Podstawy Teleinformatyki
Rozpoznawanie obrazu z gry w Warcaby oraz wizualizacja stanu gry na komputerze

Marcin Orczyk Natalia Popielarz Piotr Wołyński

14 czerwca 2018

Marcin Orczyk	126804	marcin.orczyk5@gmail.com
Natalia Popielarz	126803	natalia.popielarz@student.put.poznan.pl
Piotr Wołyński	126832	piotr.wolynski@student.put.poznan.pl

Spis treści

1 Charakterystyka projektu

1.1 Opis projektu

Projekt zakłada stworzenie programu do cyfrowej wizualizacji stanu gry w Warcaby, którego zadaniem będzie przeniesienie stanu planszy fizycznej do komputera oraz weryfikacja poprawności ruchów wykonywanych przez graczy. Nad planszą do gry w Warcaby umieszczona będzie kamera. Moduł do rozpoznawania obrazu z kamery wyszuka pozycje pionków i przekaże je do modułu weryfikacji, który sprawdzi, czy wykonano poprawny ruch. Jeśli tak, dane o ruchu przekazywane zostaną do modułu wizualizacji, w przeciwnym wypadku gracze otrzymają informację o wykonaniu błędnego ruchu. Moduł wizualizacji będzie przedstawiać aktualny, poprawny stan planszy fizycznej w postaci cyfrowej.
Realizowany program pozwoli na zrewolucjonizowanie transmisji meczów w Warcaby. Dzięki programowi będzie istniała możliwość transmisji rozgrywek na żywo dla osób, które posiadają łącze internetowe o niskiej przepustowości, limity transmisji danych bądź urządzenia mobilne o małej wydajności (np. z aplikacjami napisanymi w języku Java). Dzięki temu oprogramowaniu, osoby takie będą mogły w czasie rzeczywistym (o ile transmisję danych w sieciach pakietowych opartych o protokoły Transmission Control Protocol/User Datagram Protocol /Internet Protocol można nazwać transmisją w czasie rzeczywistym) śledzić postępy ulubionych zawodników. Program cechuje się wyjątkowo niskim zapotrzebowaniem na zasoby sprzętowe i sieciowe, ponieważ umożliwia przesyłanie jedynie informacji o planszy (takich jak rozmieszczenie pionków czy informacje o pionkach takich jak ich kolor i rodzaj — czy jest to zwykły pionek, czy damka). Nie jest wymagana transmisja całego obrazu, co jest realizowane w klasycznych systemach telewizji internetowej. Dzięki temu nie jest konieczne posiadanie szybkiego dostępu do Internetu ani nie trzeba angażować procesora komputera w dekodowanie sygnału audio i wideo.

1.2 Opis gry

Program umożliwia wizualizację planszy, na której rozgrywany jest mecz w Warcaby. Na potrzeby programu, klasyczne zasady gry zostały uogólnione tak, aby w grę mogły grać jednocześnie trzy osoby. Dwie spośród osób wchodzących w skład drużyny rozgrywają między sobą tradycyjną partię gry. Zadaniem trzeciej osoby jest podtrzymanie kamery tak, aby znalazła się ona nad planszą, a krawędzie planszy były równoległe do krawędzi obrazu. Dodatkowym celem grającej pary osób jest zakończenie gry, zanim osoba podtrzymująca kamerę nie będzie mogła dalej wykonywać swojego zadania z powodu bólu ręki.W przypadku braku trzeciej osoby trzymającej kamerę niezbędne jest posiadanie statywu lub innego mechanizmu umożliwiającego utrzymanie kamery nad planszą.

1.3 Uzasadnienie wyboru tematu

Temat projektu został wybrany zgodnie przez wszystkich członków zespołu, głównie ze względu na wspólne zainteresowanie grami komputerowymi oraz planszowymi. Projekt ten łączy w sobie tematykę gier, przetwarzania obrazów, programowania reguł i zasad oraz integracji różnych technologii takich jak C#, Python

oraz istniejące aplikacje webowe. Dzięki realizacji programu, członkowie zespołu mogli znacznie rozwinąć nabytą na innych zajęciach wiedzę związaną z przetwarzaniem i analizowaniem obrazu pochodzącego z internetowych kamer RGB (w szczególności tak zwanego obrazu ruchomego, składającego się z ciągłego strumienia klatek obrazu).

1.4 Moduły projektu i podział prac

Z uwagi na złożoność projektu, konieczne jest wyodrębnienie na etapie projektowania modułów, z których powinna składać się końcowa aplikacja. Projekt będzie składać się z następujących modułów:

- Moduł odpowiedzialny za zdigitalizowanie stanu planszy fizycznej.
- Moduł odpowiedzialny za weryfikację logiki gry.
- Moduł odpowiedzialny za wyświetlenie gry w aplikacji webowej.[1]

1.5 Narzędzia

Powodzenie realizacji projektu w dużej mierze zależy od zespołu, który go tworzy, jak również od narzędzi, którymi będą posługiwać się programiści oraz wszystkie osoby zaangażowane w tworzenie aplikacji. Ze względu na preferencje członków zespołu oraz ich indywidualne doświadczenie wybrane zostały następujące narzędzia:

- Język programowania C#.
  - Środowisko Visual Studio 2017.
  - Biblioteka „OpenCV".
- Język programowania Python.
  - Biblioteka „Selenium".
- System kontroli wersji Github.com.
- Edytor tekstu Overleaf języka TeX (Dokumentacja).

2 Wymagania projektu

Przed przystąpieniem do prac nad projektem konieczne jest określenie wymagań, jakie powinna spełniać finalna aplikacja.

2.1 Wymagania funkcjonalne

Wymagania funkcjonalne skupiają się na funkcjonalnościach oferowanych przez aplikację takich jak:

- Cyfrowa wizualizacja rzeczywistej rozgrywki.
- Weryfikacja poprawności ruchów gracza.

[1] Uwaga: Do wizualizacji zostanie wykorzystana gotowa aplikacja www.kurnik.pl

2.2 Wymagania niefunkcjonalne

Wymagania niefunkcjonalne określają właściwości całej aplikacji. Wynikają z potrzeb użytkownika, a także preferencji twórców. Aby aplikacja działała poprawnie konieczne jest spełnienie określonych warunków. Niezbędne wyposażenie użytkownika to:

- Kamera internetowa.
- Plansza do gry w Warcaby klasyczne (8 x 8) oraz pionki w 4 różnych kolorach (niedozwolone kolory biały i czarny).
- System operacyjny Windows 10.
- Monitor o rozdzielczości minimalnej 1080 x 1920.

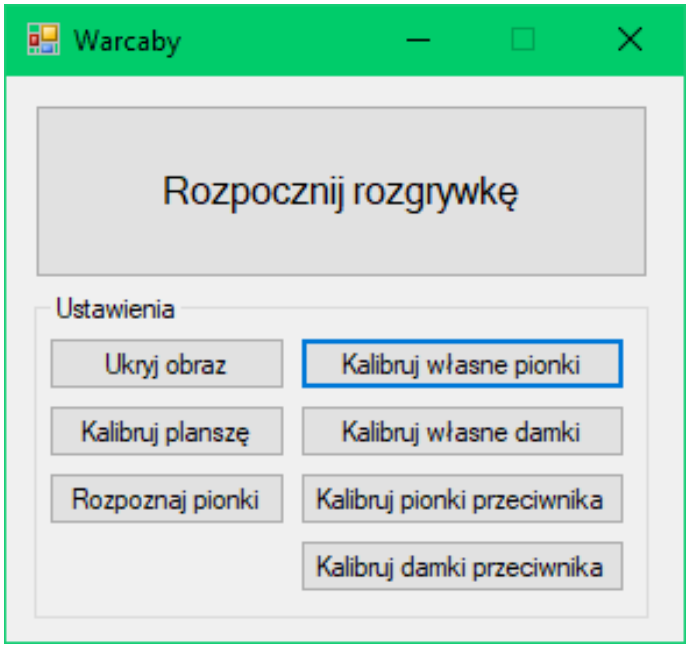
3 Architektura rozwiązania

Aplikacja składa się z trzech modułów. Moduł wizyjny przy pomocy kamery odczytuje pozycje pionków na planszy i przekazuje stan planszy do modułu walidacyjnego. Po pozytywnej walidacji moduł wizyjny przekazuje dany ruch do modułu wyświetlającego rozgrywkę.

4 Opis implementacji

Program został przygotowany z wykorzystaniem programu Visual Studio, a większość testów została przeprowadzona na systemie Windows 10 (gdzie zostały wykonane zrzuty ekranu, wchodzące w skład niniejszej dokumentacji).

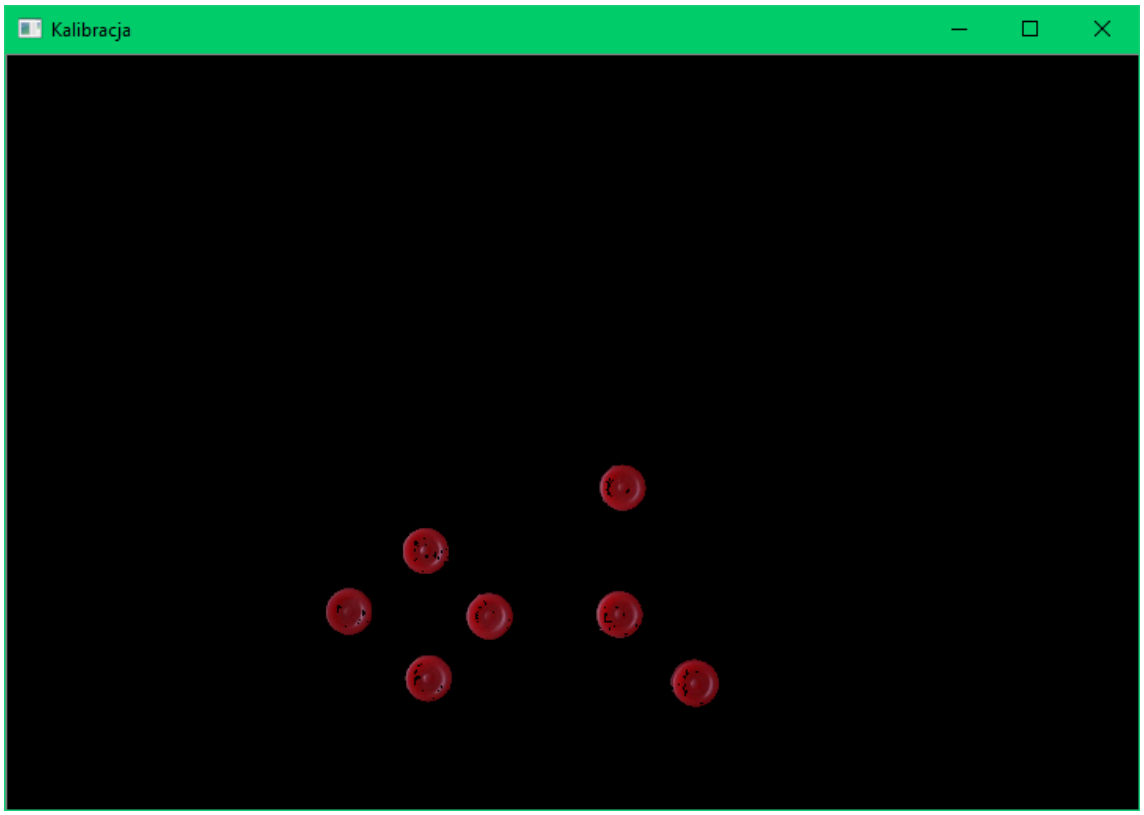
Po uruchomieniu aplikacji wyświetlane jest proste okno główne, przedstawione na rysunku 1, zawierające kilka przycisków.



Rys. 1: Główne okno programu.

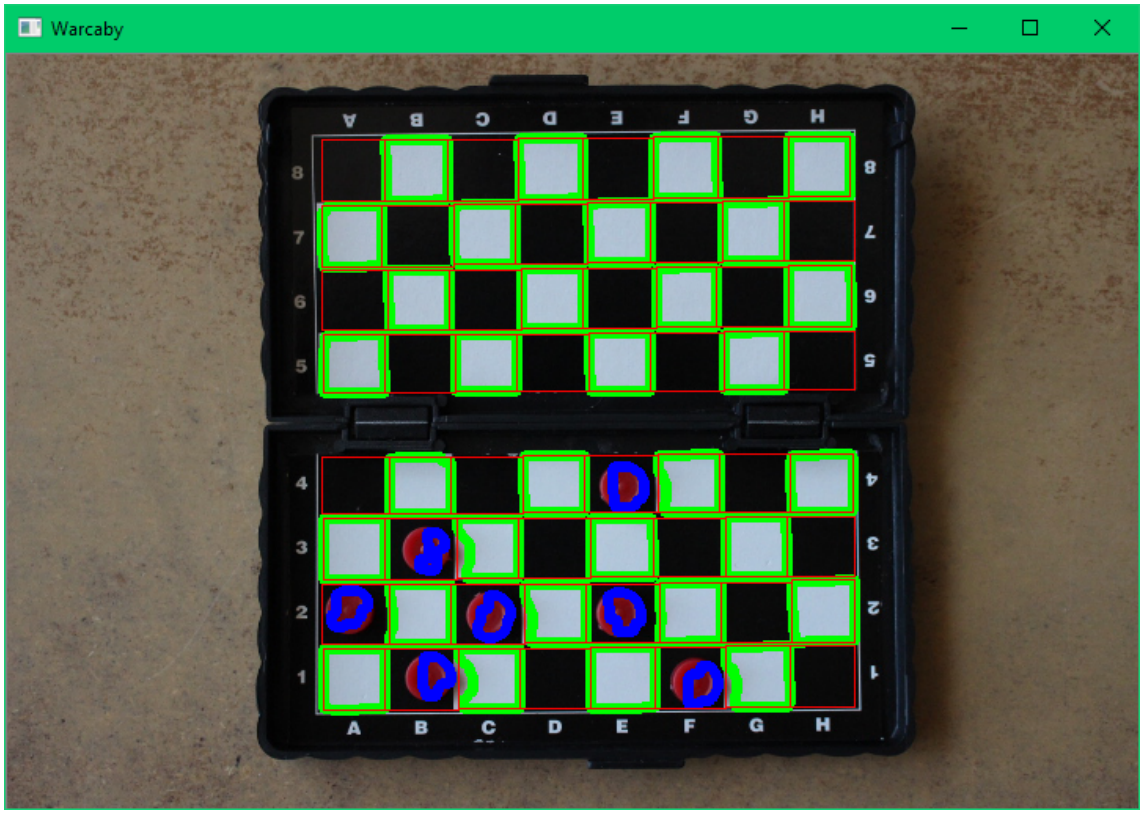
Najważniejszym przyciskiem, który, z powodu swojej wielkości, od razu rzuca się w oczy, jest przycisk służący do rozpoczęcia nowej rozgrywki. Po kliknięciu tego przycisku, w programie uruchamiana jest funkcja mająca za zadanie przygotować środowisko do wyświetlania stanu gry. Funkcja ta, między innymi, inicjalizuje moduł służący do sprawdzania poprawności ruchu oraz uruchamia dwa nowe procesy. W procesach tych rozpoczęte zostaje wykonywanie programu interpretera języka Python. W każdym z tych procesów uruchamiany jest jeden plik ze skryptem Pythona. Zadaniem skryptów jest uruchomienie przeglądarki Google Chrome, połączenie się z witryną kurnik.pl i wyświetlanie na niej rozgrywki. Po inicjalizacji następuje wejście do głównej pętli programu. W pętli tej najpierw następuje odczytanie z obrazu z kamery aktualnego stanu planszy i rozmieszczenia pionków. Następnie sprawdzana jest poprawność danych — czy zawartość planszy została odczytana poprawnie (nie wystąpiły problemy, takie jak np. zasłonięcie ręką planszy) oraz czy ruchy wykonane przez graczy były poprawne w sensie zasad gry. Ponieważ poprawne odczytanie obrazu potrafi sprawić wiele problemów, w celach testowych wyświetlane jest dodatkowe okienko, przedstawiające odczytany stan planszy, czyli rozmieszczenie pionków oraz ich kolory, przez co można określić, z jakiego rodzaju pionkiem program ma do czynienia (do którego gracza należy pionek oraz czy jest to zwykły pionek, czy damka).

Pod przyciskiem umożliwiającym rozpoczęcie analizowania rozgrywki, wyświetlona jest sekcja umożliwiająca skonfigurowanie programu. Na samej górze, po lewej stronie znajduje się przycisk o nazwie „Pokaż obraz” lub „Ukryj obraz” (nazwa zmienia się w zależności od stanu programu). Kliknięcie przycisku powoduje wyświetlenie okna zawierającego podgląd na żywo obrazu z kamery, np. taki jak na rysunku 2.



Rys. 2: Obraz po poprawnej kalibracji czerwonych pionków.

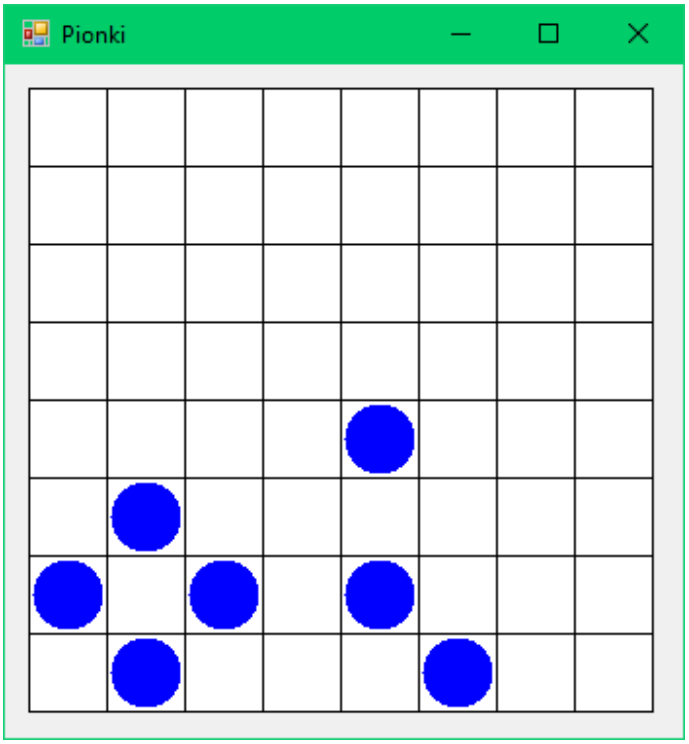
Na rysunku 3 zaznaczone są informacje o rozpoznanych konturach — różnymi kolorami zaznaczone są kontury poszczególnych pionków oraz pól białych. Poza tym, za pomocą czerwonych, cienkich linii wyświetlone zostają granice pól.



Rys. 3: Rozpoznawanie konturów.

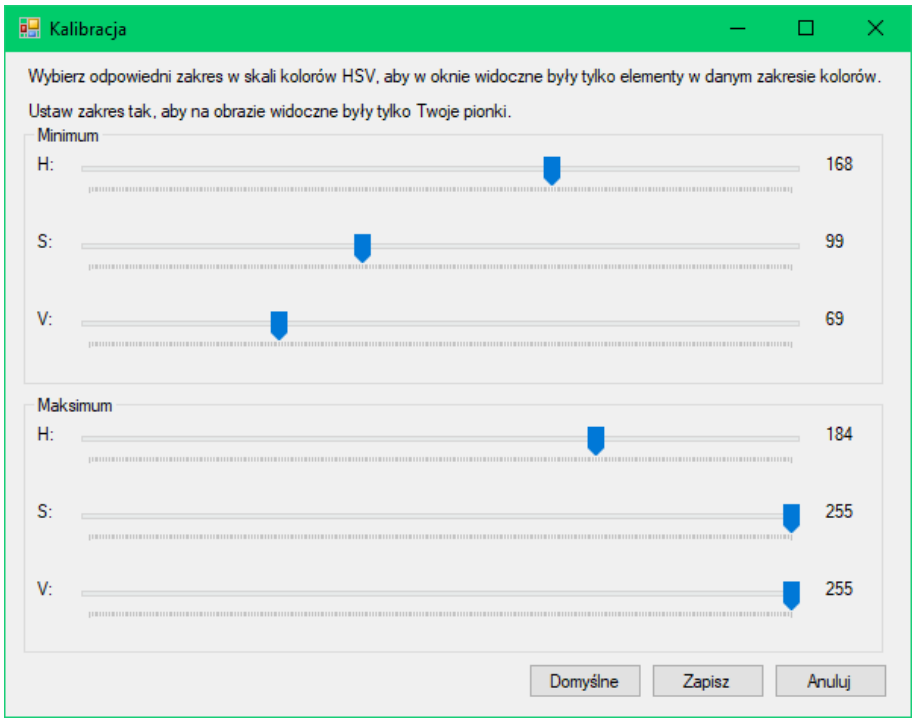
Granice te są obliczane na podstawie rozpoznanych konturów pól białych i są używane do wyznaczenia, na którym dokładnie polach znajdują się konkretne pionki. Specyfika biblioteki OpenCV powoduje, że zwykłe zamknięcie okna staje się czynnością trudną. Po kliknięciu przyciski „Zamknij", umieszczony w prawym górnym rogu okna, użytkownik klikający spodziewa się, że okno zostanie zamknięte. Tak się jednak nie dzieje. Okno wprawdzie rzeczywiście znika, jednak natychmiast pojawia się nowe, z taką samą zawartością. Dlatego, aby zamknąć okno programu, należy kliknąć przycisk Ukryj obraz, znajdujący się w oknie głównym.

Na dole lewej kolumny umieszczony został przycisk służący do rozpoznania pionków. Jeśli konfiguracja programu została przeprowadzona poprawnie i program jest w stanie wykryć pionki, po kliknięciu przycisku pojawi się okno takie jak na rysunku 4 przedstawiające aktualny stan planszy. Okno to było używane przez twórców programu do sprawdzenia poprawności implementacji algorytmów rozpoznawania obrazu, jednak jest również przydatne dla zwykłych użytkowników programu celem oceny poprawności konfiguracji.



Rys. 4: Okno rozpoznanych pionków.

W środkowym wierszu lewej kolumny przycisków znajduje się przycisk umożliwiający kalibrację planszy. Po jego kliknięciu wyświetlone zostaje okno służące do kalibracji parametrów programu, takie jak na rysunku 5.



Rys. 5: Okno kalibracji.

Przycisk ten ma takie samo działanie, jak cztery przyciski umieszczone w prawej kolumnie — ich zadaniem jest możliwość zmiany ustawień programu. Przyciski te służą odpowiednio do kalibracji planszy (mówiąc bardziej szczegółowo, do kalibracji białych pól na warcabnicy), do kalibracji pionków jednego z graczy, damek jednego z graczy, pionków drugiego z graczy oraz damek drugiego z graczy. Po kliknięciu jednego z przycisków pojawiają się dwa okna. W jednym z nich zostaje umieszczony obraz na żywo z kamery, z zastosowaniem wprowadzanych parametrów. Drugie okno zawiera sześć suwaków, służących do ustalenia zakresu kolorów HSV. Pierwsza grupa suwaków umożliwia wybranie dolnych zakresów parametrów, natomiast grupa położona poniżej umożliwia wybranie górnych zakresów parametrów. W oknie ponadto pojawia się komunikat, o tym, jakie dokładnie elementy należy skonfigurować. Zadaniem użytkownika jest takie ustawienie suwaków, aby na podglądzie obrazu z kamery wyświetlała się tylko konkretna grupa elementów (na przykład, pionki w jednym kolorze). W oknie konfiguracji dostępne są trzy przyciski. Pierwszy z nich (patrząc od lewej strony), umożliwia przywrócenie ustawień domyślnych. W takim przypadku dolny zakres parametrów ustawiany jest na zero, natomiast górny na wartość 255. Drugi przycisk umożliwia zastosowanie wybranych ustawień w programie oraz zamyka okna ustawień, oraz podglądu kamery, Ostatni przycisk, pozwala na anulowanie wprowadzonych ustawień i zamknięcie okien bez ich zapisywania.

Program umożliwia zapisanie ustawień konfiguracyjnych w pliku, który domyślnie pojawia się na pulpicie i posiada nazwę Ustawienia.dat. Plik ten jest plikiem binarnym i charakteryzuje się niewielkim zużyciem miejsca na dysku. W pliku zapisywane są kolejno zakresy kolorów (w skali HSV) dla poszczególnych rodzajów pionków oraz dla pól planszy. Plik ten jest zapisywany przy zamykaniu programu, natomiast wczytywany podczas uruchamiania aplikacji.

5 Instrukcja użytkowania aplikacji

Przed uruchomieniem programu, należy przygotować środowisko służące do gry. Należy przejść w miejsce, gdzie istnieje możliwość równomiernego oświetlenia planszy (na przykład do piwnicy, w której zainstalowane są odpowiednie źródła światła). Nie należy kierować żadnego źródła światła w stronę planszy — może to spowodować, że na planszy pojawią się odbicia, poza tym plansza będzie oświetlona nierównomiernie. W różnych miejscach planszy będzie panować różne natężenie oświetlenia, w efekcie czego elementy, które teoretycznie są w takim samym kolorze (na przykład, identyczne czerwone pionki), będą mieć inne odcienie kolorów lub nawet inne kolory, przez co kolory teoretycznie takich samych elementów zostaną rozpoznane jako znacząco różniące się, przez co może nie być możliwe prawidłowe analizowanie planszy przez program. Najlepszym rozwiązaniem jest zastosowanie silnego źródła światła i rozproszenie promieni świetlnych w pomieszczeniu. Wszelkie cienie czy inne problemy ze światłem mogą wpływać na niepoprawne działanie programu.

Do rozgrywki należy przygotować odpowiednią planszę oraz pionki. Wymagane jest, aby połowa pól na planszy (w klasycznych warcabach mających kolor biały) oraz pionki miały różne kolory, tak aby możliwe było rozpoznanie ich na podstawie kolorów. Wymagane jest więc, aby w kadrze kamery znalazły się elementy o co najmniej sześciu różnych kolorach — pola na warcabnicy, cztery kolory pionków (pionki obu graczy oraz należące do tych zawodników damki) oraz co najmniej jeden dodat-

kowy kolor, w którym będą pozostałe pola na warcabnicy oraz tło (na przykład blat stołu lub inny element, na którym została umieszczona plansza). Zalecanym sposobem przygotowania stanowiska jest zastosowanie planszy posiadającej pola białe i czarne, pionków w czterech różnych kolorach oraz czarnego tła. Przed uruchomieniem programu, na planszy należy rozmieścić (w dowolnym porządku) co najmniej po jednym pionku w każdym kolorze, a nad planszą należy umieścić kamerę RGB (tak, aby krawędzie planszy były prostopadłe do krawędzi kadrów obrazu). Po wykonaniu czynności, na komputerze można uruchomić prezentowane oprogramowanie w celu przeprowadzenia kalibracji oraz oficjalnego uruchomienia programu.

Kamerę nad planszą należy ustawić tak, aby krawędzie planszy były równoległe do krawędzi obrazu przechwytywanego z kamery. Kolejnym warunkiem jest aby w lewym górnym rogu planszy znajdowało się pole białe. Po uruchomieniu programu wyświetli się okno główne aplikacji. W oknie tym można dokonać konfiguracji programu, tak aby dostosować go do posiadanej kamery oraz do panujących aktualnie warunków oświetleniowych. Na początku, należy przeprowadzić kalibrację obrazu przez kliknięcie na przyciski „Kalibruj planszę", „Kalibruj własne pionki", „Kalibruj pionki przeciwnika" oraz „Kalibruj damki przeciwnika". Po kliknięciu każdego z tych przycisków wyświetlą się dwa okna — jedno zawiera podgląd obrazu z kamery, natomiast drugie suwaki umożliwiające zmianę parametrów. Zadaniem użytkownika jest takie ustawienie suwaków, aby na obrazie z kamery pozostały widoczne tylko wybrane elementy (pionki oraz damki, w przypadku kalibracji planszy białe pola na warcabnicy). Program umożliwia zapisanie wprowadzonych ustawień, dzięki czemu nie trzeba ich zmieniać po każdym uruchomieniu programu (jednak oczywiście, jeśli program zostanie ponownie uruchomiony w innych warunkach oświetleniowych, wtedy wybrane ustawienia mogą stać się nieaktualne, przez co może wystąpić konieczność ponownej konfiguracji programu).

W oknie konfiguracji istnieje też możliwość wyjścia z okna bez zapisywania konfiguracji lub przywrócenia ustawień domyślnych, jeśli użytkownik zgubi się podczas wprowadzania ustawień i nie będzie w stanie znaleźć prawidłowych wartości. Po poprawnej konfiguracji każdego z pięciu elementów, istnieje możliwość przetestowania wprowadzonych parametrów. W tym celu, należy kliknąć przycisk „Pokaż obraz". Zostanie wtedy wyświetlony podgląd z kamery z zaznaczonymi konturami rozpoznanych elementów. Aby zamknąć okno podglądu, należy kliknąć przycisk „Ukryj obraz" (jest to ten sam przycisk, który posłużył do otwarcia okna, jednak ze zmienionym opisem wyświetlanym na elemencie). Okna wyświetlane przez bibliotekę graficzną nie zamykają się przy próbie zamknięcia ich przyciskiem „X" znajdującym się w prawym górnym rogu ekranu. Można również wybrać i kliknąć przycisk „Rozpoznaj pionki", dzięki czemu powinno wyświetlić się okno wyświetlające zawartość planszy. Jeśli okno nie zostanie wyświetlone lub rozmieszczenie pionków nie zgadza się z rzeczywistym stanem planszy, oznacza to, że kalibracja nie została przeprowadzona poprawnie. W takim przypadku, należy powtórzyć proces kalibracji.

Jeśli wszystkie operacje powiodą się, należy przygotować planszę do rozgrywki. Na planszy należy rozmieścić pionki na czarnych polach tak, by na górze planszy znajdowały się pionki gracza rozpoczynającego rozgrywkę, a na dole pionki drugiego gracza. Należy ponownie sprawdzić, czy w lewym górnym rogu planszy znajduje się białe pole.

Dopiero teraz można kliknąć na duży, wyróżniający się przycisk „Rozpocznij rozgrywkę". Po chwili uruchomią się dwa okna przeglądarki Google Chrome, w któ-

rych wczytana zostanie strona www.kurnik.pl, służąca do wizualizacji rozgrywki. W oknach przeglądarki, aplikacja zaloguje się na dwa różne konta użytkownika we wspomnianym serwisie, po czym nastąpi połączenie wspomnianych dwóch użytkowników. Od tego momentu możliwa będzie gra w Warcaby na fizycznej planszy. Przesuwanie pionki zostaną rozpoznane przez program oraz wyświetlone w serwisie kurnik.pl.

6 Podsumowanie

Celem pracy było stworzenie zautomatyzowanego systemu umożliwiającego wizualizację rozgrywki w Warcaby. Zadanie udało się wykonać całkowicie. Istotnym i bardzo rozwijającym elementem naszej pracy była integracja wielu technologii takich jak programy napisane w językach C#, Python czy gotowych aplikacji webowych. Wbrew naszym przewidywaniom nie przysporzyło to większych problemów, co potwierdza czas integracji naszych modułów, który wyniósł 2.5 godziny. Elementem dodatkowym, który w przyszłości warto byłoby dodać, jest automatyczna konfiguracja doboru parametrów HSV progowania obrazu w celu prawidłowego rozpoznawania pionków oraz planszy.

6.1 Napotkane problemy

Realizowany program służy do analizy i rozpoznawania obrazu z kamery. Zadanie to wydaje się łatwe, zwłaszcza że użyta zostaje gotowa biblioteka OpenCV. Okazuje się jednak, że przetwarzanie obrazu jest procesem bardzo złożonym i skomplikowanym. Jednym z problemów jest wybór sprzętu. Stosowane powszechnie kamery internetowe RGB potrafią zapewnić obraz o jedynie przeciętnej jakości. Ponadto, same kamery mają nieszczególnie zadowalające parametry sprzętowe. Między innymi, podczas nagrywania obrazu w pomieszczeniu, gdzie jest niezbyt duże natężenie światła, na obrazie pojawia się wyraźny szum ISO. Do tego dochodzi fakt, że obraz przesyłany między kamerą a komputerem za pośrednictwem portu Universal Serial Bus (pol. Uniwersalna Magistrala Szeregowa) jest kodowany za pomocą algorytmów kompresji stratnych, z zastosowaniem niskich parametrów (np. niska wartość prędkości transmisji bitów).

Wszystko to powoduje, że obraz, który trafia do algorytmów programu, jest mocno zaszumiony. Powoduje to trudności z rozpoznawaniem obrazu — jeśli np. na planszy znajduje się pionek w jednolitym, czerwonym kolorze, w przetwarzanym obrazie sąsiednie piksele będą różnić się kolorem. W efecie czego, podczas procesu kalibracji nie da się ustawić parametrów tak, aby z obrazu wyciąć inne piksele niż piksele pionków. Na kalibrowanych pionkach pojawiają się czarne, wycięte piksele, które posiadają inny kolor niż rzeczywisty kolor pionka (z powodu szumów i strat związanych z kompresją).

Kolejnym problemem jest wysoki poziom abstrakcji bibliotek i automatyczna konfiguracja parametrów kamery. Ustawienia automatyczne są bardzo przydatną funkcjonalnością urządzeń optycznych, takich jak kamery czy aparaty cyfrowe. Dzięki nim, nie trzeba znać się zbyt dobrze na fotografii, ani, w przypadku posiadania takich umiejętności, nie trzeba poświęcać dużej ilości czasu na ustawienie wszelkich parametrów i można szybko wykonać zdjęcie. Fotografowie zapewne zgodzą się z twierdzeniem, że bardzo przydatna jest możliwość ręcznej zmiany parametrów

w używanym urządzeniu, dzięki czemu można zrobić udane zdjęcie, w przypadku gdy automatyka zawodzi lub ustawia parametry poprawne, lecz niechciane przez fotografa (jeśli np. fotograf chce uzyskać na zdjęciu specjalne efekty). Podczas tworzenia programu, zespół programistów spotkał się z problemem automatycznej konfiguracji parametrów optycznych internetowej kamery RGB. Jeden z członków zespołu próbował przeprowadzić konfigurację obrazu pochodzącego z kamery. Chwilę przed tym, zanim konfiguracja została ostatecznie wybrana, w kamerze zmienił się balans bieli. Z tego powodu, wszystkie elementy na obrazie miały zupełnie inne odciennie kolorów. Spowodowało to, że precyzyjnie ustawiane parametry stały się bezużyteczne i konieczna stała się ponowna konfiguracja programu.

Najważniejszym czynnikiem, wpływającym na jakość działania programu jest oświetlenie. Jego istnienie oraz kolor, natężenie i właściwości światła są w stanie w drastyczny sposób wpłynąć na działanie programu. Załóżmy, że na stole wewnątrz pomieszczenia leży książka w białej okładce. Poszczególne punkty wchodzące w skład powierzchni okładki znajdują się w różnej odległości od okna. Na książkę pada rozproszone światło, wpadające do pokoju przez odsłonięte okno. W oknie znajduje się roślina liściasta, która częściowo zasłania wpadające światło. Wszystko to powoduje, że okładka jest oświetlona w sposób wysoce nierównomierny. Człowiek na takie różnice nie zwraca uwagi, dla niego książka ma po prostu białą okładkę. Oprogramowanie komputera nie potrafi jednak stwierdzić, jaki kolor w rzeczywistości ma dany obiekt, i czy różnica koloru wynika np. z posiadania przed przedmiot zróżnicowanego koloru, czy po prostu na różne punkty obiektu pada światło o różnym natężeniu, ponadto pojawiają się cienie.

Wszystko to powoduje, że analizowane obiekty muszą być oświetlone równomiernie, nie mogą posiadać żadnych różnic w oświetleniu ani cieni. Poza tym, parametry światła nie mogą zmieniać się w czasie - jeśli skonfiguruje się program dla pewnych wartości oświetlenia, zmiana tych wartości powoduje, że program traci możliwość wykrycia pionków (program np. nie będzie mógł rozpoznać innych odcieni skonfigurowanych kolorów) Uzyskanie takich warunków może być bardzo trudne - nie powinno się polegać na naturalnym świetle, ponieważ jest ono zmienne w czasie. Powodem jest ruch obrotowy Ziemi wokół własnej osi. Sprawia to, że w ciągu doby plansza jest oświetlana światłem o różnym kolorze oraz o natężeniu, który zależy od pory dnia czy położenia planszy w przestrzeni względem kierunków świata czy innych przedmiotów, mogących wpływać na jakość i kierunek padania światła. Uniemożliwia to poprawne ustawienie parametrów programu lub powoduje konieczność częstej ich zmiany. Dlatego zaleca się oświetlenie planszy i pionków za pomocą sztucznego źródła światła, które jest niezmienne, i niedopuszczenie do pomieszczenia, w którym odbywa się eksploatacja programu, światła pochodzącego z innych, niestabilnych źródeł. Źródło światła powinno być tak umieszczone i skierowane, aby w odpowiedni sposób oświetlało warcabnicę oraz pionki i nie wprowadzało zbędnych zakłóceń do obrazu.

6.2 Dlaczego C++ jest lepszy od C#?

Podczas wyboru narzędzi projektowych niezbędnych do realizacji zadania, zespół miał duży dylemat, czy wybrać język C++, czy C#. Zdecydowano się ostatecznie na rozwijanie umiejętności programowania w języku C# z chęci poznania różnic względem języka C++ oraz utwierdzenia się w myśli, że język C++ jest lepszy.

Za wyższością języka C++ przemawiają następujące argumenty: Język C++ uczy pełnego zrozumienia działania komputera oraz pamięci w komputerze poprzez jawne używanie wskaźników na obiekty, oraz referencji.

Podczas prac związanych z projektowaniem programu, były uwzględnione możliwości stworzenia programu w jednym z wielu różnych języków programowania; jedną z propozycji był język C++. Wybór na C# padł z powodu propozycji stworzenia modułu do weryfikacji ruchów właśnie w tym języku. Gdyby moduł do rozpoznawania obrazu powstał w C++, cała aplikacja byłaby zbyt zróżnicowana pod względem użytych technologii.

W przypadku modułu do rozpoznawania obrazu, w zasadzie nie byłoby różnicy między wyborem C# a C++. Używana w programie biblioteka OpenCV posiada możliwość użycia zarówno w języku C++ (w którym jest napisana), jak i w C# (za pomocą nakładki EmguCV). Między innymi, ze względu na podobną składnię obu języków, a także z powodu użycia praktycznie tych samych funkcji, kod programu, niezależnie od tego, w jakim języku by powstał, wyglądałby bardzo podobnie. O wyborze C# przesądziła jednak łatwość stworzenia interfejsu użytkownika. Tworzona aplikacja jest rozbudowanym programem, poza tym rozpoznawanie obrazu jest złożonym procesem, zależnym od wielu czynników zewnętrznych, dlatego program został wyposażony w bogate możliwości konfiguracji. Wymagało to stworzenia rozbudowanego interfejsu użytkownika, co jest znacznie łatwiejsze (i wymaga przeznaczenia na tę czynność zdecydowanie mniej czasu) w języku programowania C# niż w C++.

Zaletą języka C++ jest możliwość jego użycia na wielu platformach — na wielu systemach operacyjnych (jak np. Linux czy Windows) oraz na wielu różnych platformach sprzętowym (począwszy od prostych, ośmiobitowych mikrokontrolerów rodziny ATtiny, przez bardziej rozbudowane mikroprocesory rodziny ARM, aż po wydajne i złożone procesory architektury x64. Podczas opracowywania języka C++, jego autorzy bez wątpienia musieli pójść na wiele kompromisów — np. wbudowany mikrokontroler nie ma nawet części takich możliwości graficznych, jak np. system Windows z wydajną kartą graficzną i pakietem DirectX. Zapewne z tego właśnie powodu oraz z powodu chęci zachowania małych rozmiarów narzędzi niezbędnych do rozwoju oprogramowania w tym języku, twórcy biblioteki standardowej języka C++ nie zdecydowali się na umieszczenie w niej pewnych elementów, takich jak funkcje służące do tworzenia graficznych interfejsów użytkownika oraz funkcje umożliwiające współpracę z tymi interfejsami. Oznaczałoby to konieczność znalezienia dodatkowych bibliotek graficznych.

Pewną trudnością jest fakt, że język C++ jest wprawdzie językiem wysokiego poziomu (w porównaniu z na przykład, Asemblerem albo językiem maszynowym), jednak istnieją też języki posiadające wyższy poziom abstrakcji niż C++. Wykonanie interfejsu użytkownika w tym języku byłoby zapewne procesem znacznie bardziej złożonym i czasochłonnym, niż przy użyciu innych technologii. Jedną z rozważanych możliwości było wykorzystanie wspomnianej już biblioteki OpenCV, która posiada możliwość tworzenia graficznych interfejsów użytkownika. Narzędzia te są proste w obsłudze, jednak zapewniają jedynie niewielką kontrolę nad wynikowym interfejsem. Innym przykładem biblioteki, z którą miał już do czynienia co najmniej jeden z członków zespołu programistycznego tworzącego omawiany program, jest Windows API. Biblioteka ta jest dość trudna i skomplikowana w obsłudze, przez co stworzenie w niej interfejsu mogłoby w zasadzie być oddzielnym modułem, przygotowanym

przez jedną z osób wchodzących w skład zespołu (w takim przypadku, składającego się z czterech członków).

Ostatecznie, po bardzo długich i burzliwych dyskusjach (podczas których wystąpiły również pewne straty materialne, związane z użyciem argumentów fizycznych), jako oficjalna technologia wykonania programu wybrany został język C# (znany również pod nazwami, takimi jak C Sharp lub C Płotek), działający na platformie .NET Framework. Dzięki jego zastosowaniu, tworzenie aplikacji przebiegło względnie szybko i sprawnie. Poza tym nie trzeba było przejmować się kwestiami takimi jak na przykład wydajność — stworzenie biblioteki OpenCV w języku C++ (pierwotnie w języku C) zapewnia wystarczająco wysoką wydajność, której najprawdopodobniej nie udałoby się uzyskać, gdyby bibliotek przetwarzania obrazu OpenCV została napisana od razu w języku C#.

6.3 Podział prac

Realizacja projektu byłaby niemożliwa, gdyby nie pełne zaangażowanie wszystkich członków zespołu. Każdy wywiązał się rzetelnie ze swoich zadań i aktywnie wspierał resztę zespołu w trakcie trwania prac nad projektem. Podział zadań przedstawiał się następująco:

- Marcin Orczyk.
  - Moduł do rozpoznawania obrazu z kamery i tworzenie cyfrowego odpowiednika planszy fizycznej.
- Natalia Popielarz.
  - Moduł logiki gry w Warcaby i weryfikacji poprawności ruchu.
- Piotr Wołyński.
  - Moduł wizualizacji stanu gry na komputerze.

6.4 Cele zrealizowane

Dzięki systematycznej pracy zrealizowaliśmy wszystkie zamierzone cele. Aplikacja działa.

6.5 Perspektywa rozwoju

Aplikację w przyszłości można rozwinąć o dodatkowe funkcjonalności np. dodanie zautomatyzowanego ustawiania parametrów ekspozycji w taki sposób, aby nie trzeba konfigurować ich ręcznie. Innym usprawnieniem aplikacji mogłoby być obsługa długich bić, bez konieczności wykonywania każdego pomniejszego bicia pojedynczo.