

POLITECHNIKA WROCŁAWSKA  
WYDZIAŁ ELEKTRONIKI

---

KIERUNEK:                   Informatyka (INF)  
SPECJALNOŚĆ:               Inżynieria internetowa (INT)

**PRACA DYPLOMOWA  
MAGISTERSKA**

Rozproszona detekcja głębi obrazu  
na podstawie stereowizji

Distributed depth detection  
based on stereo vision

AUTOR:  
inż. Adrian Frydmański

PROWADZĄCY PRACĘ:  
dr hab. inż. Henryk Maciejewski

OCENA PRACY:



*Oldze*



# Spis treści

<b>Wstęp</b>	<b>3</b>
<b>1 Opis teoretyczny</b>	<b>5</b>
1.1 Stereowizja . . . . .	5
1.1.1 Model kamery perspektywicznej . . . . .	5
1.1.2 Odwzorowanie w układzie kanonicznym i niekanonicznym . . . . .	6
1.1.3 Problem odpowiedniości i dysparycja . . . . .	6
1.1.4 Obliczanie współrzędnych . . . . .	8
1.1.5 Korekcja zniekształceń . . . . .	9
1.1.6 Kalibracja . . . . .	10
1.1.7 Dopasowanie . . . . .	12
1.1.7.1 Cechy obrazu do dopasowywania . . . . .	13
1.1.7.2 Problemy przy dopasowywaniu i założenia upraszczające .	14
1.1.8 Rekonstrukcja mapy głębi . . . . .	14
1.2 OpenCV . . . . .	15
1.2.1 Algorytm dopasowywania bloków w OpenCV . . . . .	16
1.2.2 Filtrowanie WLS . . . . .	16
1.3 Zrównoleglenie obliczeń przy pomocy wielowątkowości . . . . .	16
<b>2 Implementacja</b>	<b>19</b>
2.1 Kalibracja układu pomiarowego . . . . .	20
2.1.1 Kolekcja danych do kalibracji . . . . .	20
2.1.2 Kalibracja pojedynczej kamery . . . . .	20
2.1.3 Kalibracja układu kamer . . . . .	20
2.2 Generowanie mapy głębi . . . . .	22
2.2.1 Tryb jednowątkowy . . . . .	23
2.2.2 Tryb wielowątkowy . . . . .	23
2.2.3 Synchronizacja i komunikacja między wątkami . . . . .	24
2.2.4 Napotkane problemy . . . . .	25
2.3 Pomiar czasu . . . . .	28

<b>3 Eksperyment</b>	<b>29</b>
3.1 Plan eksperymentu . . . . .	29
3.2 Przebieg i wyniki eksperymentu . . . . .	29
3.3 Analiza wyników . . . . .	33
<b>4 Podsumowanie i wnioski</b>	<b>35</b>
<b>Bibliografia</b>	<b>37</b>
<b>Spis rysunków</b>	<b>39</b>
<b>Spis tabel</b>	<b>41</b>

# Wstęp

Stereowizja jest techniką obrazową umożliwiającą rekonstrukcję scen trójwymiarowych. Głębię uzyskuje się poprzez porównanie obrazu z dwóch kamer. Operacja ta jest zadaniem skomplikowanym obliczeniowo, gdyż wymaga znalezienia odpowiadających sobie punktów na obu obrazach. Dodanie do płaskiego obrazu gębi ma zastosowanie między innymi w robotyce (wykrywanie przeszkód), detekcji obiektów w przestrzeni, mapowaniu terenu i tworzeniu trójwymiarowego modelu rzeczywistego obiektu. Praca ta dotyczy możliwości przyspieszenia tychże obliczeń poprzez zrównoleglenie ich.

Celem jest stworzenie systemu pozwalającego na generowanie mapy gębi na podstawie obrazów stereowizyjnych, a następnie zbadanie, jaki wpływ na szybkość działania ma rozproszenie obliczeń. System ten będzie bazował na bibliotece OpenCV (odczyt danych z kamer i tworzenie mapy gębi) i działał w środowisku Linux.

Praca polega na wykonaniu następujących zadań:

- stworzenie programu do kalibracji stereowizyjnego układu kamer,
- stworzenie programu wykrywającego gębię w parze obrazów stereowizyjnych,
- zrównoleglenie działania programu,
- analiza rezultatów przed i po zrównolegleniu obliczeń.

Dokument obejmuje wstęp teoretyczny opisujący stereowizję, użytą bibliotekę graficzną i wykorzystywane algorytmy. Następnie opisana jest implementacja. W końcowej części przedstawione są osiągnięte wyniki badań, ich analiza i wnioski.



# Rozdział 1

## Opis teoretyczny

### 1.1 Stereowizja

Stereowizja umożliwia rekonstrukcję sceny na podstawie obrazów pozyskanych z układu co najmniej dwóch sensorów optyczno-elektronicznych. Porównując obrazy sceny wykonane z dwóch punktów widokowych, można wyodrębnić informację o głębi, badając względne pozycje obiektów w obu obrazach.

W tradycyjnej stereowizji dwie kamery, przesunięte poziomo względem siebie, są używane do uzyskania dwóch różnych widoków na scenie, w sposób podobny do ludzkiego widzenia obuocznego. Porównując te dwa obrazy, względną informację o głębokości można uzyskać w postaci mapy dysparcji, która koduje różnicę w poziomych współrzędnych odpowiednich punktów obrazu. Wartości w tej mapie różnic są odwrotnie proporcjonalne do głębokości sceny w danym miejscu.

Aby człowiek mógł porównać te dwa obrazy, muszą one zostać nałożone na urządzenie stereoskopowe, przy czym obraz z prawej kamery jest pokazywany prawemu, a z lewej lewemu oku obserwatora.

W komputerowym systemie wizyjnym wymagane jest kilka etapów wstępniego przetwarzania.

1. Obraz musi być najpierw niezniekształcony, tak aby usunąć zniekształcenia wynikające z układu optycznego kamery i zniekształcenia spowodowane nierównym ustawieniem urządzeń w układzie. Zapewnia to, że obserwowany obraz jest zgodny z projekcją idealnej kamery otworkowej w układzie kanonicznym.
2. Para obrazów musi zostać porównana na wspólnej płaszczyźnie, co określa się mianem rektyfikacji.
3. Miara, która porównuje dwa obrazy jest minimalizowana. Tworzona jest mapa dysparcji, czyli różnic pomiędzy pozycją poszczególnych pikseli obrazu.
4. Opcjonalnie, otrzymana mapa różnic jest wyświetlana w chmurze punktów 3D. Wykorzystując parametry projekcyjne kamer, chmurę punktów można obliczyć tak, aby zapewniała pomiary w znanej skali.

#### 1.1.1 Model kamery perspektywicznej

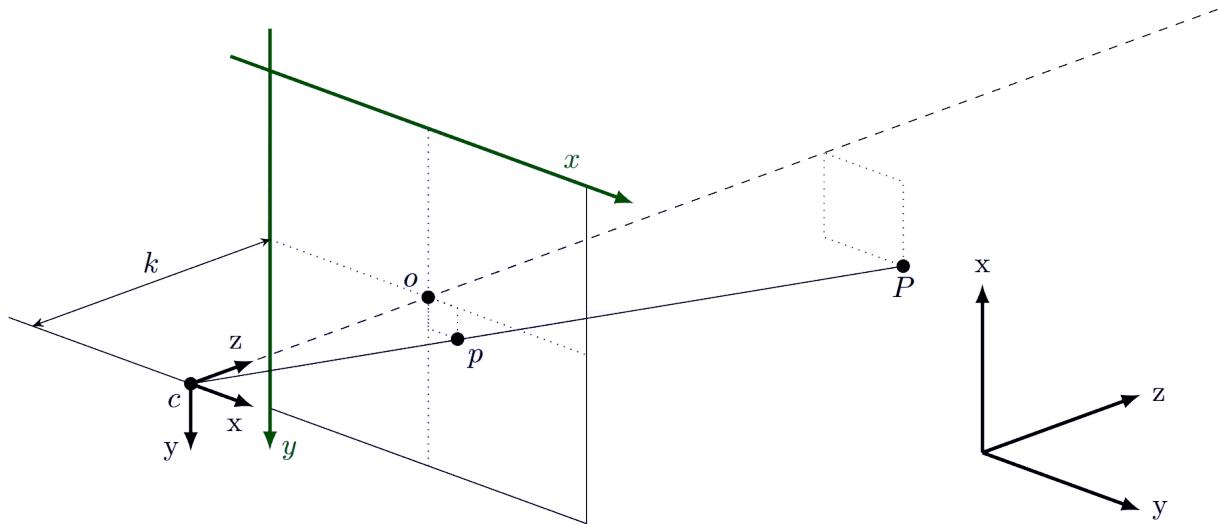
Mając dany punkt  $P = (X, Y, Z)$  w układzie współrzędnych kamery, współrzędne jego obrazu  $p = (x, y)$  w detektorze kamery można wywieść z podobieństwa trójkątów:

$$x = k \frac{X}{Z}$$

$$y = k \frac{Y}{Z}$$

$$z = k$$

Głębina wpływa na współrzędne  $x, y$  – widzenie stereoskopowe wystarczy do widzenia głębi i nie ma potrzeby interpretacji obrazu.



Rysunek 1.1 Model kamery perspektywicznej [10]

### 1.1.2 Odwzorowanie w układzie kanonicznym i niekanonicznym

Obrazy punktów  $P_1$  i  $P_3$  są rzutowane w jednym punkcie w prawym obrazie. Aby uzyskać wzajemnie jednoznaczne odwzorowanie współrzędnych przestrzeni  $(X, Y, Z)$  konieczna jest informacja o rzucie punktów przestrzeni na więcej niż jeden obraz. [3]

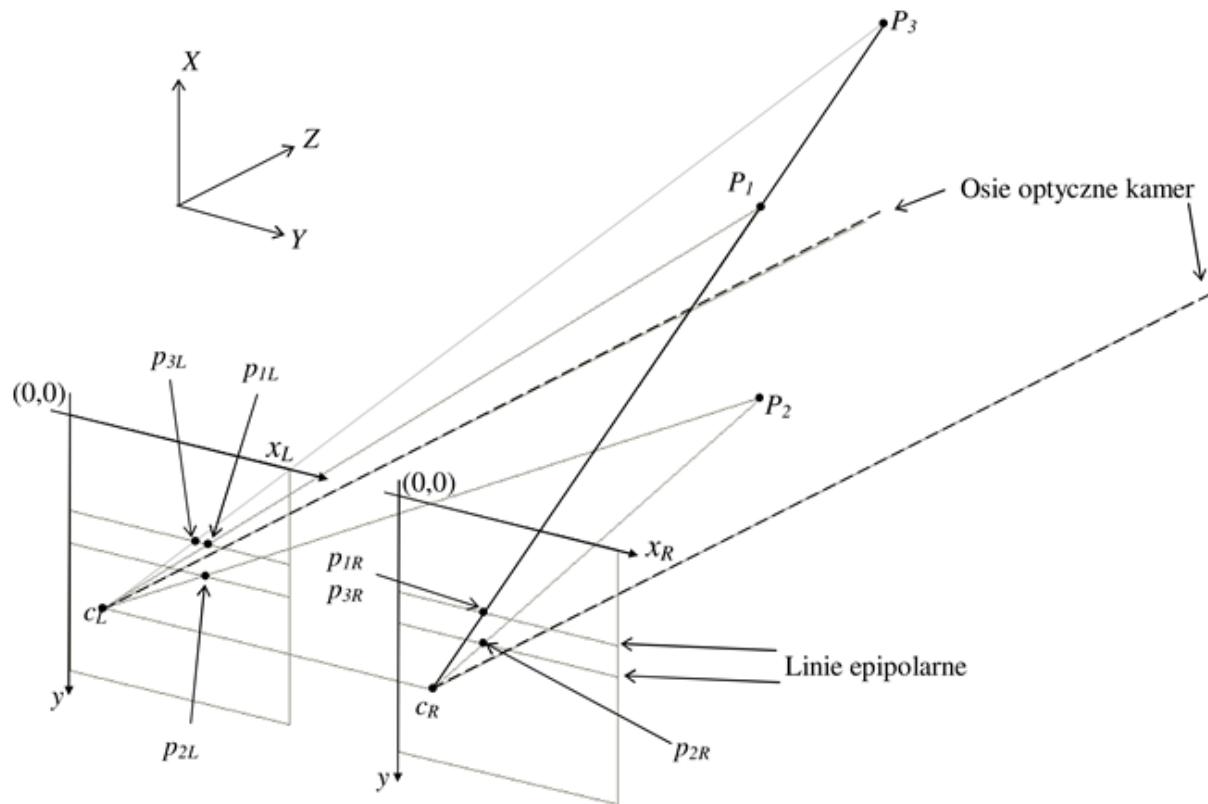
Linią epipolarną dla punktu  $p_R$  ( $p_L$ ) z obrazu prawej (lewej) kamery jest linia będąca obrazem w kamerze lewej (prawej) prostej wychodzącej z ogniska  $c_R$  ( $c_L$ ) prawej (lewej) kamery i przechodzącej przez  $p_R$  ( $p_L$ ). Zgodnie z tą definicją punktowi  $p_{1R}$  z prawego obrazu można przyporządkować linię epipolarną z lewego obrazu, na której leżą punkty będące obrazami punktów z przestrzeni leżących na prostej, która przecina ognisko  $c_R$  i punkt  $p_{1R}$ . Dla kanonicznego układu kamer liniami epipolarnymi są proste równoległe do wierszy obrazu. [11]

Dla niekanonicznych układów kamer wszystkie linie epipolarne nie są równoległe względem siebie i przecinają się w punkcie epipolarnym. [11]

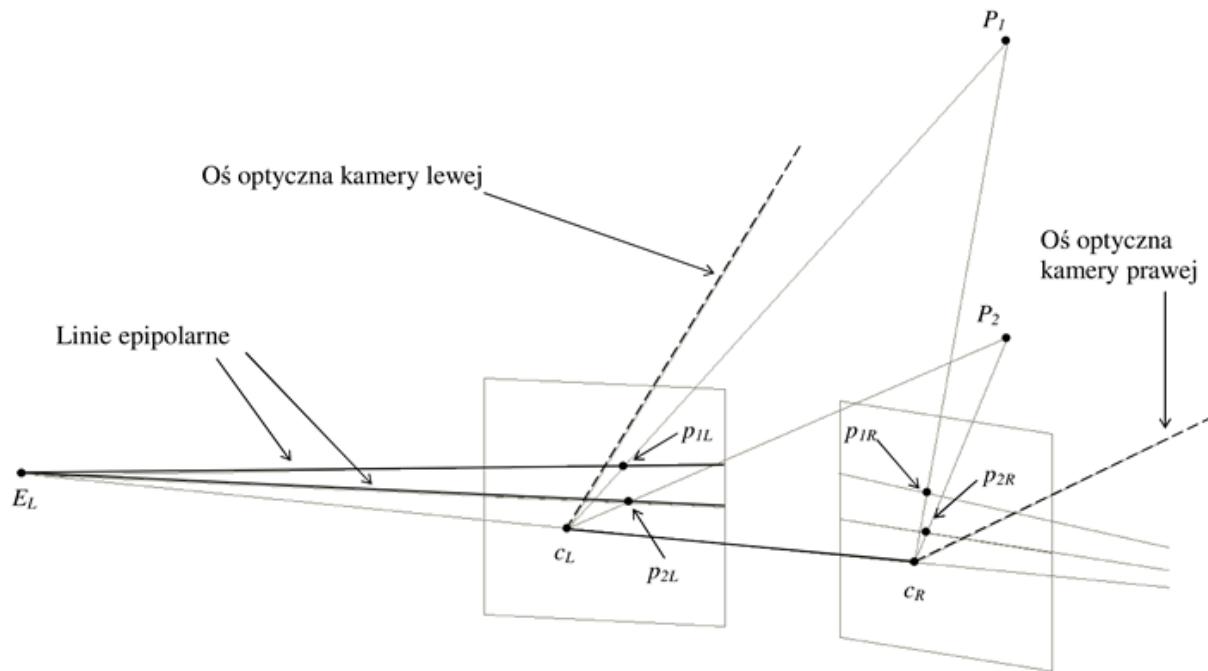
### 1.1.3 Problem odpowiedniości i dysparycja

Główным zadaniem podczas rekonstrukcji sceny trójwymiarowej z obrazów stereowizyjnych jest rozwiązywanie tzw. problemu odpowiedniości (ang. *correspondence problem*), polegającego na znalezieniu współrzędnych punktów  $p_L, p_R$ , czyli rzutów danego punktu w przestrzeni  $P(X, Y, Z)$  w obrazach lewej i prawej kamery. [11]

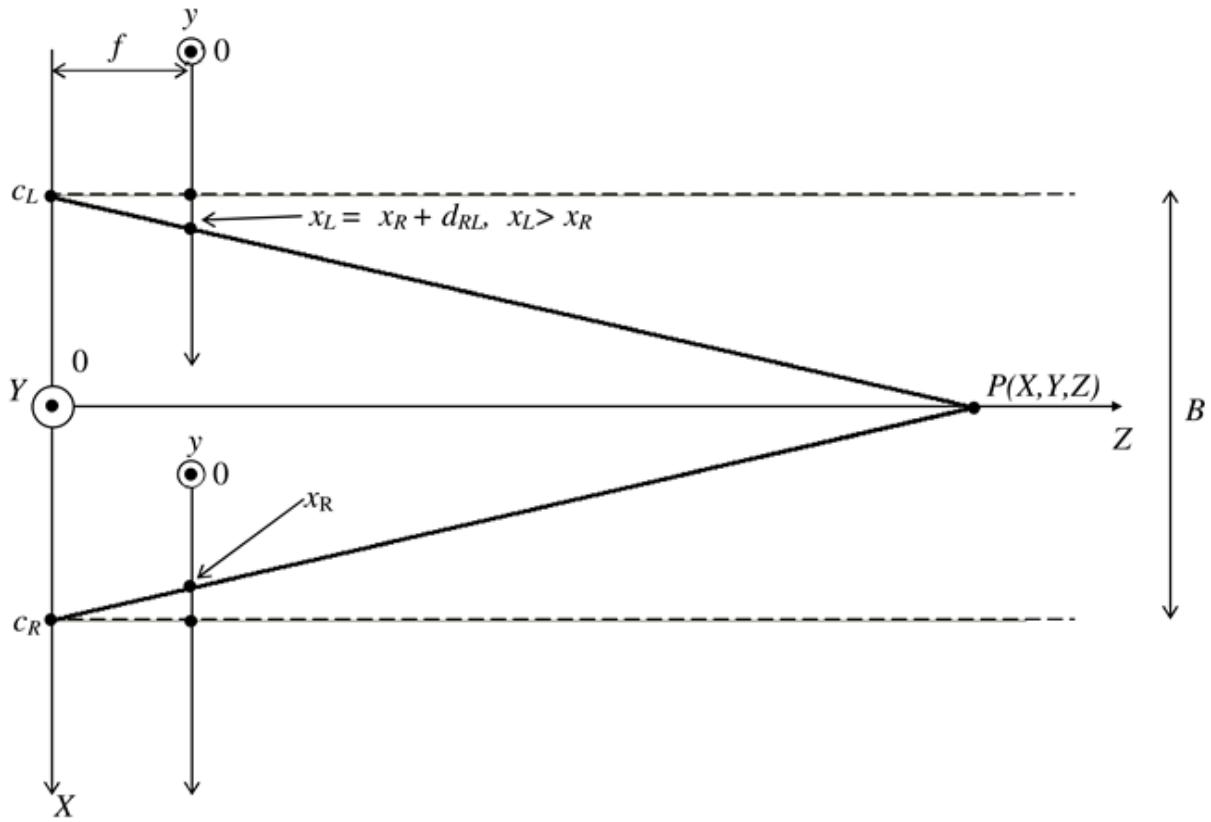
W kanonicznym układzie kamer problem ten upraszcza się, bo współrzędne  $y$  obrazów punktu w przestrzeni są identyczne ( $y_L = y_R$ ). Z tego faktu wynika, że poszukiwanie punktu  $p_L$  odpowiadającego  $p_R$  może być ograniczone do linii przebiegającej wzdłuż pojedynczego wiersza o znanej współrzędnej. Rozwiązyując zadanie odpowiedniości wyznacza



Rysunek 1.2 Odwzorowanie geometryczne punktów przestrzeni trójwymiarowej ( $X, Y, Z$ ) na płaszczyzny przetworników obrazowych kamer usytuowanych w tzw. układzie kanonicznym (osie kamer równoległe, współrzędne  $Z$  ognisk kamer identyczne) [10]



Rysunek 1.3 Odwzorowanie geometryczne punktów przestrzeni trójwymiarowej ( $X, Y, Z$ ) na płaszczyzny przetworników obrazowych kamer usytuowanych w tzw. układzie niekanonicznym (osie kamer nierównoległe, współrzędne  $Z$  ognisk kamer różne) [10]



Rysunek 1.4 Odpowiedniość i dysparcja [10]

się tzw. obraz dysparcji, czyli różnicy współrzędnych  $x$  wynikających z wzajemnego przesunięcia obrazów każdego z punktów przestrzeni  $(X, Y, Z)$  w obu kamerach. Obliczając dysparcję przyjmuje się jeden z dwóch obrazów jako obraz odniesienia (na rysunku obraz prawy). Jeśli punktowi  $p_R$  z obrazu odniesienia mającemu współrzędne  $(x_R, y_R)$  odpowiada punkt z drugiego obrazu,  $p_L$ , mający współrzędne  $(x_L, y_L)$  (gdzie  $y_R = y_L$ , co wynika z wcześniejszych założeń), to przesunięcie (dysparcja) między obrazem punktu z prawej i lewej kamery jest równa

$$d_{RL} = x_L - x_R.$$

#### 1.1.4 Obliczanie współrzędnych

W każdym stereowizyjnym układzie kamer dysparcja  $d_{RL}$  jest zawsze nieujemna,  $x_L \geq x_R$ . Dla każdego punktu przestrzeni znajdującego się w nieskończonej odległości od kamer  $x_L = x_R$ . Zachodzi tu zależność między parą punktów na obrazach ( $p_R, p_L$ ) i punktem w przestrzeni  $P(X, Y, Z)$ ). Współrzędne punktu  $P(X, Y, Z)$ , który jest „obserwowany” w obrazach każdej z kamer w układzie kanonicznym wylicza się używając następujących wzorów:

$$\begin{aligned} d_{RL} &= x_L - x_R, \\ X &= \frac{B \cdot (x_R + x_L - 2 \cdot x_0)}{2 \cdot d_{RL}}, \\ Y &= \frac{B \cdot y}{d_{RL}}, \\ Z &= \frac{B \cdot f_p}{d_{RL}}, \end{aligned}$$

gdzie:

$B$  – odległość między osiami optycznymi kamer, tzw. baza układu kamer,

$f_p$  – ogniskowa kamer podana w liczbie pikseli,

$x_0$  – współrzędna środka obrazu, przez którą przechodzi oś kamery. [11]

Współrzędne  $X, Y, Z$  odnoszą się do układu współrzędnych o początku znajdującym się w połowie odcinka, który łączy ogniska kamer. Na podstawie wzoru na  $Z$  można zauważyć, że poznanie dysparcji jest decydującym elementem przy rekonstrukcji sceny trójwymiarowej. Kiedy znane jest wzajemne przesunięcie obrazów punktu w każdej z kamer, ogniskowa i baza układu stereowizyjnego możliwe jest wyznaczenie odległości (tzw. głębi) punktu z przestrzeni trójwymiarowej. Gęsta mapa dysparcji tworzona jest, kiedy wyznacza się dysparcję dla każdego punktu sceny trójwymiarowej, który widziany jest przez obie kamery. [11]

### 1.1.5 Korekcja zniekształceń



Rysunek 1.5 Korekcja zniekształceń [10]

W rzeczywistym świecie trudno zbudować kanoniczny układ kamer. Ponadto kamery takie odbiegają od zakładanego w teorii modelu kamery otworkowej. Aby przekształcić układ obrazów w kanoniczny należy zastosować odpowiednie rzutowanie. Aby znaleźć parę macierzy przekształcających obrazy z obu kamer należy wykonać kalibrację układu ze względu na dwa rodzaje parametrów kamer:

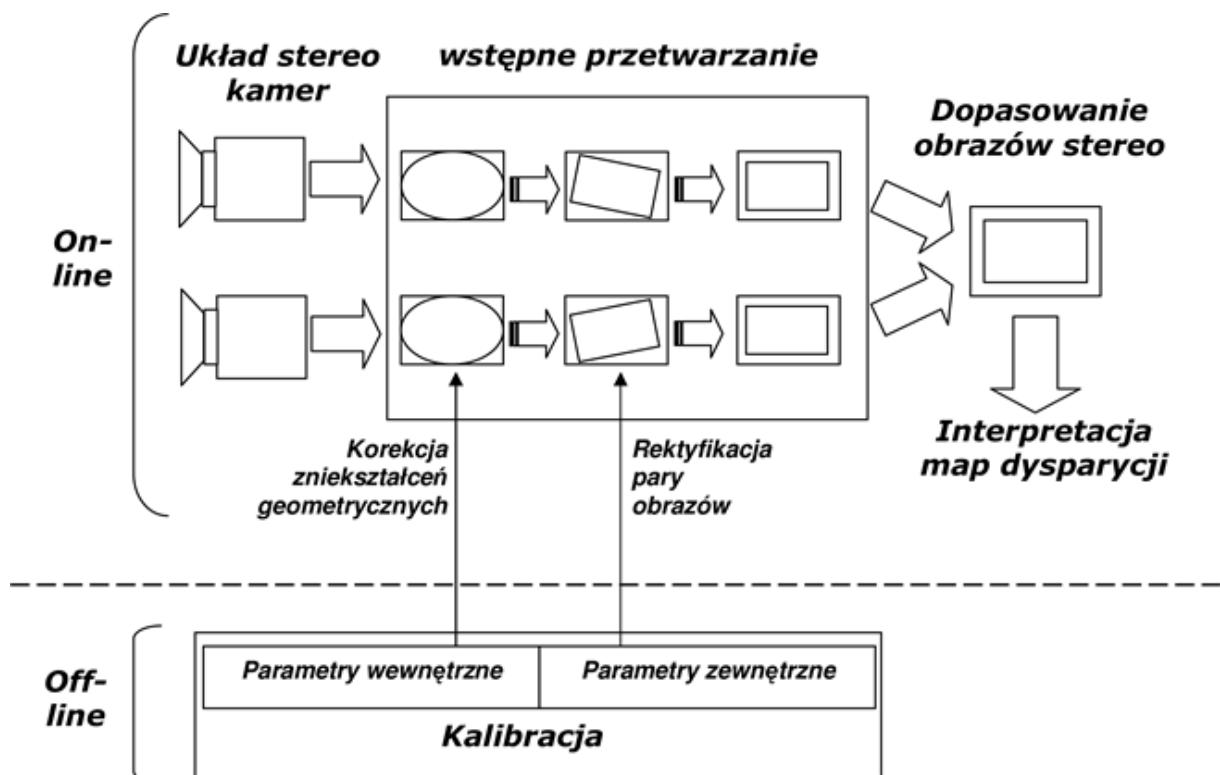
- wewnętrzne,

– zewnętrzne.

Parametry zewnętrzne mówią o rozmieszczeniu kamer względem siebie, albo względem wybranego zewnętrznego układu współrzędnych. Dla pojedynczej kamery definiuje je trójelementowy wektor  $T$  określający przesunięcie między układem współrzędnych związanym z kamerą ( $X_c, Y_c, Z_c$ ), a arbitralnie wybranym zewnętrznym układem współrzędnych ( $X, Y, Z$ ) oraz macierz obrotu  $R$  określająca obrót między osiami tych układów współrzędnych. Parametry wewnętrzne to ogniskowa kamery i parametry mówiące o zniekształceniach.

Wyznaczenie tych wszystkich parametrów pozwala zredukować zniekształcenia powstające w układach optycznych kamer i wynikające z nieidealnego rozmieszczenia ich w układzie stereowizyjnym.

### 1.1.6 Kalibracja



Rysunek 1.6 Przetwarzanie obrazów z kamer [10]

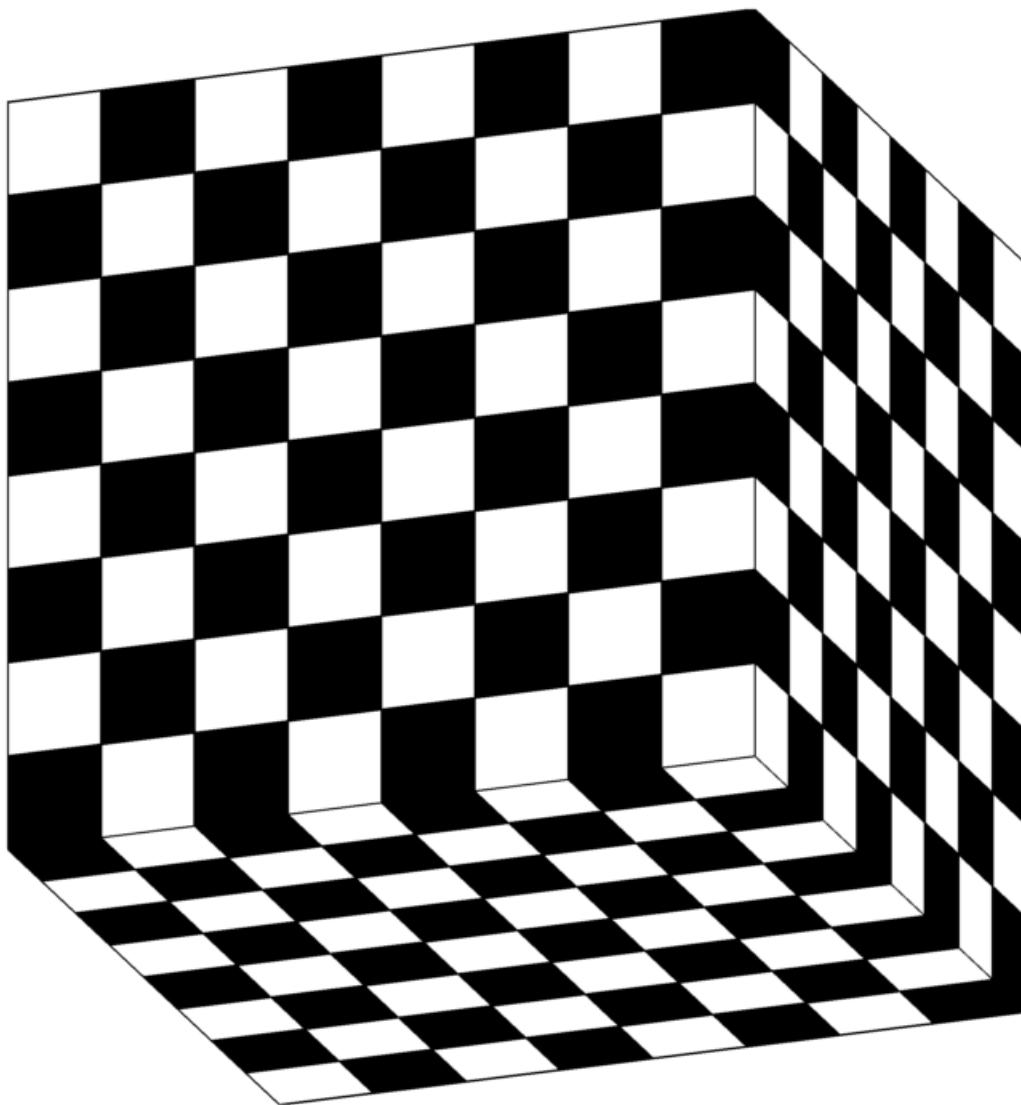
Dla poprawnej korekcji zniekształceń należy wykonać kalibrację zarówno poszczególnych kamer, jak i całego układu – uzyskać parametry przekształceń wewnętrzne i zewnętrzne.

Każda z kamer powinna zostać skalibrowana tak, by po zastosowaniu odpowiednich przekształceń, uzyskać obraz jak najbardziej zbliżony do modelowego obrazu z kamery otworkowej.

Układ kamer kalibrowany jest w celu uzyskania obrazu z układu kanonicznego w układzie niekanonicznym.

Kalibracja bezpośredniej transformacji liniowej (ang. *Direct Linear Transformation calibration*) wykorzystuje korelacje pomiędzy punktami świata i punktami obrazu z kamery do oszacowania parametrów kamery. W szczególności kalibracja DLT wykorzystuje

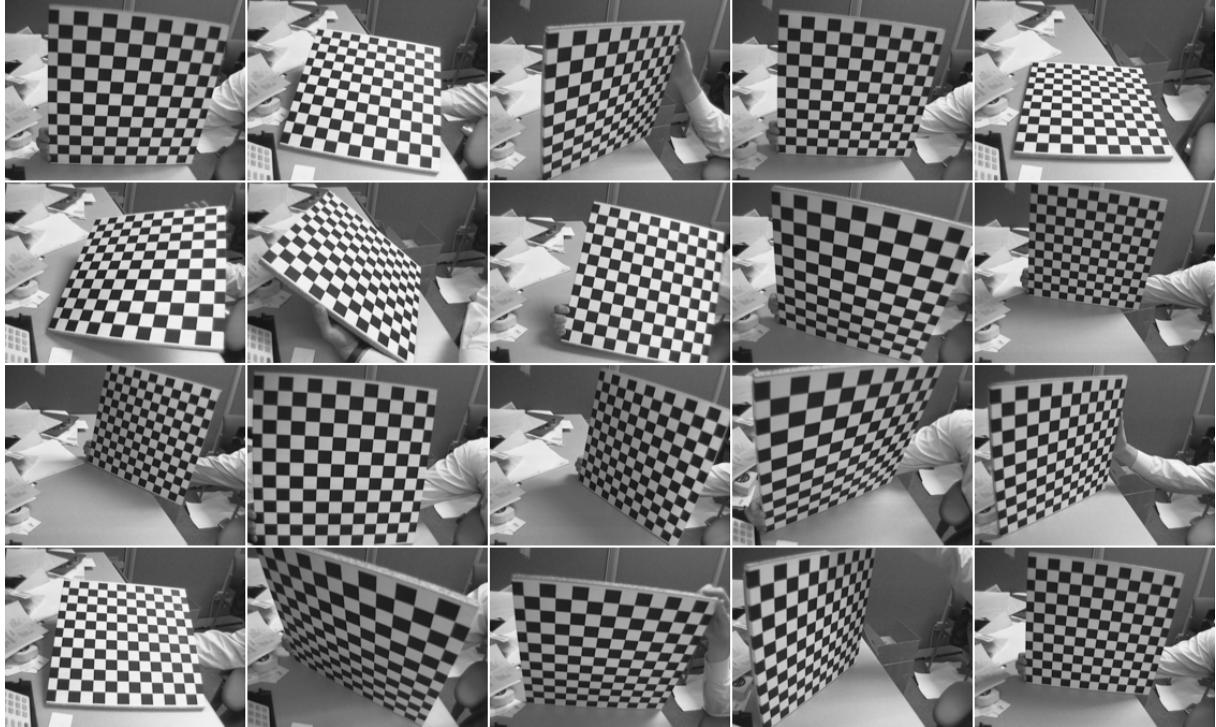
fakt, że perspektywiczny model kamery otworkowej definiuje zbiór relacji podobieństwa, które można rozwiązać za pomocą bezpośredniej transformacji liniowej. Aby zastosować to podejście, wymagane są dokładne współrzędne nie zdegenerowanego zbioru punktów w przestrzeni 3D. Powszechnym sposobem na osiągnięcie tego celu jest skonstruowanie zestawu do kalibracji kamer zbudowanego z trzech wzajemnie prostopadłych szachownic. Ponieważ narożniki każdego kwadratu są jednakowo od siebie oddalone, łatwo jest obliczyć współrzędne 3D każdego z narożników z uwzględnieniem szerokości każdego kwadratu. Zaletą kalibracji DLT jest jej prostota. Dowolne kamery mogą być kalibrowane poprzez rozwiązanie pojedynczego układu równań liniowych. Praktyczne zastosowanie kalibracji DLT jest jednak ograniczone koniecznością użycia precyzyjnego układu do kalibracji 3D oraz faktem, że w celu uniknięcia niepewności wymagane są niezwykle dokładne współrzędne 3D owego układu szachownic.



Rysunek 1.7 Układ szachownic do kalibracji kamery z użyciem DLT (widok perspektywiczny) [5]

Kalibracja wielopłaszczyznowa jest wariantem autokalibracji kamery, która pozwala na obliczenie parametrów kamery z dwóch lub więcej widoków powierzchni płaskiej. Metoda

ta kalibruje kamery poprzez rozwiązywanie konkretnego jednorodnego systemu liniowego, który rejestruje jednorodne relacje między wieloma widokami perspektywicznymi tej samej płaszczyzny. To podejście jest popularne, ponieważ w praktyce bardziej naturalne jest rejestrowanie wielu widoków pojedynczej powierzchni płaskiej – jak szachownica – niż konstruowanie precyzyjnego urządzenia do kalibracji 3D, zgodnie z wymaganiami kalibracji bezpośredniej transformacji liniowej. Poniższe rysunki pokazują praktyczne zastosowanie kalibracji wielopłaszczyznowej kamery z wielu widoków szachownicy.



Rysunek 1.8 Widok szachownicy z różnej perspektywy w kalibracji wielopłaszczyznowej [5]

### 1.1.7 Dopasowanie

Korzystając z obrazów po rektyfikacji wyszukiwane są odpowiadające sobie piksele w obrazie lewym i prawym oraz wyznaczana jest wartość dysparcji dla danej pary pikseli. Celem jest uzyskanie mapy dysparcji. W kanonicznym układzie kamer problem jest o wiele prostszy. Jest on kluczowy dla rekonstrukcji trójwymiarowej. [10]

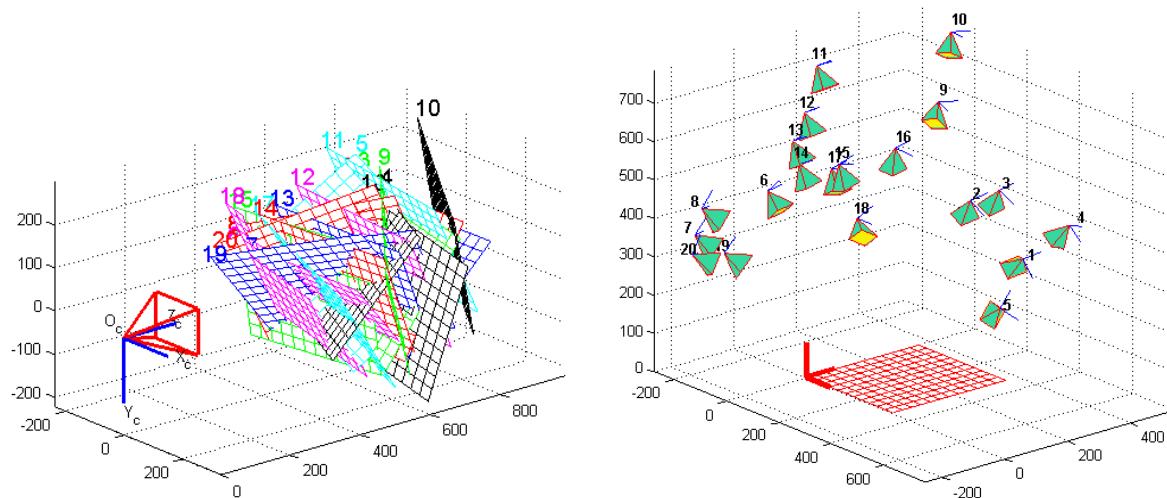
Algorytm pozwalający wyznaczyć gęstą mapę dysparcji dla danej pary obrazów jest skomplikowany i kosztowny obliczeniowo bo dla każdego punktu pierwszego obrazu należy znaleźć odpowiadający mu punkt w obrazie drugim. [11]

Całość polega na wyznaczeniu współrzędnych punktów  $(x_l, y_l)$  i  $(x_r, y_r)$  dla danego punktu w przestrzeni. Wynikiem tego zadania jest różnica współrzędnych wynikająca z wzajemnego przesunięcia obrazów danego punktu w przestrzeni w dwóch kamerach. Obliczając dysparcję jeden z obrazów przyjmuje się jako obraz odniesienia.

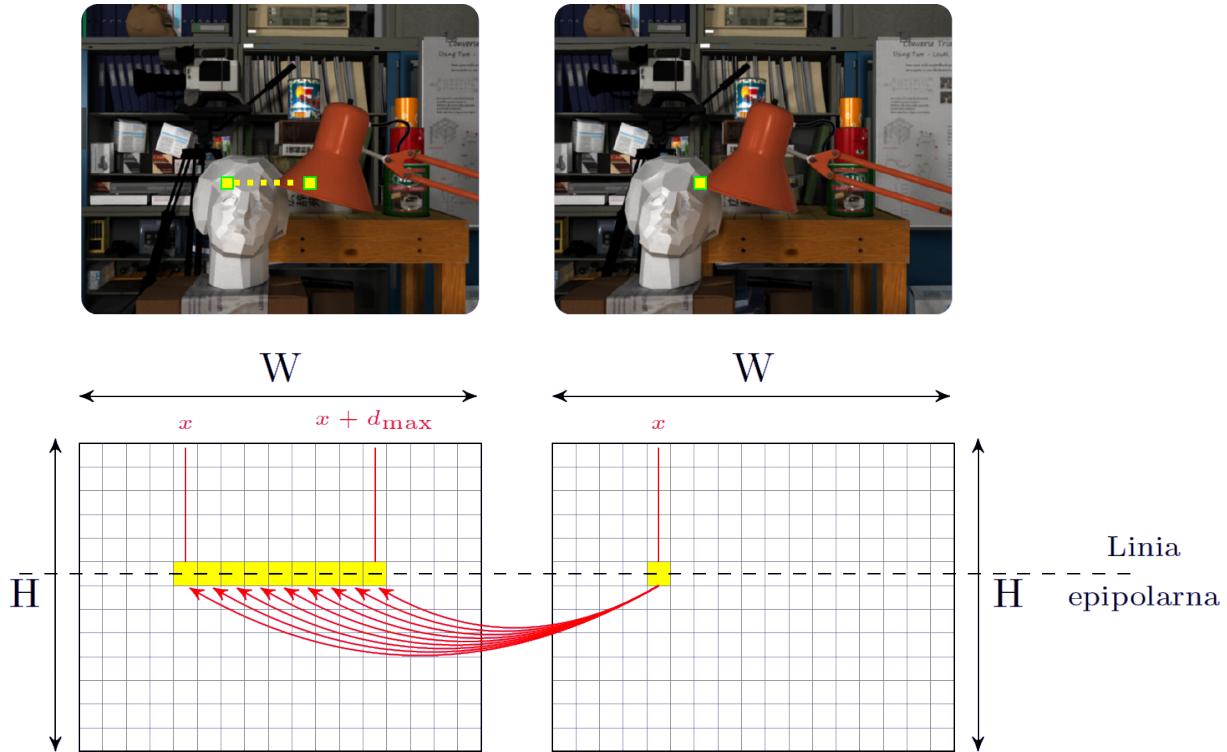
Funkcja znajdująca odległość (przesunięcie) od punktu na jednym obrazie do korespondującego punktu na drugim obrazie może mieć następującą postać:

$$d^* = \min_d |f_r(x, y) - f_l(x + d, y)|$$

gdzie  $f_r$  i  $f_l$  przyjmują wartość cechy punktu we współrzędnych  $x$  ( $x + d$ ) i  $y$ .



Rysunek 1.9 Kalibracja wielopłaszczyznowa; po lewej układ z kamerą w centrum i szachownicami w różnych pozycjach, po prawej układ z szachownicą w centrum i kamerami w różnych pozycjach [5]



Rysunek 1.10 Problem dopasowania obrazów z obu kamer [10]

#### 1.1.7.1 Cechy obrazu do dopasowywania

Algorytm dopasowujący te same elementy na dwóch obrazach może to robić zwracając uwagę na:

- jasność punktów obrazu (metody korelacyjne),
- kierunek i wartość gradientu na krawędziach,
- parametry krawędzi (krzywizna, długość, orientacja),
- parametry sylwetek.

### 1.1.7.2 Problemy przy dopasowywaniu i założenia upraszczające

Podczas próby dopasowania do siebie dwóch obrazów można napotkać następujące problemy:

- niedoskonałość pomiaru światła,
- szum,
- powierzchnie odblaskowe,
- zniekształcenia perspektywiczne,
- jednolite powierzchnie,
- powtarzalne wzory,
- obiekty przezroczyste,
- przesłanianie/nieciągłość.

Dla zminimalizowania stopnia skomplikowania obliczeń przyjmuje się pewne założenia upraszczające:

- korespondujący piksel na linii epipolarnej w drugim obrazie,
- jednoznaczność,
- symetryczność,
- jasność piksela w jednym obrazie jest taka sama lub bliska jasności odpowiadającego piksela w drugim obrazie,
- cechy geometryczne obiektów na obu obrazach różnią się nieznacznie (długość, orientacja, itp.).

### 1.1.8 Rekonstrukcja mapy głębi

Poniższe grafiki przedstawiają mapę dysparcji i wizualizację głębi badanego obrazu.



Rysunek 1.11 Mapa dysparcji surowa, po odfiltrowaniu, zwizualizowana w 3D [10]

Dysparcja jest różnicą położenia punktów na dwóch obrazach – matrycach dwóch sensorów optycznych. Wyraża się ją wzorem

$$d = |x_l - x_r|.$$

Istnieje ścisły związek pomiędzy głębią, a dysparcją. Im mniejsza jest różnica w położeniu punktu na dwóch obrazach, tym ów punkt jest dalej. Analizując trójkąty podobne otrzymujemy następujące zależności:

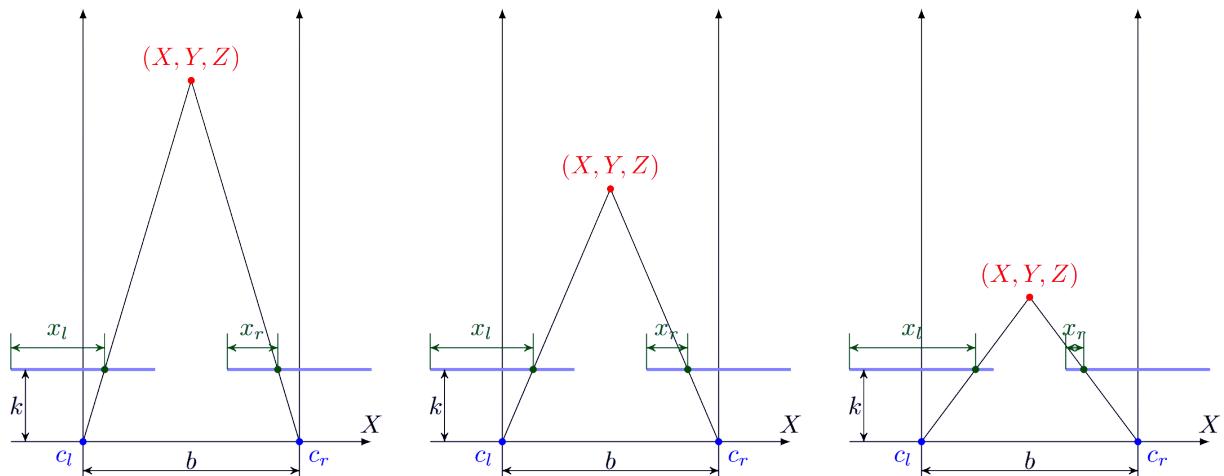
$$\frac{b}{Z} = \frac{(b + x_r) - x_l}{Z - k} \rightarrow Z = \frac{b \cdot k}{x_l - x_r} = \frac{b \cdot k}{d},$$

$$X = Z \cdot \frac{x_r}{k},$$

$$Y = Z \cdot \frac{y_r}{k},$$

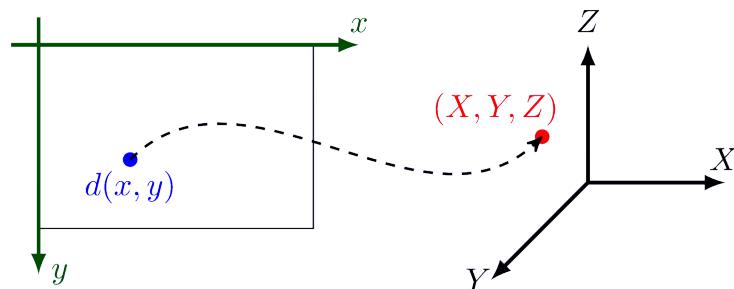
$$Z = \frac{k \cdot b}{d}.$$

Poniższa grafika przedstawia zależność odległości punktu (głębii) i różnicy we współrzędnych dwóch obrazów tego punktu – dysparycji.



Rysunek 1.12 Dysparycja, a głębia [10]

Korzystając z mapy dysparycji i parametrów uzyskanych z kalibracji wyznacza się mapę głębi. Punktowi na dwuwymiarowej płaszczyźnie obrazu dodaje się trzecią zmienną, umieszczając go w przestrzeni trójwymiarowej.



Rysunek 1.13 Przypisanie punktowi trzeciej zmiennej i umieszczenie go w przestrzeni trójwymiarowej [10]

## 1.2 OpenCV

OpenCV (ang. *Open Source Computer Vision Library*) jest biblioteką graficzną wydawaną na licencji BSD. Posiada interfejs w C++, Pythonie i Javie. Napisana w C/C++ biblioteka może korzystać z obliczeń wielordzeniowych. Dzięki OpenCL, biblioteka może używać sprzętową akcelerację.

Pliki źródłowe i instalatory można pobrać z oficjalnej strony [9], gdzie również dostępna jest dokumentacja [8].

Projekt ten bazuje na bibliotece OpenCV w wersji 4.0.1.

### 1.2.1 Algorytm dopasowywania bloków w OpenCV

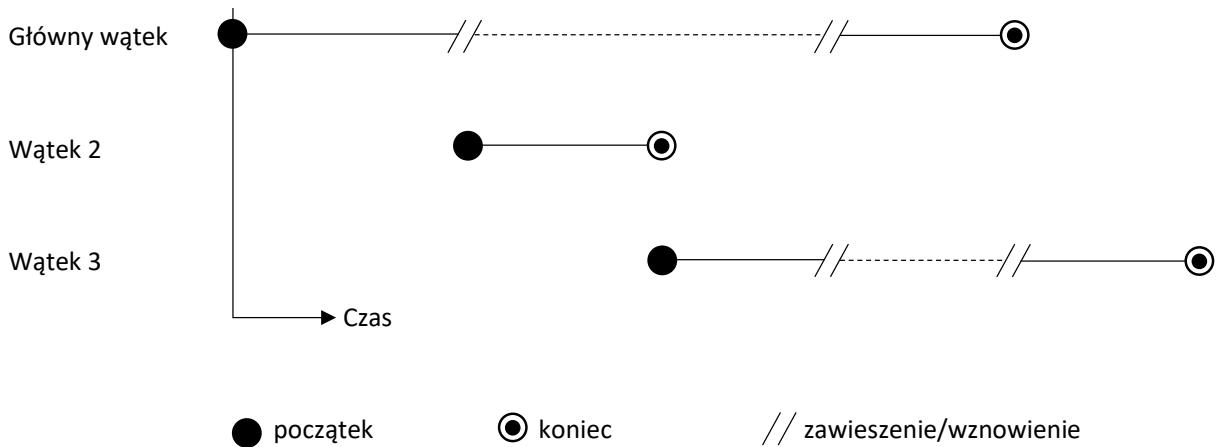
Algorytm dopasowywania bloków (ang. *Stereo Block Matching*) szacuje dysproporcję w danym punkcie, porównując mały region w tym punkcie z regionami analogicznymi wyodrębnionymi z drugiego obrazu [2]. Zaimplementowane w OpenCV dopasowywanie bazuje na tej metodzie [6].

Algorytm dopasowania blokowego przeszukuje jeden obraz w poszukiwaniu najlepiej odpowiadającego mu regionu dla wzorca w drugim obrazie. Wyszukiwanie odbywa się, jak wspomniano we wstępnie teoretycznym, na linii epipolarnej – obszar poszukiwań przesuwany jest wzdłuż niej i dopasowywany do wzoru w pierwszym obrazie. [2]

### 1.2.2 Filtrowanie WLS

Obraz odczytany z sensorów optycznych jest obarczony szumem wynikającym z niedoskonałości matryc światłoczułych. Zaszumienie jest jednym z wymienionych wyżej problemów utrudniających znalezienie korespondujących punktów w parze obrazów. Filtrowanie polega na usunięciu szumu z obrazu. Użyty w projekcie filtr mapy dysparycji – `cv::ximgproc::DisparityWLSFilter` – oparty jest na filtrze WLS (ang. *Weighted Least Squares*) będącym uogólnieniem metody najmniejszych kwadratów i regresji liniowej [4, 12]. Filtrowaniu poddawana jest mapa dysparycji.

## 1.3 Zrównoleglenie obliczeń przy pomocy wielowątkowości



Rysunek 1.14 Schemat działania wielowątkowości

Obliczenia równoległe to forma obliczeń, gdzie wiele instrukcji jest wykonywanych jednocześnie. Są one możliwe przy użyciu:

1. wielowątkowości,
2. wieloprocesowości,
3. przetwarzania rozproszonego,
4. GPGPU (wykonywanie obliczeń ogólnego przeznaczenia za pomocą procesora karty graficznej, ang. *General-Purpose Computing on Graphics Processing Units*) [13].

Wielowątkowość (ang. *multithreading*) jest cechą systemu operacyjnego, dzięki której w ramach procesu wykonywane jest kilka zadań nazywanych wątkami. Te zadania to

kolejne ciągi instrukcji wykonywane do pewnego stopnia niezależnie. Wszystkie wątki w obrębie jednego procesu współdzielą tę samą wirtualną przestrzeń adresową zawierającą wykonywany kod programu i wszystkie jego dane. Poniższa grafika wizualizuje ogólną zasadę działania wielowątkowości.

Wielowątkowość charakteryzuje pewne cechy.

- Wszystkie wątki wykonują się w obrębie jednego procesu – jeden proces dysponuje wieloma instancjami wykonawczymi (wątkami).
- Wątki umożliwiają przetwarzanie współbieżne, np. gdy zachodzi potrzeba wykonania wielu zadań jednocześnie. Wiąże się to ze zwiększeniem wydajności przetwarzania, o ile oczywiście istnieją zasoby sprzętowe do tego przeznaczone (wiele procesorów jednordzeniowych lub pojedynczy procesor wielordzeniowy). W niektórych przypadkach użycie wątków może doprowadzić do obniżenia wydajności, ponieważ najczęściej konieczne jest przy tym zastosowanie odpowiednich mechanizmów synchronizacji (np. semafory).
- Wszystkie wątki w obrębie procesu współdzielą tę samą wirtualną przestrzeń adresową i korzystają z tych samych zasobów systemowych.
- Komunikacja między wątkami, w odróżnieniu od komunikacji międzyprocesowej, jest prosta do wykonania. W programach wielowątkowych wystarczy odwoływać się do tych samych zmiennych i obiektów. Komunikacja międzyprocesowa wymaga zastosowania mechanizmów IPC (ang. *InterProcess Communication*).
- Współdzielenie wirtualnej przestrzeni adresowej może nieść zagrożenie, kiedy „wadliwy” wątek może zagrozić egzekucji całego procesu.
- Każdy wielowątkowy system operacyjny zapewnia swoiste metody synchronizacji wątków, które z powyższych przyczyn muszą być obowiązkowo zaimplementowane.

Wielowątkowe systemy operacyjne to m.in. Windows 95, Windows NT i ich następcy, BeOS, Unix i systemy bazujące na jądrze Linux. W tym projekcie zdecydowano się na wielowątkowość ze względu na prostą implementację i jej znajomość. Całość zaprojektowano pod kątem działania na systemach Linux.



## Rozdział 2

# Implementacja

W tym rozdziale opisana jest implementacja systemu generowania mapy głębi z po- dzialem na główne elementy tegoż systemu:

- kalibracja,
- klasyczne generowanie mapy głębi,
- wielowątkowe generowanie mapy głębi,
- pomiar szybkości obliczeń.

Cały projekt dostępny jest w publicznym repozytorium `git` pod adresem <https://github.com/Adrian94F/StereoOnSteroids>.



Rysunek 2.1 Architektura systemu

Pod względem architektury w systemie można wydzielić dwa podstawowe moduły odpowiedzialne za:

1. kalibrację układu kamer,
2. generowanie mapy glebi:
  - (a) jednowątkowe,
  - (b) wielowątkowe.

Zrównoleglenie obliczeń zastosowano w generowaniu mapy glebi przez podział obrazów do przetworzenia na części i rozdzielenie ich pomiędzy wątki (co jest opisane w dalszej części). Czas wykonywania każdej z operacji jest mierzony i zapisywany celem porównania wersji klasycznej i zrównoległej.

## 2.1 Kalibracja układu pomiarowego

Do właściwego działania każdy układ kamer wymaga kalibracji. Osobno zaimplementowano kalibrację pojedynczej kamery i kalibrację całego układu.

### 2.1.1 Kolekcja danych do kalibracji

Dane użyte do kalibracji zostały zebrane przez fotografowanie szachownicy o układzie 9x6, składającej się z kwadratowych pól o boku 2,65 cm, ułożonej pod różnym kątem i w różnych odległościach od kamery. Zdjęcia wykonano w mocnym oświetleniu celem polepszenia jakości przechwytywanych obrazów.

By utworzyć kolekcję danych do kalibracji (par obrazów pozyskanych z lewej i prawej kamery), należy wykonać następujące kroki:

1. ustawić ścieżkę do katalogu z pozyskanymi obrazami w zmiennej `imPath` (ustawianej w pliku `header.hpp`),
2. skompilować i uruchomić program `read`,
3. zapisywać kolejne zdjęcia po odpowiednim ułożeniu szachownicy w kadrze przez wciskanie dowolnego klawisza (oprócz `ESC`),
4. wcisnąć `ESC` celem zakończenia zapisywania i wyjścia z programu.

### 2.1.2 Kalibracja pojedynczej kamery

Kalibracja odbywa się przez uruchomienie programu `calibrate`. Z lokalizacji obrazów opisanej w poprzednim punkcie pobierane są obrazy dla lewej, a następnie prawej kamery. Dane kalibracyjne każdej z kamer zapisywane są w plikach z rozszerzeniem `.yaml`.

### 2.1.3 Kalibracja układu kamer

Uruchomienie programu `calibrate-stereo` pozwala na użycie zapisanych danych kalibracji poszczególnych kamer i pozyskanych obrazów i skalibrowanie całości, jako układu, by uzyskać macierze przekształceń obrazów pozyskanych z kamer w obrazy pozyskane z kanonicznego układu idealnych modeli kamer. Dane kalibracyjne układu kamer również zapisywane są w pliku z rozszerzeniem `.yaml`.

Dla sprawdzenia poprawności działania jest możliwość uruchomienia dodatkowego programu, `undistort_rectify`, który otworzy plik do kalibracji stereo i obrazy ze wskazanej przez zmienną `imPath` lokalizacji, a następnie przekształci je i zapisze zmodyfikowane ich kopie z nową nazwą w tym samym katalogu.



Rysunek 2.2 Zdjęcie przed przekształceniem



Rysunek 2.3 Zdjęcie po przekształceniu

## 2.2 Generowanie mapy głębi

Mapa głębi generowana jest na żywo z uzyskanych z układu kamer obrazów. W podstawowej wersji generowanie odbywa się bez zrównoleglenia. Stworzona na potrzeby tej pracy zrównoleglona wersja ma pozwolić na przyspieszenie tego procesu i zwiększenie liczby generowanych klatek na sekundę.

Uruchomienie programu `disparity` pozwala na rozpoczęcie generacji mapy głębi na żywo. Odczytuje on wspomniany wcześniej plik konfiguracyjny (`calibStereo.yaml`), a następnie obrazy z kamer – aż do momentu wciśnięcia klawisza ESC. Wciśnięcie go powoduje zatrzymanie przetwarzania w danym trybie i przejście do kolejnego. Pierwszy tryb to jednowątkowy, kolejne wielowątkowe z ustalonymi liczbami tworzonych wątków. Po uruchomieniu wyświetcone zostanie okno ze skorygowanym obrazem z kamer i wyliczoną mapą głębi.

Do testowania poprawności działania użyto obrazów testowych pochodzących z dokumentacji OpenCV [7].



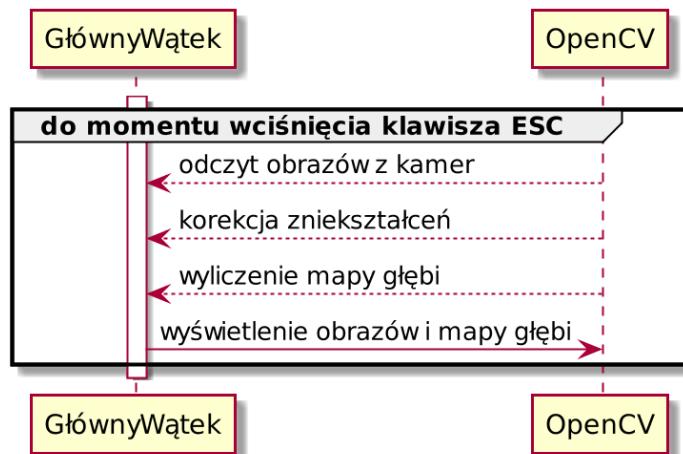
Rysunek 2.4 Obrazy testowe (lewy na górze, prawy na dole) [7]



Rysunek 2.5 Wzorcowa mapa głębi [7]

### 2.2.1 Tryb jednowątkowy

W trybie jednowątkowym działanie programu jest proste. W pętli odczytywane są obrazy z kamer, usuwane są z nich zniekształcenia, a następnie wyprostowane obrazy wyświetlane razem z mapą głębi.



Rysunek 2.6 Schemat działania programu w wersji jednowątkowej

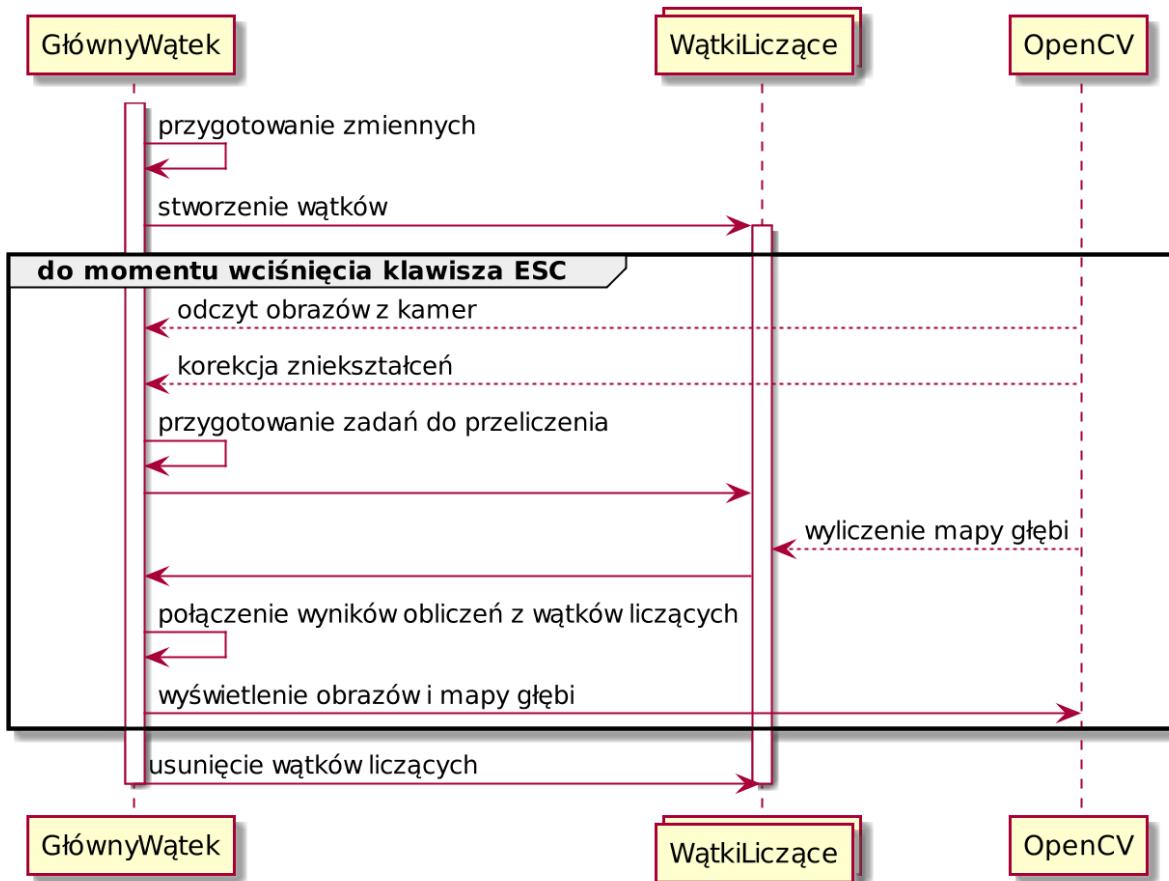
### 2.2.2 Tryb wielowątkowy

W trybie wielowątkowym główny wątek odpowiada za pobranie obrazów, przygotowanie zadań, połączenie rezultatów i wyświetlenie ich. Wątki liczące po uruchomieniu oczekują na informację z wątku głównego o gotowości zadań do przeliczenia (lub o konieczności zakończenia działania).

Liczba zadań jest równa liczbie wątków – każdy z wątków oblicza zadanie o odpowiednim, przydzielanym przy tworzeniu wątku, numerze.

Do tworzenia wątków użyto wątków z biblioteki standardowej – `std::thread`. Zadania do przeliczenia, ich statusy i wyniki przechowywane są w kontenerach `std::vector`. Wątek główny ustawia status na `TODO` i czeka, aż wątki liczące ustawią statusy swoich

zadań na Done. Wątki liczące czekają, aż status zadania zmieni się na TODO (bądź Abort – wtedy kończą swoje działanie). Po otrzymaniu zadania wyliczają mapę dysparacji odpowiednich fragmentów obrazów wejściowych. Gdy je uzyskają, wstawiają je do wektora rezultatów i ustawiają status zadania na Done. Następnie w wątku głównym odbywa się łączenie wyników obliczeń. Po połączeniu są one wyświetlane.



Rysunek 2.7 Schemat działania programu w wersji wielowątkowej

### 2.2.3 Synchronizacja i komunikacja między wątkami

Synchronizacja między wątkami jest konieczna. Wątek roboczy może przetwarzać swoje zadanie wtedy, kiedy zostanie przygotowane. Wątek główny może łączyć wyniki zadań, kiedy wszystkie wątki robocze zgłoszą ich gotowość. Celem synchronizacji użyto zmiennych warunkowych – `std::condition_variable` – w dwóch miejscach:

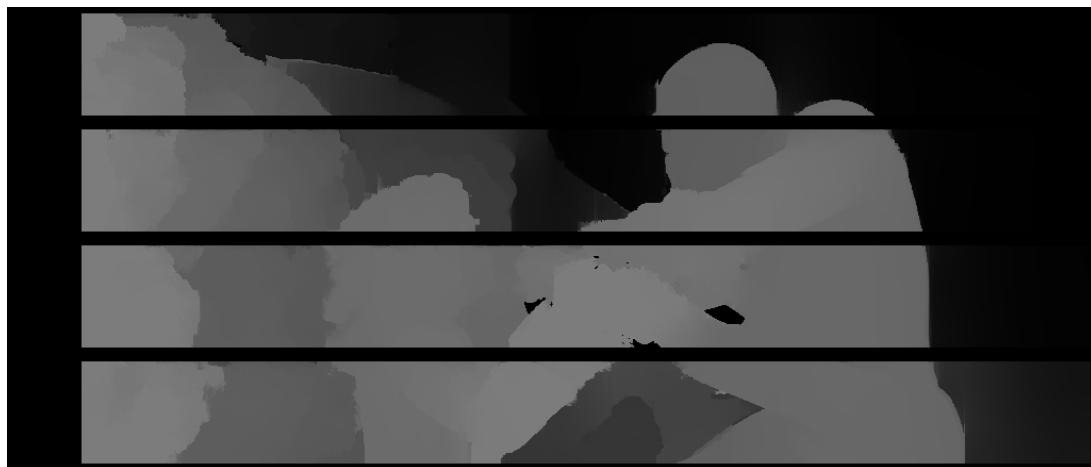
- w wątku roboczym przy oczekiwaniu na nowe zadanie,
- w wątku głównym przy oczekiwaniu na realizację zadań w wątkach roboczych.

Po przygotowaniu zadania dla danego wątku wywoływana jest w głównym wątku metoda `notify_all()`, a każdy z wątków roboczych sprawdza, czy jego zadanie zostało przygotowane do przeliczenia. Sytuacja wygląda analogicznie w przypadku skończenia przetwarzania zadania. Jego status jest ustawiany na `Done`, wątek roboczy powiadamia wątek główny o zmianie statusu, a ten czeka, aż wszystkie wątki ustawią statusy swoich zadań na `Done`.

### 2.2.4 Napotkane problemy

Podczas tworzenia wielowątkowego systemu napotkano na pewne problemy i podjęto próby rozwiązania ich.

Generowana mapa głębi posiada czarne krawędzie. Czarne krawędzie wynikają z użycia algorytmu dopasowywania bloków – do generowania mapy głębi. Przy krawędziach nie można dopasować bloku, kiedy wychodzi on poza obraz. W takich miejscach dysparcja ustawiana jest na 0, co w wynikowym obrazie (w przypadku tej pracy, gdzie nie zmieniano kolorów, w jakich mają być widoczne rezultaty) widać w postaci pikseli w kolorze czarnym. Celem wyeliminowania tego problemu fragmenty obrazów do przeanalizowania są przygotowywane z odpowiednim marginesem, a od wynikowej mapy głębi odcinany jest powstały czarny margines.



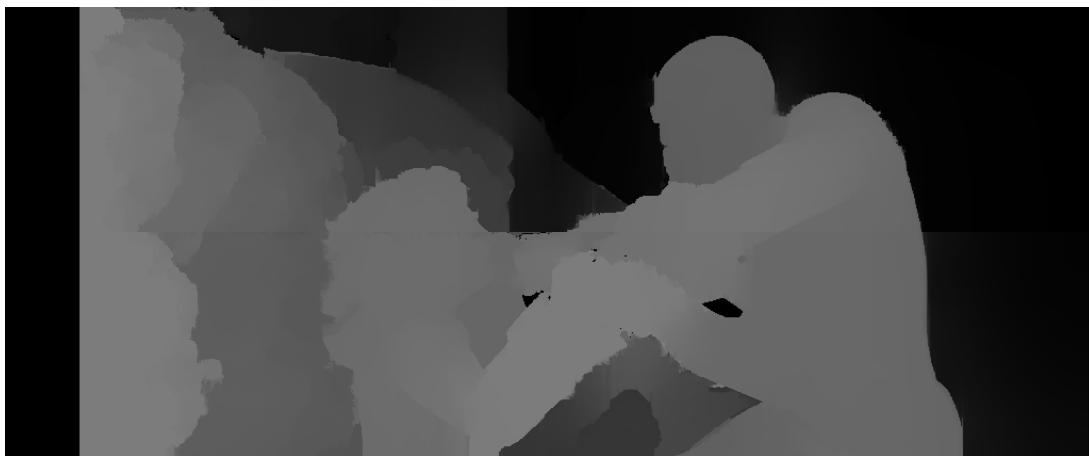
Rysunek 2.8 Mapa dysparcji w trybie z czterema wątkami roboczymi



Rysunek 2.9 Wygenerowana mapa głębi dla danych testowych bez użycia wielowątkowości



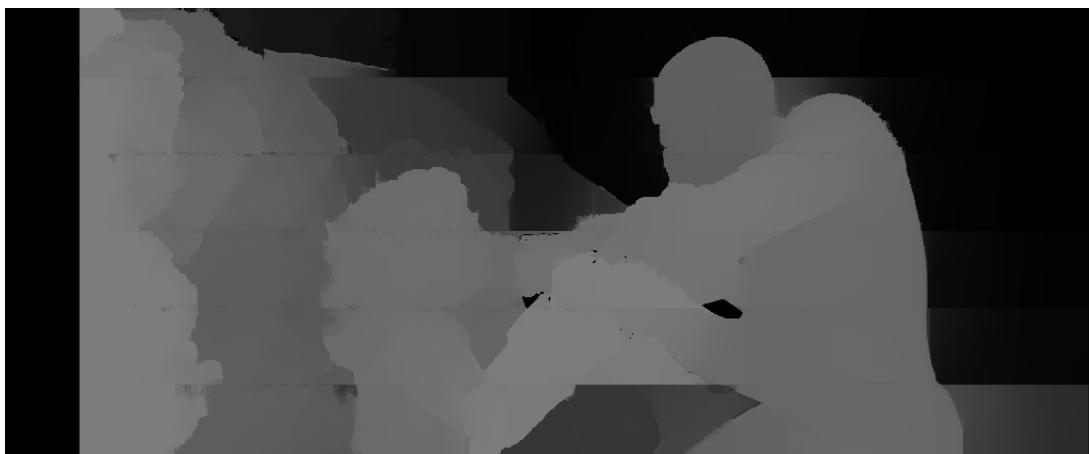
Rysunek 2.10 Wygenerowana mapa głębi dla danych testowych z użyciem wielowątkowości, z podziałem na 1 zadanie, po zredukowaniu czarnych obszarów



Rysunek 2.11 Wygenerowana mapa głębi dla danych testowych z użyciem wielowątkowości, z podziałem na 2 zadania, po zredukowaniu czarnych obszarów



Rysunek 2.12 Wygenerowana mapa głębi dla danych testowych z użyciem wielowątkowości, z podziałem na 4 zadania, po zredukowaniu czarnych obszarów



Rysunek 2.13 Wygenerowana mapa głębi dla danych testowych z użyciem wielowątkowości, z podziałem na 6 zadań, po zredukowaniu czarnych obszarów



Rysunek 2.14 Wygenerowana mapa głębi dla danych testowych z użyciem wielowątkowości, z podziałem na 8 zadań, po zredukowaniu czarnych obszarów



Rysunek 2.15 Wygenerowana mapa głębi dla danych testowych z użyciem wielowątkowości, z podziałem na 16 zadań, po zredukowaniu czarnych obszarów

Generowane fragmenty po usunięciu czarnych krawędzi i złączeniu nie są identyczne na brzegach (widać nierówne „przejścia” pomiędzy fragmentami), co widać coraz bardziej

wraz ze wzrostem liczby fragmentów, z których „sklejona” ma zostać wynikowa mapa dysparcji. Wynika to z zastosowania post-filtrowania wygenerowanych częściowych map głębi. [7] Zmiana tego zachowania wymagałaby zmian w architekturze systemu i łączenia ze sobą dodatkowych danych przy okazji łączenia ze sobą fragmentów map głębi.

## 2.3 Pomiar czasu

Aby zapewnić dokładny pomiar czasu wykonywania poszczególnych operacji (w szczególności generowania mapy głębi), wykorzystano zegar wysokiej rozdzielczości z biblioteki standardowej – `std::chrono::high_resolution_clock`. Klasa ta reprezentuje zegar z najkrótszym okresem tykania zapewnionym przez implementację. Może być aliasem `std::chrono::system_clock`, `std::chrono::steady_clock` lub innego, niezależnego zegara [1].

# Rozdział 3

## Eksperiment

### 3.1 Plan eksperimentu

Poniżej pisany jest plan eksperimentu,

1. Przeprowadzić kalibrację zbudowanego układu kamer.
2. Generować mapę głębi bez zrównoleglenia.
3. Generować mapę głębi w wersji ze zrównolegleniem, dzieląc obliczenia na 1-20 zadań.

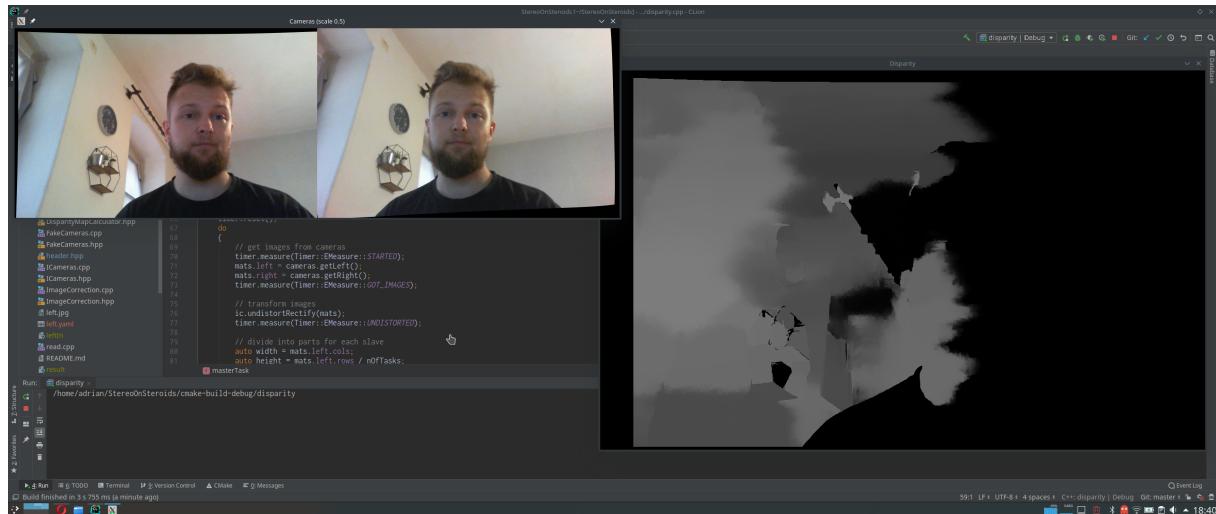
Do eksperimentu użyto komputera Lenovo E540 z zainstalowanym systemem operacyjnym Manjaro KDE w wersji 18.03. Posiadany przez niego procesor, Intel Core i5-4200, charakteryzuje się 2 rdzeniami (4 wątkami), bazową częstotliwością 2,50 GHz (zwiększaną do 3,10 GHz w trybie Turbo) i 3 MB cache'u. Wielkość pamięci RAM zainstalowanej w komputerze wynosi 16 GB. W układzie kamer użyto Microsoft LifeCam HD-3000. Kamery te dają obraz w rozdzielcości 1280 x 720 px i z szybkością 30 klatek na sekundę.



Rysunek 3.1 Skonstruowany układ kamer

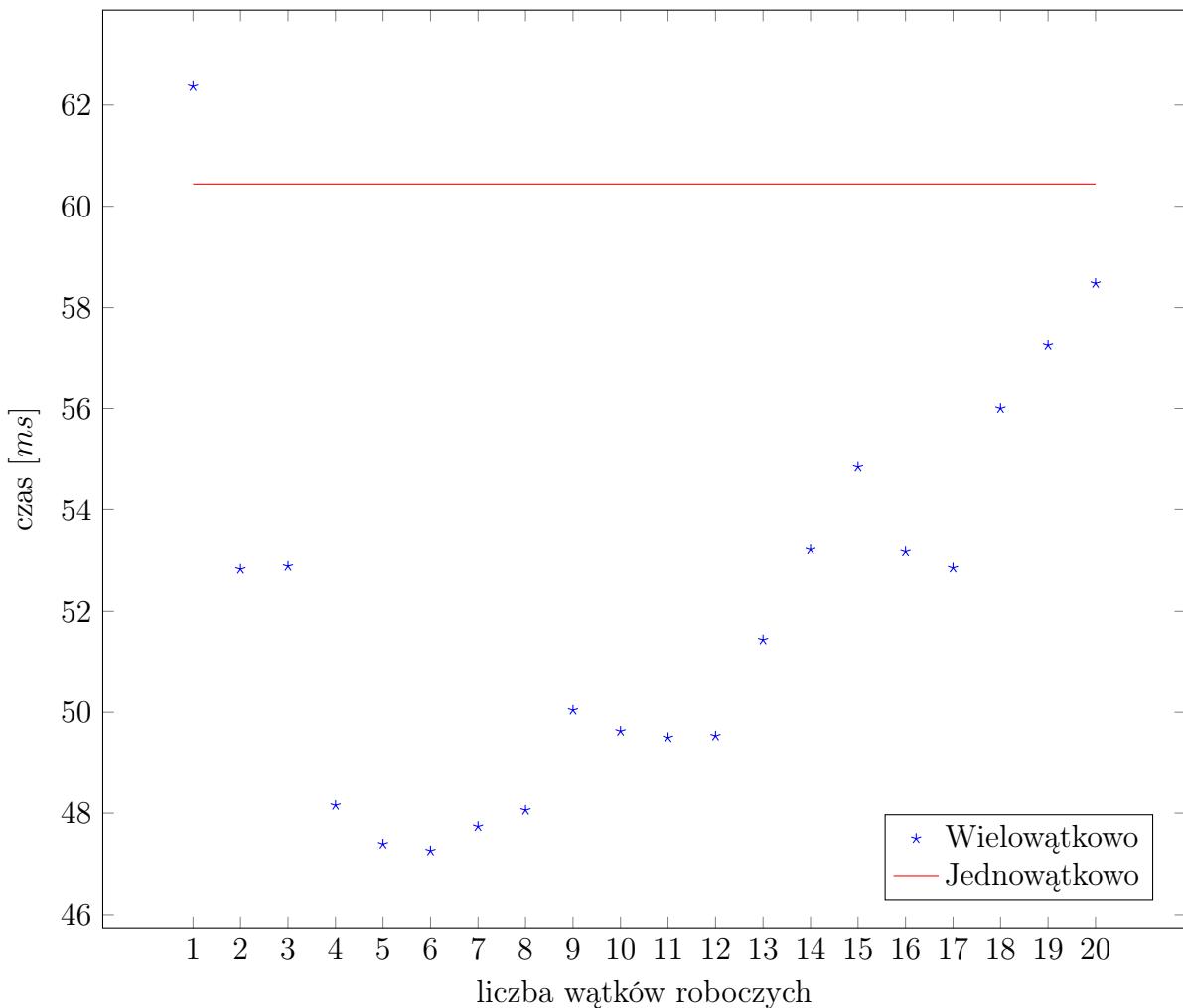
### 3.2 Przebieg i wyniki eksperimentu

Zrzut ekranu przedstawia uruchomiony program testowy. Generowane mapy głębi nie posiadają czarnych pasów dzięki zastosowaniu opisanej we wcześniejszym rozdziale korekcji przy wyznaczaniu zadań.



Rysunek 3.2 Zrzut ekranu podczas trwania eksperymentu

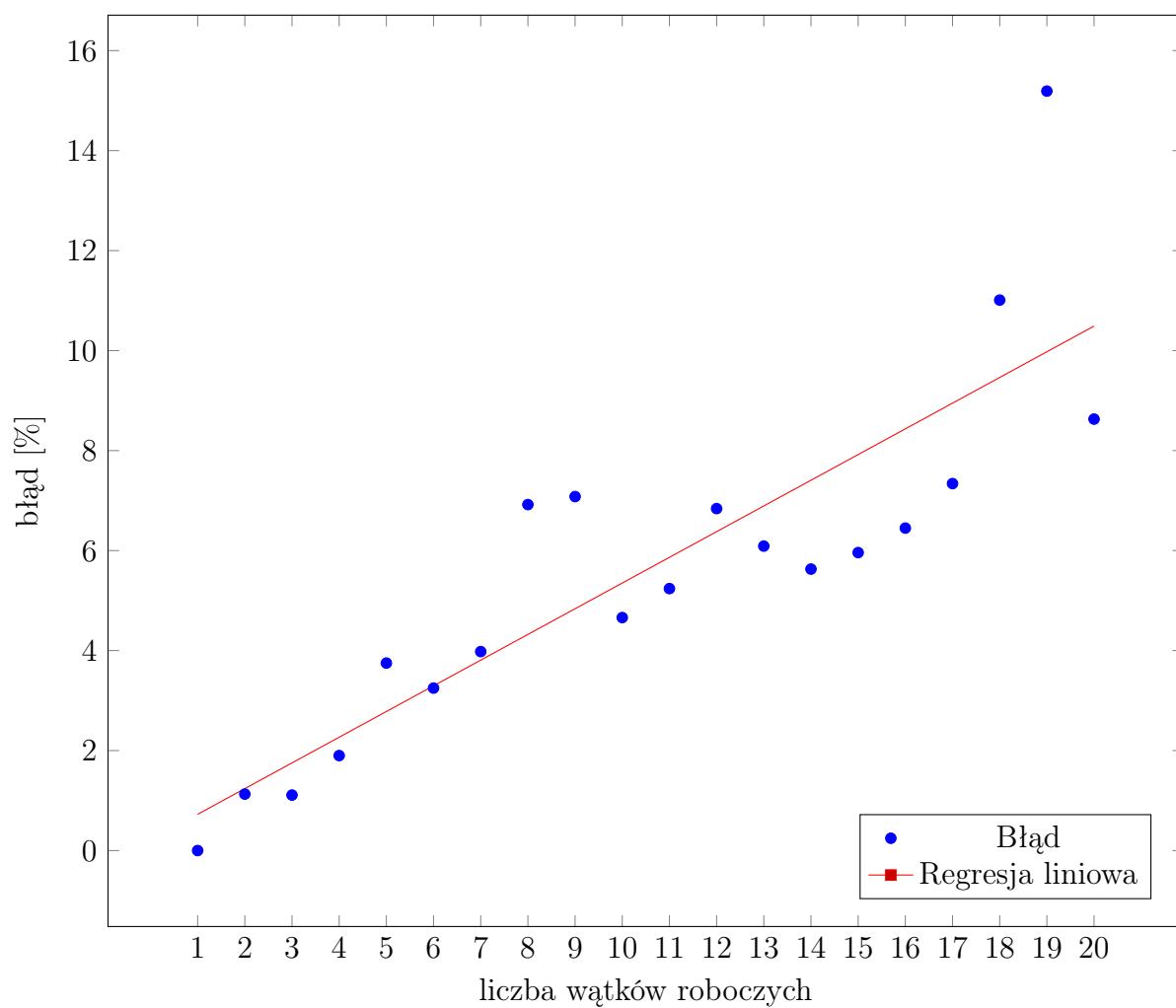
Dalej przedstawione są wyniki przeprowadzonego eksperymentu. Jest to wykres czasu liczenia mapy dysparycji i tabela z czasami działania w zależności od wersji.



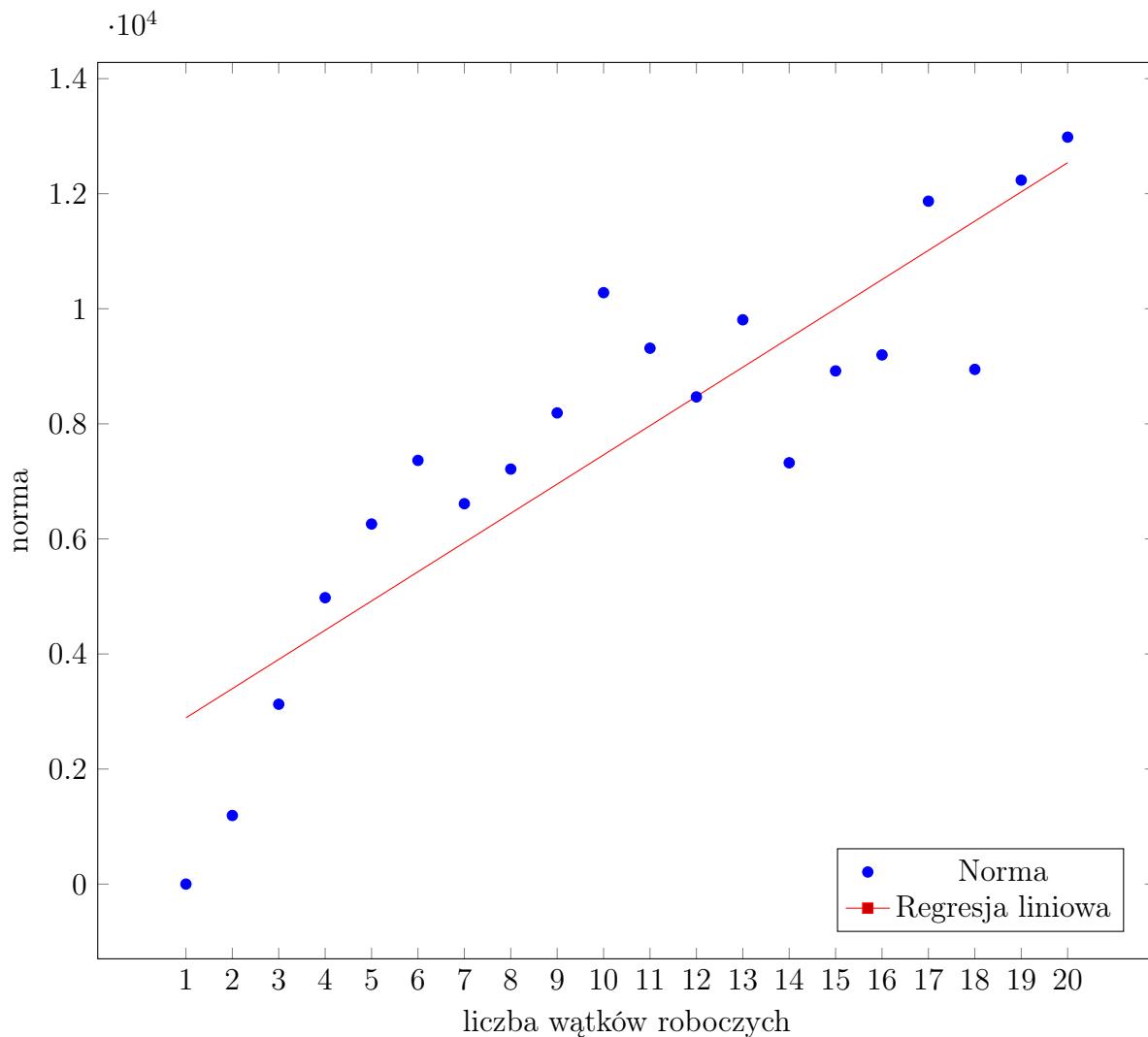
Rysunek 3.3 Wykres czasu liczenia mapy dysparycji w zależności od liczby wątków liczących w wersji wielowątkowej i w wersji jednowątkowej

Tabela 3.1 Średnie czasy operacji (zaokrąglone do  $1\mu s$ ) w wersji jednowątkowej (brak osobnych wątków liczących) i wielowątkowej w zależności od liczby równolegle wykonywanych zadań

Wątki	Podział operacji na zadania	Przygotowanie zadania	Wykonanie zadania	Wydzielenie zadania	Wyświetlenie zadania	Czas CPU	Suma
Brak	2,130	22,786	—	—	60,439	2,121	3,675
1	1,553	22,813	0,291	62,073	0,002	62,366	1,770
2	1,495	22,706	1,102	51,580	0,147	52,829	1,859
3	1,623	22,408	1,164	51,496	0,227	52,886	1,835
4	1,482	22,626	4,335	43,506	0,313	48,154	1,779
5	1,525	22,405	7,620	39,354	0,413	47,386	1,797
6	1,552	22,006	11,670	35,124	0,458	47,252	1,750
7	1,483	21,926	13,071	34,154	0,509	47,734	1,791
8	1,557	22,201	16,487	30,933	0,637	48,057	1,787
9	1,538	22,113	18,716	30,704	0,620	50,041	1,815
10	1,538	22,473	22,098	26,790	0,735	49,623	1,826
11	1,625	22,113	23,207	25,515	0,774	49,496	1,793
12	1,547	21,900	24,230	24,475	0,824	49,529	1,778
13	1,613	22,176	26,424	24,118	0,891	51,434	1,835
14	1,593	22,049	27,572	24,684	0,957	53,213	1,830
15	1,601	22,414	29,869	23,969	1,015	54,853	1,886
16	1,626	21,992	30,601	21,556	1,014	53,172	1,848
17	1,555	22,134	29,915	21,865	1,073	52,853	1,799
18	1,600	21,958	33,945	20,984	1,071	56,000	1,801
19	1,610	22,140	35,726	20,391	1,143	57,260	1,843
20	1,683	21,647	35,161	22,120	1,195	58,476	1,829



Rysunek 3.4 Wykres błędu wygenerowanej wielowątkowej mapy dysparycji w zależności od liczby wątków liczących w wersji wielowątkowej względem wzorca z wersji jednowątkowej przy zastosowaniu post filtrowania generowanych fragmentów



Rysunek 3.5 Wykres normy różnicy wygenerowanej wielowątkowo mapy dysparycji i wzorcowej – wygenerowanej w wersji jednowątkowej – w zależności od liczby wątków liczących w wersji wielowątkowej przy zastosowaniu post filtrowania generowanych fragmentów

### 3.3 Analiza wyników

Generowana z obrazu z kamer mapa dysparycji może być niewyraźna z powodu niedostatecznej jakości sensorów optycznych używanych kamer. Niewystarczająca jakość mogła spowodować niedostatecznie dobrą kalibrację lub zbyt duże zaszumienie obrazu w warunkach zastanych podczas eksperymentu.

Na podstawie wyników można zauważyc wzrost czasu przetwarzania przy przejściu na wersję wielowątkową (w przypadku 1 wątku roboczego). Wiązać się to może z dodatkowym narzutem spowodowanym przez stworzone wątki i ich synchronizację. Największy wzrost wydajności w wersji wielowątkowej widoczny jest przy wykorzystaniu 6 wątków liczących mapę dysparycji.

Widoczne są „schodki”. Procesor użyty w badaniu posiada 2 rdzenie i 4 wątki. Owe „schodki” mogą wynikać z użytego procesora. Dla 1 wątku roboczego (czyli w sumie 2 wątków) widoczny jest spadek wydajności w porównaniu do wersji jednowątkowej. 2 i 3 wątki robocze (3 i 4 wątki ogółem) powodują spadek czasu wykonywania obliczeń o 15%. W uzy-

skanych wynikach zaobserwować można fakt, że największy wzrost wydajności w tym wypadku jest przy zastosowaniu 6 wątków liczących (7 ogółem) i dla takiej liczby procesor najlepiej radzi sobie z przełączaniem między nimi. Liczenie dysparcji jest tu mniejsze o 22% od tego z wersji jednowątkowej, a generowanie całej klatki szybsze o 17%.

Rosnący wraz z liczbą wątków roboczych błąd wynika z post filtrowania częściowych wyników, a nie całej mapy głębi. Filtrowanie powinno być przeprowadzone na wynikowej, stworzonej ze złączonych fragmentarycznych wyników obliczeń, mapy głębi.

## Rozdział 4

### Podsumowanie i wnioski

W ramach tej pracy stworzono system do generowania mapy głębi na podstawie obrazów z kamery stereowizyjnej bazującej na dwóch połączonych ze sobą na stałe kamerach.

Zauważono, że na użytym komputerze czas obliczania mapy dysparcji był najmniejszy w przypadku uruchomienia wersji wielowątkowej z 3 wątkami roboczymi.

Zauważalny wzrost wydajności mógłby być lepiej widoczny, gdyby zadanie było bardziej czasochłonne, a wyświetlanie obrazu niezależne od przetwarzania poszczególnych klatek. Prawdopodobnie w przypadku, kiedy zastosowano by kamery o lepszej jakości i większej rozdzielczości, obliczanie map dysparcji wymagałoby więcej czasu i różnicę w wydajności byłaby bardziej zauważalna.

Wzrost błędu wraz ze wzrostem liczby wątków liczących został zauważony pod koniec prac nad projektem. Można go uniknąć poprzez przeprojektowanie architektury systemu i zastosowanie post filtrowania wynikowej mapy głębi zamiast poszczególnych fragmentów obliczanych w obrębie jednego wątku.

Użycie lepszej jakości kamer mogłoby pozytywnie wpłynąć na wyjściową mapę głębi. Prawdopodobnie poprzez zastosowanie odszumiania pobieranych z kamer obrazów można by osiągnąć mapę głębi w lepszej jakości.



# Bibliografia

- [1] cppreference. std::chrono::high\_resolution\_clock. [https://en.cppreference.com/w/cpp/chrono/high\\_resolution\\_clock](https://en.cppreference.com/w/cpp/chrono/high_resolution_clock), 2018.
- [2] A. Fusiello. Stereo matching: an overview. <http://www.diegm.uniud.it/fusiello/teaching/mvg/stereo.pdf>.
- [3] A. G.-H. im. Stanisława Staszica w Krakowie. Kalibracja stereowizyjnego systemu wizyjnego z użyciem pakietu matlab. [http://home.agh.edu.pl/~hyla/Downloads/ASiUT/Cwiczenie\\_2/ASiUT\\_2.pdf](http://home.agh.edu.pl/~hyla/Downloads/ASiUT/Cwiczenie_2/ASiUT_2.pdf), 2011.
- [4] Minitab. Weighted regression. <https://support.minitab.com/en-us/minitab/18/help-and-how-to/modeling-statistics/regression/supporting-topics/basics/weighted-regression/>.
- [5] B. moore81. Chessboard detection. [https://en.wikipedia.org/wiki/Chessboard\\_detection](https://en.wikipedia.org/wiki/Chessboard_detection), 2014.
- [6] OpenCV. cv::stereobm class reference. [https://docs.opencv.org/4.0.1/d9/dba/classcv\\_1\\_1StereoBM.html](https://docs.opencv.org/4.0.1/d9/dba/classcv_1_1StereoBM.html).
- [7] OpenCV. Disparity map post-filtering. [https://docs.opencv.org/4.0.1/d3/d14/tutorial\\_ximgproc\\_disparity\\_filtering.html](https://docs.opencv.org/4.0.1/d3/d14/tutorial_ximgproc_disparity_filtering.html).
- [8] OpenCV. Dokumentacja opencv. <https://docs.opencv.org/>.
- [9] OpenCV. Pliki źródłowe i instalatory. <https://docs.opencv.org/releases.html>.
- [10] J. Ratajczak. Cyfrowe przetwarzanie obrazów i sygnałów, wykład 12. <http://rab.ict.pwr.wroc.pl/~jr/cpois/wyklady/wyklad12.pdf>, 2015.
- [11] D. Rzeszotarski, P. Strumiłło, P. Pełczyński, B. Więcek, A. Lorenc. System obracowania stereoskopowego sekwencji scen trójwymiarowych. <http://www.eletele.lodz.pl/programy/sv/publikacje/pdf/12.pdf>.
- [12] SAS. Visualize a weighted regression. <https://blogs.sas.com/content/iml/2016/10/05/weighted-regression.html>, 2016.
- [13] D. M. P. Sha'Kia Boggan. Gpus: An emerging platform for general-purpose computation. arl-sr-154, u.s. army research lab. <https://www.arl.army.mil/arlreports/2007/ARL-SR-154.pdf>, 2007.



# Spis rysunków

1.1	Model kamery perspektywicznej [10] . . . . .	6
1.2	Odwzorowania geometryczne punktów przestrzeni trójwymiarowej ( $X, Y, Z$ ) na płaszczyzny przetworników obrazowych kamer usytuowanych w tzw. układzie kanonicznym (osie kamer równoległe, współrzędne $Z$ ognisk kamer identyczne) [10] . . . . .	7
1.3	Odwzorowania geometryczne punktów przestrzeni trójwymiarowej ( $X, Y, Z$ ) na płaszczyzny przetworników obrazowych kamer usytuowanych w tzw. układzie niekanonicznym (osie kamer nierównoległe, współrzędne $Z$ ognisk kamer różne) [10] . . . . .	7
1.4	Odpowiedniość i dysparycja [10] . . . . .	8
1.5	Korekcja zniekształceń [10] . . . . .	9
1.6	Przetwarzanie obrazów z kamer [10] . . . . .	10
1.7	Układ szachownic do kalibracji kamery z użyciem DLT (widok perspektywiczny) [5] . . . . .	11
1.8	Widok szachownicy z różnej perspektywy w kalibracji wielopłaszczyznowej [5] . . . . .	12
1.9	Kalibracja wielopłaszczyznowa; po lewej układ z kamerą w centrum i szachownicami w różnych pozycjach, po prawej układ z szachownicą w centrum i kamerami w różnych pozycjach [5] . . . . .	13
1.10	Problem dopasowania obrazów z obu kamer [10] . . . . .	13
1.11	Mapa dysparycji surowa, po odfiltrowaniu, zwizualizowana w 3D [10] . . . . .	14
1.12	Dysparycja, a głębia [10] . . . . .	15
1.13	Przypisanie punktowi trzeciej zmiennej i umieszczenie go w przestrzeni trójwymiarowej [10] . . . . .	15
1.14	Schemat działania wielowątkowości . . . . .	16
2.1	Architektura systemu . . . . .	19
2.2	Zdjęcie przed przekształceniem . . . . .	21
2.3	Zdjęcie po przekształceniu . . . . .	21
2.4	Obrazy testowe (lewy na górze, prawy na dole) [7] . . . . .	22
2.5	Wzorcowa mapa głębi [7] . . . . .	23
2.6	Schemat działania programu w wersji jednowątkowej . . . . .	23
2.7	Schemat działania programu w wersji wielowątkowej . . . . .	24
2.8	Mapa dysparycji w trybie z czterema wątkami roboczymi . . . . .	25
2.9	Wygenerowana mapa głębi dla danych testowych bez użycia wielowątkowości . . . . .	25
2.10	Wygenerowana mapa głębi dla danych testowych z użyciem wielowątkowości, z podziałem na 1 zadanie, po zredukowaniu czarnych obszarów . . . . .	26
2.11	Wygenerowana mapa głębi dla danych testowych z użyciem wielowątkowości, z podziałem na 2 zadania, po zredukowaniu czarnych obszarów . . . . .	26

2.12 Wygenerowana mapa głębi dla danych testowych z użyciem wielowątkowości, z podziałem na 4 zadania, po zredukowaniu czarnych obszarów . . . . .	26
2.13 Wygenerowana mapa głębi dla danych testowych z użyciem wielowątkowości, z podziałem na 6 zadań, po zredukowaniu czarnych obszarów . . . . .	27
2.14 Wygenerowana mapa głębi dla danych testowych z użyciem wielowątkowości, z podziałem na 8 zadań, po zredukowaniu czarnych obszarów . . . . .	27
2.15 Wygenerowana mapa głębi dla danych testowych z użyciem wielowątkowości, z podziałem na 16 zadań, po zredukowaniu czarnych obszarów . . . . .	27
3.1 Skonstruowany układ kamer . . . . .	29
3.2 Zrzut ekranu podczas trwania eksperymentu . . . . .	30
3.3 Wykres czasu liczenia mapy dysparcji w zależności od liczby wątków liczących w wersji wielowątkowej i w wersji jednowątkowej . . . . .	30
3.4 Wykres błędu wygenerowanej wielowątkowo mapy dysparcji w zależności od liczby wątków liczących w wersji wielowątkowej względem wzorca z wersji jednowątkowej przy zastosowaniu post filtrowania generowanych fragmentów . . . . .	32
3.5 Wykres normy różnicy wygenerowanej wielowątkowo mapy dysparcji i wzorcowej – wygenerowanej w wersji jednowątkowej – w zależności od liczby wątków liczących w wersji wielowątkowej przy zastosowaniu post filtrowania generowanych fragmentów . . . . .	33

# Spis tabel

3.1 Średnie czasy operacji (zaokrąglone do $1\mu s$ ) w wersji jednowątkowej (brak osobnych wątków liczących) i wielowątkowej w zależności od liczby równolegle wykonywanych zadań . . . . .	31
--	----