

Rozpoznawanie stanu rozgrywki w grze planszowej Catan

Magdalena Wiechczyńska, 132337

Piotr Tomaszewski, 136821

1 Temat i opis rozwiązania problemu

Program na podstawie zdjęcia planszy gry Catan rozpoznaje stan rozgrywki, tzn. ułożenie heksagonalnych pól reprezentujących surowce oraz rozstawienie pionków graczy (gracza czerwonego, niebieskiego i pomarańczowego).

W pierwszym kroku stara się wyizolować na zdjęciu samą planszę, aby pominać w przetwarzaniu elementy nie należące do gry. Następnie po kolejne typy pól, zaczynając od tych najbardziej wyróżniających się. Pola znajdywane są na podstawie selekcji pikseli należących do zadanego przedziału wartości HSV. W celu ułatwienia poszukiwania kolejnych, już znalezione pola na bieżąco są usuwane ze zdjęcia poddawanego przetwarzaniu. W przypadku identyfikacji niektórych pól zbliżonych kolorystycznie koniecznym jest skalowanie saturacji zdjęcia, dodatkowo zastosowano również filtr medianowy.

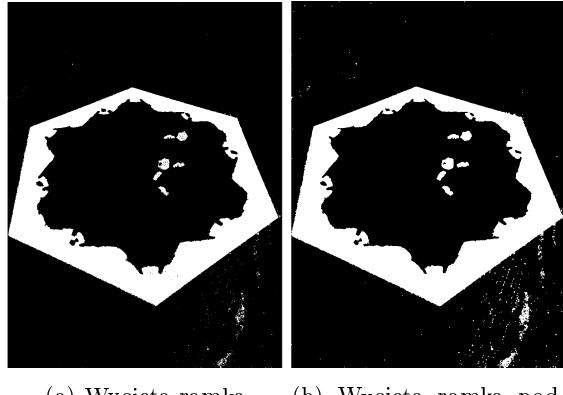
Pionki graczy znajdują się analogicznie. Miasta i osady są odróżniane od dróg na podstawie podobieństwa kształtu do koła (miasta mają okrągłą podstawę, osady natomiast, pomimo nieregularnego kształtu, stanowią dość dobrze przybliżenie koła).

2 Działanie programu

2.1 Znajdowanie planszy

Na początku program próbuje ustalić położenie planszy i usunąć wszelkie elementy do niej nienależące.

Wiedząc, że plansza do gry jest zawsze otoczona ramką przedstawiającą wodę, wyszukujemy pikseli, których wartość odcienia (Hue) w przestrzeni HSV należy do odpowiedniego przedziału. Następnie, w celu usunięcia dziur uzyskany obraz poddajemy dylatacji.

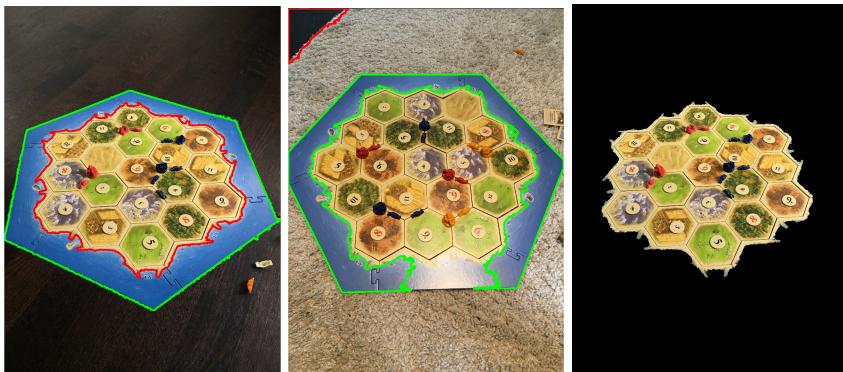


(a) Wycięta ramka (b) Wycięta ramka pod-
dana dylatacji

Figure 1: Znajdowanie ramki

Na tak przetworzonym obrazie szukamy konturów. Zależy nam na znalezieniu wewnętrznego konturu planszy. Na większości zdjęć jest to drugi kontur pod względem obejmowanego pola powierzchni. Jednak, w przypadku gdy ramka zdjęcia jest prześwietlona, oba konury łączą się w jeden. Dlatego w celu ustalenia, który kontur jest konturem wewnętrznym sprawdzamy czy na danym zdjęciu drugi pod względem wielkości kontur znajduje się wewnątrz największego.

Następnie zastępujemy kolorem czarnym wszystkie piksele, które nie znajdują się wewnątrz tego konturu.



(a) Dwa największe kontury (b) Kontury łączą się ze sobą (c) Wycięte tło

Figure 2: Wycinanie tła

2.2 Znajdowanie pionków

2.3 Rozpoznawanie pionków

Na każdej masce uzyskanej w poprzednim kroku szukamy konturów. Dla każdego z nich wyznaczamy otoczkę wypukłą (convex hull), ponieważ pionki wykonane

są z błyszczącego plastiku i trudno jest zapewnić, aby w całości były zaznaczone na masce.

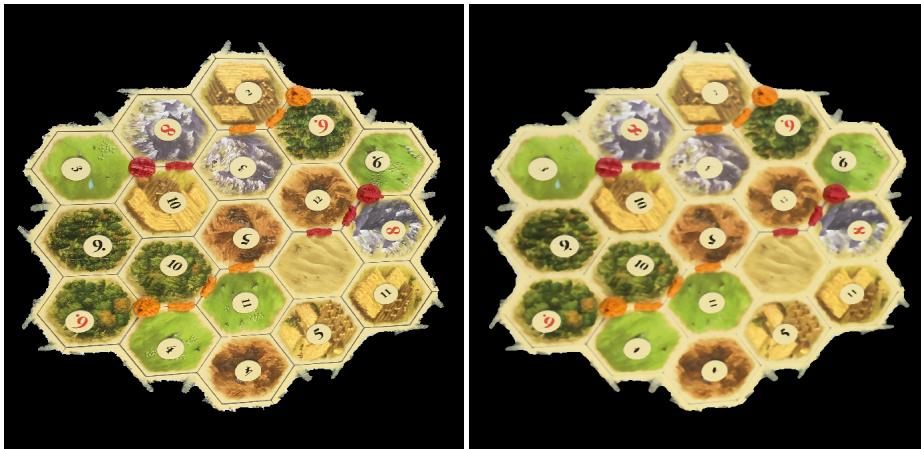
Następnie w każdą otoczkę, której powierzchnia znajduje się w akceptowalnym przedziale wpasowywana jest elipsa. Liczymy stosunek półosi wielkiej do półosi małej elipsy. Dla miast i osad stosunek ten jest bliski 1, natomiast drogi są bardziej podłużne, więc dla nich ten stosunek jest niższy.

Jeżeli jednak powierzchnia otoczki danego konturu przekracza akceptowalną wartość, może to oznaczać, że na tym obszarze znajdują się w rzeczywistości dwa pionki. Dlatego ten kontur rysujemy na nowej, czystej masce. Maskę następnie poddajemy erozji i rekurencyjnie wywołujemy na niej funkcję rozpoznającą pionki. Robimy tak do czasu, aż powierzchnia otoczki znajdzie się w akceptowalnym przedziale.

Dla każdego zidentyfikowanego pionka wyznaczany jest centroid, który następnie zaznaczany jest na obrazie.

2.4 Znajdowanie i rozpoznawanie pól

W pierwszej kolejności stosujemy filtr medianowy z maską 15×15 . Sprawi to, że kolory zostaną wygładzone i staną się mniej różnorodne w obrębie poszczególnych pól.



(a) Plansza przed nałożeniem filtru

(b) Plansza po nałożeniu filtru

Figure 3: Filtr medianowy

Następnie odszukujemy najbardziej wyróżniający się typ pola, łąkę z owcami. Dokonujemy selekcji pikseli na podstawie ich wartości H, S i V (sprawdzamy czy wszystkie wartości znajdują się w akceptowalnych przedziałach).

Przykładowo, oto przedziały znormalizowanych wartości H, S, V, które lokalizują pola z owcami:

$$H = [0.15, 0.3], S = [0.4, 1], V = [0.6, 1]$$

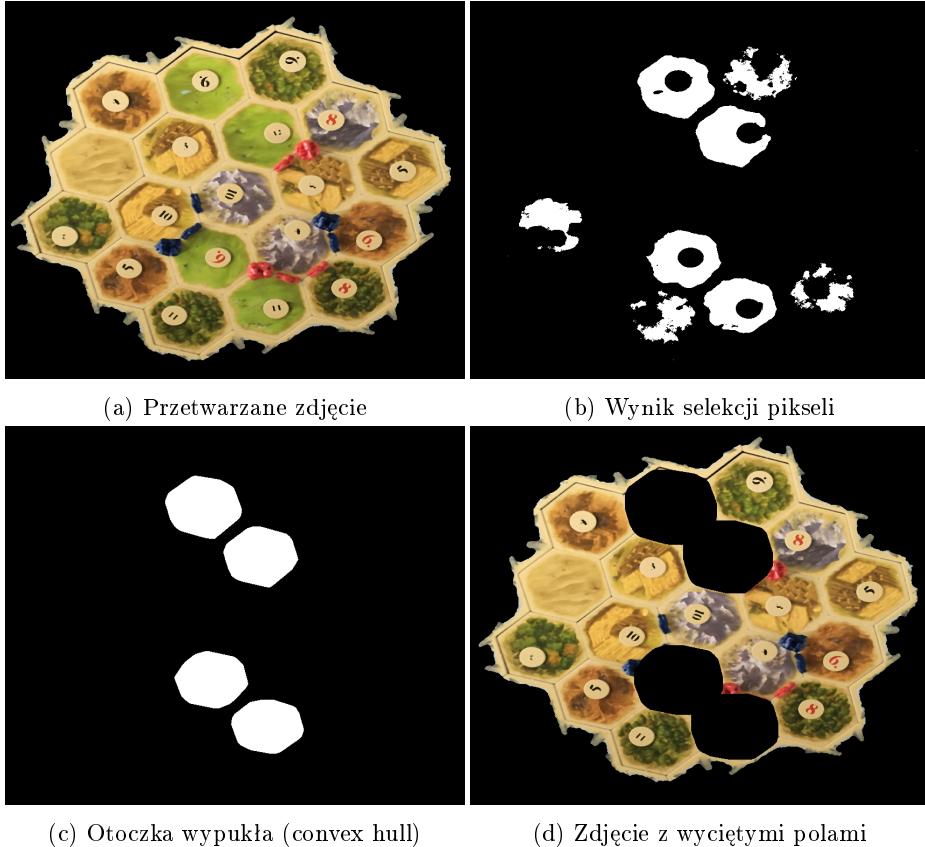


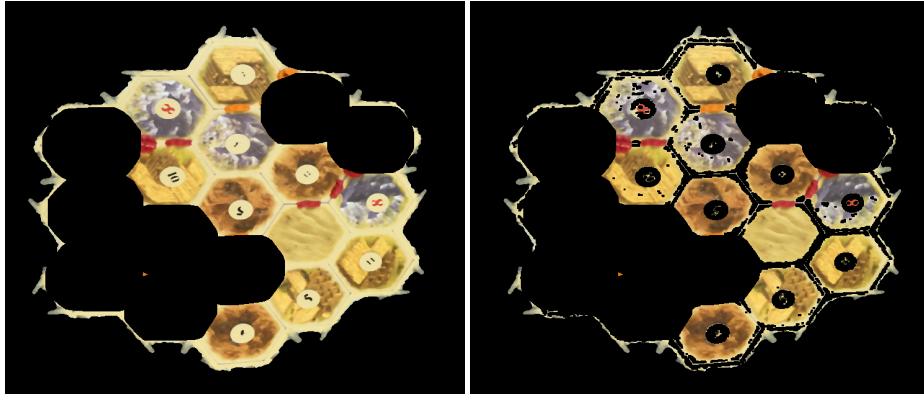
Figure 4: Selekcja pikseli na podstawie akceptowalnych przedziałów wartości.

Następnie, w zależności od aktualnie rozważanego typu pola, maska poddawana zostaje dylacji/erozji. Na masce odszukujemy kontury, które następnie sortowane są pod względem pola powierzchni.

Wykorzystujemy fakt, że liczba poszczególnych typów pól na planszy jest stała (są zawsze cztery łąki, trzy pola z górami itd.). Z tego powodu rozważamy jedynie x pierwszych konturów z posortowanej listy. Dla każdego z tych konturów znajdowana jest otoczka wypukła (convex hull). Na jej podstawie obliczany jest centroid i wyrysowywana jest kropka reprezentująca dane pole.

Dodatkowo, otoczki umieszczane są na nowej masce, która jest następnie poddawana dylacji. Tak utworzona maska wycina z przetwarzanego obrazu pola, które zostały już odnalezione. Ma to ułatwić odnajdywanie następnych typów pól.

Po odnalezieniu łąk w sposób analogiczny lokalizowane są lasy. Przed wyszukiwaniem gór staramy się dodatkowo wyciąć obramowanie pól, aby zminimalizować możliwość odnalezienia konturów w tych miejscach (ze względu na swoją jasność).

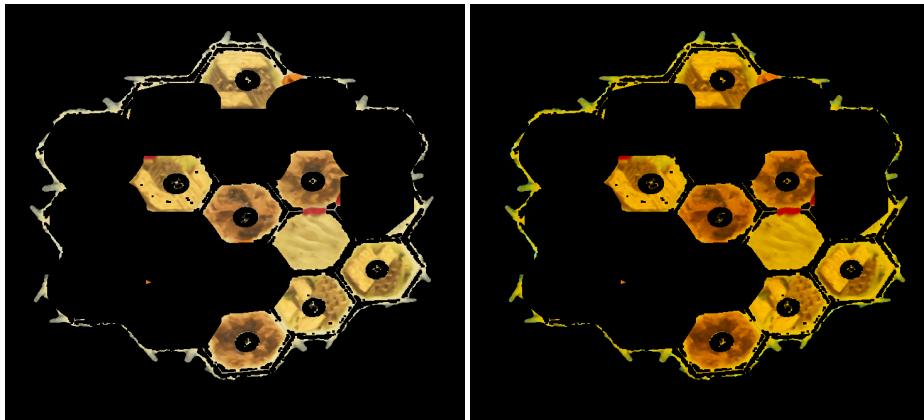


(a) Zdjęcie po usunięciu lasów

(b) Odcięte obramowanie

Figure 5: Odcięcie obramowania planszy

Ostatnie pola do identyfikacji są zdecydowanie trudniejsze, ze względu na podobieństwo kolorystyczne, jednak po zastosowaniu 5-krotnego skalowania saturacji obrazu glina wystarczająco wyróżnia się na tle pozostałych pól, aby móc z powodzeniem wykorzystać wyżej przedstawiony algorytm.



(a) Przed

(b) Po

Figure 6: Skalowanie saturacji, odszukanie gliny staje się łatwiejsze

Odróżnienie pustyni od łąk jedynie na podstawie koloru nie jest wystarczającym rozwiążaniem - delikatna różnica w oświetleniu pól może zaważyć o wyniku działania programu. W tym celu wykorzystywany jest fakt, że na pustyni nigdy nie znajdzie się okrągły krążek z cyfrą, z racji tego, że pustynia reprezentuje brak surowca. Z tego powodu za jednym razem znajdujemy wszystkie pięć pozostałych w grze pól (cztery pola ze zbożem i pustynia) oraz uznajemy za pustynię ten kontur, dla którego pole powierzchni jest największe.

3 Przedstawienie wyników

3.1 Zdjęcia łatwe



(a) Brak pionków



(b) Liczby na planszy

Figure 7: Przykłady zdjęć łatwych

3.2 Zdjęcia średnie

3.3 Zdjęcia trudne



(a) Prześwietlone zdjęcie

Figure 8: Przykłady zdjęć trudnych

4 Podsumowanie wyników

Program jest w bardzo małym stopniu wrażliwy na obecność dodatkowych elementów na zdjęciu. Krytyczna istotna jest kwestia oświetlenia - mocne rozblaski, światło pod zbyt ostрыm kątem (cienie) oraz nierównomierne i zbyt słabe oświetlenie mogą zaburzyć uzyskiwane wyniki.

Dodatkowo program opiera się na powtórnych iteracjach algorytmu znajdywania pól, co powoduje, że jeżeli na wczesnym etapie został popełniony błąd, tj. źle zaklasyfikowaliśmy jedno z pól, problem będzie narastał dla kolejnych typów pól. Aby wyeliminować ten problem, należałoby usprawnić metodę wycinania wody, która rozpoznawałaby zaistnienie rozblasku i uwzględniała go w wycięciu.



Figure 9: Rozblask na planszy - woda nie zostaje w pełni wycięta, w miejscu rozblasku algorytm znajduje zboże

Jeżeli oświetlenie jest odpowiednie, program całkiem dobrze radzi sobie ze zdjęciami zrobionymi pod kątem. Trzeba jednak wziąć pod uwagę sytuacje, w których pionki z powodu nachylenia zaczynają stawać się jedną całością, bądź zostały źle postawione, tzn. dotykają się. W niektórych przypadkach erozja

jest w stanie rozdzielić takie pionki, w innych nie. Aby wyeliminować ten problem, należałoby zaimplementować o wiele bardziej skomplikowany sposób identyfikacji pionków, bazujący nie tylko na ich kształcie i kolorze, ale także na ich ułożeniu względem heksagonów.



(a) Zbyt duży kąt zdjęcia zlał ze sobą pionki (b) Złe rozstawienie - pionki dotykają się

Figure 10: Problemy z pionkami