams params z funkcji cvExtractSURF, może być 64-elementowym lub 128-elementowym wektorem punktów. Jeżeli descriptors zostanie ustawiony na NULL, wówczas deskryptory nie będą obliczane,
- storage – magazyn pamięci przechowujący punkty charakterystyczne i deskryptory,
- params – zestaw parametrów przechowywanych w strukturze CvSURFParams. Najważniejsze z nich to:
  - *typedef struct CvSURFParams*

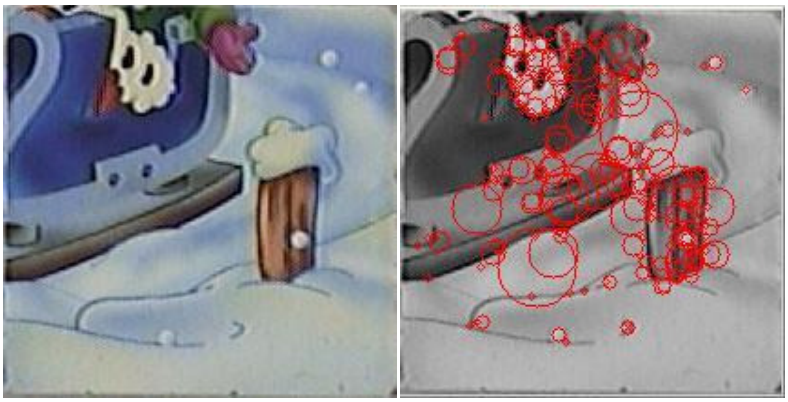
    {

    int extended; // 0 oznacza podstawowe deskryptory (64 elementowy każdy)
    1 oznacza rozszerzone deskryptory (128 elementowy każdy)

    double hessianThreshold; // próg hesjanowy, cechy posiadające wartość hesjana większą od tego progu zostaną wyznaczone

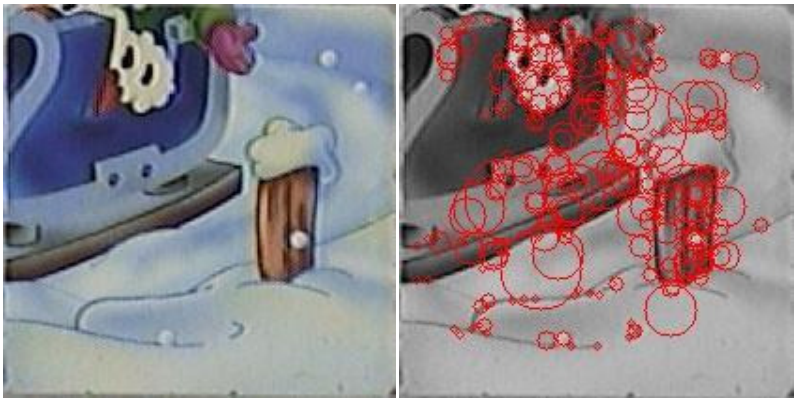
    }

W projekcie postanowiono wykorzystać metodę SURF jako detektor i jednocześnie deskryptor. Wybrano zatem także użycie parametru wyjściowego descriptors.
Ilustracja 6.2.2 pokazuje efekt działania ekstrakcji cech dla progu hesjanowego 660 oraz deskryptorów rozszerzonych: *CvSURFParams params = cvSURFParams(660, 1).*



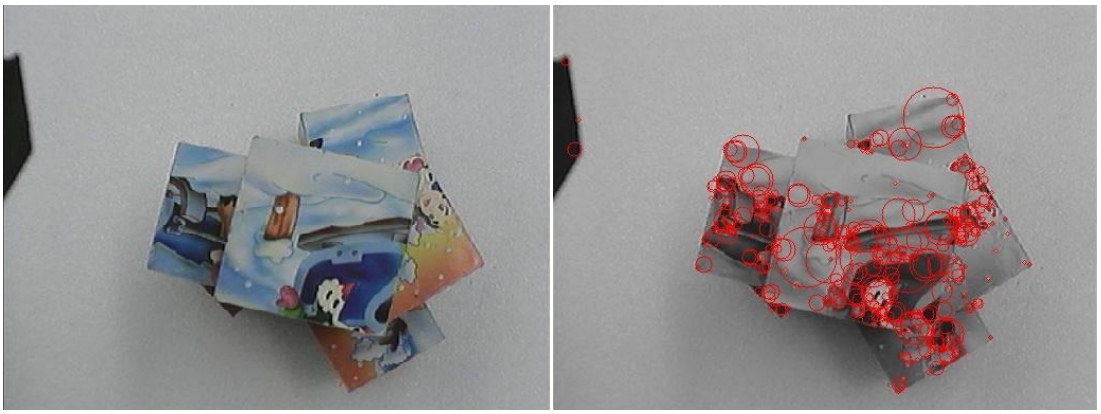
Rys.6.2.2 Ekstrakcja cech zdjęcia wzorcowego dla progu hesjanowego 660

Otrzymano 239 punktów charakterystycznych. Dla porównania przy wyborze progu hesjanowego 360 uzyskano 313 punktów charakterystycznych tak jak to pokazano na rys 6.2.3.



Rys.6.2.3 Ekstrakcja cech zdjęcia wzorcowego dla progu hesjanowego 360

W identyczny sposób wyznaczono punkty charakterystyczne dla zdjęcia obserwowanej sceny z kamery manipulatora i otrzymano 409 cech, tak jak pokazano na rys. 6.2.4.



Rys.6.2.4 Ekstrakcja cech zdjęcia obserwowanej sceny dla progu hesjanowego 660

Kolejnym krokiem było porównanie obu zdjęć i znalezienie punktów charakterystycznych pasujących do siebie. Następnie należało wyznaczyć cztery narożniki klocka na obrazie obserwowanej sceny. Do tego celu wykorzystano funkcję:

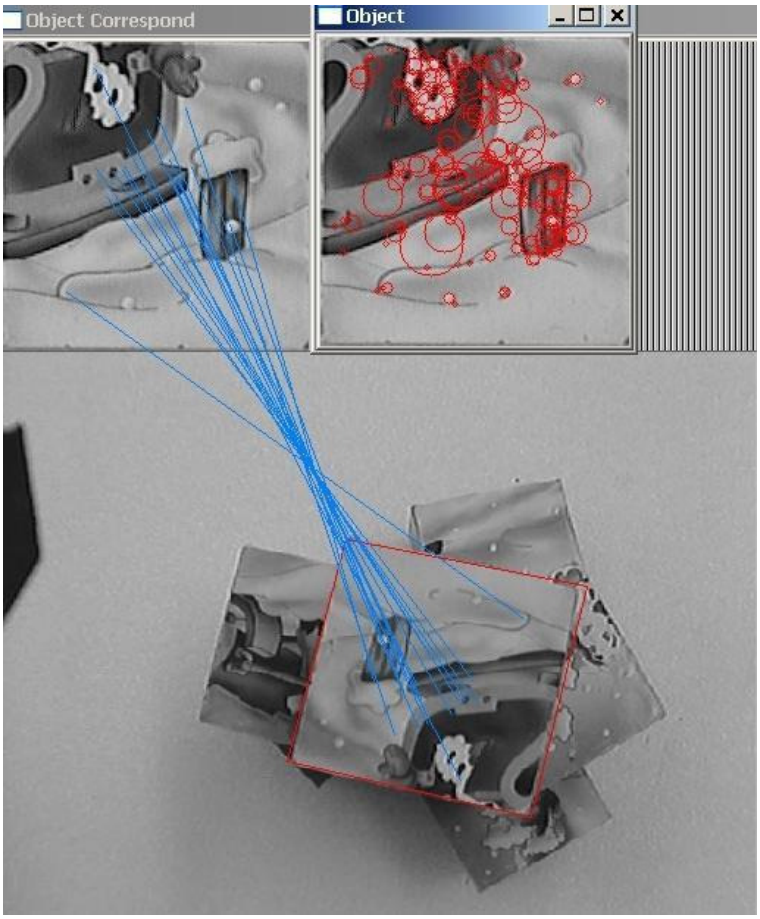
locatePlanarObject (const CvSeq objectKeypoints, const CvSeq* objectDescriptors,*
const CvSeq imageKeypoints, const CvSeq* imageDescriptors,*
const CvPoint src_corners[4], CvPoint dst_corners[4])

Parametrami tej funkcji są:

- objectKeypoints – punkty charakterystyczne zdjęcia wzorcowego,
- objectDescriptors – deskryptory punktów charakterystycznych zdjęcia wzorcowego,
- imageKeypoints – punkty charakterystyczne zdjęcia obserwowanej sceny,
- imageDescriptors – deskryptory punktów charakterystycznych zdjęcia obserwowanej sceny,
- src_corners[4] – współrzędne narożników zdjęcia wzorcowego,
- dst_corners[4] – parametr wyjściowy, tu będą znajdowały się współrzędne czterech narożników poszukiwanego obiektu na zdjęciu obserwowanej sceny.

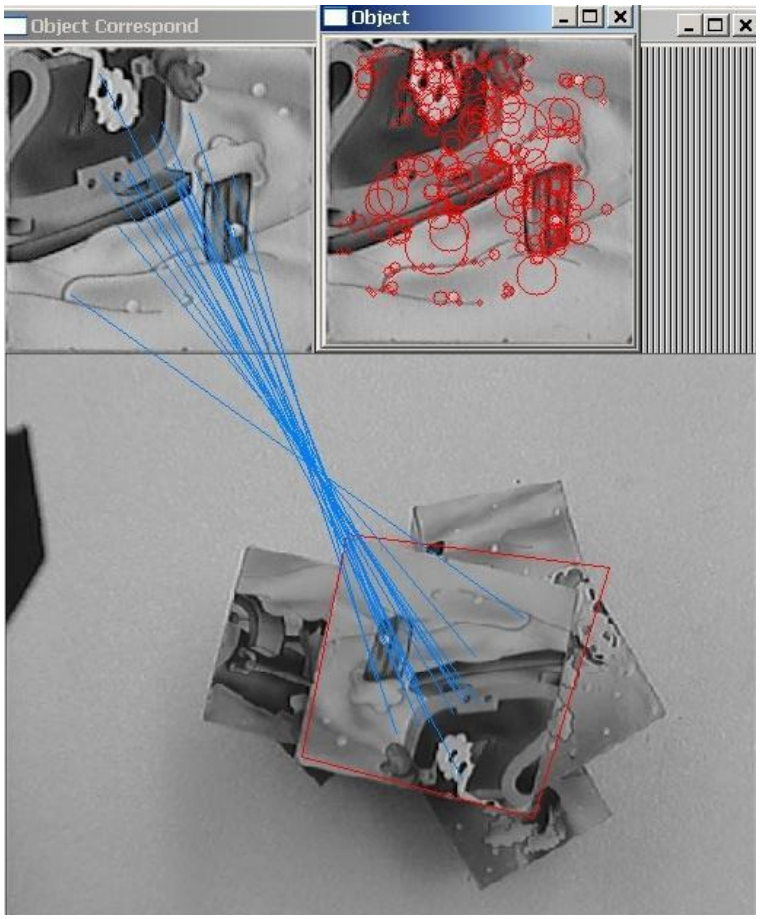
Funkcja ta, po porównaniu obu zdjęć i dopasowaniu cech odpowiadających sobie, znalazła poszukiwany klocek, a właściwie przekształcenie homograficzne pomiędzy płaszczyzną klocka ze zdjęcia wzorcowego a płaszczyzną klocka ze zdjęcia obserwowanej sceny. Znając macierz homografii i rozmiar klocka, czyli współrzędne narożników zdjęcia wzorcowego, funkcja poprzez odpowiednie obliczenia zwróciła poszukiwane dane wyjściowe. Na rys. 6.2.5 pokazany jest efekt działania zastosowanych operacji.

W celu lepszego zaobserwowania poprawnego wykrycia, znaleziony klocek został zakreślony czerwonymi liniami.



Rys.6.2.5 Wykrycie klocka na zdjęciu obserwowanej sceny dla progu hesjanowego 660

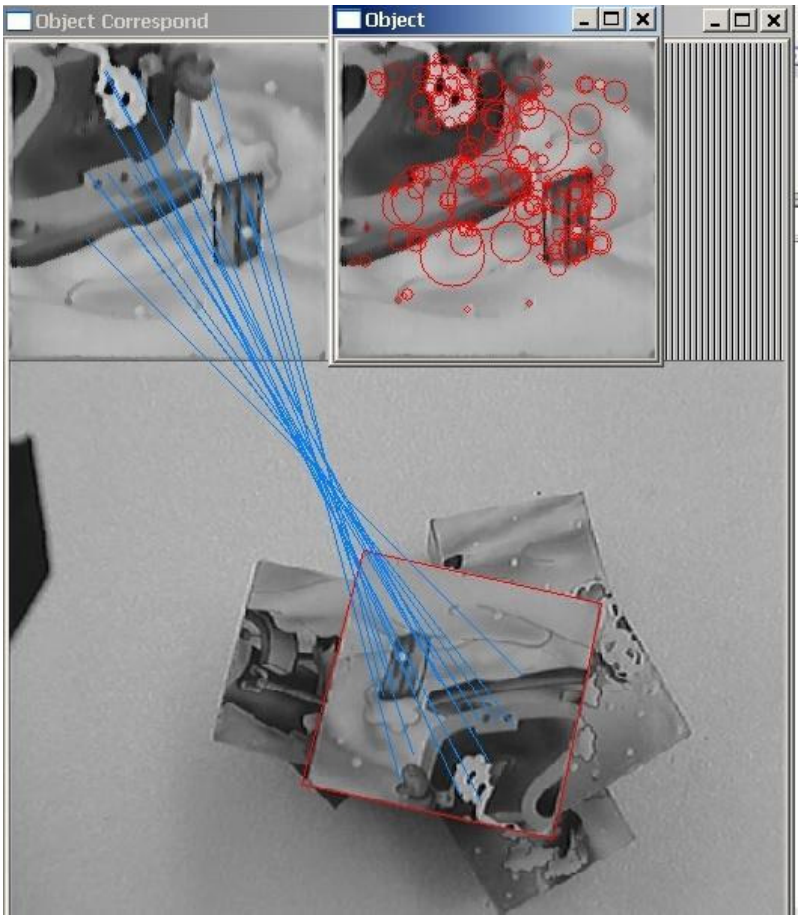
Wartość progu hesjanowego ma wpływ na skuteczność wykrywania obiektów. Przykładowo przeprowadzenie wykrycia na podstawie tych samych zdjęć, jednak z zastosowaniem progu 360 daje efekt tak pokazano na rys. 6.2.6



Rys.6.2.6 Wykrycie klocka na zdjęciu obserwowanej sceny dla progu hesjanowego 360

Dalsze badania pokazały, że na skuteczność wykrycia ma wpływ wiele czynników zewnętrznych, takich jak oświetlenie, stopień oddalenia kamery, znaczna zmiana położenia kamery, jakość przechwyconego obrazu, jak również stopień zróżnicowania cech na obrazie, który ma być wykryty. Lepsze wyniki dopasowania zaobserwowano przy zastosowaniu filtru medianowego. Zachowując te same warunki zewnętrzne i próg hesjanowy, lecz stosując filtrację obu obrazów, zarówno wzorcowego jak również obserwowanej sceny, otrzymano znacznie lepsze dopasowanie tak jak to pokazano na rys. 6.2.7. Dokładne dopasowanie to także przyszłe dokładne zlokalizowanie obiektu i chwycenie go. W trakcie doboru parametrów, przy zmianie warunków zewnętrznych oraz poszczególnych ścianek klocków pojawiały się także przypadki, w których wykrycie w ogóle nie występowało.
Należało zatem odpowiednio dopasować stopień filtru i próg hesjanowy aby zminimalizować błąd polegający na nie wykryciu klocka.

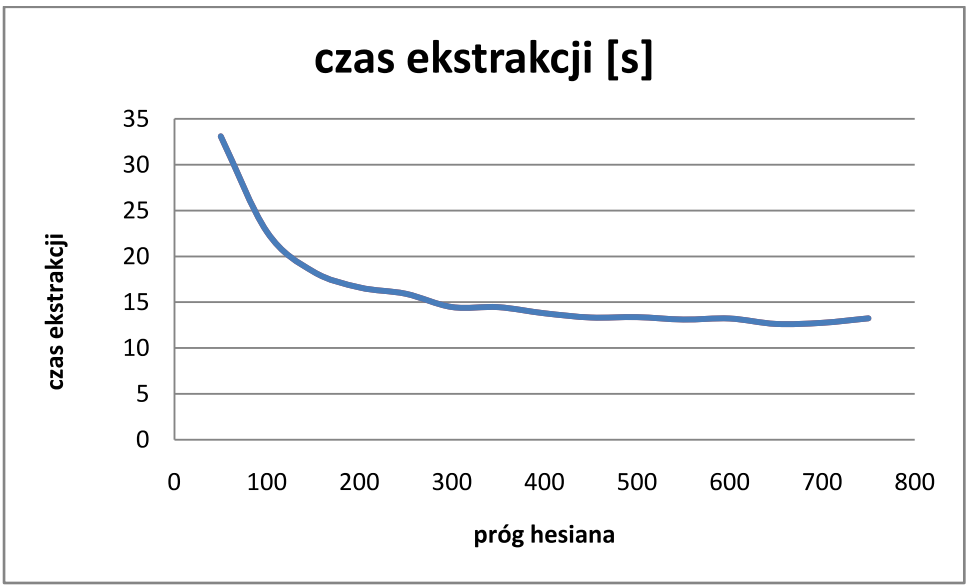
Zrealizowano to poprzez napisanie odpowiedniej aplikacji, która miała za zadanie, w trybie automatycznym, sprawdzenie przy której wartości progu hesjanowego skuteczność wykrywania klocka była najwyższa oraz przy której najkrócej trwała ekstrakcja i porównanie cech.



Rys.6.2.7 Wykrycie klocka dla progu hesjanowego 360 i z zastosowaniem filtru medianowego

Dla każdej wartości progu wykonywano 100 zdjęć obserwowanej sceny, z zachowaniem niezmienionych warunków otoczenia oraz z tej samej pozycji kamery. Obiektem poszukiwań został wybrany klocek ze ścianką przedstawiającą najbardziej jednostajną ilustrację, tak żeby był to klocek najtrudniejszy do wykrycia. Wykresy 6.2.8 i 6.2.9 przedstawiają wyniki eksperymentu przy zmianie progu hesjanowego co 50. W zakresie progu 250 – 350 zanotowano najwięcej sukcesów jednocześnie przy czasie ekstrakcji odpowiadającym ustabilizowanemu poziomowi.

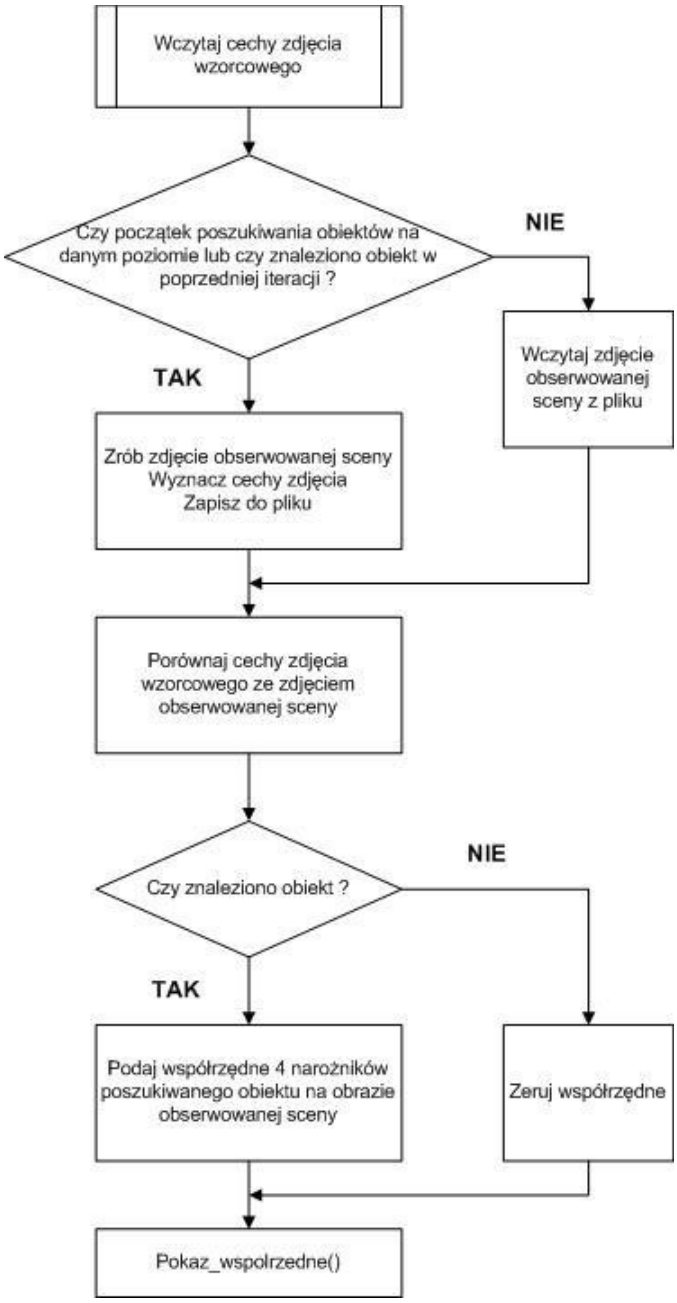
Rys.6.2.8 Liczba wykryć klocka na 100 zdjęć w zależności od progu hesjanowego



Rys.6.2.9 Czas ekstrakcji cech dla 100 zdjęć w zależności od progu hesjanowego

6.3. Organizacja funkcji *wykryj_objekt()*

Uproszczony schemat blokowy algorytmu funkcji *wykryj_objekt()* przedstawiono na rys 6.3.1. Funkcja *wykryj_objekt()* wraz z funkcją *pokaz_wspolrzedne()* są wywoływane w projekcie cyklicznie aż do momentu rozpoznania wszystkich klocków znajdujących się na zdefiniowanej warstwie. Gdy to nastąpi robot w odpowiedniej kolejności dekomponuje klocki z danej warstwy, a następnie obniża wysokość ramienia, tak aby możliwe było rozpoczęcie rozpoznawania klocków na kolejnej, niższej warstwie struktury.



Rys.6.3.1 Uproszczony schemat blokowy funkcji wykryj_objekt()

Na początku funkcji wykryj_objekt() wczytywane są z pliku wyznaczone wcześniej cechy kolejnego zdjęcia wzorcowego. Pod pojęciem cech, kryją się punkty charakterystyczne i deskryptory tych punktów. Dane te są pobierane z pliku, ponieważ ekstrakcja cech zajmuje więcej czasu niż wczytanie wyznaczonych wcześniej cech z pliku. Następnie sprawdzany jest warunek czy to jest początek poszukiwania obiektów na danej warstwie lub czy w poprzedniej iteracji znaleziono obiekt. Jeżeli któryś z tych warunków jest spełniony wówczas przechwytywane jest zdjęcie obserwowanej warstwy struktury, wyznaczane są cechy tego

zdjęcia i zapisywane do pliku. Jeśli któryś z tych warunków nie jest spełniony, pobierane są cechy zdjęcia obserwowanej sceny z pliku (wyznaczone w poprzedniej iteracji).

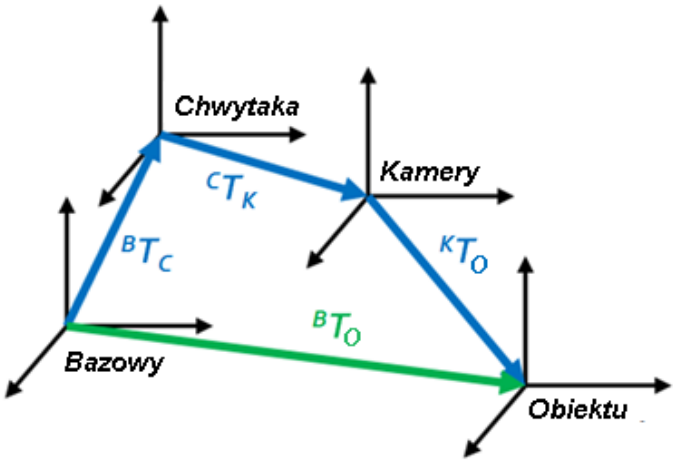
Kolejny krok to porównanie zdjęcia wzorcowego ze zdjęciem obserwowanej sceny. Jeżeli poszukiwanie zakończy się sukcesem, wtedy zwracane są współrzędne 4 narożników aktualnie poszukiwanego obiektu na obrazie obserwowanej sceny. W przeciwnym przypadku 4 narożniki są zerowane.

Warunki sprawdzane w algorytmie są wynikiem założenia, że poszukiwanie obiektów na danej warstwie ma się odbywać na podstawie jednego zdjęcia. Założenie to uzasadniono szybszym działaniem analizy obrazu. Każdorazowa ekstrakcja cech zdjęcia obserwowanej sceny wiązałaby się ze znacznym wydłużeniem działania programu. W przypadku znalezienia obiektu w kolejnej iteracji ponownie przechwytywane jest zdjęcie obserwowanej sceny i poszukiwany jest ponownie obiekt z poprzedniej iteracji, ponieważ założono, że wyniki lokalizacji powinny być uśrednione. Ma to na celu wyeliminowanie losowych, mniej dokładnych wykryć. Jednak ponownie ze względu na ograniczenie czasu przetwarzania obrazu, przyjęto tylko 3 składowe do średniej. Obliczanie średniej dokonuje się w kolejnej funkcji pokarz_wspolrzedne() i proces ten dokładnie zostanie przybliżone w kolejnym rozdziale.

Głównymi zadaniami funkcji z następnego rozdziału jest obliczenie położenia i orientacji wykrytych obiektów trójwymiarowej struktury w układzie bazowym robota oraz dokonanie sprawdzenia czy wykrycia obiektów są prawidłowe.

7. Lokalizacja i klasyfikacja rozpoznanych obiektów *(Paweł Golba)*

Wyznaczono już parametry wewnętrzne kamery, macierz przekształcenia kamery względem chwytaka ${}^{C}T_{K2}$, współrzędne 4 narożników poszukiwanego klocka. Te wszystkie dane wraz ze znajomością rzeczywistego rozmiaru klocka, są wystarczające aby można było zlokalizować klocek, czyli poznać jego położenie i orientację w układzie bazowym robota. To zagadnienie jest podobne do koordynacji układu oko–ręka z tą różnicą, że teraz poszukiwana jest transformacja układu obiektu względem układu bazowego robota ${}^{B}T_{O}$ tak jak to pokazano na rys. 7.1.



Rys. 7.1. Lokalizacja obiektu w układzie bazowym robota

7.1. Wyznaczenie macierzy ${}^{B}T_{O}$

Zgodnie z rys. 7.1. poszukiwaną macierz można obliczyć według poniższej zależności:

$$ {}^{B}T_{O} = {}^{B}T_{C} \cdot {}^{C}T_{K} \cdot {}^{K}T_{O} \quad (7.1.1), \text{ gdzie:} $$

${}^{B}T_{O}$ – macierz transformacji układu obiektu względem układu bazowego

${}^{B}T_{C}$ – macierz transformacji układu chwytaka względem układu bazowego

${}^{C}T_{K}$ – macierz transformacji układu kamery względem układu chwytaka

${}^{K}T_{O}$ – macierz transformacji układu obiektu względem układu kamery

Macierz ${}^{B}T_{C}$ została obliczona w analogiczny sposób jak w rozdziale 3, z tym że zmianie uległy dane związane z aktualną pozycją chwytaka, czyli pozycją z której wykonywane jest aktualnie zdjęcie obserwowanej sceny.

Macierz ${}^{C}T_{K}$, to znana macierz która jest wynikiem koordynacji układu oko–ręka. Do wyznaczenia macierzy ${}^{K}T_{O}$ posłużyła funkcja z OpenCV:

void cvFindExtrinsicCameraParams2(
const CvMat object_points,*
const CvMat image_points,*
const CvMat intrinsic_matrix,*
const CvMat distortion_coeffs,*
CvMat rotation_vector,*
CvMat translation_vector*
);

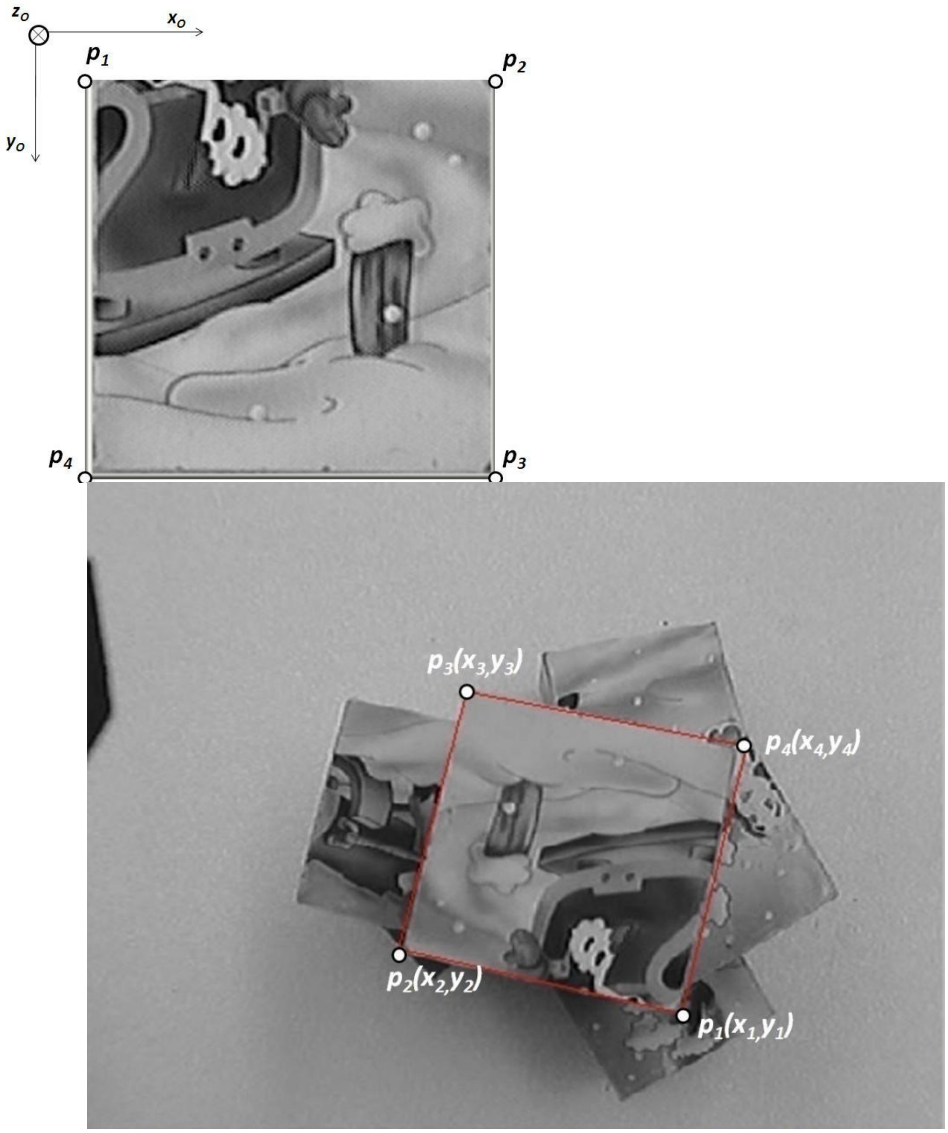
Funkcja ta poszukuje parametrów zewnętrznych, czyli rotacji i translacji obiektu planarnego względem układu kamery. W rozdziale 2 tym obiektem planarnym była szachownica, tu natomiast zastępuje ją płaszczyzna ścianki klocka. Działanie tej funkcji przybliżają jej parametry:

- *object_ points* – parametr wejściowy, macierz 4 x 3 zawierająca rzeczywisty rozmiar klocka. Są to współrzędne 4 narożników obserwowanej ścianki klocka podane w mm. Układ współrzędnych klocka pokazany jest na rys. 7.1.1. Ponieważ zmierzony bok sześcianu wynosi 55 mm, macierz ta ma zawsze postać:

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 55 & 0 \\ 55 & 55 & 0 \\ 55 & 0 & 0 \end{bmatrix}$$

- *image_points* – parametr wejściowy, macierz 4 x 3 zawierająca rozmiar klocka w pikselach z obrazu obserwowanej sceny. Są to współrzędne 4 narożników poszukiwanego klocka wyznaczone w procesie wykrywania obiektu metodą SURF, patrz rys. 7.1.1:

$$\begin{bmatrix} x_1 & y_1 & 0 \\ x_2 & y_2 & 0 \\ x_3 & y_3 & 0 \\ x_4 & y_4 & 0 \end{bmatrix}$$



Rys. 7.1.1 Współrzędne 4 narożników wykrytego klocka z oznaczeniem układu współrzędnych klocka

- *intrinsic_matrix* – parametr wejściowy, macierz parametrów wewnętrznych kamery, wyznaczona podczas kalibracji kamery:

$$M = \begin{bmatrix} 773.82598 & 0 & 362.21323 \\ 0 & 747.94159 & 227.53996 \\ 0 & 0 & 1 \end{bmatrix},$$

- *distortion_coeffs* – parametr wejściowy, wektor zniekształceń układu optycznego kamery, wyznaczony podczas kalibracji kamery:

$$D = [-0.55551 \quad 0.40102 \quad 0.00708 \quad -0.00089 \quad 0.00000]$$

- *rotation_vector* – parametr wyjściowy, wektor rotacji przechowujący kąty obrotu układu współrzędnych związanego z klockiem względem układu współrzędnych kamery:

[alfa beta gamma]

- *translation_vector* – parametr wyjściowy, wektor translacji określający przesunięcie układu współrzędnych związanego z klockiem względem układu współrzędnych kamery wzdłuż osi x, y, z kamery:

[x y z]

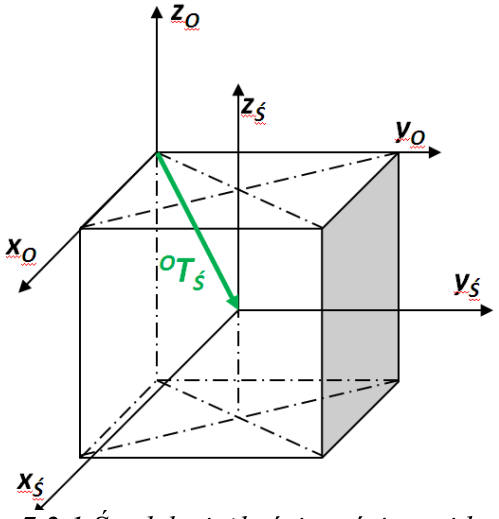
Wektor rotacji sprowadzono do postaci ogólnej macierzy rotacji za pomocą funkcji z OpenCV *cvRodrigues2,* która jako parametr wejściowy przyjmuje wektor rotacji, a jako parametr wyjściowy zwraca macierz rotacji. Powstałą rotację i translację obiektu względem kamery należało złożyć do postaci macierzy we współrzędnych homogenicznych:

$$^{K}\boldsymbol{T_O} = \begin{bmatrix} a_x & b_x & c_x & x \\ a_y & b_y & c_y & y \\ a_z & b_z & c_z & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Podstawienie powyższych macierzy do wzoru (7.1.1) doprowadzi do osiągnięcia transformacji układu klocka względem układu bazowego robota.

7.2. Chwycenie klocka – współrzędne i orientacja chwytaka

Należało jeszcze obliczyć współrzędne środka ciężkości klocka x, y, z w układzie bazowym robota oraz kąty pitch i roll odpowiadające za właściwe zorientowanie chwytaka względem klocka. Za pomocą tych 5 parametrów możliwe jest zadanie położenia i orientacji chwytaka manipulatora, celem chwycenia obiektu. Osiągnięcie położenia środka ciężkości sześciennej bryły o długości krawędzi 55 mm, w układzie współrzędnych związanych z narożnikiem klocka, następuje przez 3 translacje o 55/2 mm, tak jak to pokazano na rys. 7.2.1.



Rys. 7.2.1 Środek ciężkości sześciennej bryły

$$ {}^{O}T_{Ś} = \begin{bmatrix} 1 & 0 & 0 & 55/2 \\ 0 & 1 & 0 & 55/2 \\ 0 & 0 & 1 & -55/2 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

Przekształcenie ${}^{O}T_{Ś}$ to macierz transformacji układu środka ciężkości względem układu obiektu. Położenie środka ciężkości klocka w układzie bazowym, określają zatem współrzędne x, y, z macierzy ${}^{B}T_{Ś}$ będącej wynikiem równania:

$$ {}^{B}T_{Ś} = {}^{B}T_{O} \cdot {}^{O}T_{Ś} = \begin{bmatrix} a_x & b_x & c_x & x \\ a_y & b_y & c_y & y \\ a_z & b_z & c_z & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7.2.1) $$

Na podstawie tak otrzymanej macierzy transformacji ${}^{B}T_{Ś}$, wyznaczono kąty α, β, γ będące kolejno kątami obrotu wokół osi OZ, OY i OX układu bazowego robota. Poniższa zależność określa ogólną postać macierzy rotacji:

$$ R_{YPR} = R({}^{0}OX, \gamma) \cdot R({}^{0}OY, \beta) \cdot R({}^{0}OZ, \alpha) = \begin{bmatrix} a_x & b_x & c_x \\ a_y & b_y & c_y \\ a_z & b_z & c_z \end{bmatrix} = $$

$$ = \begin{bmatrix} \cos\beta\cos\alpha & \sin\gamma\sin\beta\cos\alpha - \cos\gamma\sin\alpha & \cos\gamma\sin\beta\cos\alpha + \sin\gamma\sin\alpha \\ \cos\beta\sin\alpha & \sin\gamma\sin\beta\sin\alpha + \cos\gamma\cos\alpha & \cos\gamma\sin\beta\sin\alpha - \sin\gamma\cos\alpha \\ -\sin\beta & \sin\gamma\cos\beta & \cos\gamma\cos\beta \end{bmatrix}, \quad (7.2.2) $$

Zatem:

$$ \alpha = \operatorname{atan2}[\cos\beta\sin\alpha, \cos\beta\cos\alpha] \quad (7.3.3) $$

$$ \beta = -\operatorname{asin}[-\sin\beta] \quad (7.3.4) $$

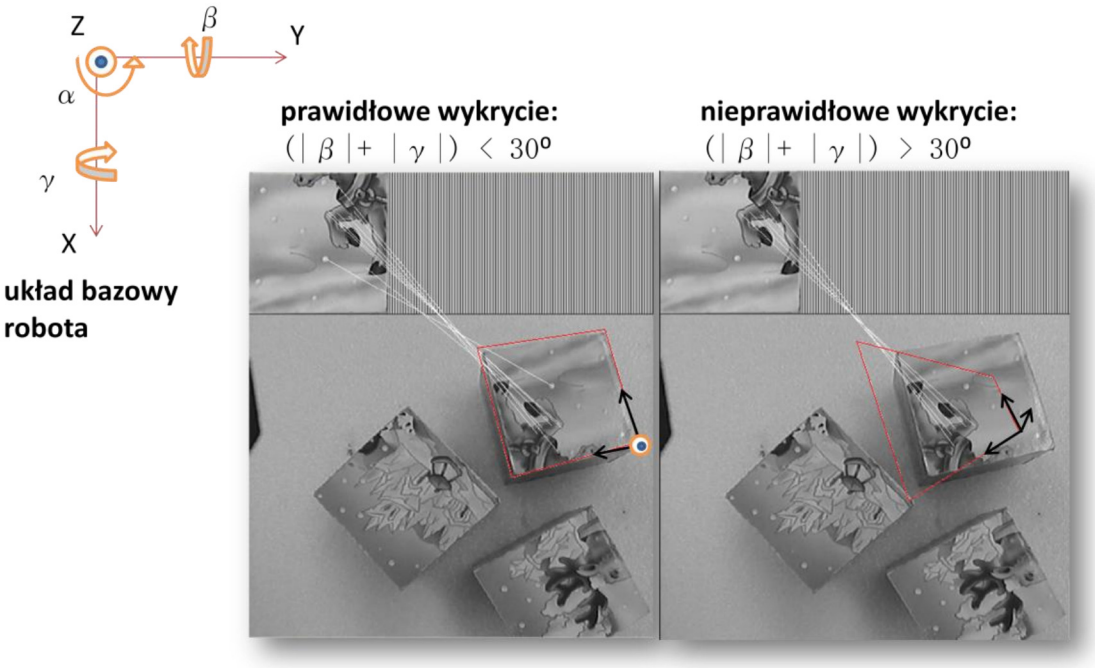
$$ \gamma = \operatorname{atan}\left[\frac{\sin\gamma\cos\beta}{\cos\gamma\cos\beta}\right] \quad (7.3.5) $$

Wzór na kąt α zwraca jego wartość z przedziału (-180° ÷ 180°). Na podstawie wartości kąta α określono wartość kąta roll chwytaka manipulatora, w celu chwycenia danego klocka. Występowanie w bezpośrednim sąsiedztwie innych klocków na danej warstwie struktury wiązało się z odpowiednim doborem kąta roll i kolejności chwytania poszczególnych klocków. Szczegóły tych operacji podane są w rozdziale 8.

Zależności określające kąty β i γ zostały uproszczone do postaci zwracających wartości tych kątów z zakresu (-90° ÷ 90°). Założono, że klocki powinny znajdować się na płaszczyźnie prostopadłej do osi z manipulatora (wartości kątów β i γ bliskie 0°). Manipulator zastosowany w projekcie posiada 5 stopni swobody, zatem chwycenie „przechylonej" sześciennej bryły w większości przypadków byłoby niemożliwe. Postanowiono zatem wykorzystać wartości kątów β i γ do sklasyfikowania wykrycia obiektu, czyli sprawdzenia czy metoda SURF prawidłowo „zaznaczyła" narożniki ścianki klocka.

7.3. Klasyfikacja wykrycia obiektu

Na rys. 7.3.1 przedstawione są 2 przykłady wykrycia obiektu. Zaznaczono układ współrzędnych bazowy robota oraz kąty obrotu wokół osi tego układu. Na zdjęciu, znajdującym się z lewej strony rys. 7.3.1, narożniki zostały wykryte prawidłowo, na drugim zaś pojawiły się pewne błędy.



Rys. 7.3.1 Klasyfikacja wykrycia obiektu

Nieprawidłowe wykrycie niesie ze sobą konsekwencje w postaci niewłaściwego obliczenia transformacji ${}^{B}\boldsymbol{T}_{\acute{S}}$. Kąty β i γ w takich przypadkach będą przybierały coraz to większe wartości, podczas gdy prawidłowe wykrycie zwróci wartości tych kątów bliskie 0. Postanowiono to wykorzystać do sklasyfikowania wykrycia obiektu, ustawiając pewien dopuszczalny próg, według poniższych zależności:

$(|\beta| + |\gamma|) < 30^0$ - prawidłowe wykrycie

$(|\beta| + |\gamma|) > 30^0$ - nieprawidłowe wykrycie

Taki sposób wykorzystania kątów β i γ nie wymagał dlatego znajomości ich wartości z przedziału ($-180^o \div 180^o$).

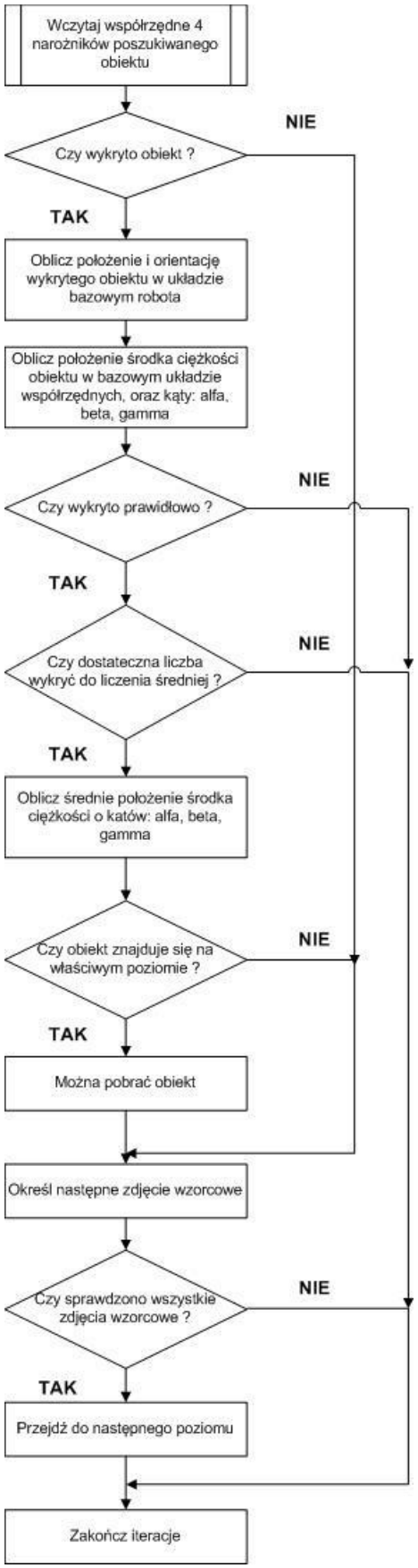
7.4. Organizacja funkcji *pokaz_wspolrzedne()*

Na rys. 7.4.1 znajduje się uproszczony schemat blokowy funkcji *pokaz_wspolrzedne()*, która jest wywoływana po funkcji wykryj_objekt(). Należy przypomnieć, że obie te funkcje działają cyklicznie, aż do momentu rozpoznania wszystkich obiektów na danej warstwie struktury. Zmianie w kolejnej iteracji podlegają wczytywane zdjęcia wzorcowe (jeżeli będą spełnione określone warunki), a właściwie cechy charakterystyczne tych zdjęć wyznaczone metodą SURF.

Na samym początku funkcji *pokaz_wspolrzedne()* wczytywane są współrzędne czterech narożników aktualnie analizowanej ścianki klocka. Następnie dokonywane jest sprawdzenie czy w strukturze *CvPoint dst_corners[4]*, przechowującej te współrzędne, znajdują się niezerowe dane, czyli czy wykryto obiekt. Jeżeli nie, oznacza to że aktualnie poszukiwanej ścianki nie ma na zdjęciu obserwowanej sceny i program przechodzi do miejsca w którym określane jest następne zdjęcie wzorcowe (zdjęcie kolejnej ścianki) i wykonywane są dalsze operacje. Przy określaniu kolejnego zdjęcia wzorcowego uwzględniane są już wykryte klocki, a zatem pomijane są zdjęcia wszystkich ścianek wykrytego już klocka.

Jeżeli wykryto obiekt, wówczas program oblicza transformację ${}^{B}\boldsymbol{T}_{O}$, wykrytego klocka, a następnie położenie środka ciężkości tego klocka w układzie bazowym robota oraz kąty α, β i γ opisane w podrozdziale 7.2.

Kolejnym krokiem jest sprawdzenie poprawności wykrycia obiektu na podstawie kątów β i γ tak jak to opisano w podrozdziale 7.3. Jeżeli metoda SURF zwróciła niepoprawne wyniki, wtedy program przechodzi na koniec funkcji pokaz_wspolrzedne() i ponownie wywoływana jest funkcja wykryj_objekt(). W przeciwnym przypadku program przechodzi dalej i zwiększany jest licznik poprawnych wykryć.



Rys. 7.4.1 Uproszczony schemat blokowy funkcji pokaz_wspolrzedne()

Następny sprawdzany warunek dotyczy liczby poprawnych wykryć. Jeżeli liczba ta nie równa się założonej, wtedy program przechodzi na koniec i ponownie uruchamiane jest wykrywanie tego samego zdjęcia wzorcowego i przechwytywane są kolejne dane do średniej. Proces ten będzie się powtarzał aż do uzyskania założonej liczby wyryć. Wtedy obliczona zostanie średnia współrzędnych środka ciężkości i kątów α, β i γ.

Na obrazie obserwowanej sceny może wystąpić taka sytuacja, w której całkowicie odsłonięta jest ścianka klocka znajdującego się na niższej warstwie struktury, dlatego po obliczeniu średniej sprawdzany jest warunek czy wykryty klocek znajduje się na właściwej warstwie. Jeżeli tak ustawiany jest znacznik w odpowiedniej tablicy, zezwalający na możliwość pobrania tego klocka. W przeciwnym przypadku program pomija tę operację i przechodzi do miejsca, w którym określane jest następne zdjęcie wzorcowe.

Ostatni warunek sprawdza czy na zdjęciu obserwowanej sceny prześledzono już wszystkie zdjęcia wzorcowe. Jeżeli tak program ustawia znacznik, który odpowiada za przejście do analizy kolejnego poziomu struktury. W przeciwnym przypadku ta operacja jest pomijana.

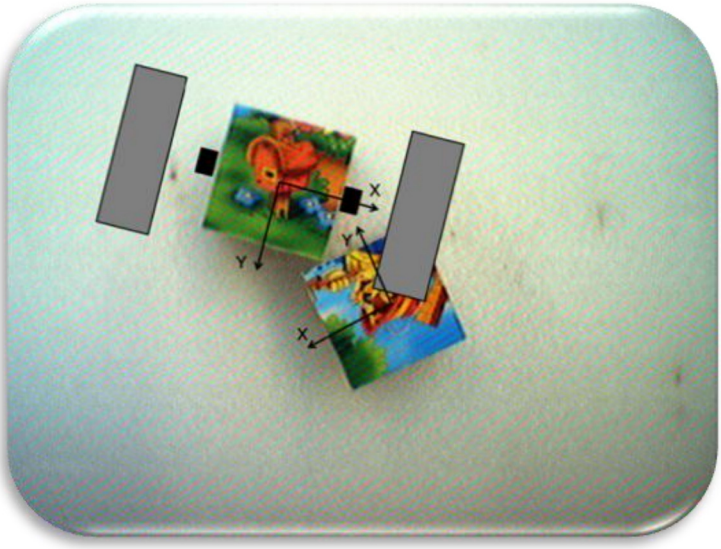
Po przeanalizowaniu danego poziomu struktury program przechodzi do funkcji dobierz_kolejnosc(), w celu wyznaczenia kolejności chwycenia poszczególnych klocków. W funkcji tej następuje także obliczenie kąta roll chwytaka, który definiuje chwycenie klocka z odpowiednią orientacją.

8. Określenie kolejności dekompozycji *(Daniel Gozdera)*

W celu przeprowadzenia odpowiedniej dekompozycji, takiej aby obiekty były pobierane z obszaru roboczego kolejno bez naruszenia pozostałej części struktury założono hierarchiczny model poziomów. Dekompozycja obiektów ze struktury jest przeprowadzana iteracyjnie, warstwami, kolejno od poziomu trzeciego do pierwszego. Funkcje odpowiedzialne za wykrywanie obrazu i lokalizacje obiektów „wykryj_obiekt()" i „pokaz_wspolrzedne()" jako wynik końcowy zwracają informacje na temat położenia środka ciężkości obiektu i jego orientacji w układzie bazowym. Kolejnym etapem jest na podstawie tych informacji odpowiedni dobór kolejności pobierania obiektów znajdującej się na jednym poziomie. Proces ten jest dość skomplikowany w przypadku gdy na jednym poziomie znajduje się więcej niż dwa obiekty. Ponadto, ważnym elementem jest też orientacja chwytaka z jaką zostanie pobrany klocek, gdyż nie zawsze jest ona dowolna. Ze względu na niewielką przestrzeń roboczą i znaczące wymiary obiektów w praktyce trudno jest ustawić na jednym poziomie cztery obiekty, w taki sposób aby możliwe było ich chwytanie. Z tego względu w dalszej części rozważań w tym rozdziale posłużono się przykładami w których w jednej warstwie znajdują się maksymalnie trzy obiekty.

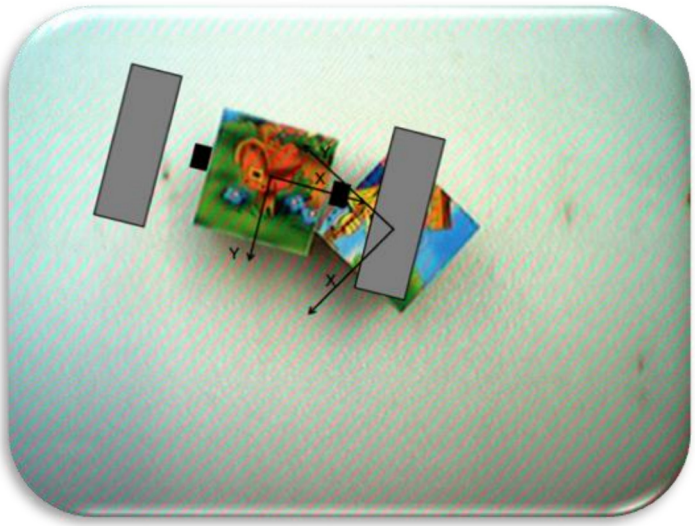
8.1. Analiza możliwości chwycenia obiektu

Chwytanie obiektów odbywa się przy użyciu chwytaka składającego się z dwóch szczęk umieszczonych symetrycznie względem osi Z chwytaka. Zdefiniowane obiekty sześcienne można chwycić przy użyciu wykorzystywanego pięcioosiowego manipulatora za jedna z dwóch par bocznych ścianek. Nie trudno więc stwierdzić iż możliwość chwycenia obiektu sprowadza się do tego czy w pobliżu ścianek które chcemy chwycić nie znajdują się inne obiekty. Aby móc analitycznie stwierdzić obecność innego obiektu w sąsiedztwie przez nas rozważanego, wykorzystano przekształcenia układów współrzędnych. Wykorzystano sprowadzenie bazowego układu współrzędnych do układu współrzędnych rozważanego przez nas obiektu. W ten sposób można łatwo określić odległość innych obiektów od ścianek analizowanego klocka sześciennego.



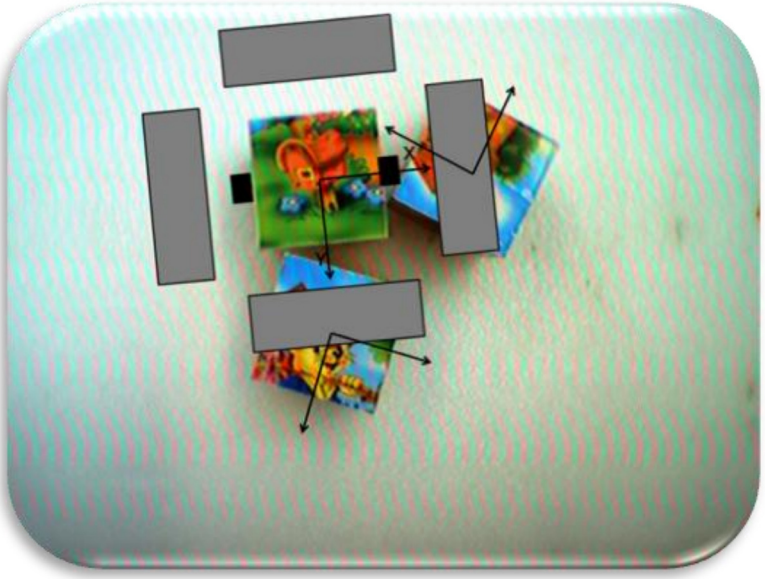
Rys. 8.1.1 Analiza możliwości chwycenia obiektu

Na rysunku 8.1.1 przedstawiono graficzną interpretację analizy chwycenia obiektu. W punktach środków ciężkości obiektów umieszczone są początki układów współrzędnych z nimi związane. Analiza dotyczy kocka znajdującego się w górnej części zdjęcia, nazywanego w dalszej części rozważań „obiektem nadrzędnym". Analizowana jest możliwość chwycenia obiektu w osi OX obiektu. Czarnymi prostokątami zaznaczono miejsca położenia chwytaka, natomiast szare prostokąty oznaczają tzw. „strefę zabronioną". Analiza chwycenia jest przeprowadzana dwukrotnie, raz dla osi OX i raz dla osi OY. Na przedstawionym rysunku środek ciężkości obiektu znajdującego się w dolnej części zdjęcia nazywanego „podrzędnym" znajduje się poza strefami zabronionymi, więc istnieje możliwość chwycenia obiektu.



Rys. 8.1.2 Możliwość chwycenia obiektu w osi OY

Na rysunku 8.1.2 przedstawiono analizę chwycenia obiektu, której wynik jest negatywny. Środek ciężkości obiektu podrzędnego znajduje się w strefie zabronionej, więc nie ma możliwości jego chwycenia w osi OX, natomiast istnieje możliwość chwycenia w osi OY.



Rys. 8.1.3 Brak możliwości chwycenia obiektu

Na rysunku 8.1.3 przedstawiono analizę chwycenia obiektu w sąsiedztwie którego znajdują się dwa inne obiekty. Widać wyraźnie iż nie ma możliwość chwycenia obiektu nadrzędnego w osi OX ani tez w osi OY. Wymiary strefy zabronionej zostały tak dobrane aby uwzględnić rozmiary obiektów, oraz dowolną orientację obiektów podrzędnych znajdujących się w sąsiedztwie obiektów nadrzędnych.

Strefy zabronione można opisać analitycznie wyrażeniem:

- W osi OX

$$\begin{cases} X \in (-40\,;\,-100) \cup (40\,;100) \\ Y \in (-40\,;\,40) \end{cases}$$

- W osi OY

$$\begin{cases} X \in (-40\,;\,40) \\ Y \in (-40\,;\,-100) \cup (40\,;100) \end{cases}$$

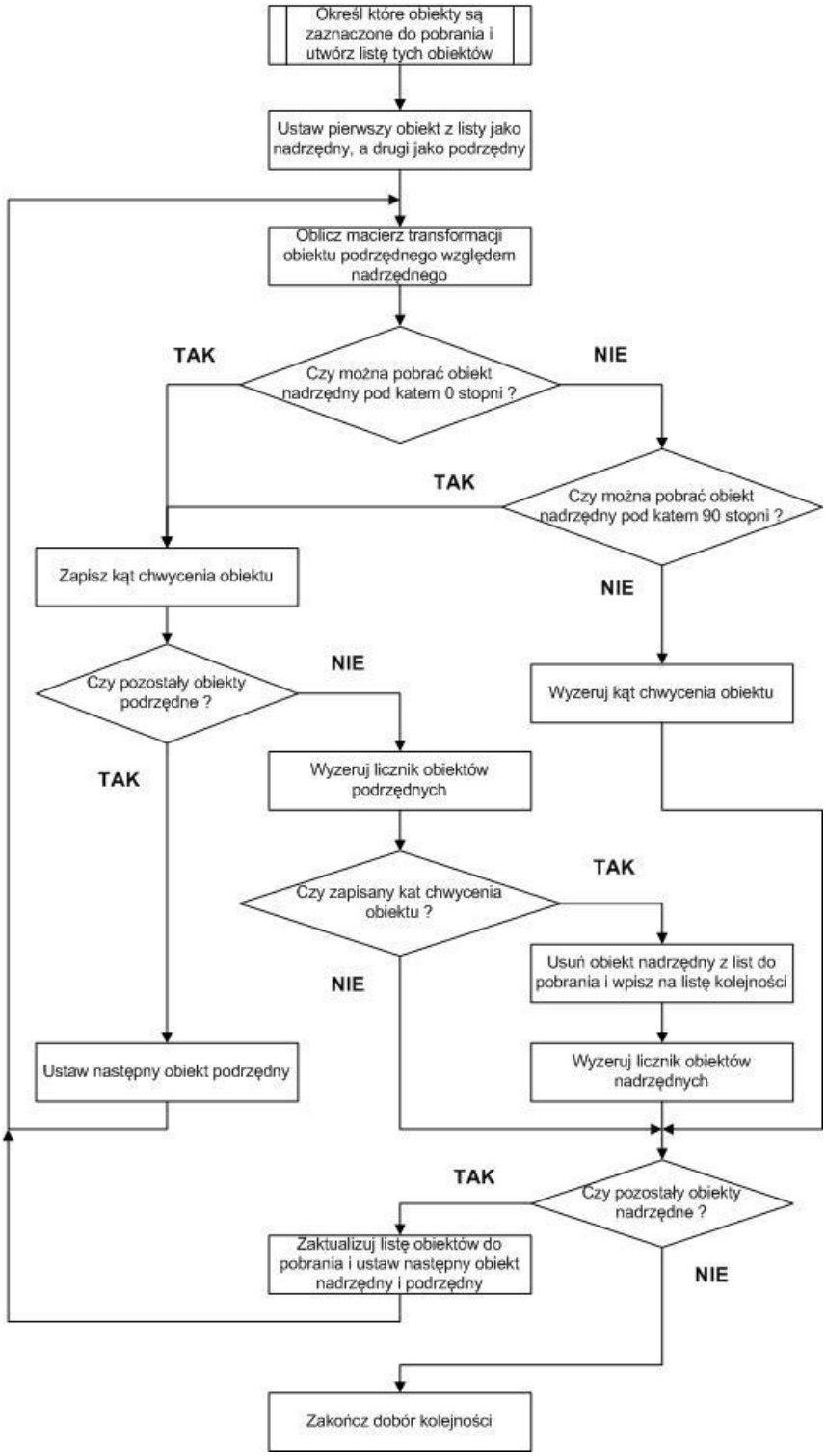
Jednostki zostały podane w milimetrach.

8.2. Algorytm doboru kolejności dekompozycji

Algorytm doboru kolejności jest dość skomplikowany ze względu na uwzględnienie w nim możliwości znalezienia się w sąsiedztwie obiektu nadrzędnego wielu obiektów podrzędnych. Został on przedstawiony w formie uproszczonej na rysunku (numer rysunku) w celu zobrazowania jego działania. Obiektem nadrzędnym jest nazywany obiekt którego możliwość chwycenia jest aktualnie analizowana, natomiast obiektami podrzędnymi te, które znajdują się aktualnie w jego sąsiedztwie. Na wstępie trzeba zaznaczyć iż funkcja doboru kolejności wymaga podania obiektów, które znajdują się na jednym, aktualnie analizowanym poziomie. Tworzona jest z nich lista obiektów które są przygotowane „do pobrania". Następnie pierwszy obiekt z tej listy jest ustawiany jako obiekt nadrzędny, a drugi jako obiekt podrzędny. Wyliczana jest macierz transformacji obiektu podrzędnego względem nadrzędnego.

W kolejnym kroku analizowana jest możliwość chwycenia obiektu nadrzędnego w osi OX przy użyciu stref niedozwolonych. Jeżeli jej wynik jest pozytywny, to obiekt zostaje uznany wstępnie jako możliwy do chwycenia i zapisana oś w której to chwycenie ma być zrealizowane. Obiekt ten zostanie oznaczony jako możliwy do chwycenia pod warunkiem że w dalszej części iteracji algorytmu (analizy możliwości chwycenia tego obiektu z innymi obiektami podrzędnymi) nie uzyskamy w tej osi wyniku negatywnego analizy. Jeśli w pierwszej części analizy w osi OX uzyskami wynik negatywny, to następuje analiza chwycenia w osi OY i to od jej wyniku decyduje czy obiekt zostanie oznaczony jako możliwy do chwycenia, i tak jak w poprzednim przypadku zostanie on w tym stanie do momentu zakończenia analiz z pozostałymi obiektami podrzędnymi jeśli nie uzyskamy w tej osi wyniku negatywnego.

Każdy obiekt nadrzędny jest poddawany analizie ze wszystkimi obiektami podrzędnymi, po czym wynik końcowy, czyli obecność oznaczenia możliwości pobrania decyduje o tym czy obiekt zostanie wyselekcjonowany.



Rys. 8.2.1 Algorytm doboru kolejności chwytania obiektów

Jeśli obiekt zostanie wyselekcjonowany to usuwany jest z listy „do pobrania” i trafia na listę „kolejności” wraz z informacja w której osi będzie następowało jego pobranie. W ten sposób jest wypełniana lista „kolejności”. Po sprawdzeniu danego obiektu nadrzędnego algorytm przechodzi do następnego, jako nadrzędny oznaczany jest drugi obiekt z listy,

pozostałe natomiast staja się podrzędnymi. Kolejno sprawdzane SA wszystkie możliwości chwycenia wszystkich obiektów z listy „do pobrania” po czym zostaje ona zaktualizowana. Proces trwa aż do momentu przeniesienia wszystkich obiektów z listy „do pobrania” do listy „kolejności”.

Cały proces doboru kolejności składa się z wielu iteracji, których liczba zależy od ilości analizowanych obiektów i sposobu ich wzajemnego ułożenia. Algorytm został zaimplementowany w języku C++ za przy użyciu pętli zagnieżdżonych i etykiet, a jego pełna analiza jest znacznie utrudniona.

9. Zaplanowanie i wykonanie ruchów manipulatora *(Paweł Golba)*

Po przeanalizowaniu danego poziomu struktury z klocków, czyli po rozpoznaniu obiektów, zlokalizowaniu i dobraniu odpowiedniej kolejności pobierania, należało pobrać klocki z danego poziomu.

W momencie zdekomponowania całej struktury, manipulator mógł przystąpić do ponownego złożenia struktury według zaplanowanego schematu. Ze względu na to, że wybrane klocki to trójwymiarowe puzzle, postanowiono że efektem końcowym będzie ułożenie przez manipulator jednego pełnego obrazu z klocków. Do ułożenia możliwe jest 6 obrazów. Przewidziano ułożenie wszystkich obrazów, pod warunkiem jednak, że klocki zostaną umieszczone w stosunku do kamery ściankami przedstawiającymi fragmenty tego samego obrazu.

Do tworzenia kodów źródłowych sterujących manipulatorem Scorbot-er 4u z poziomu języka C++ niezbędne są pliki konfiguracyjne oraz biblioteki udostępnione przez producenta. Szczegóły dotyczące konfiguracji układu oraz funkcji obsługujących zadawanie ruchów manipulatora znajdują się w opracowaniach [7, 8].

Pierwszy etap obsługi manipulatora polegał na nawiązaniu połączenia ze sterownikiem osi, a następnie dokonaniu bazowania poszczególnych osi robota. Należy zaznaczyć, iż proces ten był dość uciążliwy ze względu na długi czas jego trwania (ok. 4 min.) oraz na częste błędy przerywające aktualnie realizowane bazowanie. W przypadku wykonywania testów działania programu, kwestia bazowania pochłaniała więc znaczną część czasu.

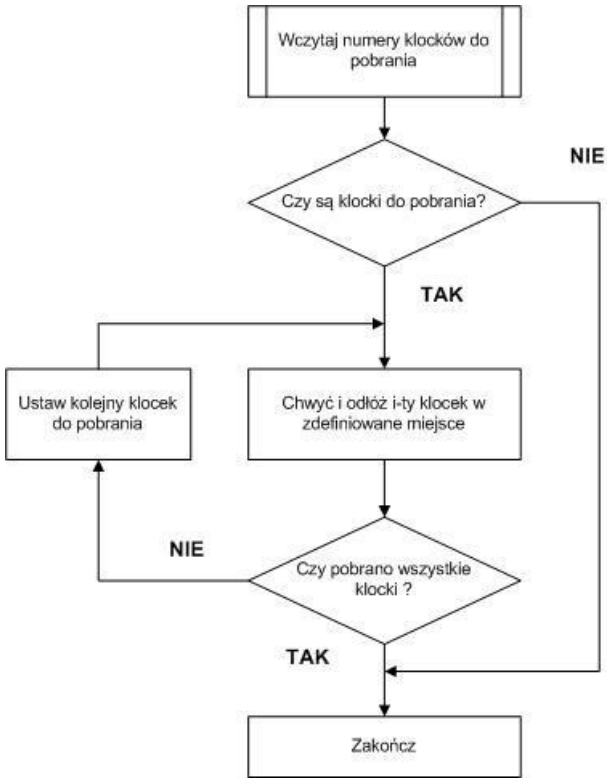
W projekcie postanowiono uprościć kwestię zadawania ruchów, tworząc specjalną funkcję, w której umieszczono szereg niezbędnych funkcji z biblioteki obsługującej sterowanie manipulatorem:

void ruch(long predkosc)

Jedynym parametrem podawanym przy wywoływaniu tej funkcji była prędkość określana w zakresie od 0 do 100. W linii poprzedzającej funkcję *ruch()* należało podać parametry określające położenie i orientację punktu roboczego chwytaka, czyli x, y, z, pitch i roll (w odniesieniu do układu bazowego robota).

9.1. Dekompozycja poziomu struktury

Funkcja pobierz() jest odpowiedzialna za realizację dekompozycji obiektów z danego poziomu struktury i jest wywoływana po funkcji dobierz_kolejnosc(). Schemat blokowy przedstawiający działanie funkcji do pobierania klocków znajduje się na rys 9.1.1.



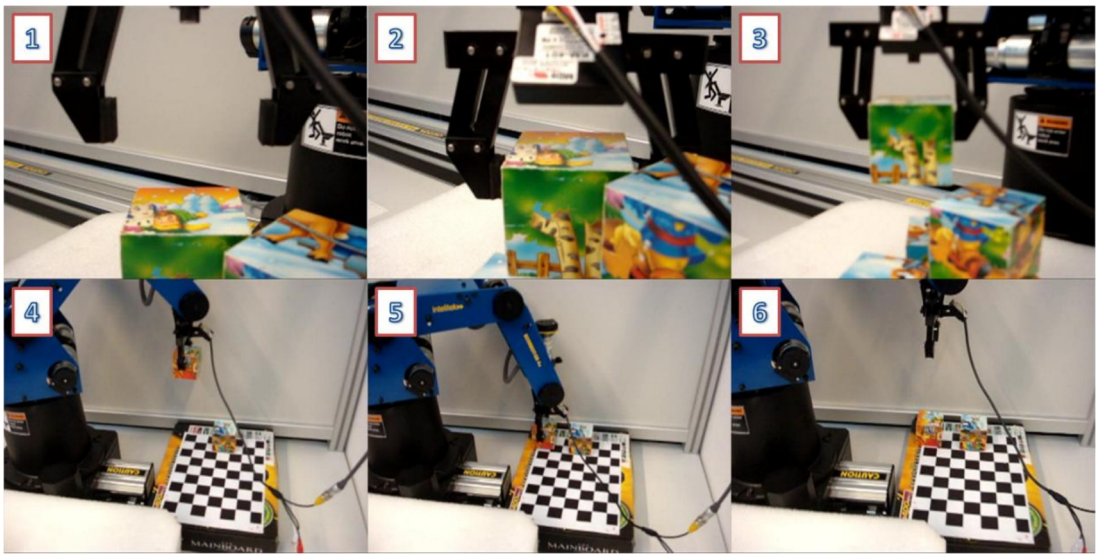
Rys. 9.1.1 Uproszczony schemat blokowy funkcji pobierz()

Jako dane wejściowe funkcja ta otrzymuje tablicę „kolejności” wraz z numerami klocków do pobrania. Następnie sprawdzane jest czy w tablicy tej znajdują się jakieś numery klocków. Jeżeli nie, funkcja kończy swoje działanie. Jeżeli tak, rozpoczynany jest proces pobierania pierwszego klocka z tablicy „kolejności”.

Szczęki chwytaka są otwarte. Ustawiane są parametry punktu roboczego chwytaka dla danego klocka i wykonywany jest ruch najpierw nad klocek z maksymalną prędkością (zdjęcie nr 1 z rys 9.1.2). Parametry x, y, z, pitch, roll dotyczące poszczególnych klocków przechowywane są w pamięci programu w odpowiednich strukturach danych. Kolejny ruchy to przemieszczenie pionowe w dół punktu roboczego chwytaka do poziomu środka ciężkości klocka, zamknięcie szczęk chwytaka i przemieszczenie klocka pionowo do góry (zdjęcia nr 2 i 3). Powinien być to punktu, od którego możliwe będzie dalsze wykonywanie ruchu poziomego w sposób bezkolizyjny z pozostałymi klockami na danej warstwie.

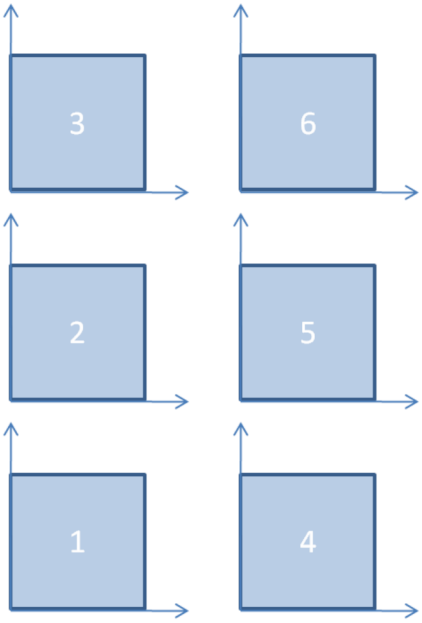
Kolejny punkt pośredni znajduje się nad pozycją do odłożenia (zdjęcie nr 4). Po jego osiągnięciu następuje powolne „postawienie” klocka, otwarcie szczęk chwytaka i odjechanie do góry (zdjęcia nr 5 i 6).

Zgodnie z algorytmem operacje te będą powtarzane, aż do momentu pobrania wszystkich klocków z danego poziomu. Poszczególne etapy pobierania i odkładania klocka przybliżają zdjęcia z rys. 9.1.2.



Rys. 9.1.2 Etapy pobierania i odkładania klocka podczas dekompozycji

Każdemu klockowi przypisano pozycję do odłożenia. Przy wyborze rozmieszczenia tych pozycji należało zachować odpowiednio duże odległości pomiędzy docelowymi obiektami, aby chwytak przy otwieraniu szczęk nie uderzał o sąsiednie klocki oraz jednocześnie aby pozycje te były dalej możliwe do osiągnięcia przez manipulator. Należy także zwrócić uwagę na bardzo ważną kwestię dotyczącą odkładania klocków z taką samą orientacją. Otóż klocki, mogą być chwytane zarówno w osi X jak i Y, zatem przy odkładaniu konieczne było uwzględnienie kąta, o jaki ewentualnie mógł być przekręcony chwytak podczas chwytania. Ilustracja 9.1.3 obrazuje poprawny szyk klocków po odłożeniu. Odłożenie z taką samą orientacją klocków ułatwiało dalsze operacje.



Rys. 9.1.3 Rozmieszczenie klocków po odłożeniu z uwzględnieniem ich orientacji

9.2. Ułożenie obrazu z klocków

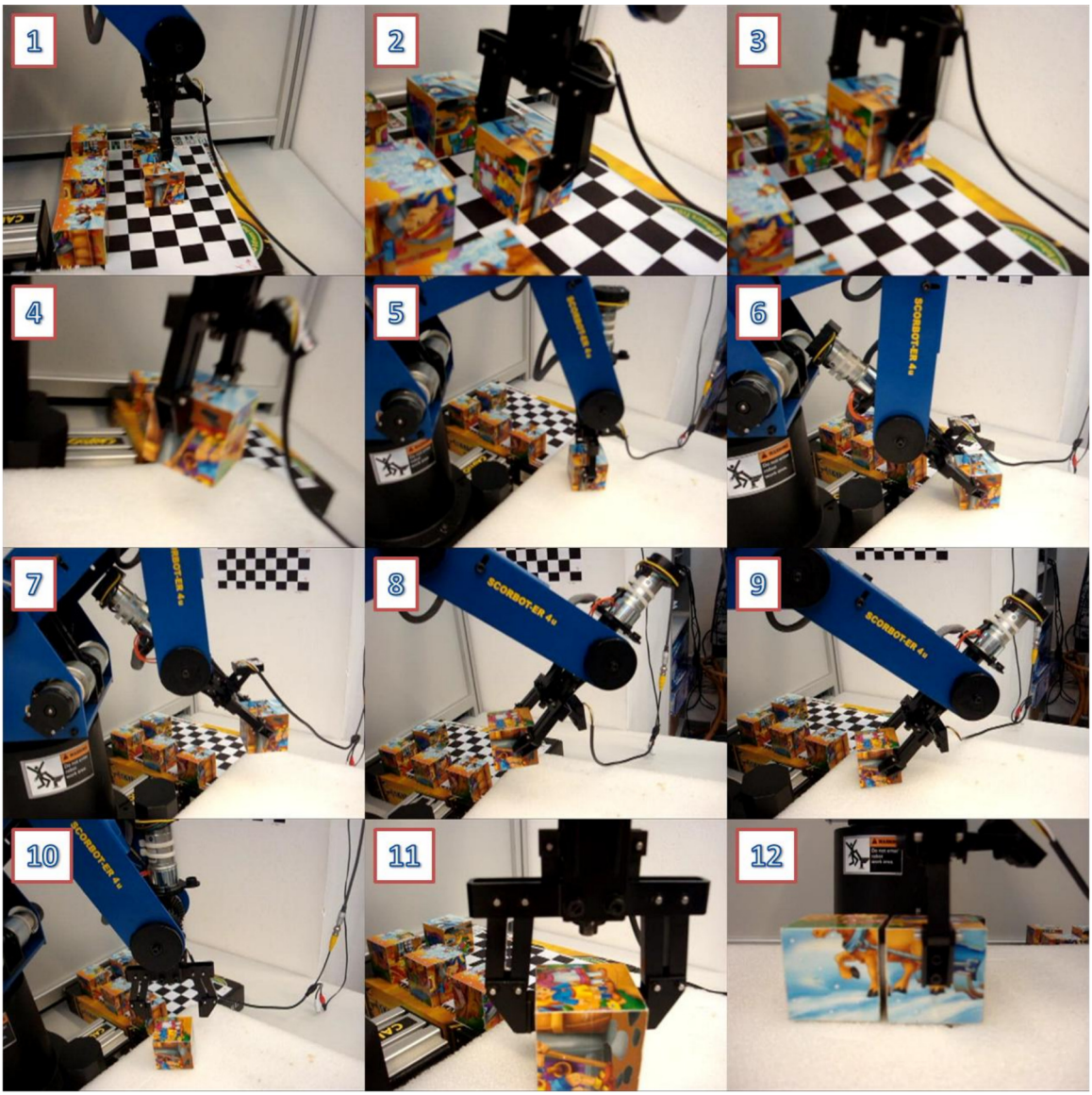
Ostatni etap projektu, polegał na zaprogramowaniu manipulatora tak, aby przeniósł i ustawił 6 klocków w zaplanowany sposób. Klocki, po ustawieniu, powinny znaleźć się możliwie jak najbliżej siebie, tak żeby utworzony przez nie obraz był jak najbardziej czytelny. Pojawił się zatem problem realizacji tego zadania za pomocą dostępnego chwytaka. Dosuwanie klocków do siebie, wiązałoby się z koniecznością zastosowania gładkiej i śliskiej podstawy. Nie byłoby także pewności czy klocek po dosunięciu znajdzie się w docelowym miejscu z odpowiednią dokładnością.

Postanowiono zatem ułożyć klocki tak, żeby oczekiwany obraz powstał w płaszczyźnie pionowej, a klocki go tworzące były ustawione w dwóch rzędach po 3 klocki – jeden rząd na drugim, tak jak to pokazano na rys. 9.2.1. Taki sposób ułożenia rozwiązywał problem dosuwania. Wystarczyło jedynie odpowiednio obrócić każdy klocek.

Funkcja, która realizuje ostatni etap projektu została nazwana układaj(). Jej algorytm jest bardzo zbliżony do algorytmu funkcji pobierz(). Także działa w pętli aż do momentu przeniesienia wszystkich klocków. W funkcji układaj() na samym początku wczytywana jest kolejność pobierania klocków, z tym że jest ona stała i wynika z konieczności ustawienia najpierw klocków z dolnego rzędu. Najwygodniejsza konfiguracja to 4, 5, 6, 1, 2, 3 (patrz rys. 9.2.1).

Rys. 9.2.1 Widok ułożonego obrazu z klocków wraz z oznaczeniem numeracji klocków

Następnie program przechodzi do wywołania sekwencji ruchów manipulatora. Zabierany jest najpierw klocek nr 4, potem 5 itd. Podobnie jak w przypadku dekompozycji klocków z poszczególnych warstw struktury, zdefiniowano odpowiednie pozycje do odłożenia dla każdego klocka. Należy zwrócić uwagę na to, że pozycje te zostały tak przyporządkowane żeby obok siebie znalazły się właściwe klocki, a w efekcie powstał pożądany obraz końcowy. Po chwyceniu danego klocka, jest on odkładany najpierw na pozycję pośrednią (zdjęcia nr 1, 2, 3, 4, 5 z rys 9.2.2), następnie obracany o 90^{o} względem poziomej osi, tak żeby ścianka z obrazkiem do układanki znalazła się w płaszczyźnie pionowej (zdjęcia nr 6, 7, 8, 9). Dopiero wtedy klocek odkładany jest w zdefiniowane miejsce (zdjęcia nr 10, 11, 12).



Rys. 9.2.2 Etapy pobierania, obracania i odkładania klocka podczas układania

W analogiczny sposób przebiegają manipulacje z pozostałymi klockami. Po odłożeniu klocka numer 3, funkcja układaj() kończy swoje działanie, układanka powinna utworzyć obraz tak jak np. na rys 9.2.1.

10. Podsumowanie i wnioski *(Daniel Gozdera)*

Wykonanie pracy wymagało poszerzenia wiedzy z zakresu programowania, przetwarzania obrazu oraz robotyki. Wiedza ta pozwoliła na całkowite osiągnięcie celów pracy sformułowanych w pierwszym rozdziale.

Realizując cele pracy zetknięto się z wieloma problemami natury technicznej. Zainstalowano kamerę bezprzewodową na ramieniu manipulatora w sposób który umożliwia obserwację całego obszaru roboczego. Kamera bezprzewodowa charakteryzująca się niewielką rozdzielczością obrazu , dużymi zakłóceniami oraz trudnością uzyskania ostrego obrazu na całej powierzchni ekranu mimo prawidłowych wyników kalibracji została zamieniona na kamerę przewodową. Dzięki temu zwiększono skuteczność rozpoznawania obiektów , oraz dokładność ich lokalizacji.

Zrealizowano koordynacje układów współrzędnych kamery i bazowego manipulatora, oraz zautomatyzowano ten proces za poprzez odpowiednią implementację w algorytmie programu. Uzyskano także prawidłowe wyniki rozpoznawania obiektów, oraz ich lokalizacji z dokładnością pozwalającą na swobodne manipulacje nimi. Opracowano algorytm pracy stanowiska i zaimplementowano go w kodzie programu.

Ze względu na małą wydajność pierwotnie napisanego programu został on przebudowany. Po uwzględnieniu pewnych uproszczeń algebraicznych napisano własną klasę języka C++ umożliwiającą szybkie obliczenia macierzowe oraz eliminującą konieczność posługiwania się wskaźnikami do zmiennych, których używanie było konieczne w funkcjach biblioteki OpenCV. Poprawiono również algorytm rozpoznawania obrazu optymalizując go pod kątem szybkości jego działania i dokładności.

Podczas optymalizowania procesu rozpoznawania obrazu zetknięto się z wieloma czynnikami, które mają negatywny wpływ na jego przebieg. Są nimi miedzy innymi: zmienne w czasie natężenie oświetlenia oraz jego źródło, różne odległości kamery od rozpoznawanych obiektów, oraz ilość zdefiniowanych szczegółów obrazu wzorcowego.

Dodatkowymi, pozytywnymi rezultatami z pracy jest zdobycie wiedzy i doświadczenia na temat projektowania i realizacji algorytmów w języku wysokiego poziomu, oraz umiejętność sterowania i programowania ruchów manipulatora z poziomu języka C++. Dodatkowo zapoznano się z funkcjami biblioteki OpenCV służącej do przetwarzania obrazu oraz wykorzystano jej możliwości w praktyce.

Osiągnięte cele pracy pozwalają wywnioskować, iż funkcjonowanie tak zrealizowanego stanowiska jest prawidłowe jedynie przy określonych założeniach, takich jak: standaryzacja rozpoznawanych obiektów, wcześniejsze zdefiniowanie ich wyglądu i rozmiarów, oraz niewielka zmienność warunków oświetleniowych stanowiska. Istotnym założeniem było także ograniczenie możliwych do rozpoznania orientacji obiektów.

Idąc w kierunku ewentualnego rozwoju pracy stanowiska oraz możliwości wykorzystania zaprojektowanego algorytmu działania w przemysłowych aplikacjach , tj. paletyzacja czy obróbka i montaż należałoby poczynić pewne udoskonalenia. Do najważniejszy z nich można by zaliczyć umiejętność automatycznej korekcji zmiennych warunków oświetleniowych, umiejętność rozpoznawania i pomiaru rozmiaru obiektów widzianych w kamerze, a także rozpoznawanie obiektów o dowolnej orientacji przy pomocy manipulatora o większej liczbie stopni swobody. W przyszłości także można by podjąć próbę wyeliminowania konieczności wcześniejszego uczenia wyglądu obiektów, tak aby program był wstanie sam określić na podstawie obrazu z kamery jakiego obiektu aktualnie szuka.

Głównymi wadami uruchomionego stanowiska są: manipulator o zaledwie pięciu stopniach swobody, wymagający bazowania przy każdym uruchomieniu i posiadający mała dokładność pozycjonowania, oraz niewielką dynamikę ruchów, kamera o małej rozdzielczości oraz niewielki obszar roboczy manipulatora.

Prezentacja działania stanowiska została utrwalona na filmie demonstracyjnym, który wraz z kodem źródłowym programu został umieszczony na dołączonej płycie CD.

Bibliografia

[1] http://www.vision.caltech.edu/bouguetj/calib_doc/ , ostatni dostęp 13.12.2010

[2] http://www.ee.pw.edu.pl/~czajewsw/studenckie/inne/opencv_doc/cv_8h.html, ostatni dostęp 13.12.2010

[3] http://www.opencv.jp/opencv-1.1.0/document/opencvref_cv.htm, ostatni dostęp 13.12.2010

[4] Gary Bradski i Adrian Kaehler, O'Reilly "Learning OpenCV", 2008

[5] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool, "SURF: Speeded Up Robust Features, Computer Vision and Image Understanding" (CVIU), Vol. 110, No. 3, pp. 346--359, 2008

[6] Lowe, David G. "Object recognition from local scale-invariant features", 7th IEEE International Conference on Computer Vision, vol. 2, pp. 1150-1157, 1999

[7] http://www.ee.pw.edu.pl/~czajewsw/studenckie/indywidualne/Scorboter.pdf, ostatni dostęp 13.12.2010

[8] http://kurser.iha.dk/eit/i4prj4/USBC-documentation.pdf, ostatni dostęp 13.12.2010