Dokumentacja projektu

Podstawy teleinformatyki

Prowadzący: mgr inż. Przemysław Walkowiak

Temat projektu: Rozpoznawanie obrazu z gry warcaby oraz wizualizacja stanu gry na komputerze

Spis treści

Opis

Celem projektu jest utworzenie systemu zapisującego przebieg gry w warcaby na podstawie informacji dostarczanych przez obraz wideo. Program ma za zadanie rozpoznać poszczególne figury - ich kolor, a także rodzaj(pion czy damka) - na planszy wraz z ich ruchami. Zestawy warcabów posiadają cechy wspólne jak plansza na bazie kwadratu, niezmienna ilość pól (64 pola ułożone 8x8 z na przemian leżącymi kolorami) taka sama ilość i rodzaj figur oraz położenie początkowe pionków.

Wybraliśmy ten projekt, ponieważ w dzisiejszych czasach rejestracja obrazu, jego przetwarzanie i analiza stają się wszechobecne, tego typu operacje stosowane są w wielu dziedzinach, takich jak medycyna, kryminologia, ale także rozrywce dostępnej dla każdego - co za tym idzie ma realny wpływ na nasze życie, zdrowie, komfort czy też bezpieczeństwo.

Członkowie zespołu i podział prac:

Rafał Adamski	
Marcin Sznura	

Tomasz Weiss	
Michał Wypchło	

Założenia projektowe

- Na podstawie obrazu wideo dostarczanego przez kamerę system ma zapisywać przebieg gry planszowej, gry w warcaby.
- Program musi wizualizować stan gry na komputerze - rozpoznawać oraz dokonywać zapisu informacji o figurach znajdujących się na planszy, a także na temat ich ruchów.
- Po wykonaniu ruchu pionka, program ma sprawdzać czy ruch został wykonany poprawnie.

Wybrane technologie

- Biblioteka OpenCV - biblioteka funkcji wykorzystywanych podczas obróbki obrazu. Oprócz faktu, że dzięki niej mogliśmy odczytać obraz z kamery w czasie rzeczywistym,, to pozwoliła nam także na rozpoznawanie oraz interpretację planszy oraz pionków.
- !! Język programowania - na ten moment C++, jednak zastanawiamy się nad zmianą na C#, ponieważ znacznie ułatwi nam to wizualizację.

Architektura rozwiązania

Interesujące problemy i rozwiązania

Problem 1. Instalacja bibliotek openCV

OpenCV to zbiór bibliotek tak rozległych że do wygodnego korzystania należało

dodać je do zmiennych środowiskowych w systemie Windows oraz do właściwości projektu w programie Visual Studio (sekcje C/C++ >> General oraz Linker >> General Input). Poprawna konfiguracja zajęła trochę czasu, jako że nie było to działanie standardowo wykonywane przy prostych projektach.

Problem 2. Dobór bibliotek openCV

OpenCV oferuje wiele obszernych bibliotek spełniających znaczną liczbę funkcji. Spośród wszystkich tych bibliotek należało wybrać te, potrzebne w naszym projekcie. Zaczęliśmy od bibliotek:

- highgui.hpp
- cv.hpp
- imgproc.hpp

które wykorzystaliśmy do śledzenia obiektów.

Problem 3. Śledzenie obiektów

Śledzenie obiektów o różnych kształtach i kolorach nie jest łatwe bez zastosowania odpowiednich sposobów, dlatego zdecydowaliśmy się na osiągnięcie tego celu poprzez wykorzystania modelu HSV(hue, saturation, value) odcień, nasycenie i wartość światła by wyodrębnić obiekty pewnego koloru i śledzić je bez zlewania ich z tłem.

Problem 4. Śledzenie wielu obiektów

Śledzenie wielu obiektów posiadających takie same grupy kolorów wymagało przechowywania ich w jakimś kontenerze danych. Zdecydowaliśmy się na użycie vectorów oraz detekcję poprzez nadawanie nazwy śledzonym obiektom poprzez pole string name w konstruktorze.

Problem 5. Detekcja planszy (wersja matematyczna)

Odczytanie pozycji pionków z powyższymi rozwiązaniami jest proste. Jednak odczytanie planszy wiążę się z większymi problemami. Pola planszy pokrywane są przez pionki. O ile pionki nie będą bardzo małe trudno będzie odczytać pod nimi pola. Czytanie drobnych linii planszy byłoby bardzo kłopotliwe. Najlepszym posunięciem jest odczytanie jednego punktu początkowego i rysowanie planszy według tego punktu. Problem: planszę można obracać i rotacja punktu

początkowego sprawiałaby rysowanie planszy w jeden z 4 możliwych sposobów. Do określenia w jaki sposób rysować planszę należałoby dodać 2 punkty odwzorowania. To rozwiązanie wiążę się jednak ze znakowaniem planszy nadmiarowymi kolorami. Obecnie pracujemy nad rozwiązaniem bardziej czystym, wykorzystującym 2 jednokolorowe punkty stanowiące naturalną ramę planszy, według których rysowana będzie plansza tworzona z dokładnych obliczeń matematycznych.

Odczytywanie planszy zrealizowaliśmy za pomocą wyznaczanie 2 charakterystycznych punktów w przeciwległych rogach. Między tymi punktami rysujemy kwadrat, który dzielimy na 8 części i matematycznie wyznaczamy pola nadając im punkt start i koniec (lewy górny róg oraz prawy dolny róg). Pozycję pionka porównujemy czy leży w granicach pola. Jeśli tak przypisujemy pionkowi to pole. Problemem są dwa: obrót planszy powoduje rysowanie planszy poza granicami faktycznej planszy oraz poprawnie obliczone pola w kwadracie otrzymamy tylko jeśli plansza będzie obrócona do kamery w stopniach 0,90,180,270,360.
Pierwszy problem (nieprawidłowe rysowanie planszy) rozwiązaliśmy poprzez porównywanie przeciwległych punktów w kwadracie planszy. Plansza zawsze rysowana jest od punktu "zero", który posiada mniejszą wartość współrzędnych x,y. Standardowo domyślny punkt pierwszy zostaje punktem zero, według którego wyprowadzane są obliczenia.Jeśli jednak plansza obrócona jest tak że niepożądany punkt jest pierwszy, punktem zero staje się punkt drugi. Problem konieczności ustawienia planszy pod kątem prostym pozostał nierozwiązany i raczej nie poczynimy tu większych postępów przy zastosowaniu metody matematycznej.

Problem 6. Wykrywanie kamery w opencv

OpenCV domyślnie identyfikuje kamerę wbudowaną jako tą z indeksem 0. W przypadku dodatkowych kamer usb, sprawa komplikuje się i wielu użytkowników ma problemy z wykryciem prawidłowego id pożądanej kamerki. Rozwiązanie tego problemu okazało się dość trywialne. W pętli for spróbowaliśmy przechwytywać

obraz z urządzeń z rożnych id z zaimplementowaną obsługą wyjątków. Dla 999 prób powiodły się tylko 3. 0-domyślna kamerka wbudowana, 700-domyślna kamerka wbudowana oraz 701- kamerka usb. Jest to rozwiązanie bardzo wyrafinowane, lecz skuteczne biorąc pod uwagę że indeks 701 trudno byłoby osiągnąć przy ręcznej próbie wylosowania odpowiedniego numeru id.

```
for (int i = 0; i <= 999; i++) {
		try {
			capture.open(i);
			capture.set(CAP_PROP_FRAME_WIDTH, FRAME_WIDTH);
			capture.set(CAP_PROP_FRAME_HEIGHT, FRAME_HEIGHT);
			capture.read(cameraFeed);
			cvtColor(cameraFeed, HSV, COLOR_BGR2HSV);
			if (capture.isOpened() == true) {
				cout << i << ": this number is ID of working camera" <<
endl;
				Sleep(60);
			}
		}
		catch (...)
		{
			if (i == 999) {
				if (!capture.isOpened() == true) {
					cout << endl << "----------------" << endl;
					cout << "Exception: " << noDeviceFound;
					cout << endl << "----------------" << endl;
				}
			}
		}
		if (capture.isOpened() == true) { break; }
	}
}
```

Instrukcja użytkowania aplikacji