Sprawozdanie z laboratorium:
Komunikacja człowiek-komputer

Przetwarzanie obrazu – odczytywanie kolorów z Kostki Rubika

19 listopada 2015

Prowadzący: dr inż. Wojciech Jaśkowski

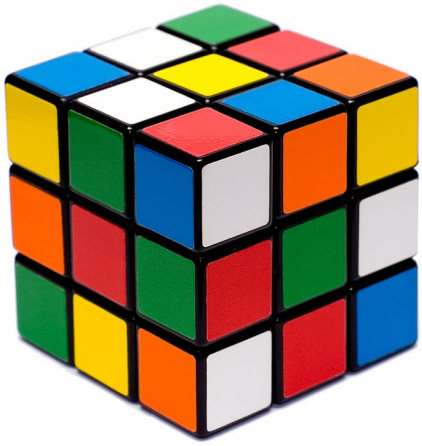
Autorzy: Maciej Olszowy 117256
Krzysztow Wędrowicz 117319

Zajęcia piątkowe, 13:30

1. Wstęp i założenia

Celem projektu było stworzenie programu odczytującego kolory Kostki Rubika z obrazu. Na zdjęciu musi być widoczna kostka zajmująca obszar o powierzchni co najmniej 500x500 pikseli. Na obrazie muszą być widoczne trzy ściany kostki w konfiguracji lewa, prawa i górna. Na każdej ścianie kostki znajduje się dziewięć kwadratów w jednym z 6 kolorów: niebieski, żółty, zielony, czerwony, pomarańczowy, biały. Kwadraty są oddzielone wyraźną czarną linią. W tle nie mogą występować bardzo ciemne obiekty.

Przykładowy akceptowany obraz:

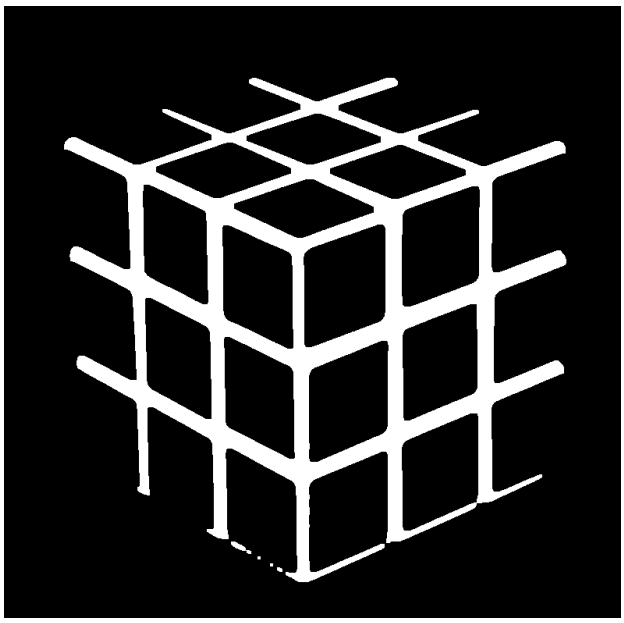


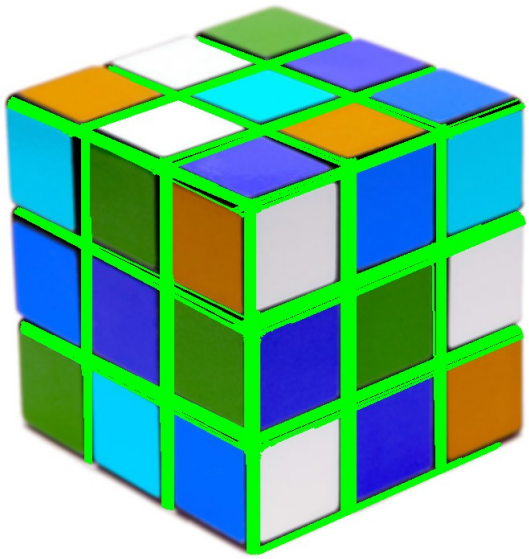
2. Opis metody

Ponieważ nie udało nam się dotrzeć do żadnych źródeł opisujących to zagadnienie, musieliśmy stworzyć własną metodę. Po wielu nietrafionych próbach udało nam się uzyskać algorytm składający się z następujących elementów.

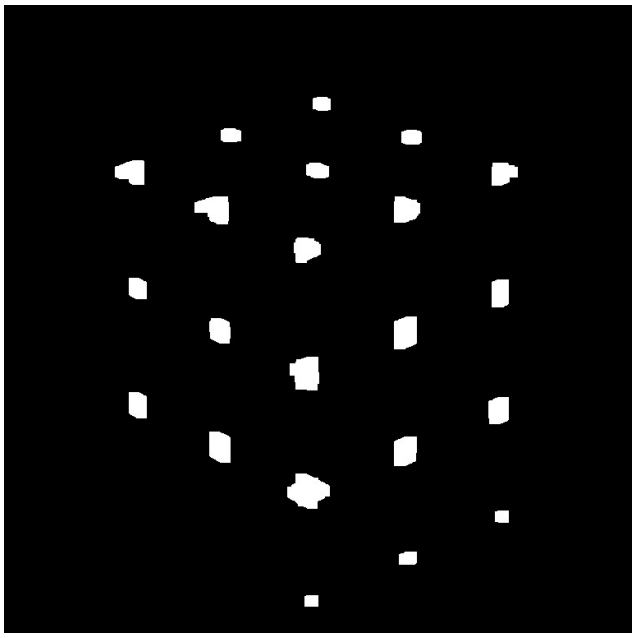
1. Rozpoznanie czarnych linii rozdzielających kwadraty.

    - Wyodrębniamy czarne elementy obrazu.
    - Znajdujemy linie przy pomocy transformaty Hougha.

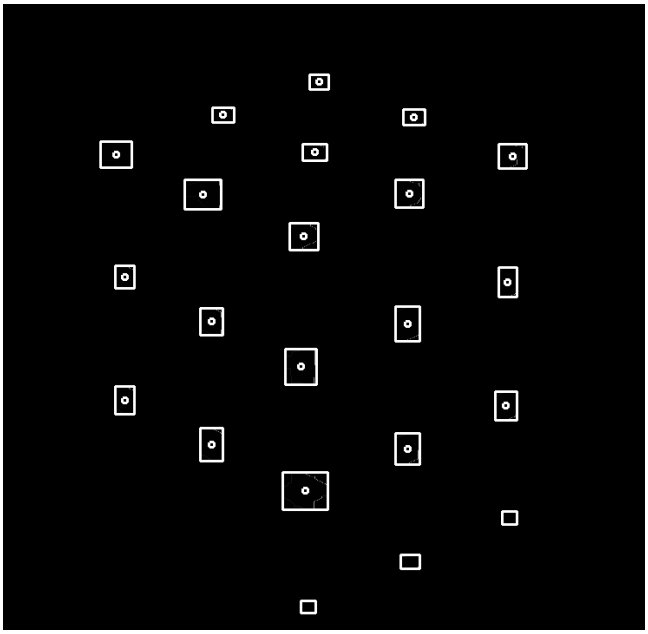




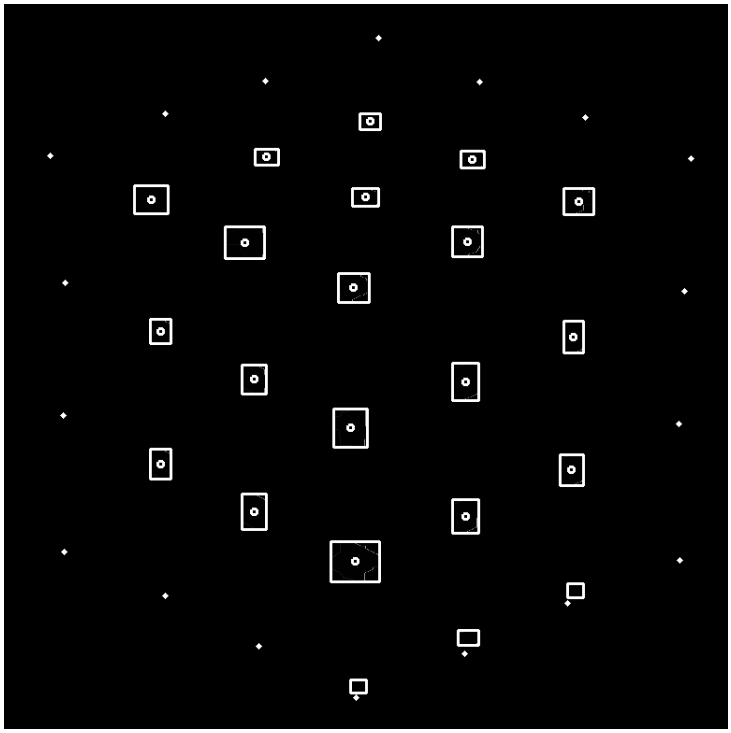
2. Obliczenie punktów przecięcia znalezionych linii.
   - Interesują nas tylko przecięcia linii, pomiędzy którymi kąt jest nie mniejszy niż 30 stopni.
   - Rysujemy znalezione punkty na nowym, pustym obrazie i używamy dylatacji aby stworzyć jednolite kształty.



3. Grupowanie punktów przecięcia.
   - Na powyższym obrazku używamy z algorytmu znajdowania konturów z biblioteki OpenCV.
   - Wybieramy 19 największych konturów.
   - Wybrane punkty zaznaczone są kropką na poniższym obrazku.



4. Obliczenie pozostałych punktów.
   - Na podstawie położenia przydzielamy wszystkie 19 punktów do odpowiedniej ściany (lewej, prawej lub górnej).
   - Dla każdej ściany na podstawie odległości między punktami obliczamy zewnętrzne punkty.
   - Obliczone punkty zaznaczone są małymi kropkami.



5. Znajdowanie pikseli do próbkowania koloru.
   - Mamy 16 punktów dla każdej ściany.
   - Iterujemy tablicę punktów każdej ściany „kwadratem”, obliczając średnią z 4 punktów ([i][j], [i+1][j], [i][j+1], [i+1][j+1]).
   - Próbkujemy kolor w wyznaczonych 9 punktach.

6. Wyznaczenie koloru piksela.
   - Do określenia koloru wykorzystujemy opis koloru w modelu HSV.
   - Tabela przedstawia wykorzystywane przez nas progi.

Kolor	**min H**	**max H**	**min V**	**max V**	**min S**	**max S**
Czerwony	165	180	110	255	50	255
Czerwony	0	10	110	255	50	255
Pomarańczowy	10	25	120	255	50	255
Żółty	26	45	120	255	50	255
Zielony	45	75	110	255	50	255
Niebieski	90	150	120	255	50	255

   - Jeśli próbka nie pasuje do żadnych danych w tabeli uznajemy, że jest to kolor biały.

3. Nierozwiązane problemy.

- Sporadyczne mylenie żółtego z pomarańczowym.
- Sporadyczne znalezienie zbyt dużej ilości przecięć prostopadłych linii, co powoduje wybranie złych punktów (często powodując brak wybrania najwyższego punktu górnej ściany kostki).

4. Ewentualne ulepszenia

Tablicę kolorów zamiast definiować na sztywno można uzyskać za pomocą histogramu. Wykrywanie 6 pików na uzyskanym wykresie oraz ich właściwe przyporządkowanie do kolorów powinno usprawnić ich rozpoznawanie, a w szczególności mylenie żółtego z pomarańczowym. Ponadto rozpoznawanie kolorów mogłoby być wykonywane po średniej z pewnego obszaru a nie jednym punkcie. Zapewniłoby to mniejsze ryzyko błędu w przypadku odblasku lub uszkodzonego piksela. Obecne rozwiązanie nie jest natomiast złe, ponieważ na początku algorytmu rozmazujemy obraz przez co powinniśmy wyeliminować większą część powyższych przypadków.

5. Podsumowanie

Jesteśmy zadowoleni z uzyskanych przez nas efektów. Poradziliśmy sobie z wieloma problemami, takimi jak różne tło zdjęcia, różne kąty nachylenia kostki, perspektywa, właściwe szeregowanie znalezionych punktów. Poprawność algorytmu w większości przypadków osiąga wymarzone 100%, tak jak w całym zbiorze testowym. Taka sytuacja otrzymuje się również na zdjęciach z internetu, pod warunkiem, że algorytm nie rozpozna za dużej liczby prostopadłych przecięć, wtedy poprawność gwałtownie spada.