Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Uczenie Maszynowe w Fotonice Obrazowej

Projekt

„Pozycjonowanie kamery”

Domian Konrad Kacper

304148

Gołębiowska Aneta

300309

Goraj Dawid

300311

Mierzejewski Daniel

304187

Strzałecki Patryk

305310

Maciej Mikulski

304188

Adam Głąbicki

289823

Jakub Frydrych

304493

Warszawa 2023

1 SPIS TREŚCI

2 ZNACZENIE PROBLEMU W UJĘCIU RYNKOWYM

Pozycjonowanie kamery odgrywa kluczową rolę w dziedzinie wizji maszynowej. Jest to proces polegający na ustalaniu orientacji i lokalizacji kamery w przestrzeni trójwymiarowej. Problem pozycjonowania kamery możemy spotkać w szeregu aspektów. Są nimi:

1. Rozwój technologii rozszerzonej rzeczywistości (ang. AR) oraz wirtualnej rzeczywistości (ang. VR). W celu zapewnienia realistycznego doświadczenia wirtualnego, konieczne jest precyzyjne pozycjonowanie kamery w przestrzeni, aby obrazy wirtualne były spójne z otoczeniem.
2. Przemysł rozrywkowy i produkcja filmowa (potocznie wirtualne studia). Wraz z rozwojem technologii rozszerzonej rzeczywistości (ang. AR), powstała koncepcja wirtualnego studia. Wirtualne studio pozwala znacząco zwiększyć atrakcyjność programu telewizyjnego przy znacznym obniżeniem kosztów produkcji. Rozwój takich projektów jak Unreal Engine [1], realizowanych przez firmę EPIC GAMES pozwala, by taka technologia trafiała i do małych twórców internetowych.

Postanowiliśmy bardziej zająć się problemem przemysłu rozrywkowego z powodu mniejszej liczby dostępnych produktów. Nasze poszukiwanie konkurencji skończyły się na znalezieniu dużej liczby bardzo drogich systemów wizyjnych, skierowanych głównie do dużych studiów telewizyjnych. Nasze rozwiązanie miało być dobrą konkurencją ze względu na niską cenę produktu, a zarazem na dostateczną dokładność pozycjonowania. Tak, aby było idealnie skrojone do twórców internetowych.

[1] https://www.unrealengine.com/en-US/virtual-production

3 OPIS PROBLEMU

Problem polega na określeniu dokładnej pozycji kamery w przestrzeni trójwymiarowej. W momencie przemieszczania kamery po studiu, jej pozycja musi być idealnie przeniesiona do wirtualnej przestrzeni tworzonej za pomocą oprogramowania typu Unreal Engine (Rysunek 1).

Rysunek 1 Zrzut ekranu z programu Unreal Engine przedstawiający pozycje kamery względem wirtualnego studia

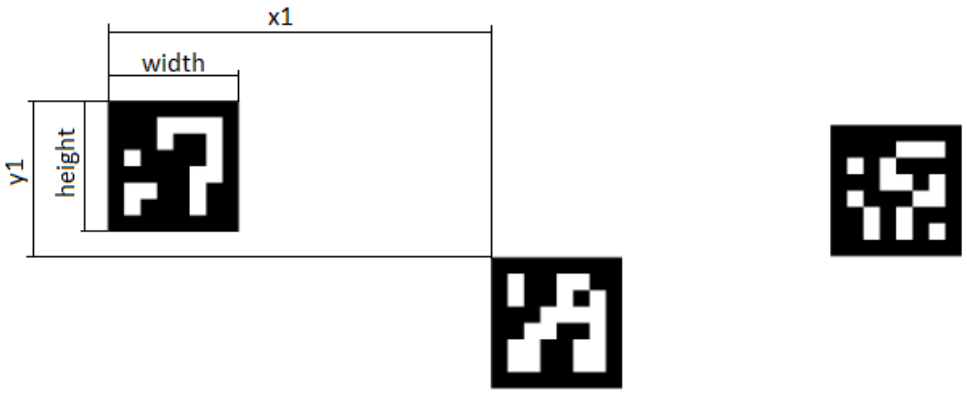
W momencie naniesienia maski na tło oraz dołączenia obrazu z rzeczywistej kamery, powstaje nam cały obraz, który do złudzenia wygląda tak, jakby osoby występujące w studiu były w innym miejscu (Rysunek 2).

Rysunek 2 Widok z kamery oraz widok po przetworzeniu obrazu z kamery po uwzględnieniu dokładnej je pozycji

Niestety, w momencie błędnego spozycjonowania kamery, może zdarzyć się tak, że obiekty wirtualne najdą na rzeczywiste, przez co iluzja ulegnie zaburzeniu. Mogą pojawić się „latające” przedmioty a także mogą one wchodzić w interakcje z aktorami.

4 PRZEGLĄD ISTNIEJĄCYCH ROZWIĄZAŃ

Stosowane obecnie układy estymacji pozycji kamery w przestrzeni 3D podzielić można na systemy pojedynczej kamery oraz systemy stereowizyjne, wykorzystujące układ wielu kamer. W przypadku układów stereowizyjnych stosowanych w mniejszych aplikacjach, powszechną praktyką jest wykorzystanie układu dwóch kamer cyfrowych o znanym wzajemnym położeniu oraz wyznaczonych parametrach wewnętrznych kamer i współczynnikach zniekształceń ich toru optycznego. Zasada działania takiego systemu opiera się na jednoczesnym rejestrowaniu obrazu z obu kamer oraz identyfikacji rzutów tych samych punktów na obrazach pozyskanych przez lewą oraz prawą kamerę. Korzystając z odpowiednich przekształceń geometrycznych, wyznaczyć można rzeczywiste współrzędne kamer. W przypadku układów pojedynczej kamery, niezbędne jest określenie cech środowiska, które kamera rejestruje. Problem ten znany jest jako problem perspektywy n punktów (*ang. Perspective-n-Point - PnP*), którego rozwiązanie polega na określeniu cech 3D na podstawie odpowiadającego obrazu 2D. Najczęściej stosowanym podejściem do rozwiązania tego problemu jest wykorzystanie dobrze zdefiniowanych znaczników, zwanych również markerami rozmieszczonymi w rejestrowanej przestrzeni. Kluczowym aspektem tego podejścia jest wykorzystanie poprawnie skalibrowanej kamery oraz dokładne zdefiniowanie wymiarów, oraz relacji przestrzennych między markerami.



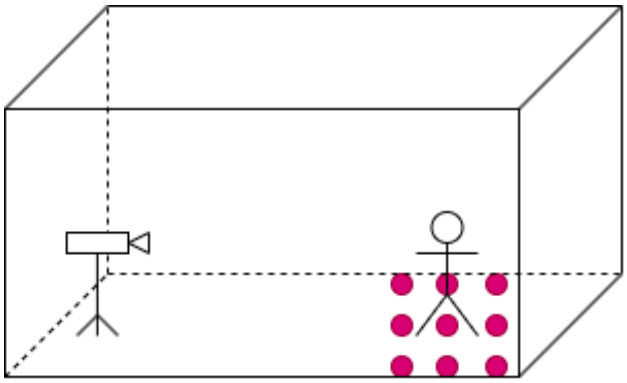
Rysunek 3. Właściwości geometryczne znaczników.

Znając parametry wewnętrzne wykorzystywanej kamery oraz geometryczne właściwości markerów, można wyznaczyć przestrzenne położenie kamery, stosując odpowiednie transformacje geometryczne. Często stosowaną praktyką jest wyposażenie układu wizyjnego w dodatkowe urządzenia pomiarowe, takie jak lasery czy akcelerometry zwiększające dokładność estymacji pozycji kamery. Wraz z rozwojem metod uczenia głębokiego, popularne stało się wykorzystanie głębokich sieci neuronowych, wspierających układy wizyjne w wyszukiwaniu odpowiadających sobie punktów w przestrzeni rejestrowanych przez układy stereowizyjne oraz detekcji znaczników wykorzystywanych w układach pojedynczej kamery.

5 PROPOZYCJA ROZWIĄZANIA PROBLEMU

5.1 WSTĘPNY OPIS ROZWIĄZANIA

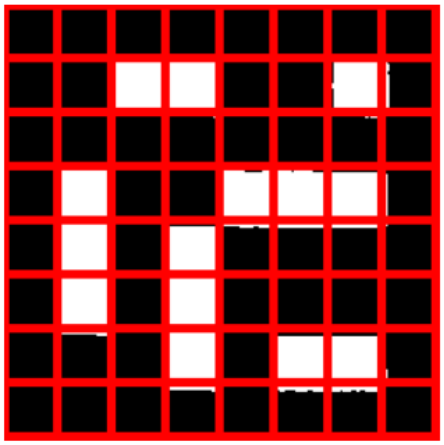
Proponujemy wykorzystać do rozwiązania problemu specjalne markery, które byłyby wykrywane poprzez nasz system i pozycjonowane względem kamery. Umieszczając znaczniki na podłodze lub suficie, znalibyśmy dokładne ich pozycje. Pozwoli to określić następnie pozycję oraz rotację kamery.



Rysunek 4. Przykładowa scena z rozmieszczonymi znacznikami.

5.2 WYKRYCIE ZNACZNIKÓW

W pierwszej kolejności sprawdzona zostanie możliwość wykorzystania dostępnego w bibliotece OpenCV modułu detekcji markerów ArUco. Moduł ten zapewnia generator znaczników o predefiniowanych rozmiarach oraz dwustopniowy detektor, którego zasada działania polega w pierwszym kroku na wyodrębnieniu z analizowanego obrazu kandydatów potencjalnie stanowiących markery poprzez adaptacyjne progowanie w celu odseparowania obiektów, których kontur aproksymuje kwadrat, a następnie analizie wyodrębnionych kandydatów poprzez ekstrakcję bitową.



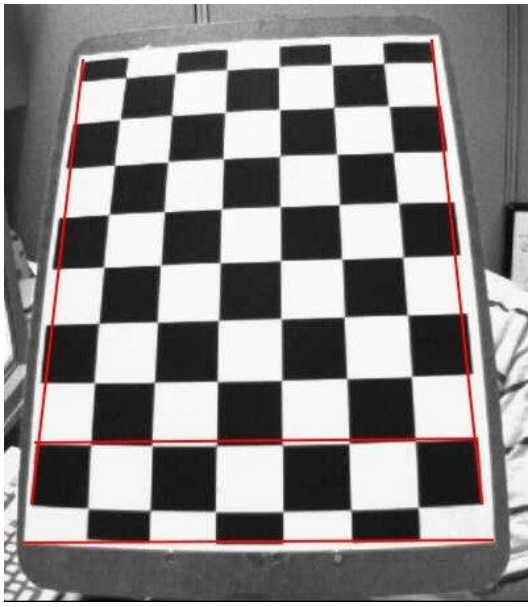
Rysunek 5. Ekstrakcja bitowa znacznika.

Dla poprawnie rozpoznanych markerów, detektor zwróci pozycję 4 narożników oraz ID danego markera. Zwrócić należy uwagę na orientację znacznika, ponieważ detektor zwraca pozycję narożników w kolejności zdefiniowanej dla domyślnej orientacji narożnika.

W przypadku, gdy wykorzystanie modułu detekcji markerów ArUco nie będzie możliwe z przyczyn technicznych, markery te zostaną zastąpione własnymi markerami w postaci jaskrawych obiektów, których kształt zostanie ustalony w trakcie implementacji modułu przetwarzania obrazu. Do detekcji tych markerów wykorzystane zostaną typowe metody wizji maszynowej, a w przypadku niewystarczającej dokładności estymacji pozycji znaczników, wykorzystany zostanie detektor w postaci modelu sieci konwolucyjnej.

5.3 Wyznaczenia pozycji kamery na podstawie znaczników oraz kalibracja kamery

Kamera zniekształca rejestrowany obraz. Zniekształcenia wprowadzane przez kamerę możemy podzielić na promieniowe oraz styczne. Zniekształcenie promieniowe powoduje, że linie proste wydają się zakrzywione. Zniekształcenie staje się tym większe, im dalej od środka obrazu znajduje się obserwowany fragment. Takie zniekształcenie można zaobserwować na obrazku poniżej. Natomiast zniekształcenie styczne występuje, ponieważ obiektyw rejestrujący nie jest ustawiony idealnie równolegle do płaszczyzny obrazowania. Dlatego niektóre obszary obrazu mogą wyglądać na znajdujące się bliżej niż są w rzeczywistości.



Rysunek 6. Zniekształcenie obrazu.

Zniekształcenia promieniowe opisuje się poniższymi równaniami:

$$x_{distorted} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_{distorted} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Zniekształcenia styczne opisuje się poniższymi równaniami:

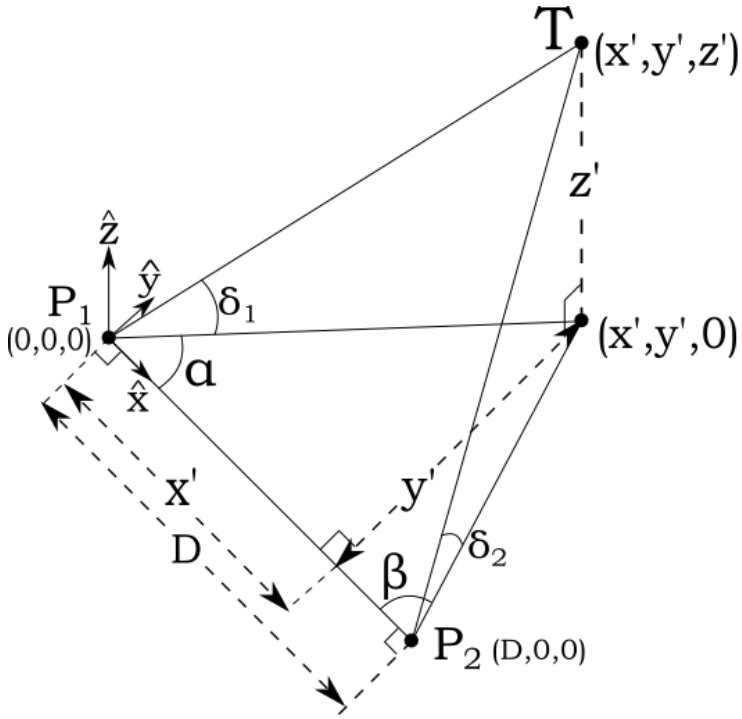
$$x_{distorted} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$
$$y_{distorted} = y + [p_1(r^2 + 2y^2) + 2p_2 xy]$$

W celu kompensacji wyżej wymienionych zniekształceń, stosuje się kalibrację kamery. Do kalibracji kamery konieczne będzie wyznaczenie współczynników zniekształcenia: p_1 , p_2 , k_1 , k_2 , k_3 oraz

tzw. macierzy kamery. Macierz kamery wyznacza się na podstawie wewnętrznych cech kamery, takich jak ogniskowe obiektywu czy centra optyczne.

$$camera\ matrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Pozycje kamery wyznacza się za pomocą triangulacji. Do triangulacji konieczna jest znajomość położenia (*x, y, z*) co najmniej dwóch znaczników:



Rysunek 7 Wyznaczenie pozycji kamery T w trójwymiarowej przestrzeni na podstawie znaczników P_1 i P_2

Położenie określa się na podstawie poniższych wzorów:

$$y' = D\frac{sin(\alpha)sin(\beta)}{sin(\alpha+\beta)}$$

$$x' = \frac{y'}{tan(\alpha)}$$

$$L_1 = \sqrt{x'^2 + y'^2}$$

$$L_2 = \sqrt{(D-x')^2 + y'^2}.$$

$$z_i' = L_i tan(\delta_i),$$

$$z' = \frac{z'_1 + z'_2}{2}$$

5.4 Wyznaczenie pozycji kamery za pomocą układu IMU

Drugim źródłem informacji o pozycji kamery będzie IMU (Inertial Measurement Unit), która składa się z trzyosiowego akcelerometru, trzyosiowego żyroskopu oraz magnetometru. Taka jednostka zostanie umieszczona na kamerze, a uzyskane z niej dane zostaną przetworzone na jej pozycję i orientację w przestrzeni.

Najpierw czujnik musi zostać poddany kalibracji. Do wyznaczenia przesunięcia należy dwukrotnie scałkować uzyskane z czujnika przyspieszenie.

$$\begin{bmatrix} \hat{x}_p(t) \\ \hat{y}_p(t) \\ \hat{z}_p(t) \end{bmatrix} = \begin{bmatrix} x_p(t_0) \\ y_p(t_0) \\ z_p(t_0) \end{bmatrix} + \int_{t_0}^{t} \begin{bmatrix} \hat{v}_{xp}(\tau) \\ \hat{v}_{yp}(\tau) \\ \hat{v}_{zp}(\tau) \end{bmatrix} d\tau$$

$$\begin{bmatrix} \hat{x}_{xp}(t) \\ \hat{y}_{yp}(t) \\ \hat{z}_{zp}(t) \end{bmatrix} = \begin{bmatrix} x_{xp}(t_0) \\ y_{yp}(t_0) \\ z_{zp}(t_0) \end{bmatrix} + \int_{t_0}^{t} \begin{bmatrix} a^N_{xm}(\tau) \\ a^N_{ym}(\tau) \\ a^N_{zm}(\tau) \end{bmatrix} d\tau$$

Powyższe wzory pokazują metodę wyznaczania zmiany położenia, gdzie $x_p(t_0)$, $y_p(t_{0,})$ $z_p(t_0)$ to położenie w chwili początkowej a $x_{xp}(t_0)$, $y_{yp}(t_{0,})$ $z_{zp}(t_0)$ to prędkość w chwili początkowej.

Pochylenie, przechylenie i obrót kamery są wyznaczane korzystając z danych żyroskopu, akcelerometru i magnetometru, w celu uzyskania mniej zaszumionych wartości. By połączyć ze sobą wyniki z tych czujników stosuje się filtr Kalmana.

5.5 UŻYCIE FILTRU KALMANA DO POPRAWIENIA OKREŚLENIA POZYCJI KAMERY ORAZ JEJ ROTACJI

W celu poprawy dokładności określenia pozycji kamery, wyniki z kamery będą połączone z wynikiem z IMU. Do połączenia wyników użyty będzie filtr Kalmana.

Filtr Kalmana [2] jest optymalnym estymatorem stanu układu dynamicznego. Należy tutaj jednak przywołać kilka założeń, które są niezbędne (choć czasem przemilczane), aby filtr działał, jak należy. Są to:

- dany jest układ dynamiczny w postaci układu równań różnicowych liniowych (dla tradycyjnego filtru) lub równań różnicowych nieliniowych (rozszerzony filtr Kalmana),
- wejścia i wyjścia układu są dostępne pomiarowo,
- zakłócenia oddziałujące na stan układu oraz szumy pomiarowe mają rozkład normalny o wartości oczekiwanej równej 0 i znanych wariancjach.
- wektory $\vec{w}, \vec{v}\ oraz\ \vec{x}$ są wzajemnie niezależne

Równania modelu:

$$\begin{cases} x_{k+1} = Ax_k + Bu_k + w_k \\ y_k = Hx_k + v_k, \end{cases}$$

x-stan układu, u- wejście, y – wyjście, k -dyskretny punkt w czasie, A – macierz przejścia, B – macierz wejścia, H – macierz wyjścia,

Równania FK można podzielić na dwie części:

część predykcyjną - dokonuje predykcji stanu w chwili k na podstawie estymaty stanu i sterowania z chwili poprzedniej:

$$\hat{x}_{k|k-1} = A\hat{x}_{k-1|k-1} + Bu_{k-1}$$
$$P_{k|k-1} = AP_{k-1|k-1}A^T + Q,$$

gdzie: $\hat{x}_{k|k-1}$ - ocena stanu a priori (przed pomiarem), $\hat{x}_{k|k}$ - wstymata stanu a posteriori (po pomiarze), $P_{k|k-1}$ - macierz kowariancji błędu predykcji, $P_{k|k}$ - macierz kowariancji błędu filtracji, Q - macierz kowariancji szumu procesowego.

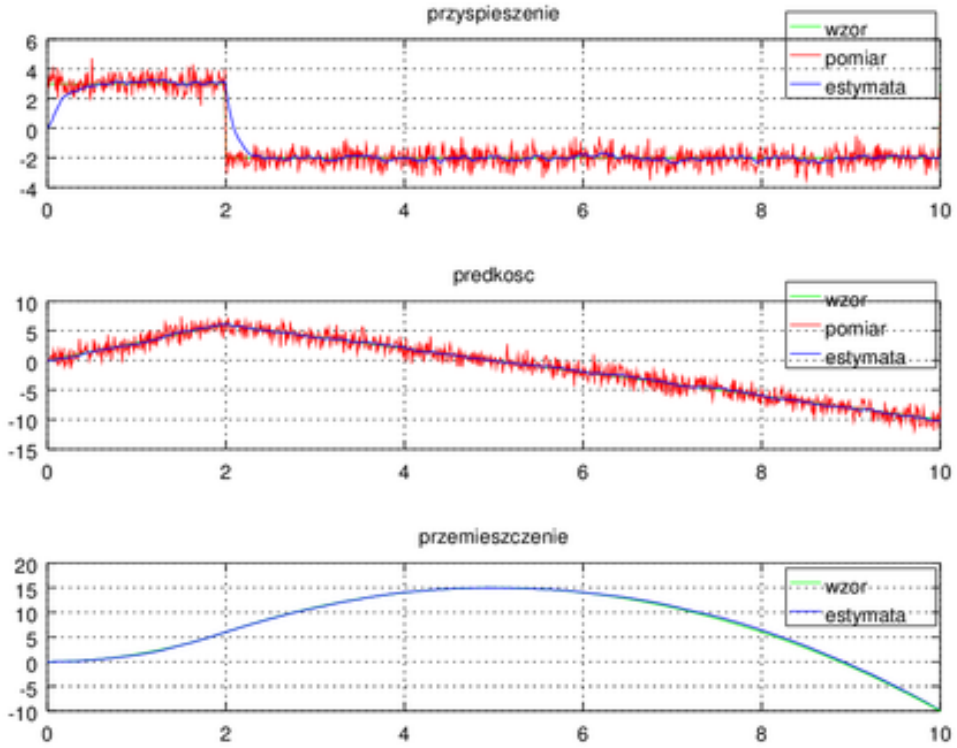
część filtracyjną - dokonuje aktualizacji estymaty stanu i macierzy błędu kowariancji stanu na podstawie pomiaru wejść w chwili obecnej:

$$K = P_{k|k-1}H^T\left(HP_{k|k-1}H^T + R\right)^{-1}$$
$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K\left(y_k - H\hat{x}_{k|k-1}\right)$$
$$P_{k|k} = (I - KH)\,P_{k|k-1},$$

gdzie: K - wzmocnienie filtru Kalmana, R - macierz kowariancji szumu pomiarowego. Wzmocnienie filtru, zależne od R jest informacją o tym, w jaki sposób informacja pochodząca z pomiarów wpływać ma na korekcję estymatu stanu. Dla dużej wartości K (małego zakłócenia stanu w R) pomiary będą miały istotny wpływ na wartość $\hat{x}_{k|k}$. Macierz I to macierz jednostkowa o odpowiednim rozmiarze.

[2] http://jtjt.pl/filtr-kalmana

Sztandarowym przykładem wykorzystania FK jest estymacja stanu ruchu obiektu. Dotyczy to zarówno ruchu liniowego jak i obrotowego, w jednym lub wielu wymiarach. Prócz odszumiania danych pochodzących z GPSu, żyroskopu czy akcelerometru, FK pozwala na fuzję danych z wielu czujników równocześnie w celu poprawy ostatecznej oceny stanu. Na rysunku poniżej przedstawiono zastosowanie filtru Kalmana w celu fuzji i odszumienia danych pochodzących z dwóch czujników.



Rysunek 8 Przykładowe zastosowanie filtru Kalmana – estymacja ruchu w jednym wymiarze.

Gotowa implementacja filtru Kalmana jest dostępna w bibliotece Pythona - ***filterpy*** [3]. Możliwa też jest samodzielna implementacja filtru Kalmana z wykorzystaniem biblioteki/środowiska umożliwiającego działania na macierzach np. Matlab czy Octave, czy też biblioteki Pythona - *numpy.*

Najwygodniejsze będzie wykorzystanie gotowej implementacji z biblioteki ***filterpy.***

[3] https://github.com/rlabbe/filterpy

6 BADANIE DOSTĘPNOŚCI DANYCH PUBLICZNYCH

6.1 WYKRYCIE ZNACZNIKÓW

https://docs.opencv.org/3.4/d5/d07/tutorial_charuco_diamond_detection.html

https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html

https://medium.com/@calle_4729/using-mathematica-to-detect-aruco-markers-197410223f62

6.2 WYZNACZENIA POZYCJI ZNACZNIKÓW ORAZ KALIBRACJA KAMERY

Kalibracji kamery:
http://marcin.kielczewski.pracownik.put.poznan.pl/POiSW12.pdf

Kalibracji kamery – kod :
https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html

Odległość obiektu od kamery – kod:
https://pyimagesearch.com/2015/01/19/find-distance-camera-objectmarker-using-python-opencv/

Triangulacja:
https://www.diva-portal.org/smash/get/diva2:1576987/FULLTEXT01.pdf

6.3 WYZNACZENIE POZYCJI KAMERY ZA POMOCĄ UKŁADU IMU

Wyznaczanie orientacji (przechylenie pochylenie obrót) artykuł:

https://medium.com/@niru5/fusion-of-accelerometer-magnetometer-data-with-gyroscope-part-2-2887261e7245

https://medium.com/@niru5/hands-on-with-rpi-and-mpu9250-part-3-232378fa6dbc

Wyznaczanie pozycji za pomocą IMU:

https://www.researchgate.net/publication/321260623_An_integrated_IMU_and_UWB_sensor_based_indoor_positioning_system

https://www.mdpi.com/1424-8220/20/16/4410

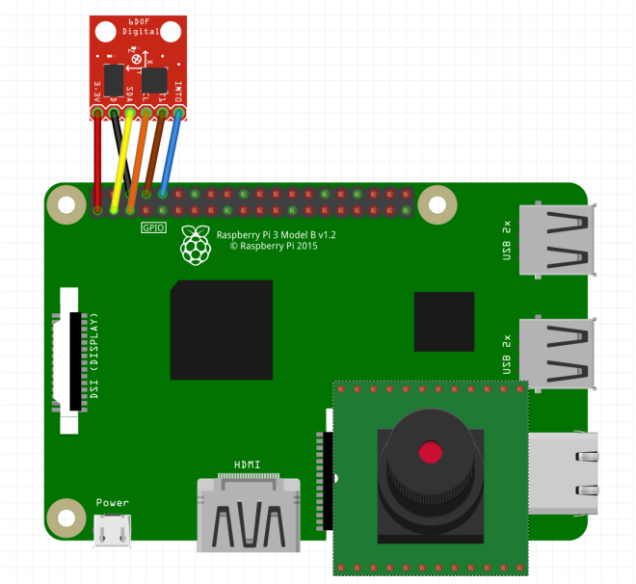
https://github.com/LibofRelax/IMU-Position-Tracking

6.4 UŻYCIE FILTRU KALMANA DO POPRAWIENIA OKREŚLENIA POZYCJI KAMERY ORAZ JEJ ROTACJI

Artykuł z rozwiązaniem:
https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=1371bd3e6834136d08e917c6e51ca9ff61bbd323
Kody z przykładami filtru Kalmana:
https://forbot.pl/blog/filtr-kalmana-w-praktyce-3-przyklady-z-kodami-id7342

Przykładowy kod wykorzystujący implementację filtru Kalmana z biblioteki *filterpy* :
https://stackoverflow.com/questions/75926943/how-to-create-kalman-filter-for-3d-object-tracking

7 UKŁAD AKWIZYCJI DANYCH/OBRAZÓW



Rysunek 9 Układ akwizycji danych

Układ akwizycji danych do pozycjonowania kamery skonstruowany został z trzech głównych składników:

- Raspberry Pi 4
- Kamery Creative Live! Cam Sync HD
- układu IMU (Inertial Measurement Unit) BNO08X

Rysunek 10 Dataflow rozwiązania

Proces ten rozpoczyna się od detekcji znaczników umieszczonych w znanych lokalizacjach. Kamera rejestruje obraz, a algorytm detekcji identyfikuje znaczniki na podstawie tych obrazów. Następnie, korzystając z macierzy kamery, system oblicza pozycję każdego znanego znacznika względem kamery. Triangulacja pozwala precyzyjnie określić trójwymiarową pozycję kamery. W celu uzyskania dokładniejszych danych dotyczących położenia kamery, system wykorzystuje informacje z układu IMU, takie jak przyspieszenia i kąty obrotu. Połączenie danych z kamery i IMU umożliwia bardziej kompleksową analizę położenia kamery w przestrzeni. Dane z kamery i IMU są następnie przekazywane przez filtr Kalmana, który eliminuje błędy pomiarowe i dostarcza bardziej precyzyjne wyniki. Ostateczne współrzędne kartezjańskie oraz informacje kątowe są przesyłane do komputera PC poprzez interfejs UART. Komputer PC wykorzystuje te dane do monitorowania i analizy położenia kamery w czasie rzeczywistym. Poszczególne kroki algorytmu zostały szczegółowo opisane w rozdziale 5.

8 PLAN TESTÓW ORAZ WERYFIKACJA ROZWIĄZANIA

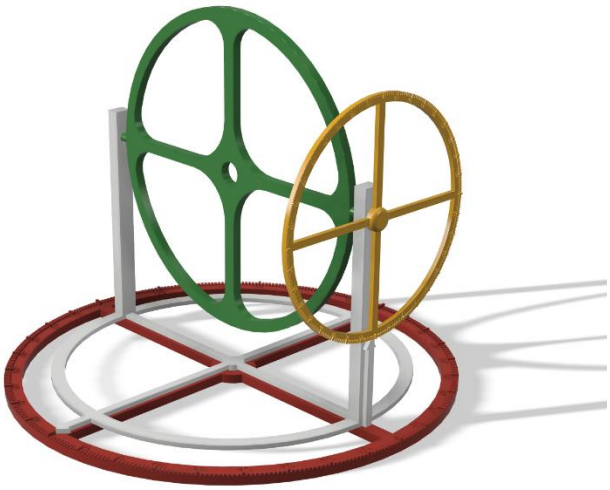
8.1 WERYFIKACJA WSKAZAŃ PRZESUNIĘCIA W UKŁADZIE KARTEZJAŃSKIM

W naszym podejściu do weryfikacji pozycjonowania kamery w układzie kartezjańskim, będziemy wykorzystywać planszę szachową jako narzędzie referencyjne. Stworzymy ją, przy pomocy zwyczajnej metrówki oraz taśmy malarskiej. Nasza plansza będzie miała znaną geometrię i precyzyjne wymiary kwadratów. Kamera zostanie umieszczona na platformie przeznaczonej do weryfikacji pozycji kątowej umożliwiającej równomierne przesuwanie jej wzdłuż szachownicy. Podczas przesuwania kamery będziemy aktywować algorytm. W celu analizy wyników będziemy zbierać informacje o każdej pozycji kamery. Następnie, opierając się na rzeczywistych wymiarach szachownicy i pozycji, obliczymy oczekiwane współrzędne punktów na planszy. W kolejnym kroku porównamy oczekiwane współrzędne z danymi zarejestrowanymi przez kamerę, obliczając błąd pozycjonowania dla każdego punktu charakterystycznego. Przeprowadzimy analizę statystyczną błędów, co pozwoli nam uzyskać ogólny pogląd na dokładność kamery w różnych obszarach przestrzeni. W przypadku wystąpienia znaczących błędów, będziemy stosować poprawki, takie jak dostosowanie parametrów kalibracyjnych kamery lub implementacja technik korekcyjnych, w tym korekcji nieliniowej. Wyniki testów będą szczegółowo udokumentowane, obejmując błędy pozycjonowania dla różnych pozycji kamery na planszy szachownicy. Przeprowadzimy testy w różnych warunkach oświetleniowych, aby ocenić wpływ na dokładność pozycjonowania kamery.

8.2 WERYFIKACJA WSKAZAŃ KĄTOWYCH

Ważnym aspektem jest zweryfikowanie precyzji pozycjonowania w współrzędnych obrotu. W tym celu opracowano koncepcję układu pomiarowego, umożliwiającego dokładne sprawdzenie położenia kamery pod różnymi kątami. Jednocześnie zapewniono możliwość utrzymania kamery w stałym położeniu podczas akwizycji danych.

Zastosowano program typu CAD (Fusion360), do stworzenia trójwymiarowego modelu składającego się z czterech elementów. Na załączonym rysunku (Rysunek 11) przedstawiono wizualizację układu, gdzie każdy element został oznaczony innym kolorem dla łatwiejszego zrozumienia. Urządzenie umożliwia precyzyjne określenie kąta obrotu z dokładnością do jednego stopnia.



Rysunek 11 Zaprojektowany układ weryfikacji odczytu położenia kątowego urządzenia

U podstawy układu zamontowano tarczę umożliwiającą obrót wokół osi Z. Dodatkowo, poprzez manipulację pokrętłem umieszczonym z boku urządzenia, można dostosować obrót kamery zgodnie z osią X. W celu przeprowadzenia testów weryfikacyjnych zgodnie z osią Y, konieczne jest zamontowanie urządzenia obróconego o 90 stopni. W centralnym położeniu urządzenia.

Cały układ został wykonany w technologii druku 3D FDM, wykorzystując do tego celu filament PLA. Aby zweryfikować poprawne działanie układu korzystając z zaprojektowanego urządzenia, przeprowadza się testy dla każdej osi obrotu, ustawiając i odczytując kąty w zakresie od 0 do 359 stopni. Układ zostanie uznany za prawidłowo działający, gdy odczyty mieszczą się w zakresie błędu pomiarowego urządzenia, czyli jednego stopnia. Wartości te należy szczegółowo udokumentować w tabeli dla każdej osi obrotu, porównując kąty ustawione z kątami odczytanymi.

9 REALIZACJA ROZWIĄZANIA

Na tym etapie rozpoczęliśmy implementacje rozwiązania. Nasz zespół podzieliliśmy na 3 grupy, jedna grupa zajmowała się implementacją wskazań IMU, druga zajęła się wyznaczeniem pozycji z kamery a trzecia zajęła się weryfikacją rozwiązania. Poniżej w podrozdziałach zostały omówione kroki oraz wyzwalania z jakimi mierzyliśmy się podczas realizacji tych zadań

9.1 UKŁAD IMU

9.1.1 Komunikacja IMU-Raspberry PI

Wybrany układu IMU pozwalał na komunikacje z zewnętrznym mikrokontrolerem za pomocą protokołu SPI bądź UART. Próby z użyciem odczytywania za pomocą protokołu UART z wykorzystaniem standardowego zbioru komend UART-RVC, wykazały znaczne wybrakowanie komend konfiguracyjnych IMU, przez co zdecydowano się wykorzystać komunikacje poprzez magistrale SPI. W tym celu poprawnie skonfigurowano device-tree systemu raspberry-os do pracy z magistralną SPI, oraz napisano aplikacje do pobierania danych oraz konfiguracji układu IMU. Kody aplikacji dostępne są na naszym repozytorium na platformie github.

9.1.2 Filtracja sygnału z układu IMU

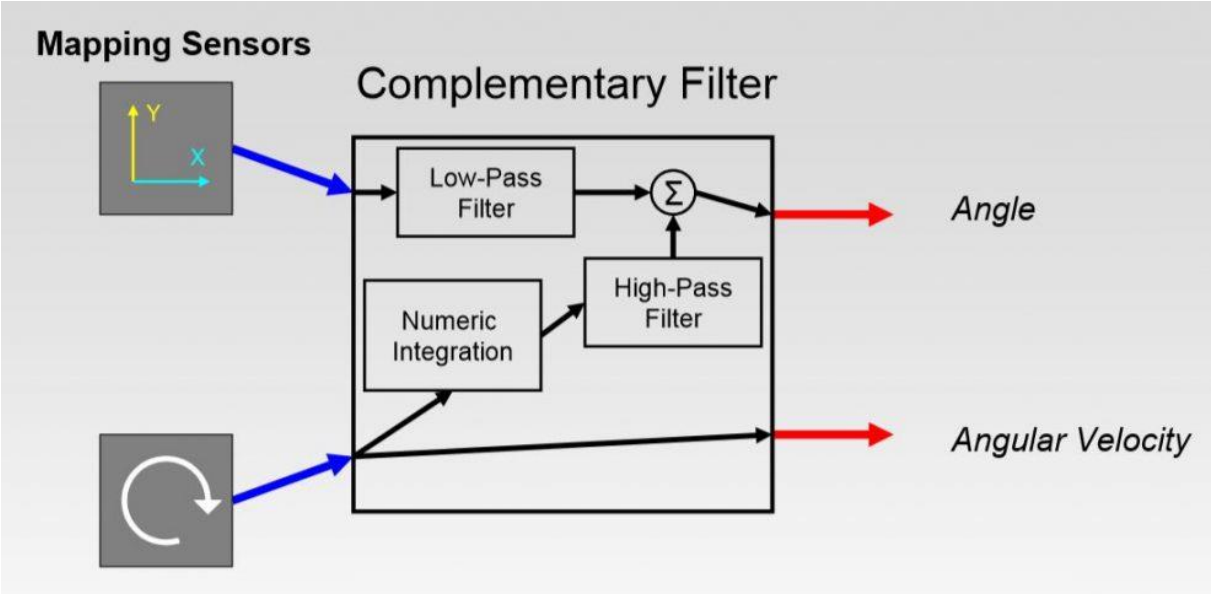
Podczas pierwszych prób wykryto konieczność filtracji sygnału. Informacje jakie udawało nam się odczytać z układu były podatne na rozmaite szumy i przejawiały się jako tzw. „szpilki”. Przez to postanowiono zastosować filtracje sygnału. Do dalszej implementacji wybrano dwa rodzaje filtrów.

9.1.2.1 Filtr Kalmana

W toku implementacji kodu przetwarzającego dane pochodzące z IMU, okazało się, że nie posiadamy doświadczenia wymaganego do poprawnej implementacji filtru Kalmana. Problematyczne okazało się zagadnienie strojenia filtru. Z tego powodu podjęto decyzję o zaniechaniu prac nad tym rozwiązaniem oraz wykorzystaniu filtra komplementarnego.

9.1.2.2 Filtr komplementarny

Prostota filtra komplementarnego spowodowała, że został on ostatecznie wybrany do filtracji danych pochodzących z IMU. Filtr taki pobiera z żyroskopu prędkości obrotowe, całkuje je, a następnie filtruje górnoprzepustowo. W ten sposób niwelowany jest dryf żyroskopu. Dane z akcelerometru filtrowane są dolnoprzepustowo, tak aby pozbyć się szumu pochodzącego z sensora. Następnie wyznaczane są z nich kąty wychylenia systemu. Dane z żyroskopu oraz akcelerometru są łączone, w wyniku czego otrzymywane jest kwaternion orientacji. Schemat działania takiego filtru przedstawiony jest na poniższej grafice:



Rysunek 11 Schemat zastosowanego filtru

Rozwiązanie wykorzystane w naszym projekcie bazuje na kodzie filtra komplementarnego z następującego repozytorium: https://gist.github.com/phausamann/721fa3df0f8ef6f4f6f24b86fdde53c0. Kod ten został zmodyfikowany, tak aby oprócz kwaternionu orientacji zwracał przefiltrowane wartości przyśpieszeń w trzech osiach. Filtr ten operuje na oknach czasowych zawierających N ostatnich próbek danych zebranych przez czujniki. Przechowywaniem wektorów danych pomiarowych, dodawaniem nowych danych oraz usuwaniem starych zajmuje się klasa nadrzędna IMU_Filter, która wywołuje w swoim działaniu klasę filtra komplementarnego.

9.1.3 Proces kalibracji układu IMU

W celu osiągania prawidłowych wskazań przez IMU, konieczna jest jego kalibracja. W celu skalibrowania żyroskopu, dokumentacja zaleca położenie układu i pozostawienie go w bezruchu. W przypadku kalibracji akcelerometru, urządzenie należy poruszyć w czterech-sześciu różnych kierunkach i utrzymać w miejscu po każdym ruchu. W przypadku kalibracji planarnej, urządzenie należy obrócić wokół własnej osi Z o co najmniej 180 stopni. Po bezskutecznym wykonywaniu niniejszych czynności okazało się, że biblioteka będąca sterownikiem IMU implementuje tylko proces kalibracji magnetometru.

9.1.4 Błędy oraz problemy przy implementacji

Podczas zapoznawania się z platformą zauważono, że przy pozostawieniu IMU w bezruchu oraz zorientowaniu go wzdłuż osi X, Y i Z, jedna ze składowych przyspieszenia przyjmuje wartość podobną do wartości przyspieszenia ziemskiego. Obracając układ o 90 stopni, składowa, na której obecne było przyspieszenie, ulegała zmianie. Wyciągnięto zatem wniosek, że IMU wykrywa przyspieszenie ziemskie, co uniemożliwiało obliczanie pozycji. W tym celu zdefiniowano następujące funkcje:

```
def quaternion_to_rotation_matrix(q):
      w, x, y, z = q
      rotation_matrix = np.array([
            [1 - 2*y**2 - 2*z**2, 2*x*y - 2*w*z, 2*x*z + 2*w*y],
            [2*x*y + 2*w*z, 1 - 2*x**2 - 2*z**2, 2*y*z - 2*w*x],
            [2*x*z - 2*w*y, 2*y*z + 2*w*x, 1 - 2*x**2 - 2*y**2]
```

```
        ])
        return rotation_matrix

def acceleration_from_imu(quaternion, gravity_vector):
        rotation_matrix = quaternion_to_rotation_matrix(quaternion)
        rotated_gravity = np.dot(rotation_matrix, gravity_vector)
        earth_acceleration = -rotated_gravity
        earth_acceleration = [earth_acceleration[2], -earth_acceleration[1],
earth_acceleration[0]]
return earth_acceleration
```

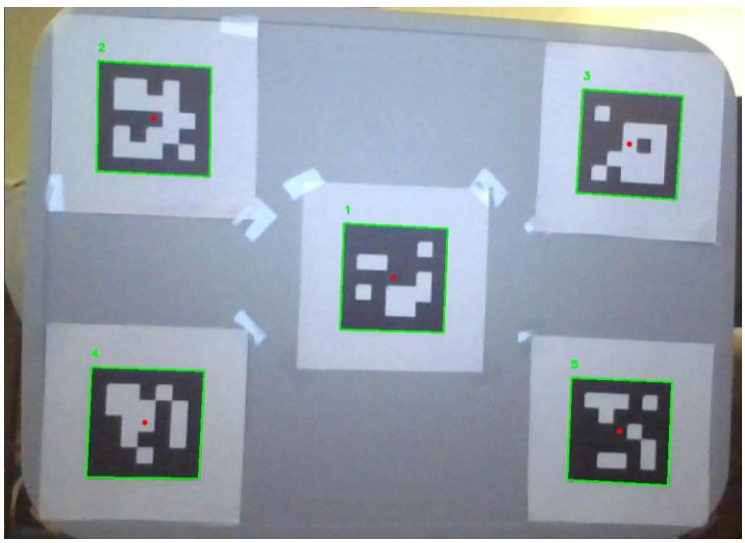
Funkcja `quaternion_to_rotation_matrix()` wyznacza macierz rotacji, na podstawie kwaternionu ustalonego przez układ IMU. Jest ona wywoływana przez funkcję `acceleration_from_imu()`, która zwraca wektor przyspieszenia z obciętą składową ziemską. Składowa ziemska została wycięta pomyślnie.

Kolejnym krokiem było wyznaczanie pozycji IMU na podstawie przyspieszenia oraz równań ruchu jednostajnie przyspieszonego. Próba wyznaczania pozycji zakończyła się niepowodzeniem. Wraz z poruszaniem układu w trzech wymiarach, pozycja nie ulegała zmianie. Postanowiono, że pozycja będzie wyznaczana za pośrednictwem kamery.

9.2 WYZNACZENIE POZYCJI Z KAMERY

9.2.1 Wykrycie znaczników

W projekcie wykorzystane zostały znaczniki ArUco z predefiniowanego słownika 5X5_100 dostępnego w bibliotece OpenCV – Contrib. Wykorzystanie predefiniowanego słownika markerów umożliwiło wykorzystanie detektora zaimplementowanego w wykorzystanej bibliotece, który dostarcza odpowiednie metody umożliwiające wykrywanie znaczników na przetwarzanym obrazie. Służy do tego metoda cv2.aruco.detectMarkers(), która jako parametry wejściowe wymaga aktualnej klatki przetwarzanego obrazu, wykorzystanego słownika markerów oraz parametrów detektora. Na podstawie podanych parametrów detektor zwraca współrzędne narożników każdego z markerów zawartych na obrazie oraz odpowiadające mu ID. Wykorzystując te informacje można zwizualizować efekt działania detektora w postaci ramek otaczających widoczne markery wraz z odpowiadającym in ID:



Rysunek 12. Rezultat wykrywania znaczników ArUco na przygotowanej tablicy.

9.2.2 Określenie pozycji znaczników

Wykorzystując informacje zwracane przez wybrany detektor można w łatwy sposób określić położenie znaczników. Detektor zwraca listę współrzędnych wszystkich narożników odpowiadających wykrytemu markerowi. Dzięki temu można iterować po wszystkich wykrytych markerach i wyznaczać potrzebne w danym momencie parametry znacznika. W kontekście realizowanego projektu szczególnie ważne było wyznaczanie wysokości i szerokości wykrytych znaczników oraz odległości między nimi w celu wyznaczenia przesunięcia układu wizyjnego względem obranego punktu. Jako punkt odniesienia wykorzystano centralny punkt znacznika o ID = 1, który znajdował się w środku przygotowanej tablicy. Do wyznaczania odległości między znacznikami wykorzystano ich współrzędne lewego górnego rogu zwracane przez detektor. Informacje te oraz parametry układu wizyjnego pozwoliły po wykonaniu niezbędnych transformacji na wyznaczenie rzeczywistych wymiarów znaczników oraz przesunięć układu wizyjnego względem obranego punktu odniesienia wzdłuż każdej z trzech osi: X, Y oraz Z.

9.2.3 Kalibracja kamery

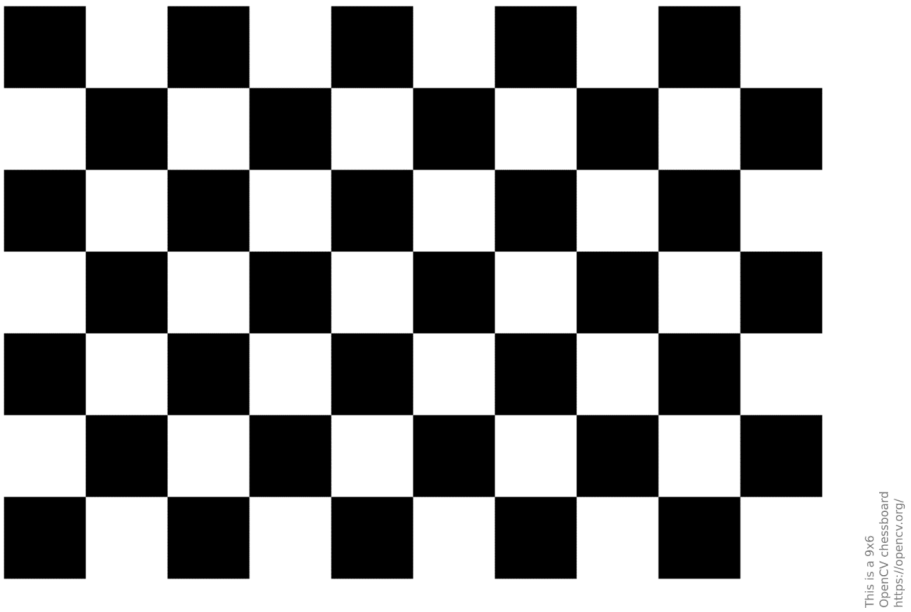
Wyznaczenie precyzyjnej pozycji kamery wymaga informacji o dokładnej wartości ogniskowej (f_x, f_y) w osi x oraz y. W tym celu należało przeprowadzić proces kalibracji kamery.

9.2.3.1 Kod kalibracji kamery

Stworzono również kod pozwalający na przeprowadzenie procesu kalibracji kamery oraz do ekstrakcji macierzy kamery, po wcześniejszym wskazaniu folderu w którym znajdują się zdjęcia. W implementacji wykorzystano bibliotekę OpenCV. Kod znajduje się na naszym repozytorium Github.

9.2.3.2 Proces kalibracji kamery

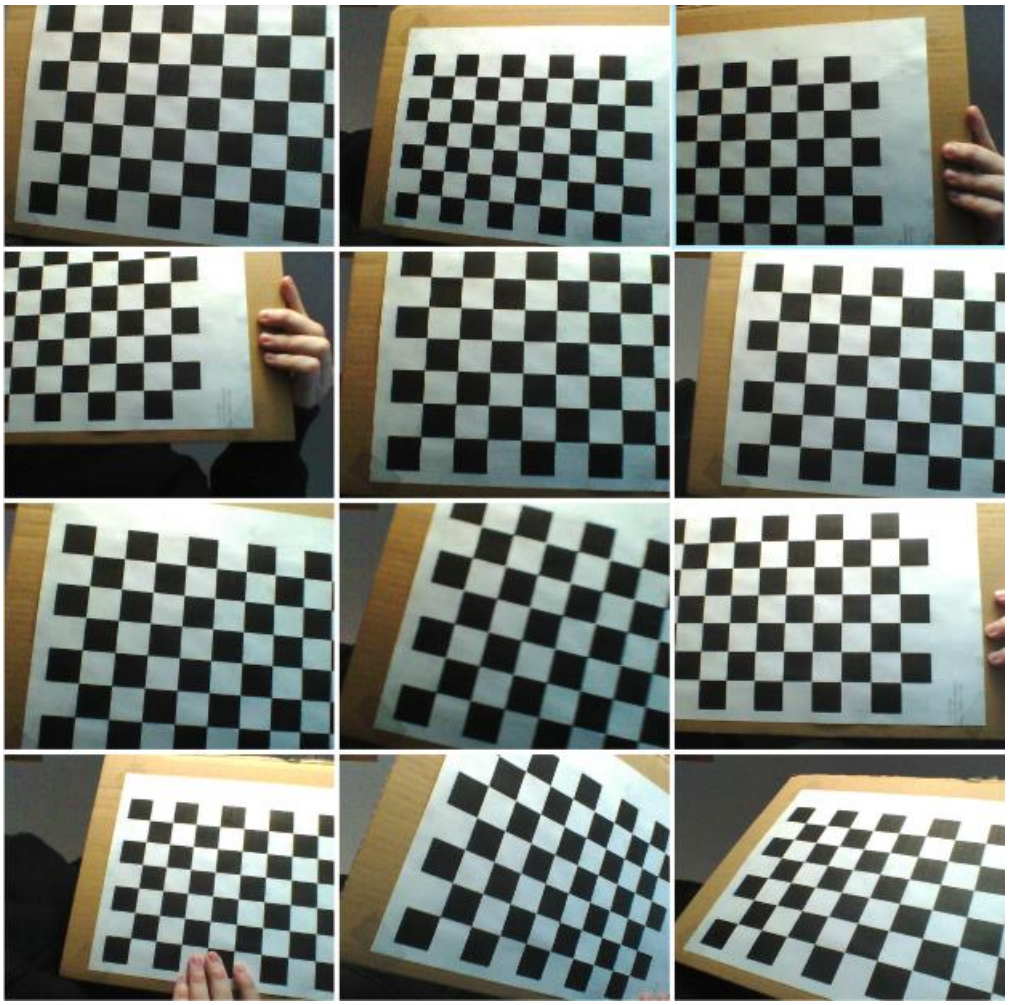
Na początku przygotowano szachownicę kalibracyjną. Szachownica kalibracyjna dostępna pod adresem https://github.com/opencv/opencv/blob/3.4/doc/pattern.png została wydrukowana i przymocowana do kawałka kartonu.



Rysunek 13 Szachownica użyta do kalibracji.

Następnie wykonano kamerą serię zdjęć kalibracyjnych (powyższa szachownica w różnych pozycjach). Zdjęcia zostały wykonane z różnym położeniem szachownicy kalibracyjnej (np. na środku, w rogach czy

przy krawędzi) i z różnymi kątami nachylenia jej względem kamery. Miało to na celu jak najlepsze wyznaczenie zniekształceń pochodzących od kulistego charakteru soczewki. Poniżej przedstawiono część z wykonanych zdjęć kalibracyjnych.

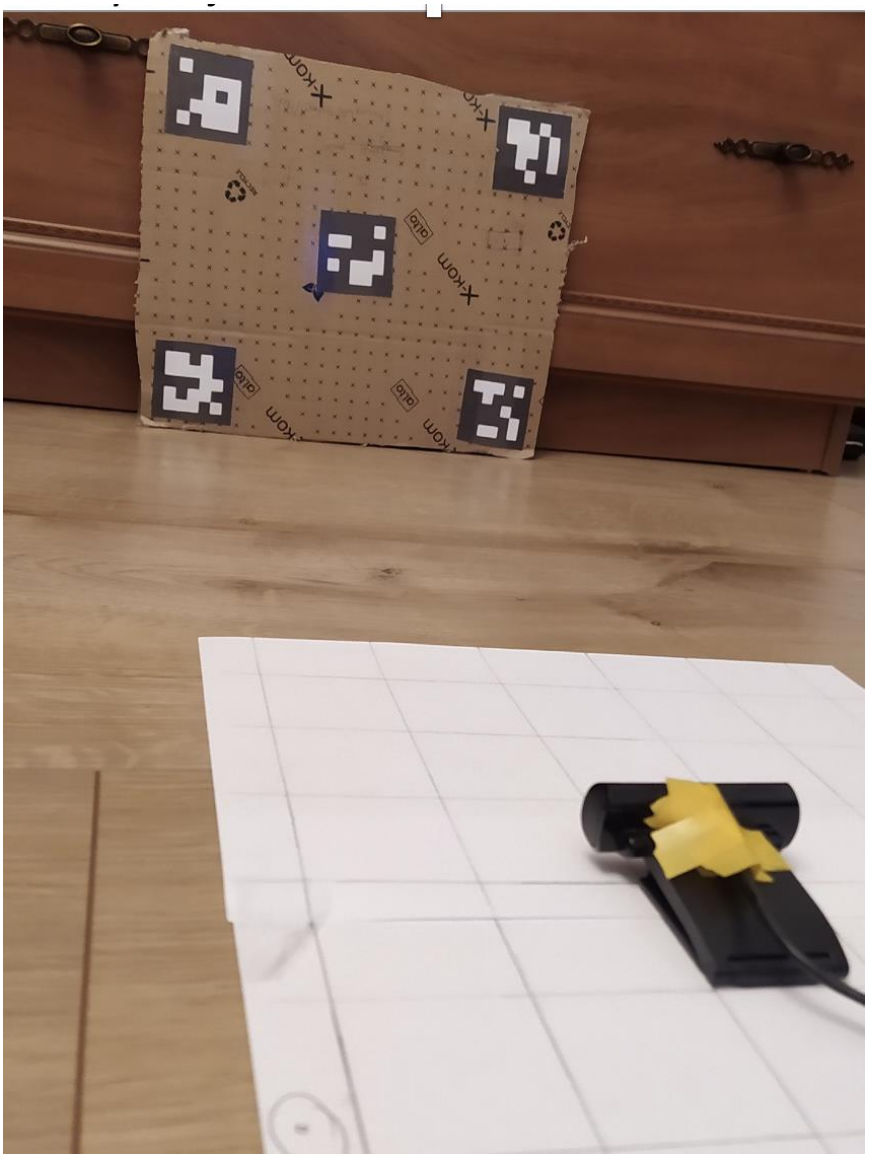


Rysunek 14 Zestawienie wybranych zdjęć kalibracyjnych.

Wykonane zdjęcia kalibracyjne zostały wykorzystane do wyznaczenia macierzy kamery. Wykorzystano skrypt opisany w podrozdziale powyżej.

10 WERYFIKACJA ROZWIĄZANIA

10.1 EKSPERYMENT 1 – ZMIANA POŁOŻENIA KAMERY



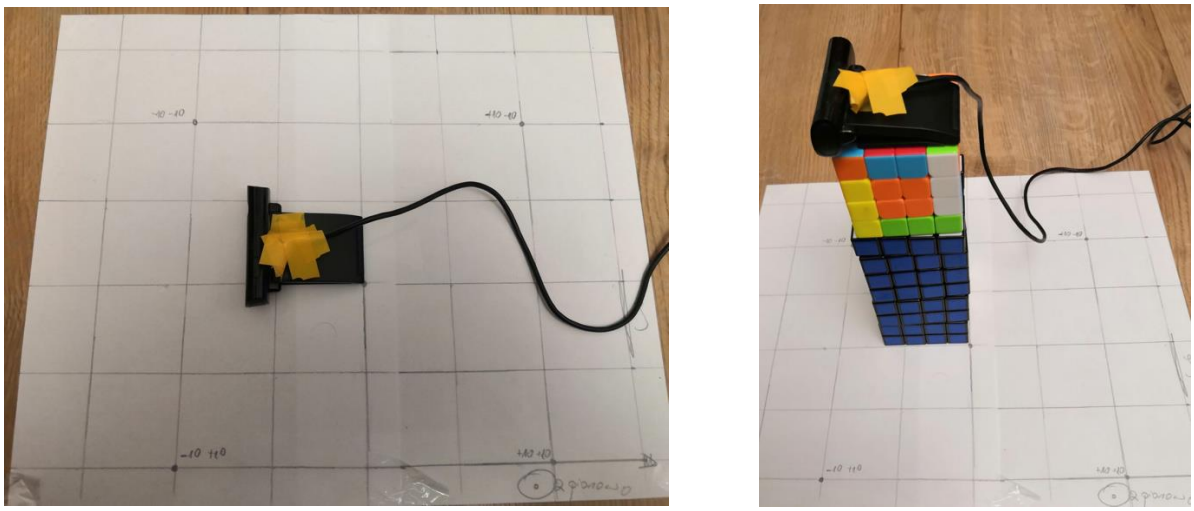
Rysunek 15 Stanowisko pomiarowe

Przebieg eksperymentu: dla niezmiennej pozycji znaczników badano wpływ zmiany pozycji kamery na wyznaczone w programie położenie.

Przebieg pomiarów:

1. Przesunięcie kamery o dany wektor
2. Odczytanie (i zapisanie) wyniku z programu
3. Powrót do (0,0,0)
4. Przesunięcie o następny wektor

Przesunięcie w płaszczyźnie XY odbywało się przy pomocy siatki odniesienia, natomiast zmiana w osi Z możliwa była dzięki umieszczeniu kamery na Kostkach Rubika o znanych wymiarach.



Rysunek 16 Zmiana pozycji kamery

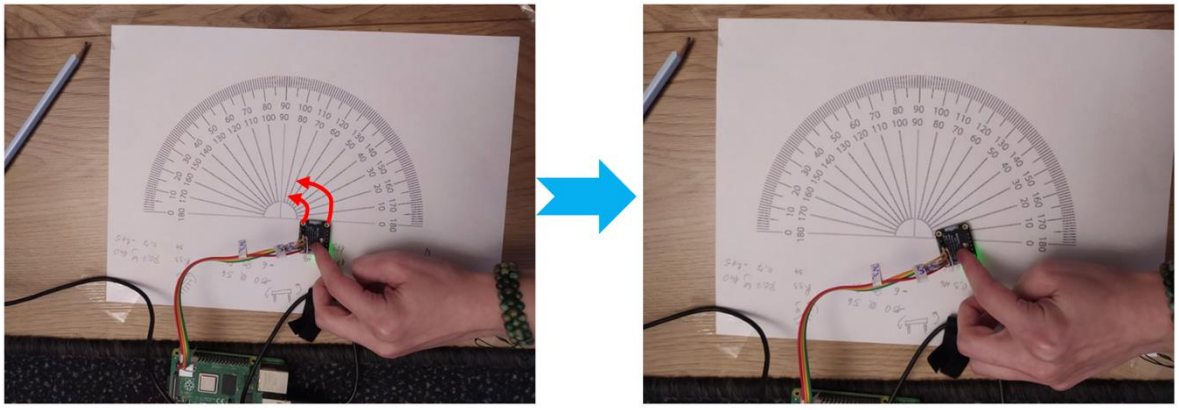
10.2 Eksperyment 2 – zmiana orientacji kamery

Przebieg eksperymentu: dla niezmiennej pozycji znaczników badano wpływ zmiany kątów nachylenia czujnika IMU na wyznaczoną w programie orientację układu

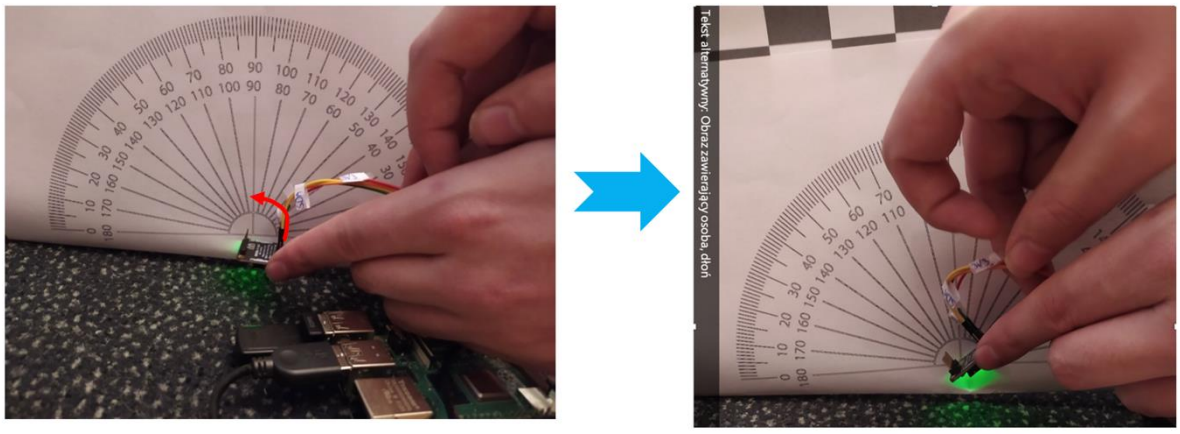
Przebieg pomiarów:

1. Obrót czujnika IMU o znany kąt
2. Odczytanie (i zapisanie) wyniku z programu
3. Powrót do kąta zerowego
4. Obrót czujnika o kolejny kąt

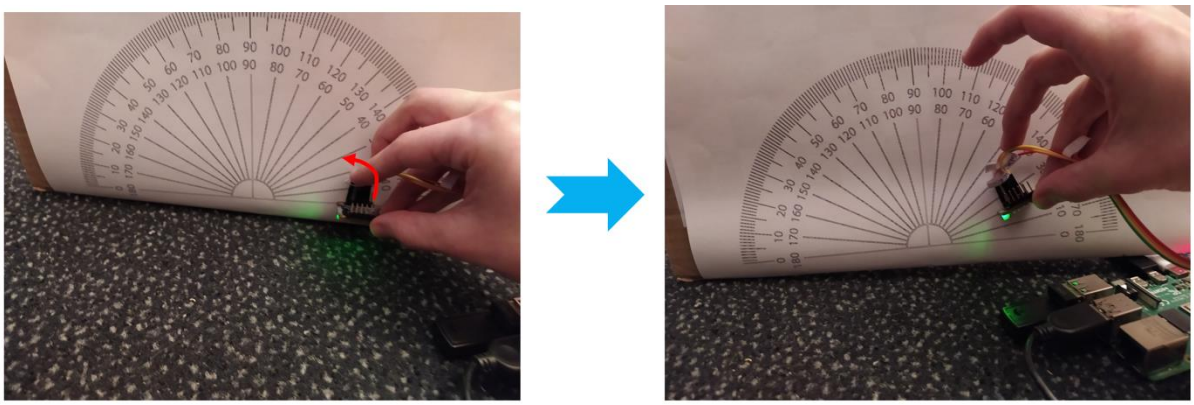
10.2.1 Pomiar Yaw:



10.2.2 Pomiar Roll:



10.2.3 Pomiar Pitch:



Rysunek 17 Zmiana kąta układu IMU

10.3 Wyniki

Wyniki pomiarów zestawiono w poniższych tabelach:

10.3.1 Odczyt kątów

Pomiar Yaw		Pomiar Roll		Pomiar Pitch	
Rzeczywisty kąt obrotu	Obliczony kąt obrotu	Rzeczywisty kąt obrotu	Obliczony kąt obrotu	Rzeczywisty kąt obrotu	Obliczony kąt obrotu
20°	23°	20°	19°	20°	23°
40°	50°	40°	39°	40°	41°
60°	73°	60°	56°	60°	60°
80°	94°	80°	78°	80°	71°

10.3.2 Odczyt pozycji

	Rzeczywisty wektor przesunięcia			Obliczony wektor przesunięcia					
				Pierwszy pomiar			Drugi pomiar		
	X	Y	Z	X	Y	Z	X	Y	Z
Przesunięcie w 1D	0	-10	0	0	-13,7	0	0	-12,6	-0,2
	0	10	0	0	10,4	0	0	10,6	-0,1
	10	0	0	9	-3,1	0	7,2	-0,9	1
	-10	0	0	-9	-2,6	-1	-8,4	-0,9	-1,2
	0	0	6	-0,7	0,8	7,1	0	-3	7,5
	0	0	13	-0,7	-0,7	16,9	-0,7	-3,3	16,7
	0	0	19	-0,7	1,2	24,6	-0,7	-5,2	24,2
Przesunięcie w 2D	-10	-10	0	-4,2	-11,2	-0,8	-7,1	-11,6	-1,6
	-10	10	0	-4,2	11,9	-0,6	-8,2	11	-1,5
	10	10	0	8,1	13,7	1,3	8,1	11,6	1,5
	10	-10	0	12	-11,4	0,8	8,1	-9,5	1,3
	10	0	19	7,7	-1,9	25,1	7	0,3	25,1
	0	10	19	0	9,9	25	0	11,8	25,2
	-10	0	19	7,7	3,1	25,3	6,1	-0,9	27,2
	0	-10	19	0	-12,4	24,4	2	-14	24,1
Przesunięcie w 3D	10	10	19	9,8	11	25,7	7	11,9	25,8
	-10	-10	19	-8,5	-13,7	23,3	-9,7	-13,9	23,5

10.4 KOMENTARZ

Po zmianie pozycji układu, obliczone przez program nowe położenie i orientacja uzyskiwane były z ok. 4s opóźnieniem, co oznacza, że nie udało się uzyskać odczytów w czasie rzeczywistym. Powodem tego może być zastosowane podczas eksperymentów Raspberry Pi, które mogło nie radzić sobie z narzuconą ilością obliczeń. Zastosowanie języka programowania Python również nie poprawiło wydajności algorytmu, język ten mimo swojej łatwości obsługi nie słynie z szybkości działania.

Ważnym wnioskiem z eksperymentów jest fakt, że wykorzystany algorytm odznacza się narastającym błędem pomiaru. Każdy kolejny powrót do pozycji (0,0,0) zwracał obliczone położenie coraz bardziej odbiegające od rzeczywistego. W przypadku pomiaru zmiany kąta nie zaobserwowano tego zjawiska, a rozbieżność obliczonego i rzeczywistego kąta prawdopodobnie wynika z niedoskonałości ręcznej metody zmiany kąta czujnika IMU.

Biorąc pod uwagę koszt wykorzystanych podzespołów oraz ograniczony czas na realizację projektu wyniki wydają się być zadowalające. W jednoznaczny sposób widać, że zaimplementowany algorytm poprawnie identyfikuje zarówno zmianę położenia, jak i orientacji układu.