

Politechnika Warszawska
Wydział Mechatroniki
Instytut Automatyki i Robotyki

Praca magisterska

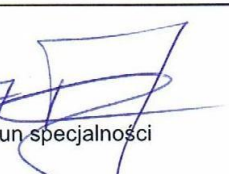
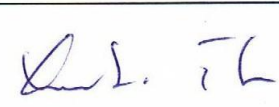
Wykonana w roku akademickim 2016/2017

Temat pracy:

**Rozpoznawanie drzwi w korytarzu i sylwetek ludzkich z
użyciem czujnika Kinect**

Prowadząca pracę: prof. dr hab. Barbara Siemiątkowska

Wykonująca pracę: Ewa Tymoszevska, grupa 87

PRACA DYPLOMOWA magisterska	
<u>Specjalność: Robotyka</u>	
<u>Instytut prowadzący specjalność:</u>	Instytut Automatyki i Robotyki
<u>Instytut prowadzący pracę:</u>	Instytut Automatyki i Robotyki
Temat pracy: <i>Rozpoznawanie drzwi w korytarzu i sylwetek ludzkich z użyciem czujnika Kinect</i>	
Temat pracy (w jęz. ang.): <i>Detection of doors in corridors and human silhouettes with Kinect sensor</i>	
<u>Zakres pracy:</u> <ol style="list-style-type: none"> 1. Przegląd istniejących rozwiązań w literaturze 2. Filtracja danych z chmury punktów w systemie operacyjnym ROS 3. Przekształcenie chmury punktów na przekrój poprzeczny w języku Python w systemie ROS 4. Zastosowanie operacji morfologicznych i algorytmu Hougha do wykrywania linii 5. Rozpoznanie linii drzwi i linii ściany na podstawie długości i koloru 6. Rozpoznanie sylwetek ludzkich za pomocą biblioteki OpenNI NITE 	
<u>Podstawowe wymagania:</u> <ol style="list-style-type: none"> 1. Znajomość programowania w językach C++ i Python 2. Znajomość systemu operacyjnego ROS, w szczególności rozwiązań dedykowanych czujnikowi Kinect 3. Znajomość zagadnień widzenia maszynowego i biblioteki OpenCV 	
<u>Literatura:</u> <ol style="list-style-type: none"> 1. Publikacje w czasopismach naukowych 2. Burrus, Nicolas, et al. Hacking the Kinect: New York, NY, Apress, 2012. 3. O'Kane, Jason M. A Gentle Introduction to ROS. Columbia, SC. 4. Dokumentacja biblioteki OpenCV 	
Słowa kluczowe: <i>Kinect, ROS, transformacja Hougha, OpenNI, OpenCV</i>	
Praca dyplomowa jest realizowana we współpracy z przemysłem?	
Tak/Nie *	
Imię i nazwisko dyplomanta: Ewa Tymoszevska	Imię i nazwisko promotora: Barbara Siemiątkowska
	Imię i nazwisko konsultanta:
Temat wydano dnia: 5/31/2017	Termin ukończenia pracy:
Zatwierdzenie tematu	
31.05.2017  Opiekun specjalności	 Z-ca Dyrektora Instytutu odpowiedzialny za sprawy dydaktyczne

Streszczenie

Pomimo ogromnych postępów robotyzacji, roboty mobilne nie zagościły jeszcze na dobre w środowiskach zamieszkiwanych przez ludzi. Budynki dostosowane do użytkowania przez ludzi są trudne w nawigacji robotów. Dlatego ważnym zagadnieniem robotyki mobilnej jest kwestia rozpoznawania drzwi i ludzi.

Do rozpoznawania tego typu elementów otoczenia użyto danych z obrazu kamery z filtrem podczerwieni (IR) - sensora Kinect. Dane te są odbierane przez program w systemie operacyjnym ROS jako rejestrowana chmura punktów i tylko punkty leżące w określonym, wąskim przedziale wysokości są wysyłane do dalszego przetwarzania. Punkty rzutowane są na płaszczyznę i w ten sposób powstaje obraz będący przekrojem poprzecznym fragmentu pomieszczenia w którym znajduje się robot. Następnie, po usunięciu szumu, celu wyznaczenia linii prostych obecnych na przekroju wykorzystywana jest transformata Hougha. W korytarzu Wydziału Mechatroniki drzwi mają jednolity kolor i rozmiar. Dlatego w tej pracy przyjęto standardową długość drzwi jako kryterium klasyfikacji. Po wykryciu segmentów linii które są potencjalnymi kandydatami (drzwi) ze względu na długość, sprawdzane są średnie wartości kolorów w przestrzeni RGB. Projekt wykorzystuje zatem rozwiązanie hybrydowe, w którym dane pobierane są w formie trójwymiarowej chmury punktów, i przetwarzane w formie dwuwymiarowego przekroju. Rozpoznawanie sylwetek ludzkich odbywa się z użyciem algorytmów biblioteki OpenNI NITE, które pozwalają na wykrycie pojawiających się postaci i śledzenie ich na mapie głębi.

Programy zostały przetestowane w warunkach korytarza Wydziału Mechatroniki. Moduł do wykrywania drzwi poprawnie rozpoznawał drzwi o ile nie były one zbyt szeroko otwarte. To ograniczenie wynikało ze sposobu w jaki program decyduje która linia jest ścianą. Moduł do wykrywania ludzi działał prawidłowo, ale miał trudności z rozpoznaniem sylwetek ludzkich kiedy czujnik Kinect był przemieszczany.

Abstract

Despite great advances of mobile robotics, robots have not yet become permanent fixtures of our homes. Navigating buildings that are made with humans in mind can be difficult for robots. That's why the issue of recognizing doors and people is important in mobile robotics, due to the possibilities of navigation and human-robot interaction.

To recognize these kinds of environment elements, Kinect depth sensing IR camera has been used. Data is received by the ROS operating system as a registered point cloud and only the points that lie in a specified, narrow range of heights are being sent for further processing. Points are projected onto a plane, and this way, an image which is a cross section of the room in which the robot currently resides is created. Next, after noise reduction, Hough transform is used to detect straight lines in the cross section image. In the Mechatronics Department corridor, doors have been found to be of constant length and similar color. That's why this dissertation has assumed the length of line as a feature used to classify a line as door. After determining the line segments which are potential door candidates based on length, average RGB values are checked to see if they match ranges observed for doors in Mechatronics department corridor. The project utilizes a hybrid solution where data is acquired as a three dimensional point and processed to form a cross section. Human silhouette detection is achieved by using OpenNI NITE library algorithms which allow for detection of humans and tracking them on a depth map.

The programs were tested in the Mechatronics Department corridor. Door detection module has correctly classified door as long as they were not wide open. This limitation is due to the nature of wall detection function which is part of door detection node. Human silhouette module has worked correctly while the Kinect sensor was stationary, but struggled to correctly track humans when the sensor was moved.

1. Wprowadzenie

Postępy w nauce i technologiach mają dla ludzkości konsekwencje dwubiegunowe – z jednej strony pozwalają na osiągnięcie celów wcześniej niemożliwych, jak np. podróż z prędkością ponaddźwiękową, czy wyprawa na dno oceanu. Z drugiej strony rozwój techniki przyczynia się do zminimalizowania czasu, który ludzie poświęcają na czynności powtarzalne i nużące takie jak zmywanie naczyń czy sortowanie listów na pocztę.

Przejęcie części zadań przez roboty w fabrykach spowodowało lepsze wykorzystanie ludzkiego potencjału pracowników w innych, bardziej zajmujących sferach. Podobny proces, dzięki rozwojowi robotyki mobilnej zajdzie być miejscach takich jak biura, instytucje, placówki medyczne czy domy mieszkalne. Roboty pracujące w takich środowiskach, aby stanowić mile widziany element budynku muszą posiadać wiedzę o własnym położeniu i możliwościach interakcji z otoczeniem, a także umiejętność wykrywania obecności ludzi.

Ważnym czynnikiem interakcji między robotem, a człowiekiem jest zdolność łączenia pojęć semantycznych z napotkanymi miejscami i obiektami [13]. Szczególnie w dynamicznym otoczeniu, robot musi dostosowywać swoją trasę do zmieniającego się otoczenia. Gdy środowiskiem jest korytarz Wydziału Mechatroniki Politechniki Warszawskiej, przydatną informacją jest obecność drzwi, ich kąt otwarcia a także obecność ludzi poruszających się przed robotem. Z podobną sytuacją mamy do czynienia w przypadku korytarza w budynku szpitala lub biura.

Powstało wiele rozwiązań problemu rozpoznawania drzwi i sylwetek ludzkich. Wiele z nich przetwarza dane z kamery RGB i wykorzystuje informacje o sposobie przemieszczania się i cechach geometrycznych. Często bardziej uniwersalnym rozwiązaniem jest użycie światła strukturalnego. Zaletą tej metody jest możliwość wykrycia drzwi niezależnie od kąta ustawienia kamery.

Wraz z rozwojem technik widzenia maszynowego wzrastać będzie obecność robotów w miejscach zajmowanych przez ludzi, np. niektóre hurtownie firmy Amazon są już dziś obsługiwane głównie przez roboty mobilne. Z drugiej strony jedynym do tej pory robotem, który zdobył popularność jako pomoc domowa jest odkurzacz Roomba. Hurtownie Amazona są zaprojektowane specjalnie dla ułatwienia robotom wykonywania ich pracy, natomiast poruszanie się po naszych domach może być dla robotów nie lada wyzwaniem. Żeby roboty mogły współistnieć z pracownikami lub mieszkańcami budynków, muszą być wszechstronnie przystosowane do poruszania się po pomieszczeniach przystosowanych do potrzeb ludzi. Niniejsza praca jest próbą rozwiązania niektórych powyższych zagadnień

2. Cel i zakres pracy

Celem projektu było zbudowanie części systemu służącego do przemieszczania się robota po korytarzu wydziału Mechatroniki Politechniki Warszawskiej. Posiadając wiedzę na temat położenia i drzwi i stopnia ich otwarcia, a także obecności ludzi przed sobą, robot potrafiłby bardziej precyzyjnie wyznaczyć trasę swojego przejazdu z jednego pomieszczenia do drugiego, omijać napotkanych ludzi lub wchodzić w nimi w interakcje, a także zmieniać „emocje” na twarzy wyświetlanej na ekranie komputera.

W zakres pracy wchodziła implementacja własnego algorytmu rozpoznawania drzwi opartego na danych z czujnika Kinect przetworzonych do postaci przekroju poprzecznego, a także wykorzystanie biblioteki OpenNI NITE do rozpoznawania sylwetek ludzkich.

Do przeprowadzenia pracy potrzebna była znajomość programowania w językach C++ oraz Python, podstawowych algorytmów przetwarzania obrazów, umiejętność pisania programów w systemie operacyjnym ROS, a także wiedza techniczna na temat czujnika Kinect i środowiska OpenNI i biblioteki NITE.

W pracy wykorzystuję następujące pojęcia:

Mapa głębi – obraz w skali szarości, w którym intensywność pikseli jest proporcjonalna do odległości danego piksela od czujnika.

Chmura punktów – nieuporządkowany zestaw punktów o wartościach (X, Y, Z) , czasem (X, Y, Z, R, G, B) , zawierających informację o położeniu i kolorze danego punktu w przestrzeni RGB. Chmura punktów powstaje po przekształceniu mapy głębi.

Światło strukturalne – wzór punktów świetlnych rzutowanych na scenę, ich zniekształcenie jest informacją o odległości danego punktu w przestrzeni rzeczywistej od czujnika.

Rejestracja – przekształcenie triangulacyjne które z użyciem obrazu RGB i mapy głębi buduje chmurę punktów z informacją o kolorze danych punktów, czyli (X, Y, Z, R, G, B) .

Struktura pracy jest następująca: po wstępie w rozdziale 3 omówię przegląd opisanych w literaturze rozwiązań problemu rozpoznawania drzwi i sylwetek ludzkich. Następnie, w rozdziale 4 przedstawię ogólny schemat algorytmu. Rozdział 5 zawiera podstawowe informacje techniczne na temat czujnika Kinect, natomiast rozdział 6, opis środowisk i bibliotek programistycznych które zostały użyte w niniejszej pracy. W rozdziale 7 szczegółowo omówiono zasadę działania algorytmu rozpoznawania drzwi a w rozdziale 8 strukturę algorytmu rozpoznawania ludzi. Rozdziały 9 i 10 zawierają opisy wyników

testowania modułów rozpoznawania drzwi i sylwetek ludzkich. W końcowej części pracy (w rozdziałach 11 - 14.) można znaleźć wnioski, podsumowanie, bibliografię i spis rysunków oraz załączników,

3. Przegląd literatury

Wiele zespołów badawczych podjęło się zadania rozpoznawania drzwi i sylwetek ludzkich. Ze względu na szybsze przetwarzanie danych i chęć opracowania algorytmów działających w czasie rzeczywistym [3][13][19], częstym rozwiązaniem jest użycie obrazu z kamery 2D. Powstały też algorytmy bazujące na czujnikach zdolnych do postrzegania głębi takich jak Kinect lub Prime Sense [11]. Większość rozwiązań zaproponowanych do dzisiaj ma swoje ograniczenia i żadne nie jest uniwersalne.

Zhang [19] zaproponował program, w który drzwi wykrywane są jako regiony których górna część jest fragmentem prostokąta, o ustalonym kolorze. Na określeniu koloru i kształtu bazuje też rozwiązanie Circirelli [3], a cechy te są wykorzystane w sieci neuronowej do klasyfikacji danego obrazu. Sekkal [16] analizuje prostokątne kształty względem zbiegających się linii perspektywy. Ten algorytm wykrywa drzwi zarówno otwarte jak i zamknięte, ale wymaga żeby kamera skierowana była w stronę korytarza. Robot ustawiony naprzeciw drzwi może ich nie zobaczyć, ponieważ w jego polu widzenia nie będzie obu linii perspektywy.

Ograniczeniem wielu metod jest brak możliwości ujęcia całych drzwi na obrazie z kamery umieszczonej na robocie [9]. Może być to przeszkodą w wąskich korytarzach, gdzie maszyna nie ma możliwości odjechania na tyle daleko od drzwi, aby widzieć całość.

Niektóre metody wykorzystują ruch otwierających się drzwi oraz ich wygląd, np Hu [10]. Najczęściej stosowanym kryterium jest analiza wektora cech za pomocą różnych technik uczenia maszynowego. Hensler [9] wykorzystuje linie pionowe, różnicę kolorów między drzwiami, a ścianą oraz obecność framugi za to Tian [17] wykorzystuje krawędzie i naroża. Murillo [13] również korzysta z układu cech geometrycznych i wyglądu. Jego rozwiązanie nie pozwala jednak na znalezienie otwartych drzwi. Algorytm zaproponowany przez Malika [12] stosuje samo-organizującą się sieć Kohonena i nie ma tej słabości.

Metody z wykorzystaniem danych z kamery zdolnej do postrzegania głębi zdają się wykazywać większą wszechstronność, ale są też wolniejsze. Borgsen [1] odnajdywał płaszczyzny drzwi, a na nich klamki i na tej podstawie rozpoznawał drzwi. To rozwiązanie nie jest przystosowane do odnajdywania zamkniętych drzwi, chociaż udaje mu się poprawnie klasyfikować drzwi z elementami szklanymi które są problematyczne z punktu widzenia światła strukturalnego, używanego przez czujniki Kinect. Klamki, jak zauważono również w innych opracowaniach wymienionych w bibliografii, mogą mieć

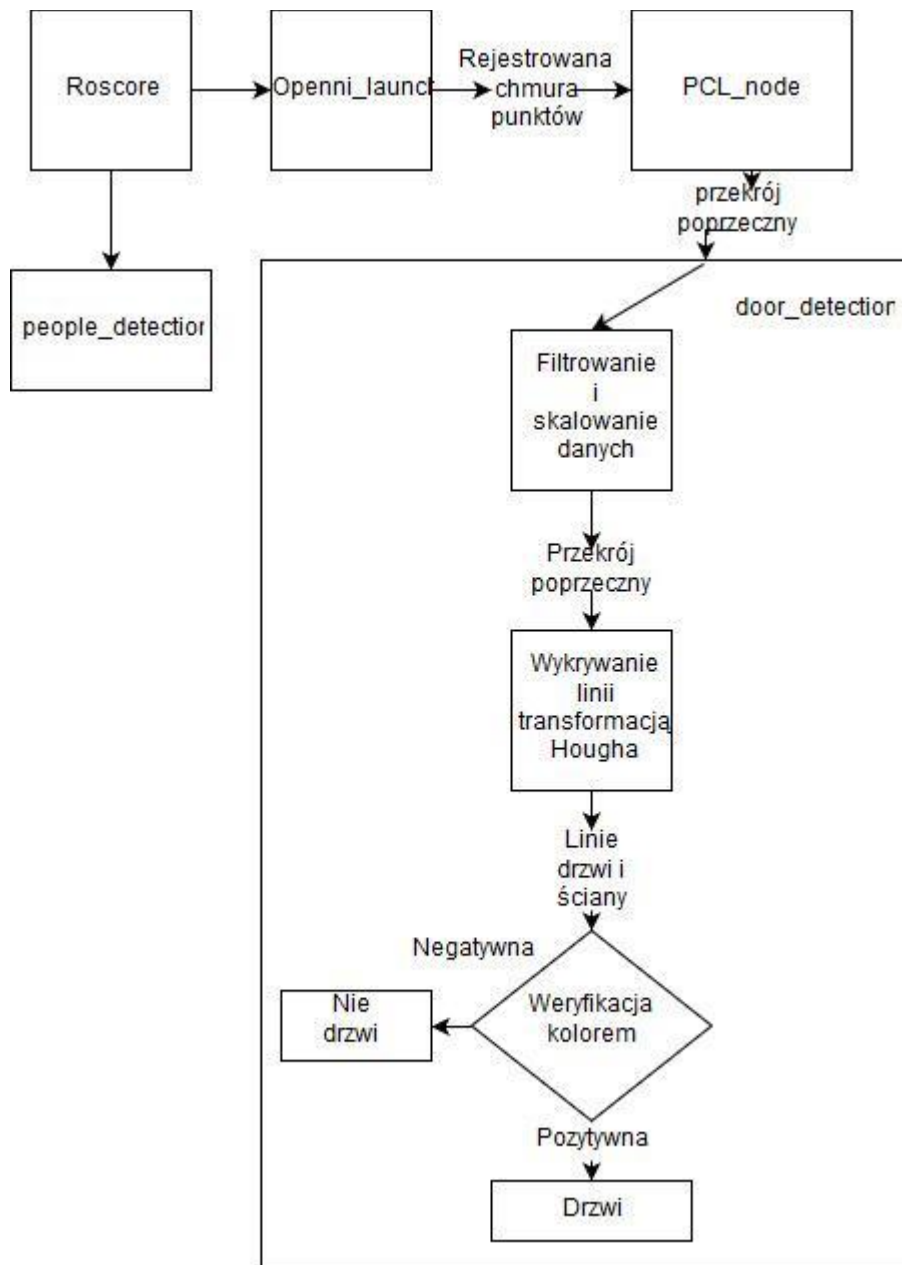
bardzo różne kształty i nie są idealnym kryterium klasyfikacji kandydatów na drzwi. Metoda ta nie działa też w czasie rzeczywistym.

Kamery zdolne do postrzegania głębi są lepiej dostosowane do zadania rozpoznawania ludzi niż zwykłe kamery. Algorytmy rozpoznawania ludzi na obrazie 2D słabo radzą sobie z grupami ludzi stojących blisko siebie [11]. Liu [11] zaproponował rozwiązanie z użyciem czujnika Kinect, które radzi sobie z tym wyzwaniem. Moduł rozpoznawania sylwetek ludzkich przedstawiony w tej pracy również jest przystosowany do rozpoznawania ludzi w grupach nawet ustawionych blisko siebie. Program Nguyena [14] do wykrywania sylwetek ludzkich bazuje na wektorze cech takich jak położenie, orientacja, wielkość. Dodatkowo analizuje cechy wyglądu, czyli kolor i teksturę, a także cechy sposobu przemieszczania się.

4. Ogólny schemat algorytmu

Na rys. 1 przedstawiono schemat projektu. Wszystkie moduły (węzły w systemie ROS) korzystają z programu roscore, który zajmuje się odbiorem i przekazywaniem wiadomości między elementami systemu.

Moduł people_detection, który zajmuje się rozpoznawaniem ludzi wymaga tylko dostępu do roscore.



Rys. 1 Ogólny schemat algorytmu użytego w projekcie

Moduł openni_launch to zestaw programów które zajmują się odbieraniem danych z czujnika

Kinect i przekazywanie ich w formacie, który może być wykorzystywany w programach ROS. Jest to moduł pobrany z zasobów społeczności systemu ROS, dlatego jego wewnętrzna architektura nie będzie omawiana. Dane publikowane są w postaci rejestrowanej chmury punktów, czyli chmury punktów z informacją o położeniu i kolorze (X, Y, Z, R, G, B).

Moduł PCL_node zajmuje się zapisem chmury punktów jako obiektu klasy PCL (patrz sekcja 4.4) po czym odfiltrowuje dane do ustalonego zakresu (X, Y, Z). Zakres danych Z wynosi 10cm przez co powstaje cienki przekrój poprzeczny obserwowanej sceny. Następnie dane chmury punktów (X, Y, Z, R, G, B) publikowane są jako wektor o powtarzającej się sekwencji (X, Y, R, G, B) dla wszystkich punktów które przeszły filtrację. Dane Z są odrzucane przez co chmura punktów może być później rzutowana na obraz 2D.

Moduł door_detection przeznaczony jest do wykrywania drzwi. Zawiera funkcje przeznaczone do filtrowania danych, przetwarzania obrazu, wykrywania linii, wybór linii drzwi i ściany, weryfikacji na podstawie powtarzalności i koloru. Moduł odbiera dane w postaci wektora (X, Y, R, G, B) i buduje dwuwymiarowy obraz binarny w którym piksele o współrzędnych (X, Y) są białe, a pozostałe piksele pozostają czarne. Na tym obrazie, będącym przekrojem poprzecznym, wykonywane są algorytmy przetwarzania obrazów i wykrywania linii. Informacje (R, G, B) są później wykorzystane do weryfikacji drzwi na podstawie barwy.

5. Czujnik Kinect

Czujnik Kinect jest popularnym dodatkiem do konsoli gier wideo XBOX 360. Urządzenie wykorzystuje emiter światła podczerwonego i kamerę na podczerwień w celu stworzenia tzw. mapy głębi.



Rys. 1 Czujnik Kinect i jego elementy

Kinect wyświetla, za pomocą promieni lasera określony układ plamek światła podczerwonego. Obraz z układem plamek jest następnie nagrywany przez kamerę z filtrem światła podczerwonego. Kropki traktowane są jak światło strukturalne. Każde zakłócenie położenia kropek jest informacją na temat odległości danego znacznika od czujnika Kinect.



Rys. 1 Obraz z kamery na światło podczerwone z widocznymi znacznikami

Użycie światła strukturalnego w zakresie podczerwonym ma następujące konsekwencje w praktycznych zastosowaniach:

- Długość fali emitowanego światła musi być utrzymywana na stałym poziomie (czujnik Kinect zapewnia to sam ogrzewając lub chłodząc diodę lasera)
- Światło naturalne ma szeroki zakres długości fali. W tym zakresie mieści się również światło o długości fali podobnej do tego, które jest emitowane przez czujnik Kinect.
- Maksymalny zasięg czujnika jest ograniczony przez moc lasera, która z kolei ograniczona jest poziomem bezpiecznym dla ludzkiego oka

Kinect odbiera obraz w rozdzielczości 1200x960, z częstotliwością 30Hz i jest on kompresowany na poziome sprzętowe. Typowy obraz wyświetlany na ekranie komputera ma rozdzielczość 640x480. Pole widzenia czujnika Kinect wynosi 58 stopni w poziomie, 45 stopni w pionie i 70 stopni po przekątnej. W odległości 2 metrów od czujnika dokładność pomiaru wynosi 3 mm w osiach X i Y oraz 1cm w osi Z (głębina) [2].

6. Elementy programistyczne

4.1 *Języki programowania*

Do napisania tej pracy potrzebna była znajomość, języka C++, a także języka Python. Język C++ charakteryzuje się dużą szybkością wykonywania obliczeń, jest jednak uważany za jeden z trudniejszych języków programowania. Python jest językiem wolniejszym, ale ze względu na prostotę składni dobrze nadaje się do prototypowania. W tej pracy koncepcja algorytmu rozpoznania drzwi zmieniała się kilkakrotnie, dlatego język Python został wykorzystany do napisania węzła `door_detection`.

W systemie operacyjnym ROS programy mogą być pisane w obu wymienionych językach programowania, nawet wtedy, gdy są elementami tego samego modułu.

4.2 *System operacyjny ROS*

System operacyjny ROS (Robot Operating System) jest narzędziem do programowania robotów. ROS powstał ze względu na fakt, że niektóre zagadnienia programowania robotów były często spotykane w wielu różnych projektach. Cechą systemu jest możliwość współdziałania i wymiany informacji pomiędzy wieloma różnymi programami jednocześnie.

Takie programy nazywane są węzłami. Węzły zorganizowane są w paczki. Informacje, którymi się dzielą nazywane są wiadomościami. Węzeł który nadaje informacje, to nadawca, a ten który je odbiera to odbiorca. Każdy węzeł może być zarówno nadawcą jak i odbiorcą wielu różnych tematów, do których publikowane są wiadomości.

ROS jest projektem o otwartym kodzie źródłowym i aktywnej społeczności w Internecie. Dzięki temu paczki i węzły implementujące rozwiązania typowych zagadnień programowania robotów są łatwo dostępne, często z bardzo liberalnymi licencjami użytkowania.

Największymi zaletami systemu ROS są wielowątkowość i możliwość wielokrotnego wykorzystania zaimplementowanych rozwiązań w projektach na całym świecie.

4.3 Środowisko OpenNI

OpenNI to Software Development Kit o otwartym kodzie źródłowym, przeznaczony do tworzenia programów przetwarzających dane z czujnika Kinect. W zasobach systemu ROS istnieje popularna paczka o nazwie `openni_launch` która, po uruchomieniu, publikuje dane z Kinecta w postaci między innymi mapy głębi, chmury punktów, rejestrowanej (z informacją RGB) chmury punktów oraz obrazu RGB.

4.3.1 Biblioteka OpenNI NITE

Biblioteka OpenNI NITE jest biblioteką pomocniczą OpenNI która posiada zaimplementowane algorytmy do rozpoznawania i śledzenia sylwetek ludzkich, a także dłoni, gestów i części ciała.

4.4 Biblioteka Point Cloud Library

Biblioteka Point Cloud Library (PCL) zawiera wiele użytecznych algorytmów do przetwarzania danych zapisanych w postaci chmur punktów. Jest biblioteką o otwartym kodzie źródłowym.

4.5 Biblioteka OpenCV

Biblioteka OpenCV jest najbardziej popularną biblioteką algorytmów widzenia maszynowego. Cechuje się szybkością działania, łatwością implementacji i dobrą dokumentacją. Jest to również biblioteka o otwartym kodzie źródłowym.

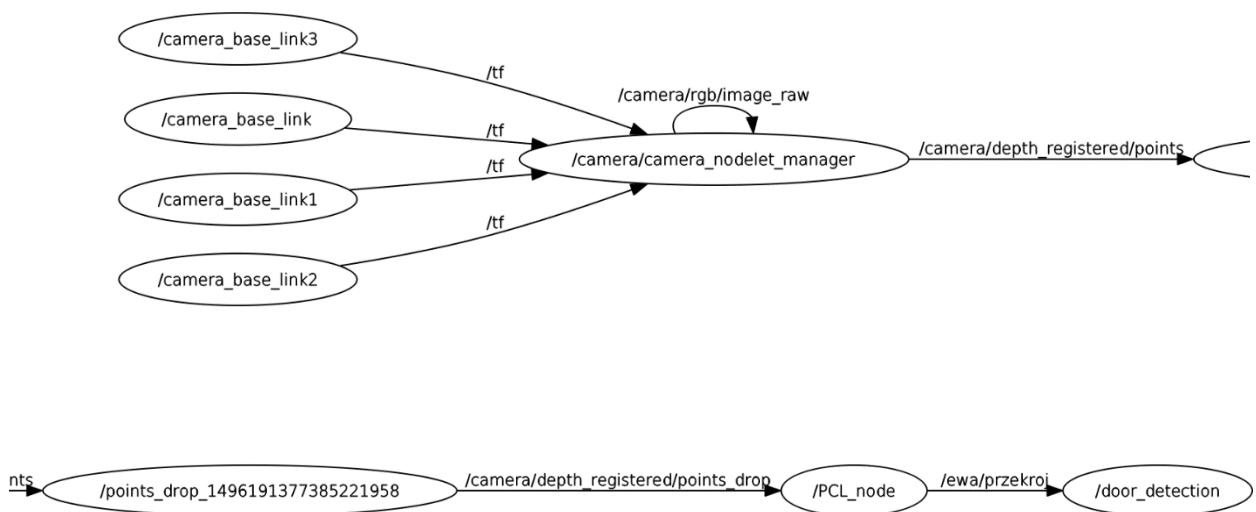
4.6 Inne biblioteki

Numpy – biblioteka w języku Python służąca do przeprowadzania obliczeń naukowych na dużych macierzach.

7. Rozpoznawanie drzwi

Moduł rozpoznawania drzwi napisany został w systemie operacyjnym ROS, w językach C++ i Python. Składa się on z programów `camera_nodelet_manager`, `points_drop`, `PCL_node` i `door_detection`. Węzły `camera_nodelet_manager` i `points_drop`... są węzłami pobranymi z zasobów społeczności systemu ROS, natomiast węzły `PCL_node` i `door_detection` zostały napisane przez autora tej pracy. W podsekcjach tego rozdziału zostanie opisana ogólna struktura rozwiązania z opisem działania wszystkich programów systemu ROS, następnie omówione zostaną kluczowe algorytmy zaimplementowane w bibliotece OpenCV, które zostały wykorzystane w module, na koniec szczegółowo opisane będą programy własne napisane dla tej pracy - moduły `PCL_node` i `door_detection`.

7.1 Ogólna struktura rozwiązania



Rys. 1 Struktura programu rozpoznawania drzwi (obraz zawinięty)

Na rys. 4 można zobaczyć, że układ węzłów w module rozpoznawania drzwi jest sekwencyjny.

Warto w nim zwrócić uwagę na następujące elementy:

- Program `/camera/camera_nodelet_manager` publikuje wiadomości do tematu o nazwie `/camera/depth_registered/points`. Węzeł ten jest częścią modułu `openni_launch` pobranej z zasobów społeczności systemu ROS. Częścią tego modułu są również programy widoczne po lewej stronie od `/camera/camera_nodelet_manager`. Temat `/camera/depth_registered/points`

przechowuje dane w typie „rejestrwana chmura punktów”, czyli chmurę punktów z wartościami RGB dla każdego punktu (X, Y, Z).

- Program `/points_drop` porzuca cztery na pięć z otrzymanych zestawów danych i publikuje tylko co piąty. Został tutaj użyty, żeby właściwy program rozpoznawania drzwi był w stanie nadążyć za tempem publikacji. Ten program jest częścią jednego z wbudowanych modułów w systemie operacyjnym ROS.
- Program `PCL_node` został napisany w języku C++ przez autorkę tej pracy. Odbiera on przerzedzone dane w postaci chmury punktów i filtruje je do wąskiego zakresu, który umożliwia później stworzenie przekroju poprzecznego. Ten program publikuje wiadomości w postaci wektora wartości (X, Y, R, G, B).
- Program `door_detection` również został napisany dla tej pracy. W tym programie zachodzi właściwe przetwarzanie obrazu przekroju poprzecznego. Program ten pokazuje okno obrazu na którym wyświetlają się efekty działania modułu wykrywania drzwi.

7.1 Algorytmy zaimplementowane w OpenCV

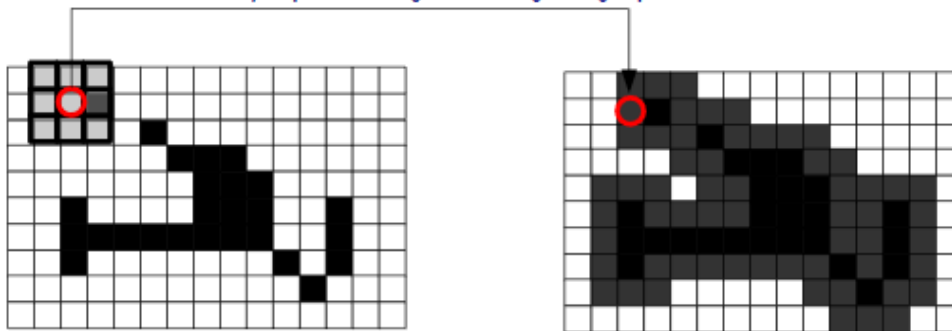
7.1.1 Dylacja

Dylacja to operacja morfologiczna. Przekształcenie jest pomocne, gdy chcemy usunąć szum i łączyć kształty które znajdują się blisko siebie.

W module rozpoznawania drzwi niniejszej pracy operacja dylacji jest konieczna do wygładzenia obrazu przekroju poprzecznego, który jest zaszumiony i posiada nieciągłości (por. rys. 6 i rys. 7). Algorytm ten jest zastosowany w węźle `door_detection` w funkcji `image_processing` (sekcja 7.3.2). Dzięki operacji dylacji i erozji (opisanej poniżej), otrzymano obraz o znacznie wyraźniejszych liniach, dzięki temu algorytm transformacji Hough’a (sekcja 7.1.3) z większą dokładnością wykrywa linie.

Operacja dylacji działa następująco: jeśli choć jeden piksel sąsiadujący z rozważanym pikselem ma wartość 1 to również i rozważany piksel przyjmuje wartość 1. W wyniku tego kształty na obrazie zwiększają pole powierzchni, łączą się blisko znajdujące się obiekty i wypełnione zostają otwory. Przykład działania operacji dylacji przedstawiony został na rys. 5. Czarne piksele są obrazem wyjściowym a szare piksele to te, które zostały dodane po dylacji.

w otoczeniu zdefiniowanym przez SE jest co najmniej 1 piksel '1'



Rys. 1 Wizualizacja rezultatów zastosowania dylacji (SE – element strukturalny).

Wynik operacji dylatacji na przekroju poprzecznym wykorzystywanym w projekcie (rys. 6) przedstawiony jest na rys. 7



Rys. 1 Przekrój poprzeczny użyty w projekcie przed zastosowaniem operacji dylacji i erozji (zamknięcia morfologicznego). Widoczne są nieciągłości i szum.

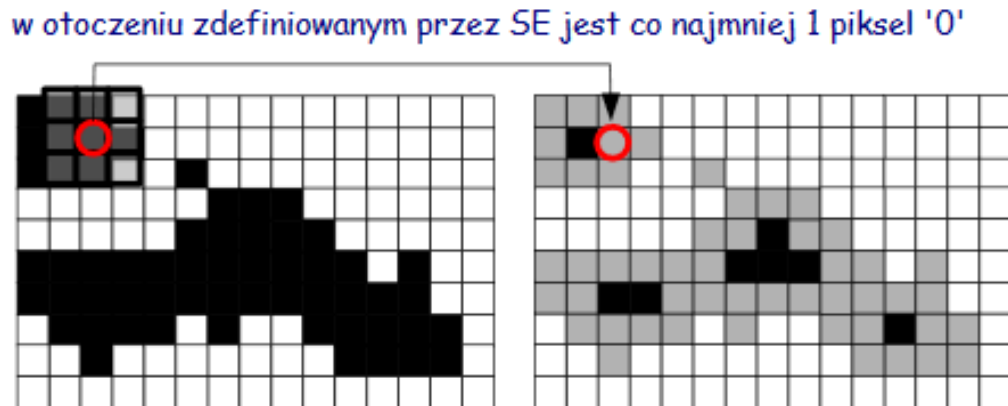


Rys. 1 Operacja dylacji wykonana na obrazie z programu do wykrywania drzwi.

7.1.2 Erozja

Erozja jest operacją morfologiczną odwrotną do dylacji. W przypadku erozji, jeśli choć jeden piksel sąsiadujący z obecnie rozważanym przyjmuje wartość 0, wtedy obecny piksel również przyjmuje wartość 0.

Rys. 8 przedstawia przykładowy efekt zastosowania operacji erozji. Czarne piksele to obraz wejściowy a szare piksele to te usunięte przez operację erozji.



Rys. 1 Wizualizacja rezultatów zastosowania erozji

Operacja dylatacji i następującej po niej erozji nazywa się zamknięciem morfologicznym. W obecnej pracy obraz wejściowy poddawany jest właśnie takiemu przekształceniu. Pomaga to w odsumianiu. Dzięki temu rozbite fragmenty linii wyznaczającej przekrój poprzeczny łączą się. Linie stają się łatwiej wykrywalne przez algorytm detekcji linii Hough'a.

Wynik operacji zamknięcia morfologicznego w obecnej pracy przedstawiono na rys. 9



Rys. 1 Operacja zamknięcia morfologicznego wykonana przez program do rozpoznawania drzwi.

7.1.3 Transformacja Hough'a

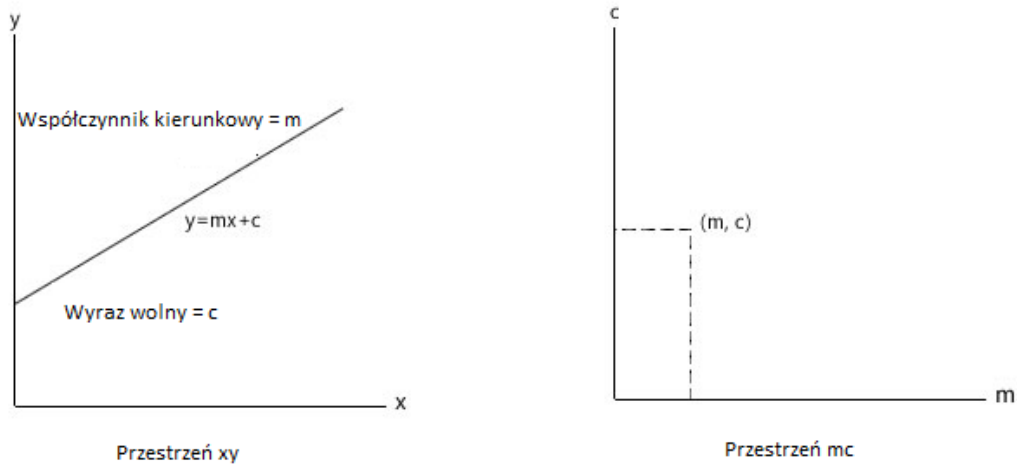
W celu znalezienia linii reprezentujących drzwi i ścianę w obrazie przekroju poprzecznego zastosowano transformację Hough'a.

Wynikiem algorytmu transformacji Hough'a w OpenCV jest zestaw odcinków dopasowanych do fragmentów linii prostych w obrazie wejściowym. Na rys 10. Przedstawiono wykorzystanie algorytmu detekcji linii Hough'a zastosowanego na przekroju poprzecznym stworzonym w węźle `door_detection`. Linie przedstawione na rysunku są już wyselekcjonowane i zweryfikowane przez kolor.

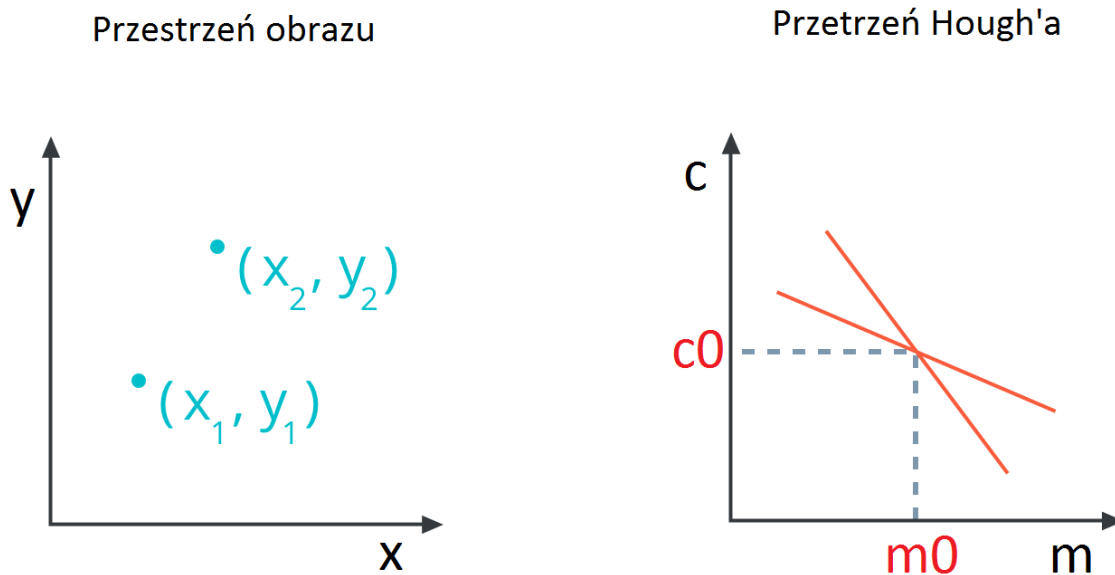


Rys. 1 Wykorzystanie transformacji Hough'a w module rozpoznawania drzwi. Linia różowa reprezentuje drzwi a linia niebieska ścianę. Na tym obrazie linie zostały wyselekcjonowane przez funkcję węzła door_detection.

Transformacja Hough'a jest zamianą przestrzeni XY na przestrzeń MC, gdzie m to współczynnik kierunkowy a c to wyraz wolny równania linii prostej. Rys 9 przedstawia wynik transformacji Hough'a dla pojedynczej linii. Linia o parametrach m i c w przestrzeni XY jest punktem w przestrzeni MC (również odwrotnie - punkt w przestrzeni XY jest linią w przestrzeni MC).

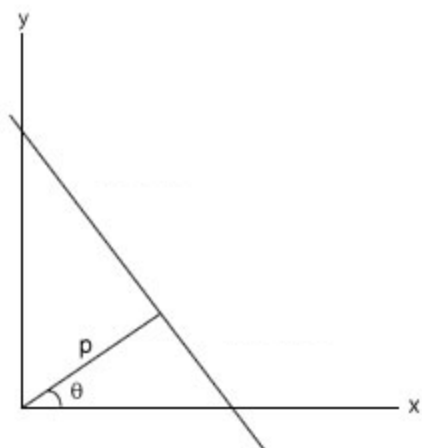


Rys. 1 Przestrzenie XY i MC



Rys. 1 Reprezentacja dwóch punktów w przestrzeni obrazu i przestrzeni Hough'a

Ponieważ współczynnik m równania linii prostej osiąga nieskończoność w przypadku linii pionowej, obliczenia transformacji Hough's wykonuje się w postaci normalnej, czyli z wykorzystaniem współrzędnych kołowych r i θ . R oznacza promień mierzony od początku układu współrzędnych i θ oznacza kąt pomiędzy promieniem a dodatnią osią X .

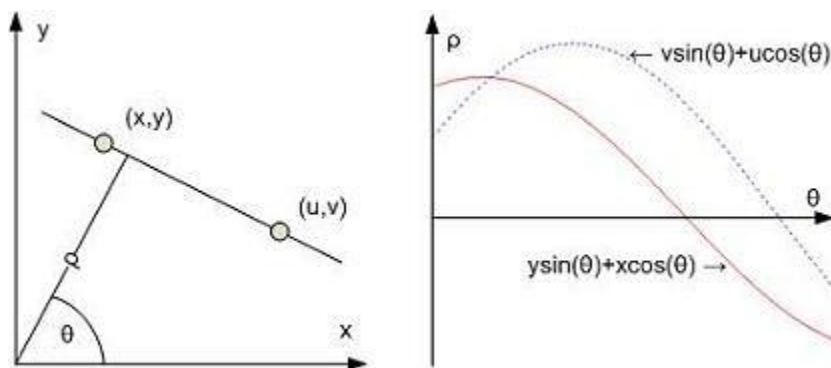


Rys. 1 Współrzędne r i θ

We współrzędnych biegunowych równanie linii prostej przyjmuje postać:

$$p = x_1 \cos \theta + y_1 \sin \theta$$

Gdzie (x_1, y_1) , to współrzędne punktu przez który przebiega linia prosta.



Rys. 1 Przekształcenie z przestrzeni XY do przestrzeni $r\theta$

Jak widać na rys. 12 punkt w przestrzeni XY staje się sinusoidą w przestrzeni $r\theta$.

Algorytm wykorzystujący do znajdowania linii prostych przekształcenie Hough'a działa w następujący sposób: budowana jest macierz przestrzeni $r\theta$ dla zakresu θ od -90 do 90 i zakresu r od 0 do maksymalnej wartości r . Rozważany jest każdy piksel w wejściowym obrazie binarnym. Jeżeli wartość piksela jest niezerowa, generowana jest jego sinusoida w przestrzeni $r\theta$.

Wartości sinusoidy w przestrzeni $r\theta$ zapisywane są w macierzy. W każdej komórce zapisana jest liczba głosów, czyli liczba sinusoid, które przechodziły przez dany punkt macierzy.

Współrzędne $r\theta$ komórek, w których znajduje się duża liczba głosów są potencjalnymi kandydatami na parametry linii prostej w przestrzeni XY.

W bibliotece OpenCV funkcja `HoughLinesP()` przyjmuje następujące argumenty:

- Image – obraz wejściowy
- Rho – rozdzielczość promienia w macierzy w przestrzeni $r\theta$ [piksel]
- Theta – rozdzielczość kątowna w macierzy w przestrzeni $r\theta$ [radian]
- Threshold – ilość głosów które musi otrzymać komórka macierzy $r\theta$ żeby jej współrzędne były uznane za parametry linii prostej [liczba całkowita]
- minLineLength – minimalna długość linii [piksel]
- maxLineGap – maksymalna odległość pomiędzy pikselami żeby mogły być uznane za część tej samej linii [piksel]

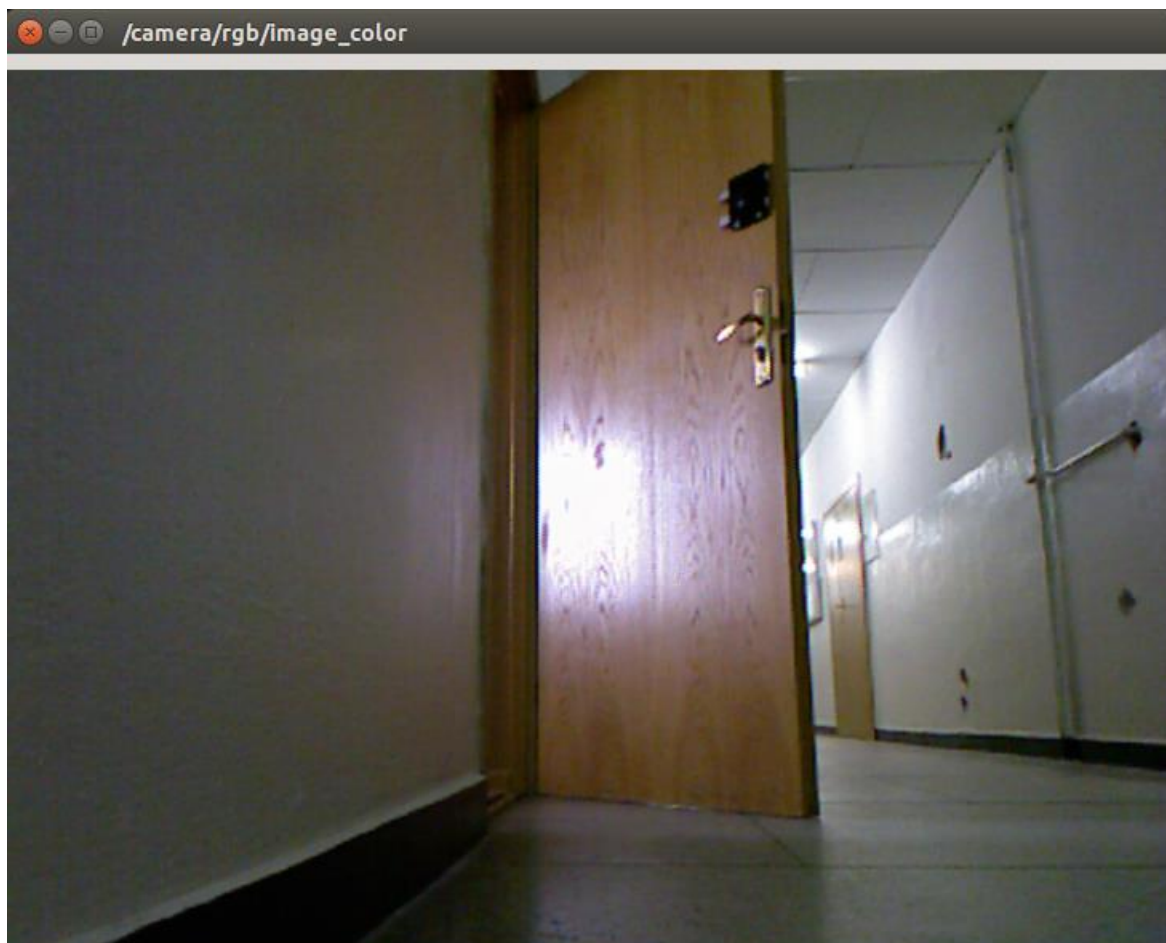
7.2 Program *PCL_node*

Program wykorzystuje dwa źródła wiedzy. Jednym z nich jest chmura punktów reprezentująca otoczenie widziane przez kamerę z filtrem podczerwieni (X, Y, Z). Drugim źródłem jest informacja o kolorze pozyskana z kamery standardowej (R, G, B). Dane te są zestawiane ze sobą w programie *camera_nodelet_manager* w procesie zwanym rejestracją. W ten sposób powstaje rejestrowana chmura punktów (X, Y, Z, R, G, B) która jest przetwarzana przez węzeł *PCL_node*. Przykładowy obraz z kamery RGB został przedstawiony na rys. 16.

Funkcją węzła jest odbieranie danych w postaci chmury punktów, zapisywanie jej w formacie PCL, odfiltrowywanie punktów które znajdują się poza ustalonymi zakresami i wysyłanie ich dalej jako wektor (X, Y, R, G, B). Zakres na osi Z jest wąski (10 cm) i po odfiltrowaniu punkty można rzutować na płaszczyznę obrazu wykorzystując współrzędne (X, Y).

Zakres wysokości wpływa na jakość obrazu przekroju poprzecznego. Kiedy zakres jest szeroki linie przekroju poprzecznego stają się grube ze względu na ustawienie czujnika Kinect, który nie jest idealnie wypoziomowany. Z drugiej strony wąski zakres wysokości zawierałby w sobie mało punktów przez co linie przekroju poprzecznego byłyby przerywane. Podczas testowania węzła stwierdzono, że zakres 10cm jest dobrym kompromisem pomiędzy grubością linii, a jej ciągłością. Linie, które są cienkie i ciągłe są lepiej wykrywane przez transformatę Hougha.

Dodatkowo zastosowano filtr usuwający punkty znajdujące się zbyt daleko od kamery. Jest to zakres wartości 4m. Dokładność czujnika Kinect spada znacznie wraz z odległością obserwowanego miejsca od czujnika, np. dla odległości 50cm dokładność wynosi 1.5mm a dla odległości 5m wynosi 5cm. Stwierdzono, że zakres 4m jest odpowiedni, żeby w polu widzenia ująć standardowe drzwi wraz z fragmentem ściany.



Rys. 1 Przykładowy obraz z kamery RGB

7.3 Program *door_detection*

Celem tego programu jest odbiór przefiltrowanych danych od programu PCL_node, stworzenie z nich obrazu 2D który reprezentuje przekrój poprzeczny oglądanej sceny, wykrycie linii prostych i wyselekcjonowanie tych które mogą być drzwiami, wybór linii która prawdopodobnie reprezentuje ścianę i weryfikacja znalezionej linii drzwi na podstawie koloru.

Węzeł door_detection wyświetla znalezione przez siebie linie i oblicza kąt pomiędzy linią drzwi i ścianą.

Węzeł ten napisany został w języku Python ze względu na możliwość szybkiego testowania pomysłów w tym języku.

Funkcje napisane na użytek tego węzła zostały opisane w następujących sekcjach:

- Funkcja data_filtering

W tej funkcji wektor który odbierany jest z węzła PCL_node zostaje rozdzielony na pięć wektorów: x, y, z, red, green, blue. Wartości x i y są skalowane tak, aby mieściły się w obrazie o rozdzielczości 640x640 i zachowywały stałą proporcję.

- Funkcja image_processing

W tej funkcji wykonywane są operacje dylacji, erozji i wykrywania linii metodą Hough'a.

Na początku inicjalizowany jest rozmiar maski dla operacji dylacji i erozji. Stwierdzono że większy rozmiar maski daje bardziej wygładzony obraz wyjściowy, ale za to bardzo spowalnia program. Dlatego ustawiono rozmiar maski jako 5 osiągając kompromis między wygładzeniem a szybkością.

W dalszej części funkcji przeprowadzane są operacje dylacji i erozji z maską o wcześniej ustalonym rozmiarze.

Z początku planowano użycia w tym miejscu detektora krawędzi Canny'ego. Obraz był jednak zaszumiony, a jego krawędzie poszarpane. Z tego powodu nie było możliwe uzyskanie krawędzi przy których dałoby się odróżnić linie o trudnym do wychwycenia przesunięciu. Takie linie otrzymuje się w przypadku zamkniętych drzwi. Pomimo, że dokumentacja OpenCV zaleca użycie detektora krawędzi Canny'ego, w przypadku tego projektu znacznie lepsze rezultaty uzyskano z grubszych linii pozostałych po operacjach dylacji i erozji.

W następnej części funkcji inicjalizowane są parametry transformaty Hougha (funkcji HoughLinesP):

- $\rho = 2.0$ – wartość rozdzielczości promienia przekracza jeden piksel ze względu na grubość linii w obrazie wejściowym
- $\theta = (\pi/180)*0.75$ – jak powyżej, nieco powiększona (względem zalecanej przez dokumentację OpenCV) wartość rozdzielczości kątowej ma wpływ na znajdowanie linii przy grubszej kresce
- $\text{threshold} = 150$ – ten próg ustalony metodą prób i błędów odfiltrowuje mało prawdopodobne linie
- $\text{min_line_length} = 40$ – minimalna długość linii pozwala na odrzucenie bardzo krótkich linii których nie spodziewano się spotkać w prostych korytarzach
- $\text{max_line_gap} = 30$ – maksymalna przerwa między pikselami. Ten parametr ustalono metodą prób i błędów

Po zdefiniowaniu wszystkich parametrów, wywoływana jest funkcja `HoughLinesP` która zwraca położenie wszystkich wykrytych linii spełniających wymienione powyżej ograniczenia.

- Funkcja `find_door_line`

Ta funkcja wynajduje linię której długość mieści się w z góry ustalonym zakresie.

Żeby uniknąć wykrywania zamkniętych drzwi na pustych ścianach, oblicza się różnicę pomiędzy współrzędnymi punktów końcowych wykrytej linii, a średnim ich położeniem w ostatnich pięciu wykrytych liniach. Punkty końcowe linii wykrytych na pustych ścianach mają tendencję do „skakania” po obrazie dlatego metoda ta dobrze nadaje się do tego zadania.

- Funkcja `bounding_box`

Funkcja ta powstała, żeby wspomóc funkcję `find_wall`, która znajduje najdłuższą linię, której punkty końcowe nie leżą w prostokącie narysowanym dookoła linii drzwi.

Funkcja `bounding_box` pozyskuje najpierw średnią wartość współrzędnych wykrytej linii drzwi. Następnie oblicza wektor jednostkowy o kierunku zgodnym z pochyleniem linii drzwi. Później oblicza dwa wektory prostopadłe do linii drzwi. Wektory te są dodawane lub odejmowane od punktów końcowych linii drzwi. Dzięki temu uzyskuje się punkty prostokąta otaczającego linię drzwi. Działanie funkcji `bounding_box` przedstawiono na rys. 15.



Rys. 1 Działanie funkcji `bounding_box`

- Funkcja `find_wall`

Funkcja ta stara się znaleźć linię ściany po tym jak znaleziono już linię drzwi. Żeby znaleźć linię która mogłaby być ścianą, szuka się linii która jest najdłuższa i której punkty końcowe nie leżą w prostokącie wyznaczonym przez funkcję `bounding_box`. Funkcja wykorzystuje moduł `Path` biblioteki `matplotlib`.

- Funkcja `color_averages`

Ta funkcja również wykorzystuje prostokąt stworzony przez funkcję `bounding_box`. Dzięki temu odnajduje wszystkie piksele, które mieszczą się wewnątrz prostokąta opasującego linię drzwi. Po odnalezieniu wszystkich tych pikseli funkcja zapamiętuje ich średnią wartość (R, G, B).

- Funkcja `final_door_verification`

Ta funkcja podsumowuje działanie poprzednich sprawdzając, czy średnie wartości kolorów mieszczą się we wcześniej ustalonych granicach dla koloru brązowego typowego dla drzwi korytarza wydziału Mechatroniki.

Jeżeli drzwi w takim kolorze zostaną odkryte, program sprawdza ile obrazach z ostatnich dziesięciu wykryto drzwi. Jeżeli jest to więcej niż 5 wtedy program wypisuje na konsoli informację o obecności drzwi oraz o kącie ich otwarcia w stosunku do wykrytej linii ściany.

8. Rozpoznawanie sylwetek ludzkich

8.1.1 Ogólna struktura rozwiązania

Program `people_detection` napisano w języku C++. Biblioteka OpenNI NITE jest dostępna tylko w tym języku. Przy budowie programu wzorowano się na innym programie dostępnym w zasobach internetowej społeczności ROS. Ten program nazywa się `openni_tracker`.

Rozpoznawanie sylwetek ludzkich z biblioteką OpenNI NITE jest bardzo proste. Można się o tym przekonać zwracając uwagę na długość kodu węzła `people_detection`. W przypadku tej pracy wystarczyło zainicjalizować klasę `xn::UserGenerator` i napisać funkcje *callback* dla dwóch sytuacji. Pierwsza sytuacja, to wykrycie nowej postaci ludzkiej. Druga sytuacja zniknięcie postaci ludzkiej z pola widzenia.

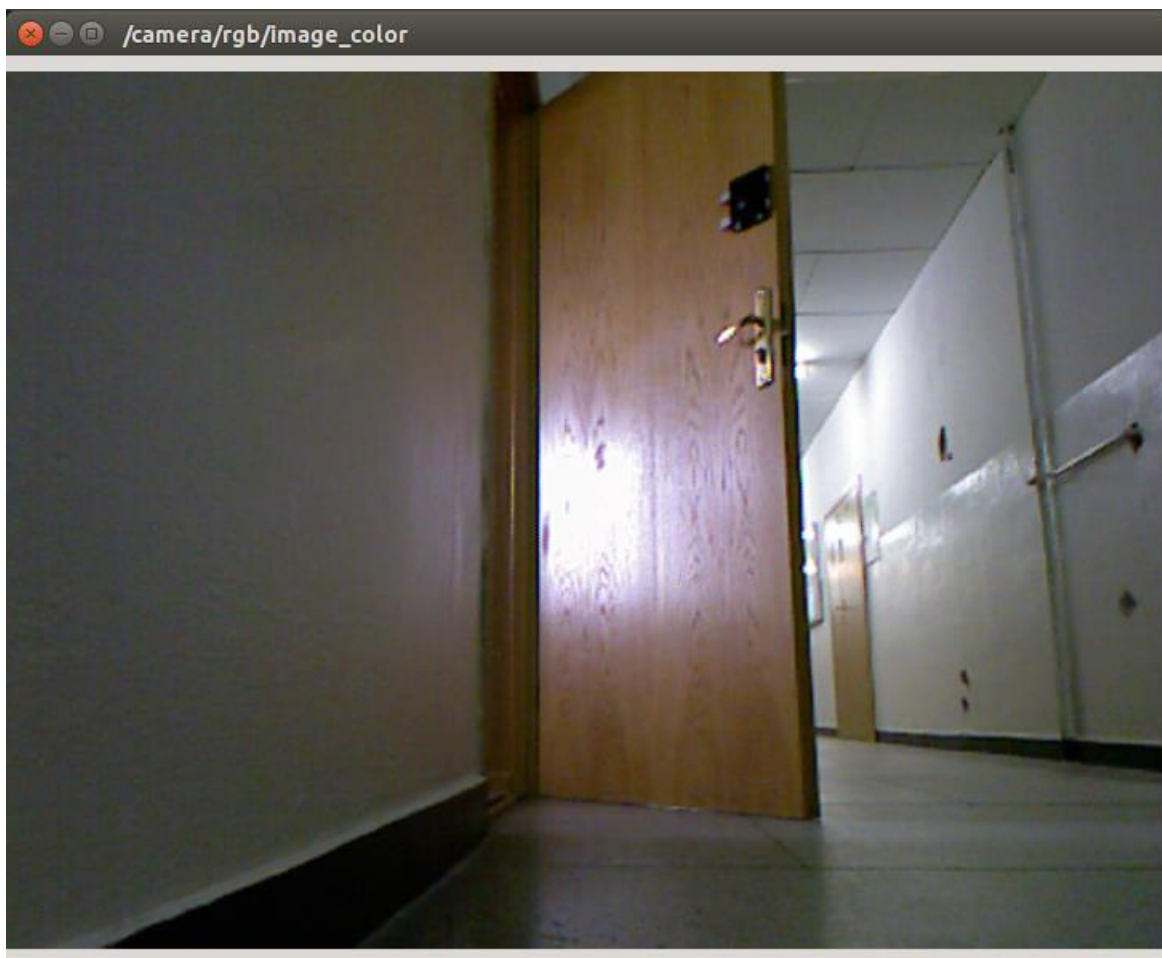
W tej pracy funkcja *callback*, która wywoływana jest w przypadku odnalezienia nowego użytkownika, zapamiętuje liczbę widzianych użytkowników podawaną przez klasę `xn::UserGenerator` i nadaje ją w temacie `/no_of_users`

9. Testy – rozpoznawanie drzwi

W poniższej sekcji wymieniono kilka przypadków próby wykrycia drzwi – w każdym z wymienionych widoków drzwi są obecne. Na końcu tej sekcji przedstawiono tabelę zawierającą wyniki testów algorytmu rozpoznawania drzwi.



Rys. 1 Półotwarte drzwi widziane od strony otwartej



Rys. 1 Widok na półotwarte drzwi od strony otwartej

Z tego widoku drzwi i ściana wykryte są prawidłowo jak widać na rys 16.



Rys. 1 Półotwarte drzwi widziane od strony otwartej, bez widoku na ścianę za nimi

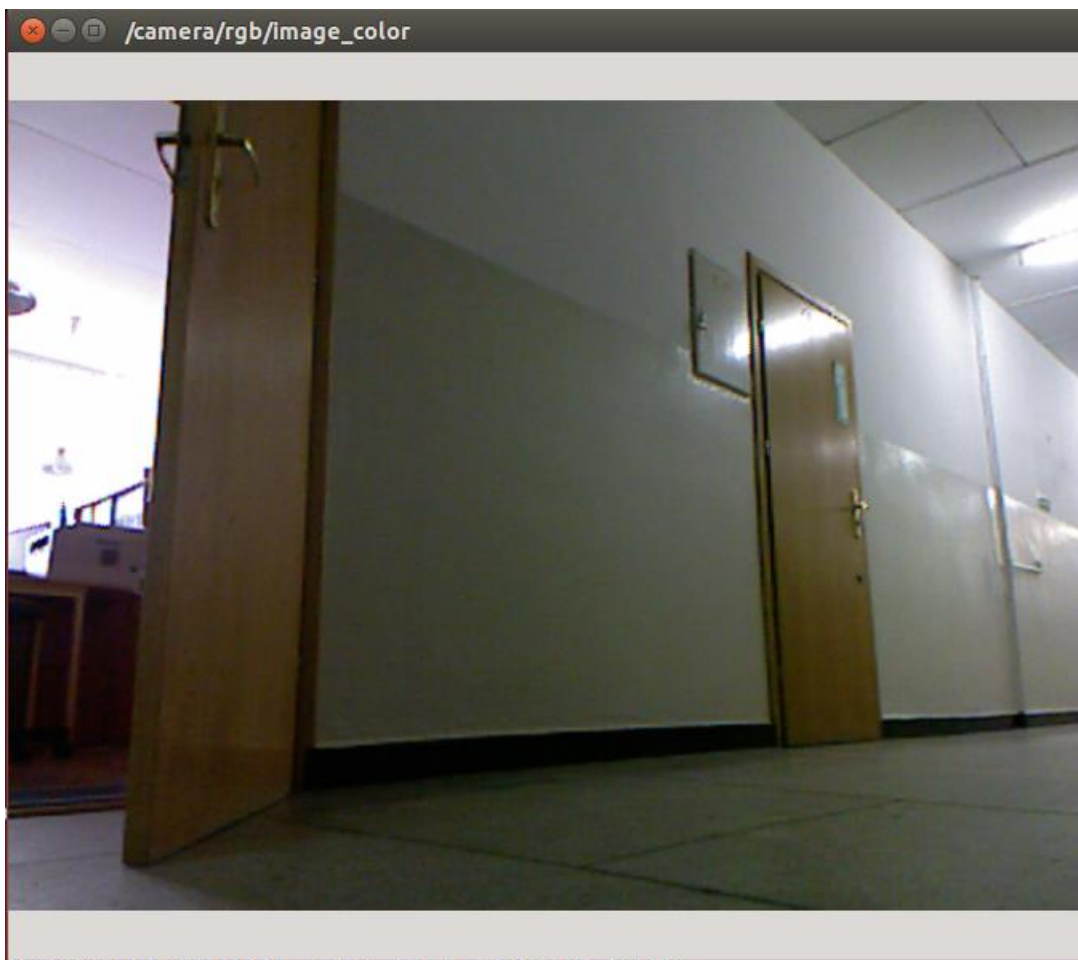


Rys. 1 Widok na półotwarte drzwi od strony otwartej

W tym ujęciu drzwi znowu rozpoznane są prawidłowo.

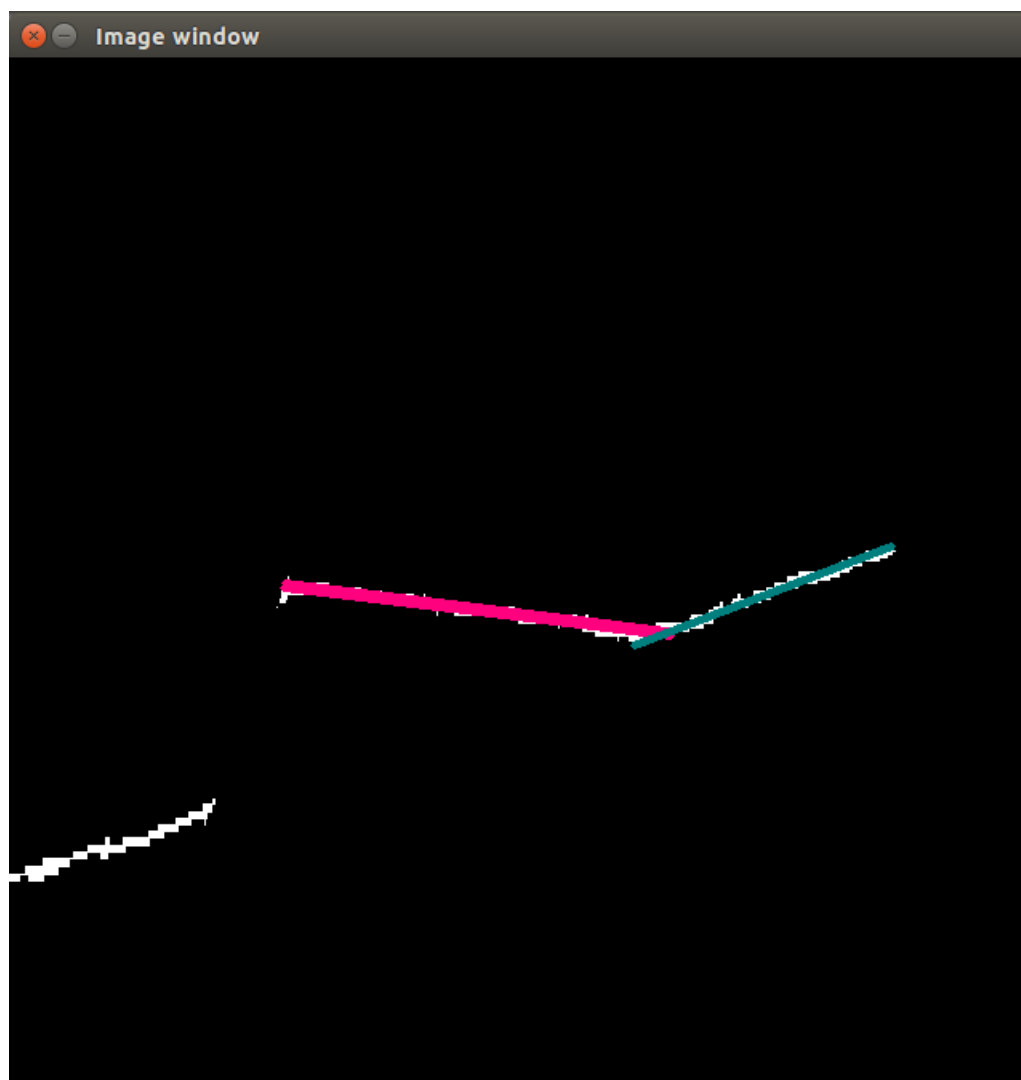


Rys. 1 Widok na lekko otwarte drzwi od strony otwartej.

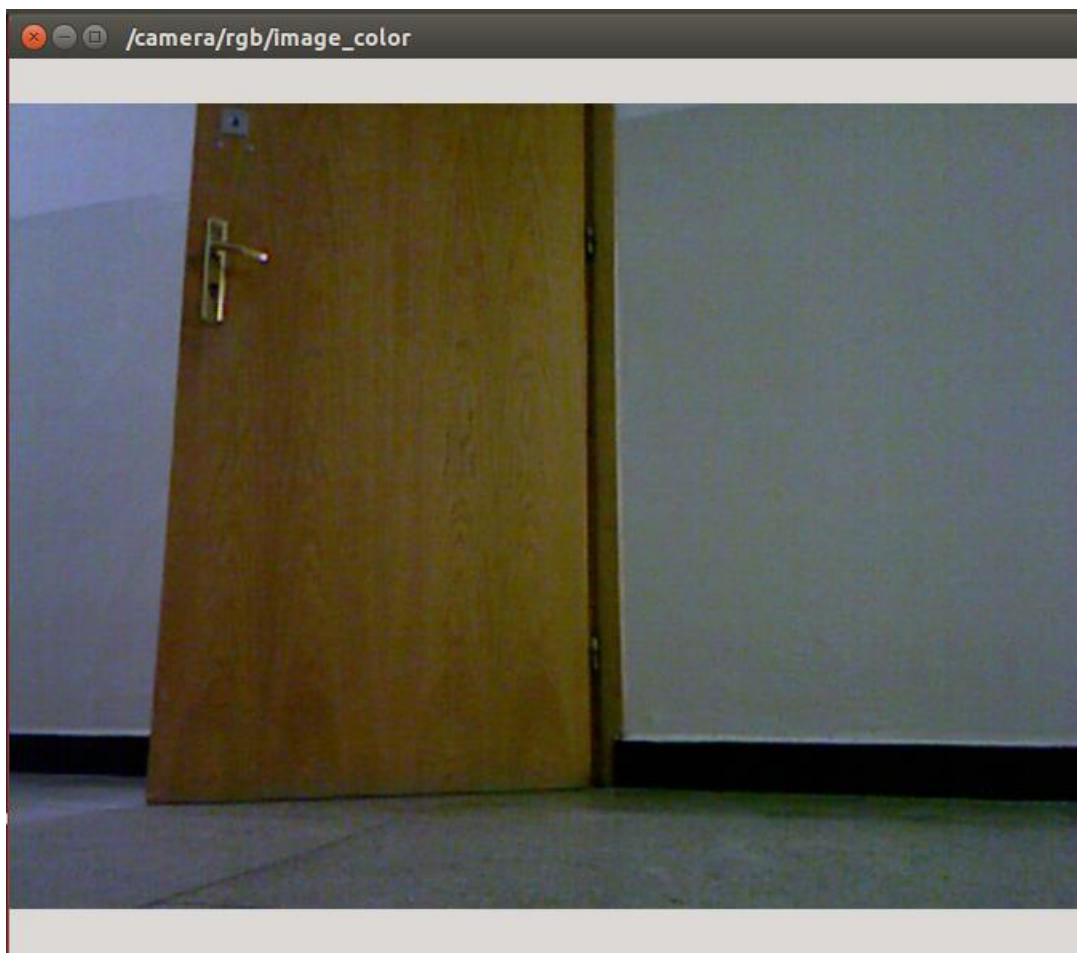


Rys. 1 Widok na lekko otwarte drzwi od strony otwartej

W tym ujęciu drzwi ponownie zostały rozpoznane prawidłowo. Choć trudno to pokazać na papierze, drzwi poruszające się, wtedy gdy były otwierane i zamykane przez ludzi również były prawidłowo rozpoznane.

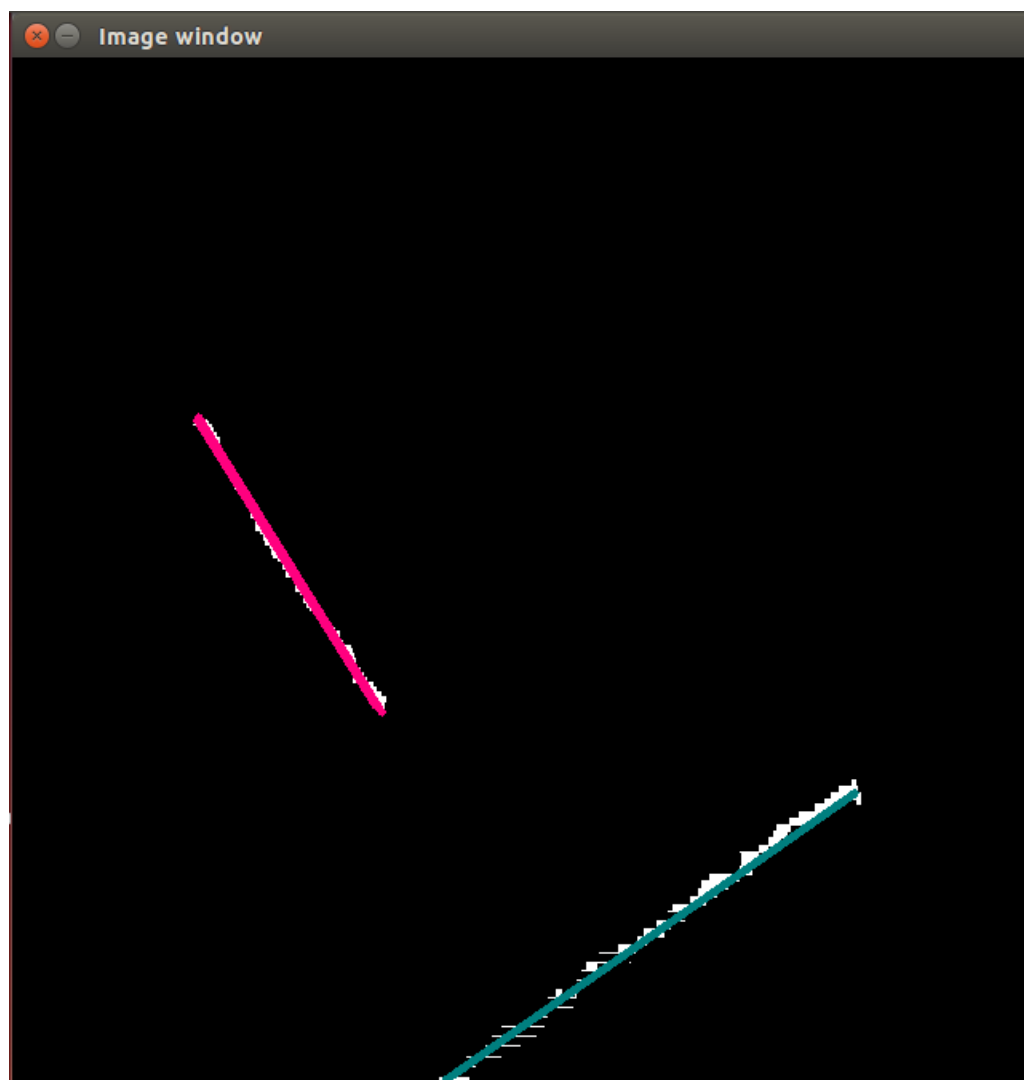


Rys. 1 Drzwi lekko otwarte widziane od strony zamkniętej

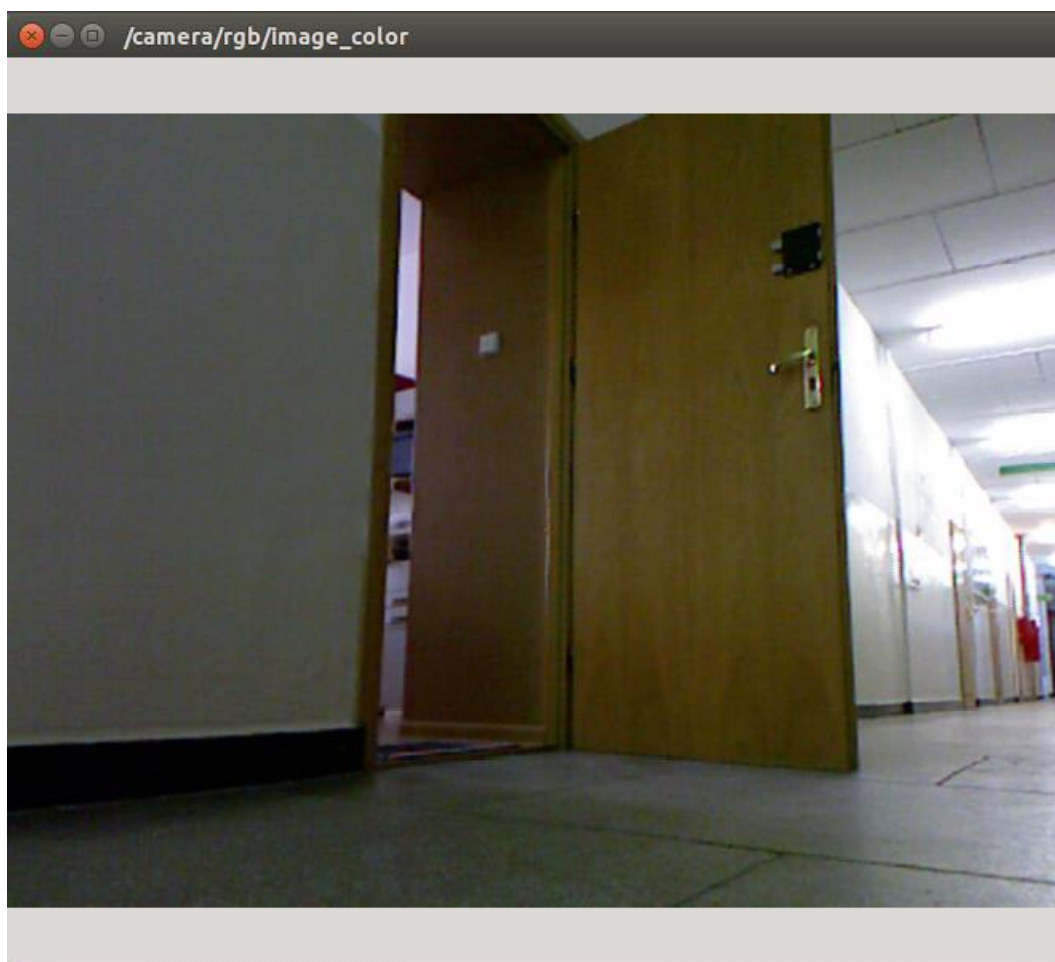


Rys. 1 Widok na drzwi lekko otwarte widziane od strony zamkniętej

W tym ujęciu drzwi również zostały prawidłowo rozpoznane.

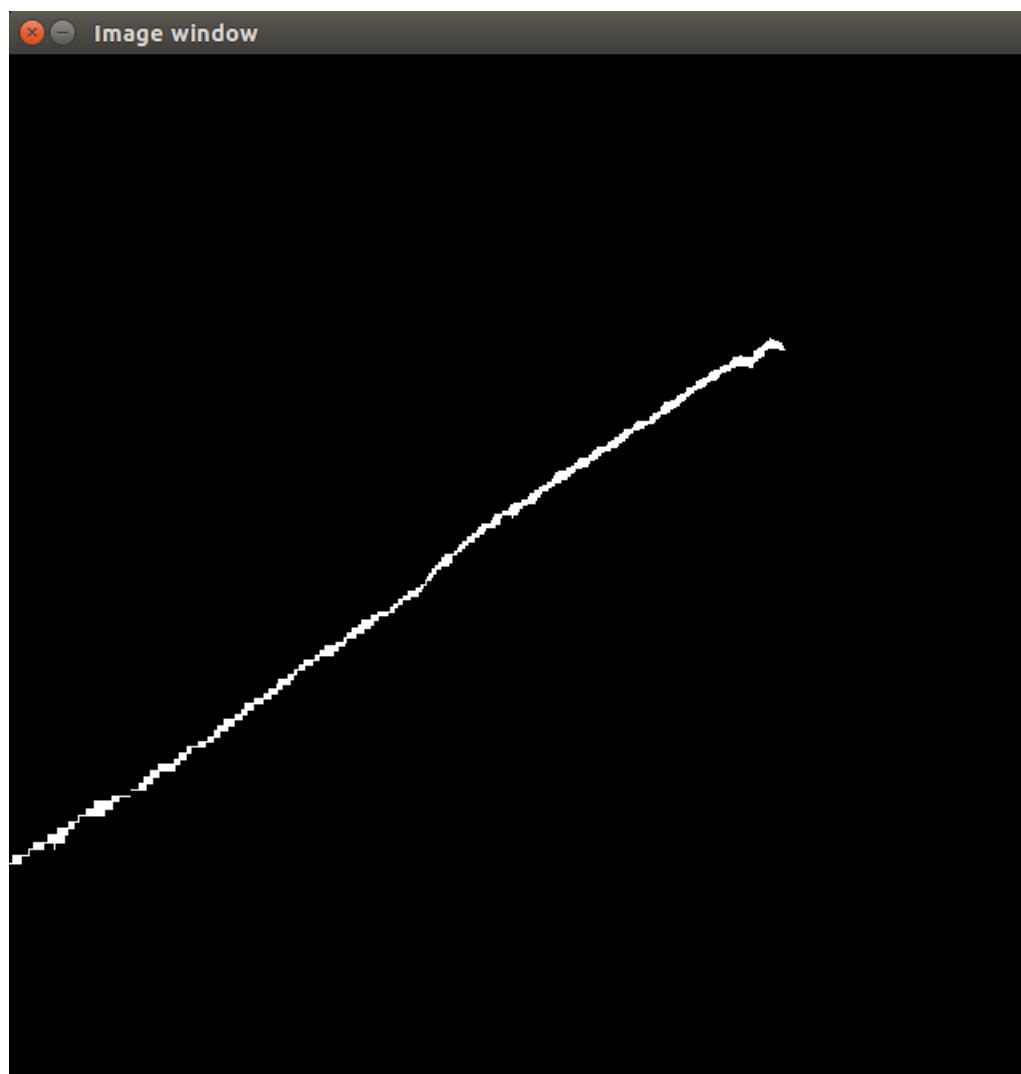


Rys. 1 Drzwi otwarte pod kątem prostym widziane od strony otwartej

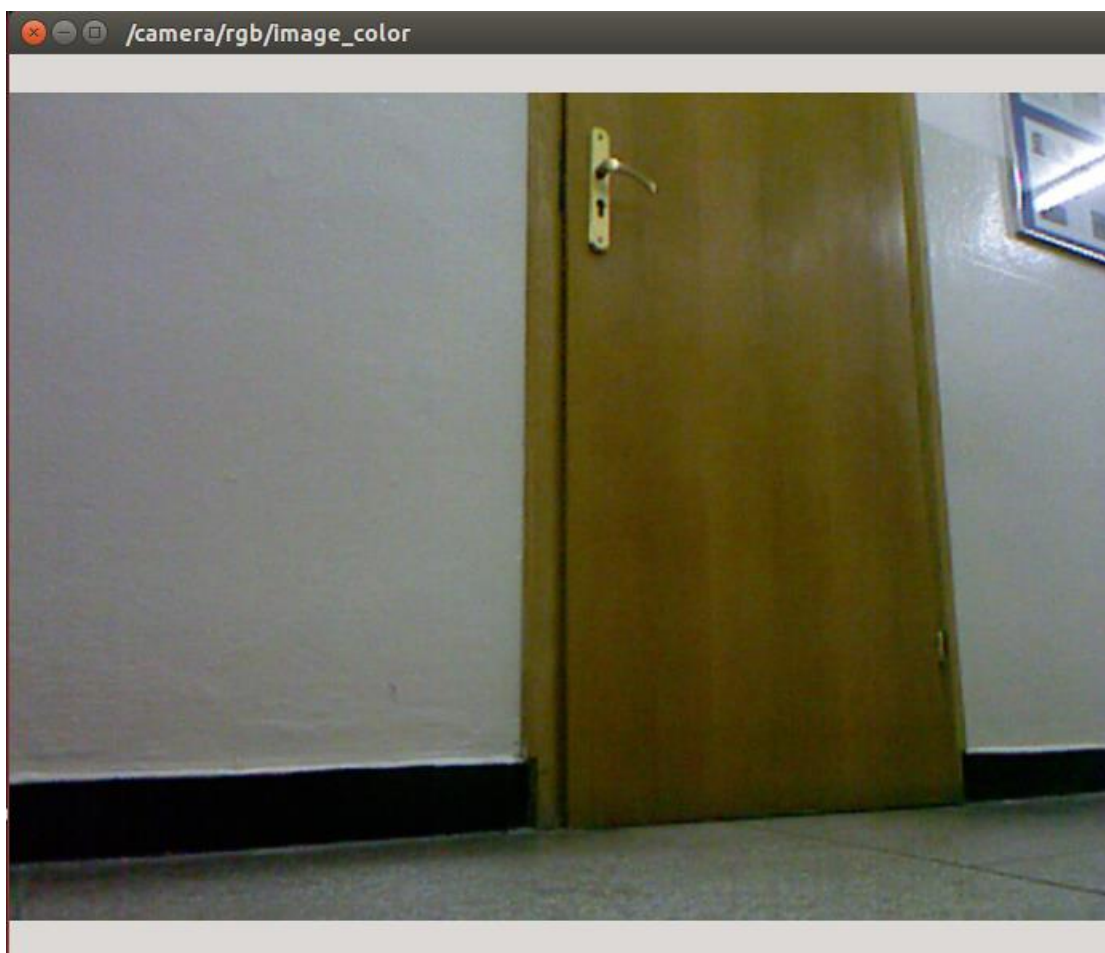


Rys. 1 Widok na drzwi otwarte pod kątem prostym od strony otwartej

W tym ujęciu drzwi nie zostały prawidłowo rozpoznane. Fragment ściany został wstępnie uznany za drzwi, a drzwi za ścianę. Weryfikacja na podstawie koloru pozwoliła odrzucić ten model, jednak drzwi nie zostały rozpoznane w miejscu w którym się znajdowały.

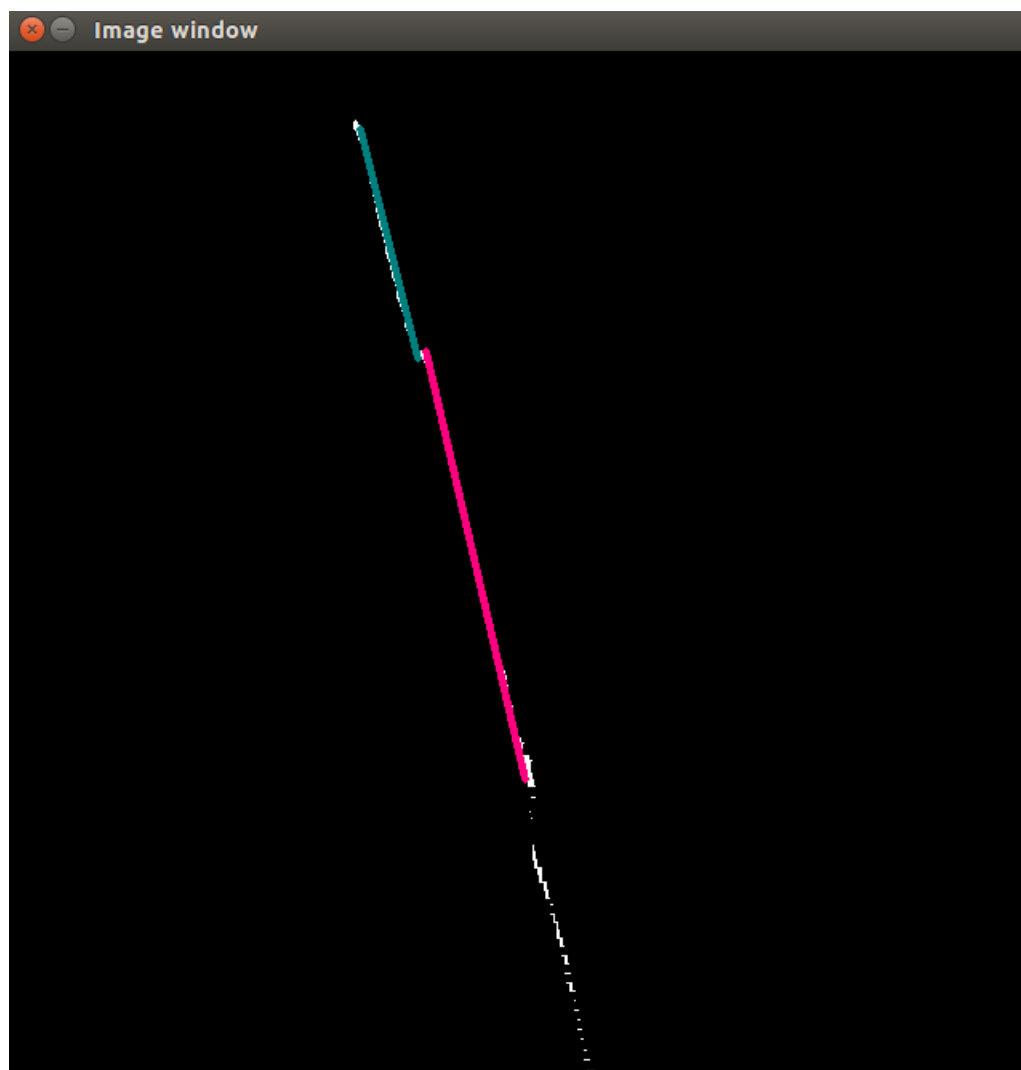


Rys. 1 Drzwi zamknięte o nierównym profilu

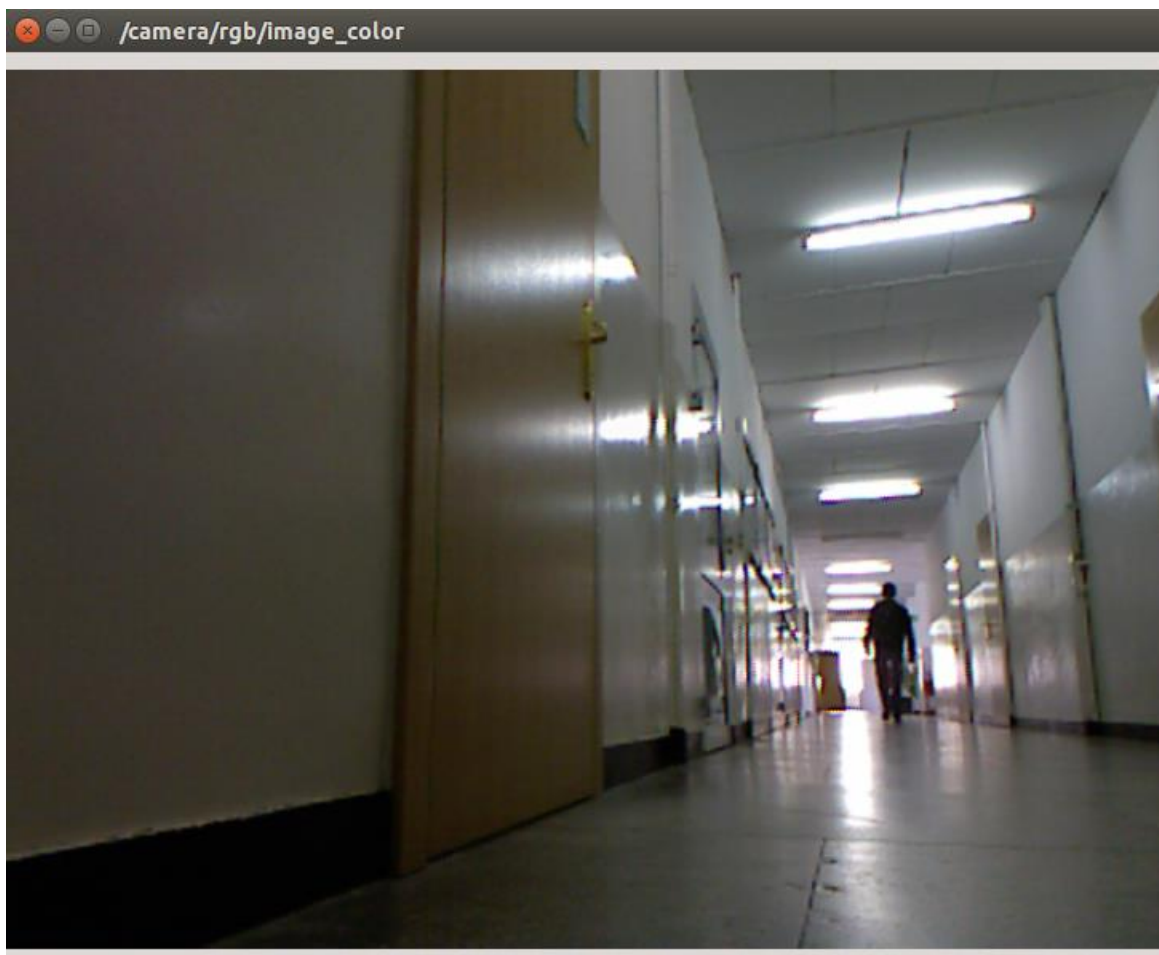


Rys. 1 Widok na drzwi zamknięte o nierównym profilu. Choć same drzwi są płaskie, to framuga znajdująca się na innym poziomie niż drzwi sprawia że linię trudno wykryć gdyż zlewają się one ze sobą

W tym widoku drzwi nie zostały zauważone. Dzieje się tak ze względu na framugę która znajduje się na innym poziomie niż same drzwi. W takim przekroju poprzecznym nawet ludzkie oko miałoby problemy z rozpoznaniem zarysu drzwi.



Rys. 1 Zamknięte drzwi o równym profilu



Rys. 1 Widok na zamknięte drzwi o równym profilu

Drzwi o równym profilu zostały zauważone.

Sformalizowane testy przeprowadzono w odległościach 1m – 2.5m pomiędzy środkiem drzwi, a czujnikiem Kinect. Dla tego zakresu odległości testowano program dla trzech różnych kątów ustawienia Kinecta w stosunku do drzwi (45, 90 i 135). Kąt 0 odpowiada czujnikowi Kinect ustawionemu równolegle do ściany skierowanemu na drzwi od strony zawiasów, kąt 90, Kinectowi patrzącemu prostopadle na drzwi, a kąt 180 Kinectowi ustawionemu równolegle do ściany, skierowanego na drzwi od strony klamki. Mówiąc inaczej, oznaczenie kąta ustawienia czujnika Kinect pokrywa się z kątem otwarcia drzwi jeśli chodzi o kierunek katowy. Natomiast punkt wokół którego obracany jest Kinect znajduje się na środku zamkniętych drzwi, a punkt wokół którego obracane są drzwi znajduje się na zawiasach.

Eksperymenty przeprowadzono sprawdzając liczbę prawidłowo wykrytych drzwi w danym zakresie kątów Kinecta i odległości od drzwi. Rubryka „Prawidłowo wykryte linie” zawiera informację o procentowej ilości prawidłowo wykrytych linii. Poprawność przypisania linii weryfikowano wizualnie i tam gdzie pojawiały się zastrzeżenia zaznaczono w rubryce „Uwagi”.

Kinect pod kątem 90 stopni.

Tabela 1. Wyniki dla czujnika Kinect pod kątem 90 stopni w odległości 2.5m od drzwi

Kąt drzwi [stopnie]	Prawidłowo wykryte linie [%]
0	43
30	90
60	86
90	27
120	85

Tabela 2. Wyniki dla czujnika Kinect pod kątem 90 stopni w odległości 2.0m od drzwi

Kąt drzwi [stopnie]	Prawidłowo wykryte linie [%]
0	25
30	84
60	14
90	3

120	1
-----	---

Tabela 3. Wyniki dla czujnika Kinect pod kątem 90 stopni w odległości 1.5m od drzwi

Kąt drzwi [stopnie]	Prawidłowo wykryte linie [%]
0	33
30	81
60	98
90	36
120	0

Tabela 4. Wyniki dla czujnika Kinect pod kątem 90 stopni w odległości 1.0m od drzwi

Kąt drzwi [stopnie]	Prawidłowo wykryte linie [%]	Uwagi
0	95	
30	97	Niewidoczna linia ściany
60	97	Niewidoczna linia ściany
90	0	Nieprawidłowo przypisana linia drzwi
120	0	Nieprawidłowo przypisana linia drzwi

Kinect pod kątem 135 stopni

Tabela 5. Wyniki dla czujnika Kinect pod kątem 135 stopni w odległości 2.0m od drzwi

Kąt drzwi [stopnie]	Prawidłowo wykryte linie [%]	Uwagi
0	10	
30	91	
60	93	
90	97	
120	12	Drzwi nie mieszczą się w polu widzenia

Tabela 6. Wyniki dla czujnika Kinect pod kątem 135 stopni w odległości 1.5m od drzwi

Kąt drzwi [stopnie]	Prawidłowo wykryte linie [%]	Uwagi
0	0	Linie niewyraźne ze względu na profil drzwi wystających z framugi
30	93	
60	97	
90	98	
120	97	

Tabela 7. Wyniki dla czujnika Kinect pod kątem 135 stopni w odległości 1.0m od drzwi

Kąt drzwi [stopnie]	Prawidłowo wykryte linie [%]	Uwagi
0	97	
30	92	Nie wykryto linii ściany ponieważ nie mieści się w obrazie
60	42	Nie wykryto linii ściany ponieważ nie mieści się w obrazie
90	0	Drzwi za blisko czujnika, obraz pusty
120	0	Drzwi za blisko czujnika, obraz pusty

Kinect pod kątem 45 stopni.

Tabela 8. Wyniki dla czujnika Kinect pod kątem 45 stopni w odległości 2.5m od drzwi

Kąt drzwi [stopnie]	Prawidłowo wykryte linie [%]	Uwagi
0	31	
30	5	
60	0	
90	27	
120	9	Drzwi daleko od czujnika, obraz bardzo niewyraźny

Tabela 9. Wyniki dla czujnika Kinect pod kątem 45 stopni w odległości 2.0m od drzwi

Kąt drzwi [stopnie]	Prawidłowo wykryte linie [%]	Uwagi
0	5	
30	1	
60	95	
90	98	
120	92	Drzwi wykryte ale czasami nieprawidłowo przypisane

Tabela 10. Wyniki dla czujnika Kinect pod kątem 45 stopni w odległości 1.5m od drzwi

Kąt drzwi [stopnie]	Prawidłowo wykryte linie [%]	Uwagi
0	3	
30	13	
60	95	Drzwi wykryte prawidłowo ale nie widać fragmentu ściany
90	98	Drzwi wykryte prawidłowo ale nie widać fragmentu ściany
120	98	Drzwi wykryte prawidłowo ale nie widać fragmentu ściany

Tabela 11. Wyniki dla czujnika Kinect pod kątem 45 stopni w odległości 1.0m od drzwi

Kąt drzwi [stopnie]	Prawidłowo wykryte linie [%]	Uwagi
0	97	
30	55	
60	0	Drzwi za blisko czujnika, obraz pusty
90	0	Drzwi za blisko czujnika, obraz pusty
120	0	Drzwi za blisko czujnika, obraz pusty

Z powyższych tabel (1-11) wynika że drzwi są najgorzej wykrywane pod kątem zbliżonym do tego pod którym ustawiony jest Kinect. Na przykład, przy Kinecie ustawionym pod kątem 135 stopni i drzwiach otartych od kątem 60 stopni (75 stopni różnicy), w odległości 1.5m wykrywalność wynosi 97%. Z drugiej strony wykrywalność przy kącie Kinecta 45 stopni i kącie drzwi 30 stopni (różnica 15 stopni) z odległości 1.5 metra wynosi 13%. W przypadku kąta drzwi zbliżonego do kąta patrzenia Kinecta kropki światła strukturalnego są od siebie zbyt oddalone, więc na wynikowym obrazie przekroju poprzecznego punkty są zbyt rozrzucone by połączyć je w linię.

Drzwi całkowicie zamknięte (0 stopni) są dobrze wykrywane w małej odległości od czujnika. Wynika to z faktu że dokładność czujnika spada wraz z odległością i dla drzwi wystających z framugi stabilna linia nie jest wykryta przy zmniejszonej dokładności.

W odległości 2.5 metra, drzwi wykrywane rzadko dla wszystkich kątów (dla kąta 135 stopni była to zerowa wykrywalność dla wszystkich kątów otwarcia drzwi tak więc zdecydowano nie umieścić tabeli dla tego kąta i tej odległości). Zalecana odległość maksymalna to 2m. Dla Kinecta oddalonego na dużą odległość drzwi wykrywane są rzadko, ponieważ obraz staje się za bardzo zaszumionym, a punkty zbyt rozrzucone żeby połączyć je w linię.

Gdy Kinect zbliżony jest za bardzo do drzwi, nie mieszczą się one w polu widzenia i nie są wykrywane (wyniki w tabeli 11 dla kątów 60, 90, 120).

10. Testy – rozpoznawanie sylwetek ludzkich

Rozpoznawanie sylwetek ludzkich z użyciem biblioteki OpenNI NITE wypadło pomyślnie. Ponieważ ta część programu nie posiada własnej wizualizacji, zostanie opisowo przedstawiona ocena działania.

W większości przypadków program nie miał żadnych trudności z rozpoznaniem sylwetek ludzkich. Zgodnie z opisem dystrybutora oprogramowania zaobserwowano następujące problemy:

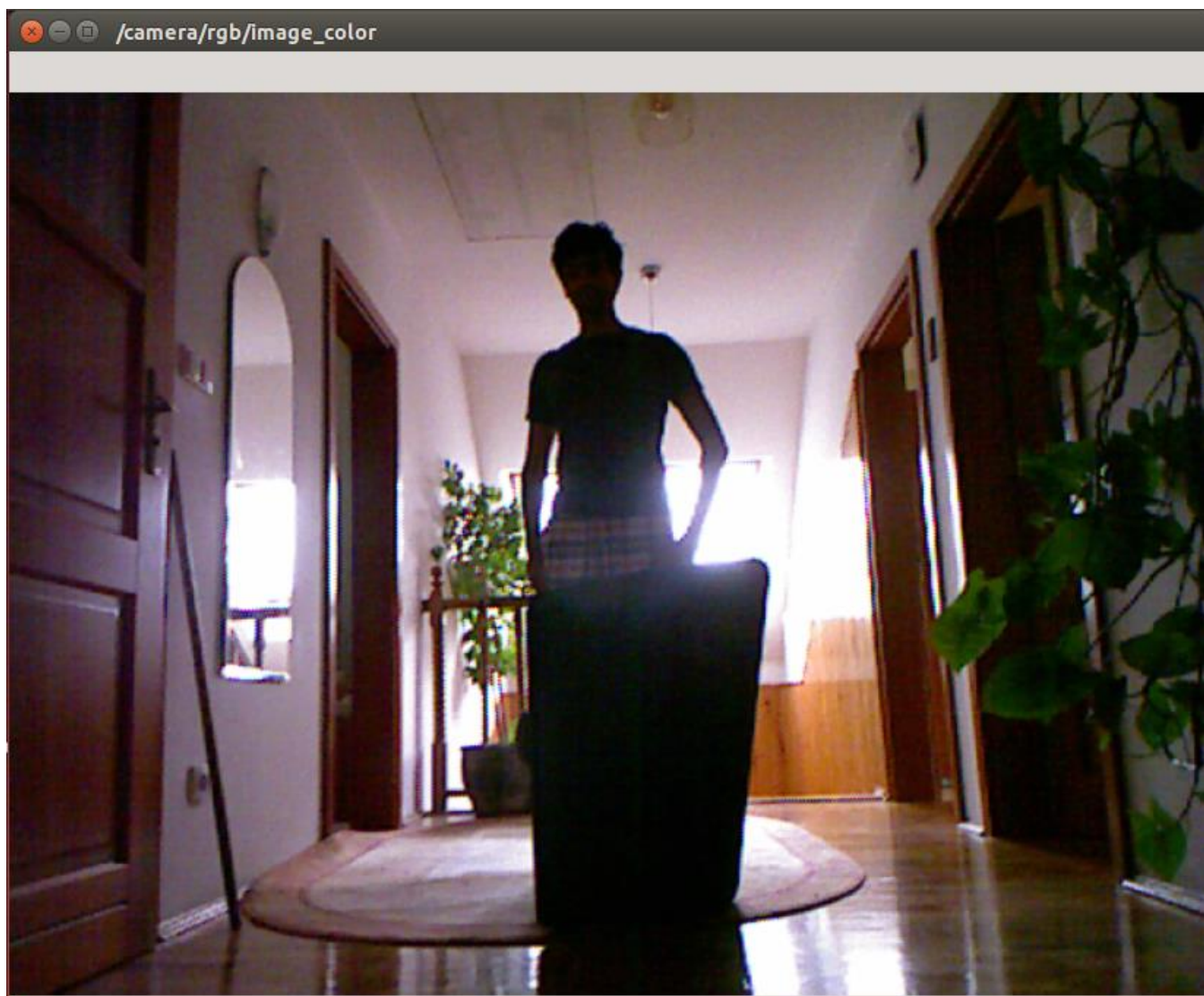
- Jeśli postać ludzka porusza się w bliskim sąsiedztwie jakiegoś obiektu to po zniknięciu użytkownika z pola widzenia ten obiekt może nadal być uznawany za postać ludzką.
- Jeżeli dwóch użytkowników porusza się dotykając, to mogą być uznani za jedną sylwetkę ludzką.
- Poruszanie czujnikiem Kinect zaburzało rozpoznanie sylwetek ludzkich.

Zauważono ponadto, że najdłuższym zakresem w którym sylwetki były rozpoznawane było 5m, pomimo że mapa głębi miała dalszy zasięg. Zdaje się to być wbudowanym ograniczeniem biblioteki.

Sylwetki ludzkie były rozpoznawane jako oddzielne kiedy stały oddzielnie w chwili wykrycia i przemieściły się w swoją stronę później. Natomiast jeśli sylwetki weszły w pole widzenia kamery zbliżone do siebie to były wykryte jako jeden użytkownik. Z tych obserwacji wynika że przypisanie obiektu jako sylwetki ludzkiej w dużej mierze zależy od sposobu poruszania się.

W tabelach 12-14 przedstawiono wyniki testów rozpoznawania postaci ludzkie w różnych odległościach i o różnym stopniu przesłonięcia. Testy wykonano w zakresie odległości 2m – 3m. W odległości mniejszej niż 2m cała sylwetka ludzka nie mieściła się w polu widzenia czujnika.

Przykładową scenę testów z sylwetką ludzką odsłoniętą w 50% pokazano na rys. 32.



Rys. 1 Przykładowa scena wykorzystana w testowaniu modułu wykrywania sylwetek ludzkich.
Sylwetka przykryta jest w 50%

Tabela 12 Wykrycie sylwetki ludzkiej dla odległości 2.0m od czujnika

Widoczność sylwetki	Sylwetka porusza się	Sylwetka nie porusza się
100%	Jest wykryta	Nie jest wykryta
75%	Jest wykryta	Nie jest wykryta
50%	Jest wykryta	Nie jest wykryta
25%	Nie jest wykryta	Nie jest wykryta

Tabela 13 Wykrycie sylwetki ludzkiej dla odległości 2.5m od czujnika

Widoczność sylwetki	Sylwetka porusza się	Sylwetka nie porusza się
100%	Jest wykryta	Nie jest wykryta
75%	Jest wykryta	Nie jest wykryta
50%	Jest wykryta	Nie jest wykryta
25%	Nie jest wykryta	Nie jest wykryta

Tabela 14 Wykrycie sylwetki ludzkiej dla odległości 2m od czujnika

Widoczność sylwetki	Sylwetka porusza się	Sylwetka nie porusza się
100%	Jest wykryta	Nie jest wykryta
75%	Jest wykryta	Nie jest wykryta
50%	Nie jest wykryta	Nie jest wykryta
25%	Nie jest wykryta	Nie jest wykryta

Sylwetka nie została wykryta w żadnym w przypadków w którym nie poruszała się. W każdej odległości poruszająca sylwetka musiała być widoczna w więcej niż 25%. Dla wartości widoczności sylwetki 25% sylwetka nie została wykryta na żadnej odległości. Wraz ze zwiększającą się odległością większy procent sylwetki musiał być widoczny żeby została wykryta.

11. Wnioski

Przetestowano programy do rozpoznawania drzwi i ludzi. Oba programy w większości sytuacji działały prawidłowo.

Program do rozpoznawania drzwi został przetestowany dla następujących parametrów: odległości czujnika Kinect od drzwi, kąta ustawienia czujnika Kinect względem zamkniętych drzwi oraz kąta otwarcia drzwi. Zauważono, że czynniki wpływającym na wykrywalność drzwi są następujące: odległość czujnika od drzwi i kąt otwarcia drzwi względem kąta ustawienia czujnika.

Żeby program działał prawidłowo czujnik Kinect nie może znajdować się bliżej drzwi niż na odległość 1.5m. Kiedy czujnik znajduje się bliżej, część drzwi nie jest objęta polem widzenia czujnika. Nawet jeżeli całe drzwi zmieszczą się w polu widzenia, czasami nie zmieści się fragment ściany. W tym przypadku nie można obliczyć kąta otwarcia drzwi.

W odległości większej niż 2m rozpoznawalność drzwi również spadała. Działo się tak ze względu na niską precyzję danych czujnika Kinect na takiej odległości. W tej sytuacji przekrój poprzeczny stawał się zbyt niewyraźny, aby wykrywać na nim linie o stabilnych właściwościach.

Rozpoznawalność drzwi również była niska, gdy kąt otwarcia drzwi był bliski kątowi ustawienia Kinecta. Powodem tego jest mała liczba kropek światła strukturalnego padająca na płaski obiekt skierowany równolegle do promieni lasera. Ze względu na mniejszą liczbę punktów, które zostały przepuszczone przez filtr, linia przekroju poprzecznego jest bardziej zaszumiona. Na takim przekroju działanie algorytmu wykrywania linii Hough'a jest utrudnione.

Moduł rozpoznawania sylwetek ludzkich został przetestowany z uwzględnieniem trzech zmiennych: odległości postaci ludzkiej od czujnika, procentowej widoczności (stopnia przysłonięcia) i informacji o ruchu postaci.

Postaci które nie poruszały się nie zostały wykryte w żadnym przypadku. Postacie poruszające były wykrywane nawet kiedy były częściowo przesłonięte. Sylwetki znajdujące się bliżej czujnika były wykrywane przy mniejszej widoczności niż sylwetki znajdujące się dalej czujnika.

Pomimo ograniczeń, oba moduły wykazały zadowalającą skuteczność w wykrywaniu drzwi i sylwetek ludzkich.

12. Podsumowanie

Praca została zakończona pomyślnie. Udało się opracować programy do rozpoznawania drzwi i sylwetek ludzkich za pomocą czujnika Kinect.

W celu wykonania tego zadania zapoznano się z budową czujnika Kinect i jego możliwościami i ograniczeniami. Dzięki temu można było opracować algorytmy które działają z dużym powodzeniem w środowisku korytarza wydziału Mechatroniki.

Do wykrywania drzwi wykorzystano transformację Hough'a wykonaną na przekroju poprzecznym chmury punktów dostarczanej przez węzeł camera_nodelet_manager.

Do wykrywania i śledzenia sylwetek ludzkich użyto gotowej biblioteki OpenNI NITE.

Metoda wykrywania drzwi jest prostsza niż inne propozycje znalezione w literaturze. Dzięki temu nie posiada niektórych ograniczeń które mają inne rozwiązania. Drzwi wykrywane są w pozycji zamkniętej i otwartej. Poruszanie drzwiami nie jest wymagane w celu ich wykrycia. Drzwi wykrywane są w pozycji statycznej i w ruchu. Drzwi nie muszą mieścić się w polu widzenia czujnika na całej wysokości. Określenie linii drzwi i ściany pozwala na łatwe obliczenie kąta otwarcia drzwi. Dzięki temu robot może podjąć decyzję o próbie przejechania przez nie.

Słabością metody zaproponowanej w tej pracy jest ograniczony zakres odległości czujnika od drzwi. Nie ma też możliwości wykrywania drzwi o rzeźbionym profilu lub z elementami szklanymi.

Moduł wykrywania sylwetek ludzkich jest bardzo skuteczny. Jego wadą jest konieczność poruszania się sylwetki, jeśli ma być wykryta. Z drugiej strony jest on w stanie wykryć sylwetki częściowo przysłonięte oraz wiele sylwetek poruszających się blisko siebie.

Zagadnienia rozpoznawania sylwetek ludzkich były i będą podejmowane wielokrotnie. Są one kluczowe dla skutecznej nawigacji robota w budynkach zajmowanych przez ludzi.

13. Bibliografia

- [1] Borgsen, Sebastian Meyer Zu, et al. “Automated Door Detection with a 3D-Sensor.” 2014 Canadian Conference on Computer and Robot Vision, 2014.
- [2] Burrus, Nicolas, et al. Hacking the Kinect: New York, NY, Apress, 2012.
- [3] Cicirelli, Grazia, et al. “Door Detection in Images Based on Learning by Components.” Intelligent Robots and Computer Vision XX: Algorithms, Techniques, and Active Vision, 2001.
- [4] Dokumentacja biblioteki OpenCV, <http://www.opencv.org>
- [5] Dokumentacja biblioteki PCL, <http://www.pointclouds.org>
- [6] Dokumentacja systemu operacyjnego ROS, <http://www.ros.org>
- [7] Dokumentacja środowiska OpenNI, <http://www.openni.ru>
- [8] Fernnndez-Caramacs, C., et al. “A Real-Time Door Detection System for Domestic Robotic Navigation.” Journal of Intelligent & Robotic Systems, vol. 76, no. 1, 2013, pp. 119–136.
- [9] Hensler, Jens, et al. “Improved Door Detection Fusing Camera and Laser Rangefinder Data with AdaBoosting.” Communications in Computer and Information Science Agents and Artificial Intelligence, 2011, pp. 39–48.
- [10] Hu, Min-Chun, et al. “Efficient Human Detection in Crowded Environment.” Multimedia Systems, vol. 21, no. 2, 2014, pp. 177–187.
- [11] Liu, Jun, et al. “An Ultra-Fast Human Detection Method for Color-Depth Camera.” Journal of Visual Communication and Image Representation, Academic Press, Inc., dl.acm.org/citation.cfm?id=2823591. Accessed 28 May 2017.
- [12] Malik, U. A., et al. “A Self-Organizing Neural Scheme for Road Detection in Varied Environments.” The 2011 International Joint Conference on Neural Networks, 2011.
- [13] Murillo, A C, et al. “Visual Door Detection Integrating Appearance and Shape Cues.” Robotics and Autonomous Systems, vol. 56, 2008, pp. 512–521.,
- [14] Nguyen, Duc Thanh, et al. “Human Detection from Images and Videos: A Survey.” Pattern Recognition, vol. 51, 2016, pp. 148–175.
- [15] O’Kane, Jason M. A Gentle Introduction to ROS. Columbia, SC.
- [16] Sekkal, Rafiq, et al. “Simple Monocular Door Detection and Tracking.” 2013 IEEE International Conference on Image Processing, 2013.
- [17] Tian, Yingli, et al. “Computer Vision-Based Door Detection for Accessibility of Unfamiliar

Environments to Blind Persons.” Lecture Notes in Computer Science Computers Helping People with Special Needs, 2010, pp. 263–270.

[18] Yu, Pengxian, and Yaxun Wei. “Human Detection and Tracking.” Proceedings of SPIE, vol. 8918.

[19] Zhang, Hai-Qiang, et al. “Door Detection without Apriori Color Knowledge.” Proceedings of the 17th World Congress The International Federation of Automatic Control, 2008, pp. 9192–9196.,

14. Spis rysunków

Rys. 1 Ogólny schemat algorytmu użytego w projekcie	11
Rys. 2 Czujnik Kinect i jego elementy	12
Rys. 3 Światło strukturalne z czujnika Kinect widziane przez kamerę na światło podczerwone	13
Rys. 4 Struktura programu rozpoznawania drzwi (obraz zawinięty)	16
Rys. 5 Wizualizacja rezultatów zastosowania dylacji.	18
Rys. 6 Przekrój poprzeczny użyty w projekcie przed zastosowaniem operacji dylacji i erozji (zamknięcia morfologicznego). Widoczne są nieciągłości i szum.	18
Rys. 7 Operacja dylacji wykonana na obrazie z programu do wykrywania drzwi.	19
Rys. 8 Wizualizacja rezultatów zastosowania erozji	20
Rys. 9 Operacja zamknięcia morfologicznego wykonana przez program do rozpoznawania drzwi.	21
Rys. 10 Wykorzystanie transformacji Hough'a w module rozpoznawania drzwi. Linia różowa reprezentuje drzwi a linia niebieska ścianę. Na tym obrazie linie zostały wyselekcjonowane przez funkcje węzła door_detection.	23
Rys. 11 Przestrzeń xy i mc	24
Rys. 12 Reprezentacja dwóch punktów w przestrzeni obrazu i przestrzeni Hough'a	24
Rys. 13 Współrzędne r i θ	25
Rys. 14 Przekształcenie z przestrzeni xy do przestrzeni $r\theta$	25
Rys. 15 Wyniki zastosowania funkcji HoughLinesP na obrazie przekroju poprzecznego uzyskanego podczas realizacji tego projektu.	27
Rys. 16 Przykładowy obraz z kamery RGB	28
Rys. 17 Działanie funkcji bounding_box	32
Rys. 18 Półotwarte drzwi widziane od strony otwartej	35
Rys. 19 Widok na półotwarte drzwi od strony otwartej	36
Rys. 20 Półotwarte drzwi widziane od strony otwartej, bez widoku na ścianę za nimi	37
Rys. 21 Widok na półotwarte drzwi od strony otwartej	38
Rys. 22 Widok na lekko otwarte drzwi od strony otwartej.	39
Rys. 23 Widok na lekko otwarte drzwi od strony otwartej	40
Rys. 24 Drzwi lekko otwarte widziane od strony zamkniętej	41
Rys. 25 Widok na drzwi lekko otwarte widziane od strony zamkniętej	42
	60

Rys. 26 Drzwi otwarte pod kątem prostym widziane od strony otwartej	43
Rys. 27 Widok na drzwi otwarte pod kątem prostym od strony otwartej	44
Rys. 28 Drzwi zamknięte o nierównym profilu	45
Rys. 29 Widok na drzwi zamknięte o nierównym profilu. Choć same drzwi są płaskie, to framuga znajdująca się na innym poziomie niż drzwi sprawia że linię ciężko wykryć gdyż zlewają się one ze sobą	46
Rys. 30 Zamknięte drzwi o równym profilu	47
Rys. 31 Widok na zamknięta drzwi o równym profilu	48