*Podstawy Teleinformatyki - dokumentacja*

JigsawSolver

Podpowiadarka do puzzli

Skład grupy:

Dominika Czarnecka

Piotr Bochra

Tomasz Fengier

Spis treści

Spis treści 2

1. Wstęp 4
   1. O projekcie 4
   2. Dlaczego ten temat? 4
2. Organizacja pracy 5
   1. Funkcjonalność 5
3. Podpowiedź 5
4. Układanie puzzli 5
   1. Wykorzystane technologie 5
5. Java Android 5
6. OpenCV 5
   1. Podział prac 6
7. Problemy i rozwiązania 7
   1. Reprezentacja puzzli w algorytmie 7
   2. Wydzielanie konturów 8
   3. Metoda podzielenia konturu na cztery części 9
   4. Normalizacja krawędzi 11
   5. Funkcja podobieństwa 13
   6. Reprezentacja położenia puzzli 15
   7. Metoda układania ramki 15
   8. Ustawianie elementów 17
   9. Metoda reprezentacji rozwiązania 18
   10. Metoda dopasowania elementu do wizerunku całego obrazu 18
   11. Metoda układania środka 19
8. Instrukcja obsługi 21
   1. Menu główne 21
   2. Ekran dopasowywania elementu do obrazka 22
   3. Ekran układania puzzli 24
9. Podsumowanie 28
   1. Zrealizowane założenia 28
   2. Niezrealizowane założenia 28
10. wydzielanie puzzli z obrazu 28
11. Dopasowywanie elementów (częściowo) 29
12. Wstęp
    1. O projekcie

JigsawSolver to aplikacja mobilna na system android. Jej zadaniem jest pomoc użytkownikowi w układaniu puzzli przez odczytywanie obrazu z aparatu telefonu. Aplikacja jest napisana w języku Java i korzysta z biblioteki OpenCV do przetwarzania obrazu.

* 1. Dlaczego ten temat?

- Użycie technologii rozpoznawania obrazu - Jest to dla nas możliwość zapoznania się z bardziej zaawansowanymi technikami przetwarzania obrazu i stosowanymi algorytmami.

- Połączenie OpenCV z Androidem - Umiejętność pisania aplikacji na ten system może być przydać się w przyszłości, ze względu na jego powszechność.

- Popularność wykorzystywanych technologii - W internecie dostępne są liczne przykłady użycia zarówno elementów systemu Android jak i biblioteki OpenCV.

- Dostępność otwartych bibliotek - OpenCV jest projektem tworzonym przez społeczność, dzięki czemu jest darmowy i szeroko stosowany. Istnieją wersje na różne rodzaje urządzeń i bindingi do różnych języków programowania.

1. Organizacja pracy
   1. Funkcjonalność

Aplikacja zakłada dwa tryby działania:

1. Podpowiedź

Użytkownik robi zdjęcie obrazu ułożonych puzzli z pudełka, a następnie może skanować pojedyncze elementy w celu dopasowania ich do wzorca. Aplikacja szuka elementu na wzorcu i pokazuje, gdzie najprawdopodobniej powinien się znaleźć.

1. Układanie puzzli

Użytkownik wysypuje puzzle i układa je obrazkiem do góry. Następnie robi zdjęcie i po wciśnięciu odpowiedniego przycisku aplikacja dopasowuje je do siebie i pokazuje obraz z ułożonymi puzzlami.

* 1. Wykorzystane technologie

1. Java Android

- Dostępność darmowych narzędzi (Android Studio).

- Powszechność tego systemu operacyjnego.

- Telefony posiadają wbudowane kamery, umożliwiają łatwe wczytywanie rzeczywistego obrazu.

1. OpenCV

- Darmowa biblioteka.

- Dostępność wersji na system Android.

- Wsparcie dużej liczby potrzebnych algorytmów.

* Dostępność dokumentacji.
  1. Podział prac

Tomasz Fengier

1. Stworzenie interfejsu użytkownika.
2. Zaprojektowanie ikon menu.
3. Zaprojektowanie reprezentacji puzzli w programie.
4. Zaprojektowanie i implementacja algorytmu wycinania puzzli z obrazu pozbawionego tła. W tym:

- Metoda podzielenia konturu na cztery krawędzie.

- Metoda określenia typu krawędzi (płaska, czy z połączeniem).

- Metoda normalizacji krawędzi.

1. Zaprojektowanie i implementacja algorytmu układania puzzli. W tym:

- Metoda układania ramki.

- Metoda układania środka.

- Funkcja podobieństwa krawędzi.

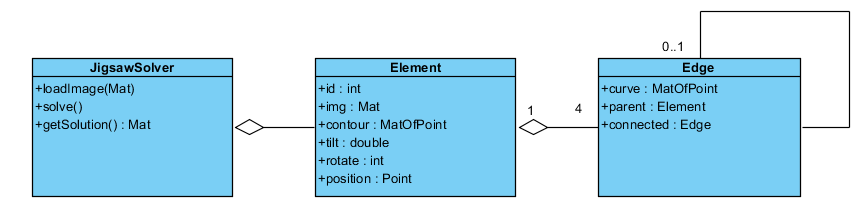
1. Metoda reprezentacji rozwiązania jako obraz.

Dominika Czarnecka

1. Implementacja widoku dopasowania.
2. Zaprojektowanie i implementacja metody dopasowania elementu do całego obrazka wykorzystując algorytm BRISK.
3. Próba wydzielania konturów puzzli wykorzystując metodę Canny.
4. Implementacja wyboru zdjęcia z galerii oraz podgląd z kamery na widoku dopasowania.

Piotr Bochra

1. Implementacja użycia wbudowanej aplikacji aparatu.
2. Próba usunięcia tła dzięki zastosowaniu progowania.
3. Zapis zdjęć w pamięci telefonu.
4. Sprawdzenie działania wydzielenia konturów dzięki metodzie Canny.
5. Problemy i rozwiązania
   1. Reprezentacja puzzli w algorytmie



Rys. 1: klasy używane w algorytmie.

Podstawową klasą służącą do układania puzzli jest JigsawSolver. Zawiera metody ekstrakcji puzzli i tworzenia rozwiązania graficznego. Metoda działania:

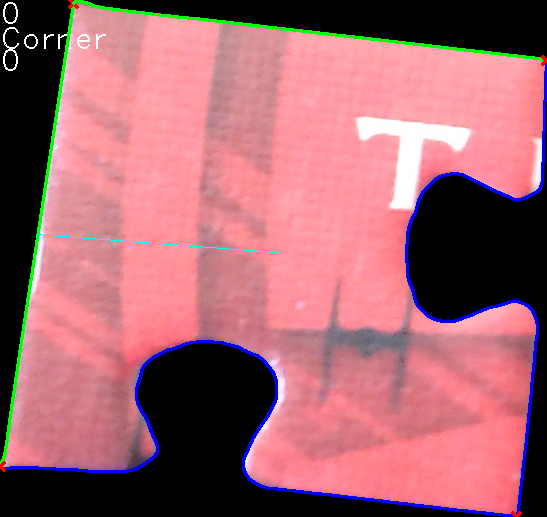
- Tworzona jest instancja obiektu dla jednego rozwiązania.

- Wywoływana jest metoda do wczytania obrazu ze zdjęcia.

- Puzzle na zdjęciu są wycinane przy wczytywaniu i dla każdego z nich jest tworzona instancja klasy Element.

- Każdy z wyciętych puzzli ma swój kontur i dane dotyczącego jego położenia.

- Konstruktor elementu zajmuje się dzieleniem go na cztery obiekty reprezentujące krawędź.



Rys. 2: wycięty element.

Wizualna reprezentacja elementu znajduje się na rys. 2. W lewym górnym rogu znajdują się kolejno: unikalny identyfikator, typ elementu (róg, krawędź, środek) i obrót. Poszczególne kontury przed normalizacją zostały zaznaczone odpowiednio na zielono dla wykrytych płaskich krawędzi i na niebiesko dla pozostałych. Punkty dzielące krawędzie zaznaczono czerwonymi markerami. Turkusowa linia wskazuje krawędź o indeksie 0.

* 1. Wydzielanie konturów

Wydzielanie działa na obrazie pozbawionym tła. Ideą algorytmu jest dla każdego z elementów utworzenie kawałka obrazu, tak aby miał on swoją graficzną reprezentację. OpenCV udostępnia funkcje z których korzysta metoda. Najpierw pozyskiwana jest lista konturów. Aby uniknąć znajdowania w środku puzzli jest używana opcja RETR\_EXTERNAL. Dla każdego z nich jest znajdowany minimalny prostokąt. Dla każdego prostokąta jest wycinany fragment obrazu, który obejmuje, a odpowiednia krzywa jest przesuwana, tak aby była pokrywała się z obrazem wycinka. Aby wygładzić kontury są stosowane trzy kroki:

- rozmazywanie obrazu wejściowego.

- prosta aproksymacja krzywej w metodzie findcontour.

- Upraszczanie poszczególnych krzywych ( w konstruktorze klasy Edge)

Kod realizujązy wydzielanie przedstawiono poniżej:

Mat gray1 = scene.clone();

Imgproc.*cvtColor*(scene, gray1, Imgproc.***COLOR\_RGB2GRAY***);

Mat gray = scene.clone();

Imgproc.*blur*(gray1, gray, **new** Size(2,2));

List<MatOfPoint> contours = **new** ArrayList<>();

Mat hierarchy = **new** Mat();

Imgproc.*findContours*(gray, contours, hierarchy, Imgproc.***RETR\_EXTERNAL***, Imgproc.***CHAIN\_APPROX\_SIMPLE***);

**int** id = 0;

**for**(MatOfPoint cont : contours) {

Rect rect = Imgproc.*boundingRect*(cont);

Mat croppedElement = **new** Mat(scene, rect);

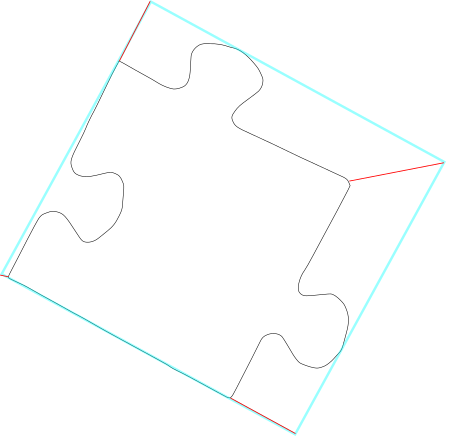
offsetContour(cont, -rect.x, -rect.y);

Element el = **this**.**new** Element(id++, croppedElement, cont);

**this**.elements.add(el);

}

* 1. Metoda podzielenia konturu na cztery części



Rys. 3: minimalny prostokąt obejmujący.

Kontur puzzla stanowi lista kolejnych punktów krzywej. Punkty te są w obrazie zawsze reprezentowane w kierunku przeciwnym do ruchu wskazówek zegara. W celu podzielenia go na dokładnie cztery części reprezentujące krawędzi stosuje się następujący algorytm:

- Na podstawie całego konturu wyznaczany jest obejmujący prostokąt o minimalnym polu ( na obrazku nr 3 zaznaczone na niebiesko)

- Iterując po kolejnych punktach konturu zostają znalezione 4 punkty będące najbliżej wierzchołków ( zaznaczone czerwonymi liniami).

- Dla każdego wycinka jest tworzony nowy obiekt.

- Ze względu na kolejność punktów konturu, obiekty reprezentujące kolejne krawędzie również idą w kierunku przeciwnym do ruchu wskazówek zegara. Na obrazku nr 3 pierwszą w kolejności krawędź połączono niebieską linią ze środkiem ciężkości kształtu.

- Przy tworzeniu krawędzi określany jest jej typ: płaska i krzywa. Zakładamy, że w klasycznych puzzlach płaskie krawędzie są zawsze skierowane na zewnątrz i nie mogą być połączone z inną krawędzią. Metodą na sprawdzenie typu krawędzi jest znalezienie minimalnego obejmującego prostokąta i porównanie jego szerokością z wysokością. Jeśli ten stosunek będzie poniżej pewnej wartości to krawędź jest określana jako płaska.

Kod do znalezienia wierzchołków będących rogami:

RotatedRect rect = Imgproc.*minAreaRect*(cont);

Point[] rectPts = **new** Point[4];

rect.points(rectPts);

**double** mindist[] = **new** **double**[4];

Arrays.*fill*(mindist, Double.***MAX\_VALUE***);

**for**(**int** i = 0; i < cont.rows(); i++) {

Point p = **new** Point(cont.get(i, 0));

**for**(**int** j = 0; j < 4; j++) {

**double** dist = *distSq*(rectPts[j], p);

**if**(mindist[j] > dist) {

corners[j] = i;

mindist[j] = dist;

}

}

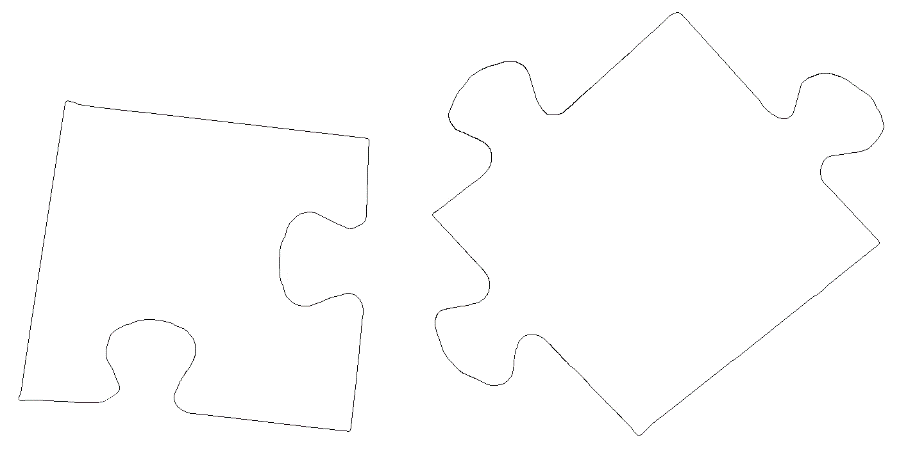
}

Arrays.*sort*(corners);

**return** corners;

* 1. Normalizacja krawędzi

Wydzielone krawędzie zawierają krzywe, których punkty składowe mają współrzędne względem lewego górnego rogu obrazka. Do złożenia puzzli konieczne jest zaimplementowanie funkcji porównującej pary krawędzi. Przykładowe kontury elementów przedstawione są na rys. 4.



Rys. 4: przykładowe kontury.

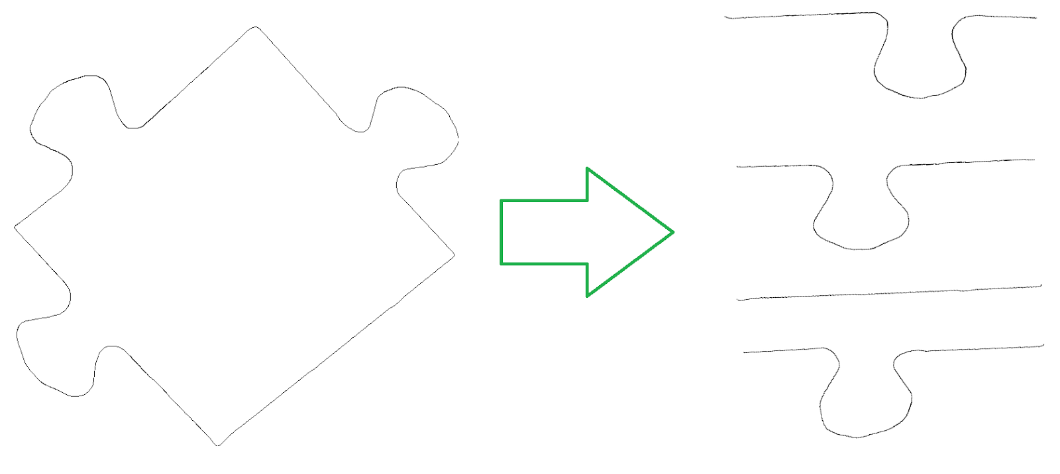
Jest to jednak znacznie utrudnione, gdy każda z nich może być dowolnie obrócona i przemieszczona względem zera układu współrzędnych. W tym celu zastosowano algorytm normalizacji krawędzi, to znaczy transpozycji ich według określonego wzorca:

- Punkt początkowy znajduje się w zerze układu współrzędnych.

- Punkt końcowy znajduje się na osi X.

- Całość krzywej zachowuje swoją długość i kształt.

- Część “zewnętrzna” krawędzi wskazuje w dół.



Rys. 5: normalizazja.

Na obrazku nr 5 przedstawiono ideę normalizacji. Ja widać wszystkie krawędzie są skierowane w dół stroną zewnętrzną. Dla każdej krawędzi realizuje ją kod:

Point p1 = **new** Point(norm.get(0, 0)); //początek

Point p2 = **new** Point(norm.get(norm.rows()-1, 0)); //koniec

**double** dist = Math.*sqrt*(*distSq*(p1, p2));

**double** cos = (p2.x - p1.x) / dist;

**double** sin = (p2.y - p1.y) / dist;

**for**(**int** i = 0; i<norm.rows(); i++) {

**double**[] pt = norm.get(i, 0);

**double** x = pt[0];

**double** y = pt[1];

pt[0] = cos\*(x-p1.x)+sin\*(y-p1.y);

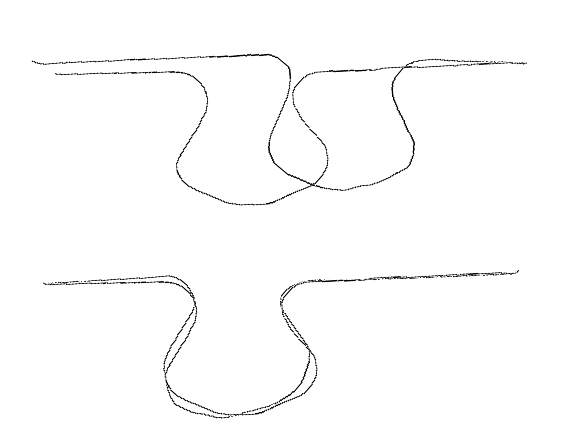
pt[1] = -sin\*(x-p1.x)+cos\*(y-p1.y);

norm.put(i, 0, pt);

}

**return** norm;

* 1. Funkcja podobieństwa



Rys. 6: porównanie jednej z krawędzi do dwóch innych.

Do porównywania par krawędzi wykorzystywana jest funkcja podobieństwa. Im bardziej są do siebie podobne, tym mniejsza jest wartość funkcji. Do porównywania są użyte krzywe znormalizowane. Wartość funkcji jest obliczana na podstawie pola figury powstałej z połączenia końca jednej krawędzi z początkiem drugiej. Wszystkie krzywe są ustawione zewnętrzną stroną w dół, więc jest konieczne przesunięcie i obrócenie jednej z nich o 180°. Tak powstała figura ma pole tym mniejsze, im bardziej pasują do siebie krawędzie.

**if**(**this**.isFlat() || e.isFlat() || **this**.parent == e.parent)

**return** Double.***MAX\_VALUE***;

MatOfPoint temp1 = **new** MatOfPoint();

**this**.curveNorm.copyTo(temp1);

MatOfPoint temp2 = **new** MatOfPoint();

e.curveNorm.copyTo(temp2);

Point end = **new** Point(temp1.get(temp1.rows()-1, 0));

**for**(**int** i = 0; i < temp2.rows(); i++) {

**double**[] t = temp2.get(i, 0);

t[0] = -t[0]+end.x;

t[1] = -t[1]+end.y;

temp2.put(i, 0, t);

}

temp1.push\_back(temp2);

**return** Imgproc.*contourArea*(temp1);

Do porównywania krawędzi stosowane są także dodatkowe zasady, sporządzone z myślą o warunkach układania rzeczywistych puzzli:

- Krawędzie należące to tego samego elementu mają maksymalną wartość funkcji. Nie można ich fizycznie połączyć.

- Krawędzie płaskie w porównaniu z dowolną inną krawędzia zawsze dają maksymalną wartość funkcji. Zakładamy, że puzzle mają wypustki pomiędzy elementami w środku (tzn poza ramką)

- Każda krawędź ma referencję na inną krawędź połączoną. Jeśli nie została ona połączona z żadną inną, to wartość wynosi null. Dodatkowo każdy element zawiera set elementów tworzących razem klaster. Dwa elementy w jednym takim zbiorze nie mogą zostać ze sobą bezpośrednio połączone, ponieważ są one już pośrednio połączone przez inne elementy.

Kod realizujący łączenie krawędzi:

**public** **void** merge(Edge e) {

HashSet<Element> set = e.parent.connected;

**this**.parent.connected.addAll(set);

**for** (Element el : set) {

el.connected = **this**.parent.connected;

}

**this**.connected = e;

e.connected = **this**;

}

* 1. Reprezentacja położenia puzzli

Do reprezentacji położenia elementu wykorzystywane są dwa główne elementy:

- Położenie jako punkt w formacie (x,y).

Przed ułożeniem puzzli nie jest znany ostateczny rozmiar, więc użycie tablicy dwuwymiarowej nie jest dobrym rozwiązaniem. Jeden z elementów jest wybierany jako odniesienie i umieszczany w (0,0). Alternatywą jest użycie listy list tworząc w ten sposób dynamiczną strukturę dwuwymiarową, jednak jest to problematyczne w przypadku, gdy chcemy dodać element z zachowaniem pewnej ilości pustych elementów pomiędzy. Taka sytuacja może wystąpić, ponieważ algorytm najpierw układa ramkę, a dopiero potem ją wypełnia. Dodatkowo nie jest możliwe tworzenie ujemnych indeksów listy. Wadą przechowywania współrzędnych w obiekcie jest konieczność przeniesienia każdego elementu przez narysowaniem rozwiązania, aby rozwiązanie w całości znajdowało się w I-szej ćwiartce układu. Elementy jeszcze nie dopasowane mają współrzędne o wartości null.

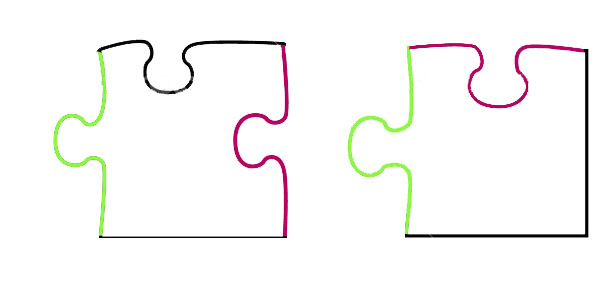
- Obrót elementu.

Algorytm zakłada, że element może być dowolnie obrócony. Krawędzie każdego elementu są numerowane. Algorytm ekstrakcji krawędzi sortuje je, tak aby zawsze były zorganizowane w kolejności kierunku wskazówek zegara. Krawędź o indeksie 0 jest obrócona względem osi x o pewien kąt. Wartość ta jest obliczana przez poprowadzenie wektora od środka ciężkości całego elementu do punktu środkowego (średnia punktów końcowego i początkowego krzywej krawędzi). kąt tego wektora jest więc kątem przechylenia elementu.

Dodatkowo każdy z elementów może znaleźć się w jednym z czterech możliwych obrotów o 90° w kierunku ruchu wskazówek zegara. Wartość ta jest reprezentowana jako liczba całkowita ze zbioru <0,3>. Można ją interpretować jako indeks krawędzi kierującej się w górę. Nieznany obrót jest reprezentowany jako liczba -1.

* 1. Metoda układania ramki

Układając puzzle zakładamy, że pewien zbiór elementów jest elementami ramki. Algorytm jest dzielony na fazę tworzenia ramki i układania elementów w środku.



Rys. 7: krawędzie porównywane w algorytmie.

Do układania ramki zastosowano następujący algorytm:

- Można zauważyć, że puzzle będące składowymi ramki mają zawsze dwie krawędzie leżące na lewo i na prawo patrząc od strony zewnętrznej elementu. Na zdjęciu nr 7 oznaczono to odpowiednio kolorem zielonym i czerwonym.

- W zależności czy jest to element w rogu, czy nie krawędź “lewa” i “prawa” mogą być obok siebie.

- Do znajdowania par najlepiej pasujących krawędzi nie trzeba porównywać każdej krawędzi elementów ramki z każdymi.

- Każda “lewa” krawędź będzie połączona zawsze z  “prawą” krawędzią i na odwrót.

- Takie podejście zapewnia znaczne ograniczenie ilości porównań.

- Tworzona jest macierz, której wiersze stanowią krawędzie “lewe”, a kolumny “prawe”.

- Poszczególne komórki to wartości funkcji podobieństwa danych krawędzi.

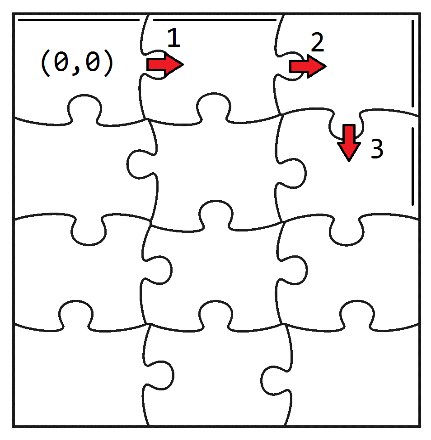
- W pętli brane jest najlepsze dopasowanie i dana para krawędzi jest łączona.

- Odpowiednia kolumna i wiersz jest usuwana z puli.

- Operacja jest powtarzana do usunięcia wszystkich krawędzi z puli. Po zakończeniu powinna powstać spójna ramka rozwiązania.

* 1. Ustawianie elementów

Mając ułożoną ramkę, to znaczy każda z odpowiednich krawędzi ma dopasowaną pasującą krawędź brany jest jeden z elementów jako punkt odniesienia i umieszczany w punkcie (0,0) i obrotem 0.



Rys. 8: przykładowy schemat ustawiania elementów.

Następnie rekurencyjnie odwiedzane są połączone elementy i w zależności od kierunku w którym następuje odwiedzenie kolejnego elementu określane jest jego położenie i obrót w odniesieniu do poprzednika. Na obrazku nr 8 zaznaczono kolejne odwiedziny elementów jako strzałki. Krawędź o indeksie 0 jest zaznaczona linią. Dla podanego przykładu kroki są następujące:

- Element w rogu ustawiany jako odniesienie.

- kolejny zostaje odwiedzony od lewej, zatem jego współrzędna x zostaje powiększona o 1. Indeks krawędzi, przez którą jest połączony to 1, zatem jego obrót wynosi 0.

- kolejny element zostaje odwiedzony od lewej. Tym razem indeks jego krawędzi łączącej to 2, zatem jego obrót to 1.

- kolejny element zostaje odwiedzony z góry, zatem jego współrzędna y będzie zwiększona o 1 względem poprzednika. Indeks łączącej krawędzi to 1, zatem taki jest też jego obrót

- Analogicznie przetwarzane są pozostałe elementy aż do odwiedzenia wszystkich.

W celu uniknięcie nieskończonej pętli używana jest kolekcja odwiedzonych wierzchołków.

* 1. Metoda reprezentacji rozwiązania

Gotowe rozwiązanie jest rysowane poprzez stworzenie nowego obrazu i naniesienie na niego każdego obrazu elementu. Metoda musi zostać wywołana po ułożeniu elementów.

- znany jest obrót elementu względem pierwszej krawędzi

- znany jest indeks krawędzi kierującej się w górę.

- Każdy z elementów ma swój punkt położenia, znajdujący się w pierwszej ćwiertce układu współrzędnych.

- Każdy z obrazów elementów jest obracany tak, aby jego krawędzie były prostopadłe do krawędzi całego obrazu i do swojego docelowego obrotu. Do tego celu wyznacza się macierz transformacji.

* 1. Metoda dopasowania elementu do wizerunku całego obrazu

Metodą dopasowania cech charakterystycznych oraz pozyskania właściwości tych cech na obu obrazach jest algorytm BRISK, który jest algorytmem dostarczonym wraz z OpenCV. Algorytm ten został wybrany ze względu na odporność na skalowanie oraz różnicę kątów między porównywanymi obrazami. Metodą porównującą cechy w celu dopasowania jest algorytm BRUTEFORCE HAMMING.

- Najpierw obrazy w formacie mat pozbawiane są kanału alfa, który utrudnia detekcję cech.

- Następnie cechy charakterystyczne obu obrazów są niezależnie wyszukiwane przez algorytm BRISK.

- W kolejnym kroku wyznaczane są obszary wokół cech, aby ułatwić porównywanie.

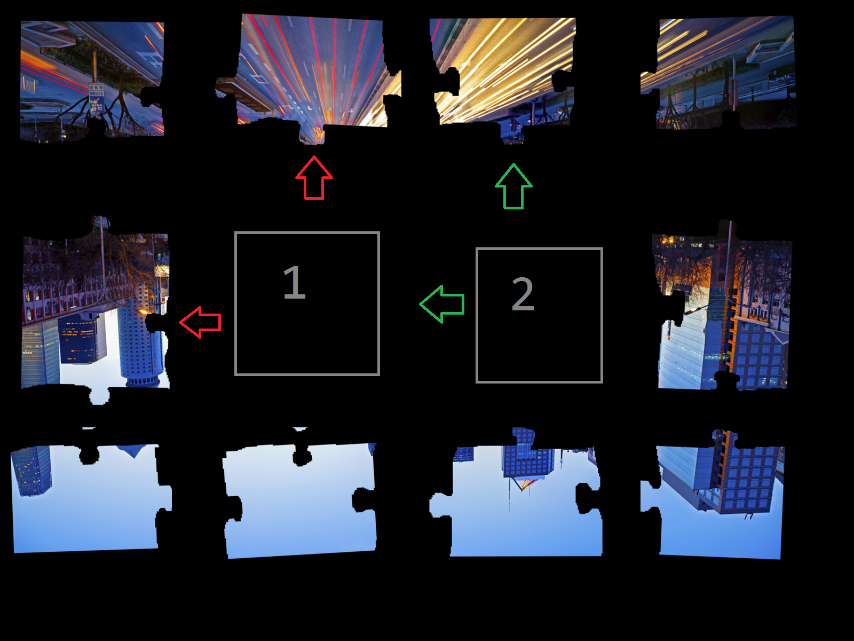
- Potem czas na porównanie metodą BRUTEFORCE wyznaczonych obszarów na obrazach.

- Po czym obliczany jest threshold jako maksymalną odległość między obszarami uznanymi za podobne. Po jego ustaleniu należy odfiltrować prawidłowe połączenia od tych przypadkowych bądź błędnych.

- Ostatnią rzeczą jest wyznaczenie czworokąta wykorzystując odfiltrowane cechy.

* 1. Metoda układania środka

Mając ułożoną ramkę pozostałe elementy mogą być iteracyjnie dopasowane do niej. Na rysunku nr 9 przedstawiono jak zostają znalezione kolejne elementy.



Rys. 9: schemat przedstawiający metodę układania środka.

Algorytm przedstawia się następująco:

- brany jest pierwszy element ze zbioru elementów środkowych (tzn. takich bez żadnej krawędzi płaskiej)

- Przestawiany jest na współrzędne w najwyższym rzędzie w skrajnym lewym położeniu

- Rozpatrywana jest kolejno każda z czterech możliwych rotacji elementu.

- Podobieństwo danego elementu w danej rotacji jest określane jako suma podobieństwa jego lewej krawędzi z prawą stroną elementu ramki i podobieństwa górnej krawędzi z krawędzią dolną elementu na górze ramki.

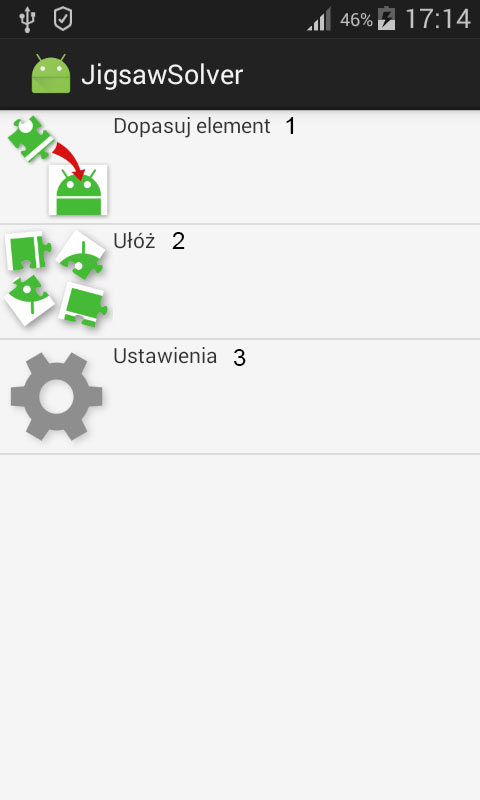
- Po sprawdzeniu każdego elementu zostaje dopasowany ten o najlepszym dopasowaniu, w określonej rotacji. Odpowiednie krawędzie są łączone.

- Dopasowany element jest usuwany z puli i dalej jest porównywane miejsce o jedno miejsce w prawo lub w następnym rzędzie, jeśli dany rząd został już wypełniony.

- Po zakończeniu działania algorytmu cały obszar ramki powinien zostać wypełniony i każdy element powinien zająć swoje miejsce.

1. Instrukcja obsługi
   1. Menu główne

W menu głównym możemy wybrać czy chcemy dopasować pojedynczy element do obrazka czy ułożyć rozsypane puzzle.  Elementy menu głównego opisane zostały na rysunku nr 10.



Rys. 10: menu główne aplikacji.

Opis rysunku  :

1. Przycisk dopasowania - otwiera ekran dopasowania elementu, w którym można odnaleźć miejsce puzzla na zdjęciu obrazka przedstawiającego obraz po ułożeniu puzzli.
2. Przycisk ułóż -  otwiera ekran układania puzzli, gdzie elementy ze zdjęcia są dopasowywane oraz układane zgodnie z obrazkiem z pudełka.
3. Przycisk ustawienia - przenosi do ekranu ustawień.

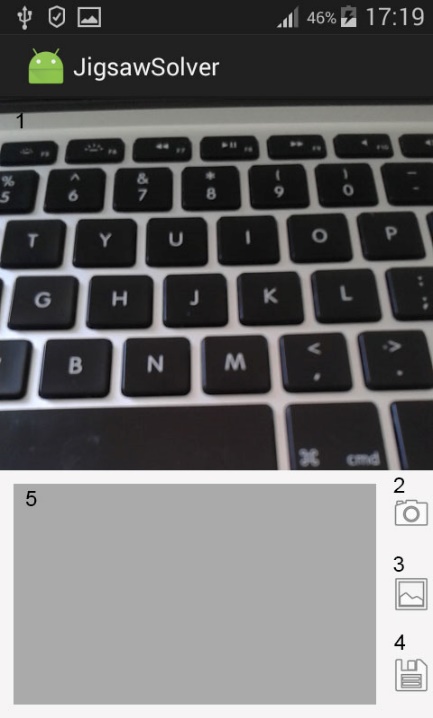
Sposób obsługi:

Należy wybrać odpowiedni przycisk w zależności od efektu jaki chce się osiągnąć.

* 1. Ekran dopasowywania elementu do obrazka

Na tym widoku można zrobić zdjęcie obrazka z pudełka na puzzle przedstawiającego ułożone puzzle, a następnie wykonać zdjęcie pojedynczego puzzla lub grupy ułożonych puzzli, aby znaleźć ich miejsce na obrazku z pudełka.

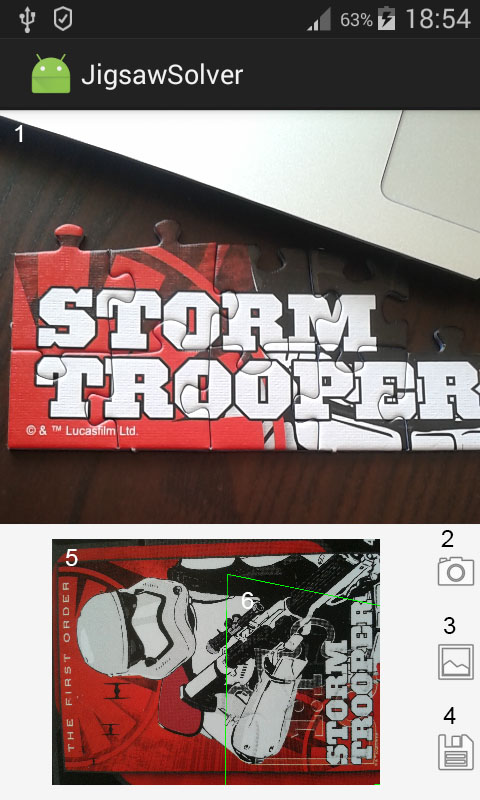
Ekran niewypełniony przedstawiony jest na rys 11, natomiast ekran z elementem oraz zaznaczonym obszarem odpowiadającym elementowi na obrazku z pudełka na rys 12 .



Rys. 11: ekran dopasowania elementu do obrazka bez wypełnionych widoków na zdjęcia.

Opis rysunku :

1. Podgląd na żywo z kamery. Po kliknięciu w dowolne miejsce podglądu kamera zatrzyma się oraz wykona zdjęcie, które będzie wyświetlane w podglądzie. Ten element przeznaczony jest dla zdjęcia pojedynczego elementu lub grupy ułożonych elementów.
2. Przycisk aparatu, po kliknięciu którego otwarte zostanie okno aparatu, w którym wykonać będzie można zdjęcie obrazka znajdującego się na pudełku lub dowolnego innego rysunku przedstawiającego widok z puzzli. Wykonane zdjęcie zostanie wyświetlone w polu nr 5.
3. Przycisk galerii. Po wybraniu tego przycisku otworzone zostanie okno z wyborem miejsca, z którego ma zostać pobrany obraz puzzli. Do wyboru jest galeria aparatu lub dokumenty na urządzeniu. Po wyborze źródła jest ono otwierane, a następnie można wybrać obraz. Wybrany obraz zostanie wyświetlony w polu nr 5.
4. Przycisk zapisu. Zapisuje aktualny obraz z pola nr 5 (jeżeli w polu znajduje się wczytany obraz). Po wybraniu przycisku wyświetla się dialog z możliwością wpisania nazwy zapisywanego obrazu. Następnie obraz zapisany zostanie w pamięci urządzenia.
5. Miejsce na obraz przedstawiający ułożone puzzle. Na znajdującym się w tym polu obraz nałożona zostanie zielona ramka czworokąta wyznaczająca  fragment obrazu odpowiadający puzlowi lub grupie puzzli z pola 1.



Rys. 12: ekran dopasowania elementu do obrazka bez wypełnionych widoków na zdjęcia.

Opis rysunku :

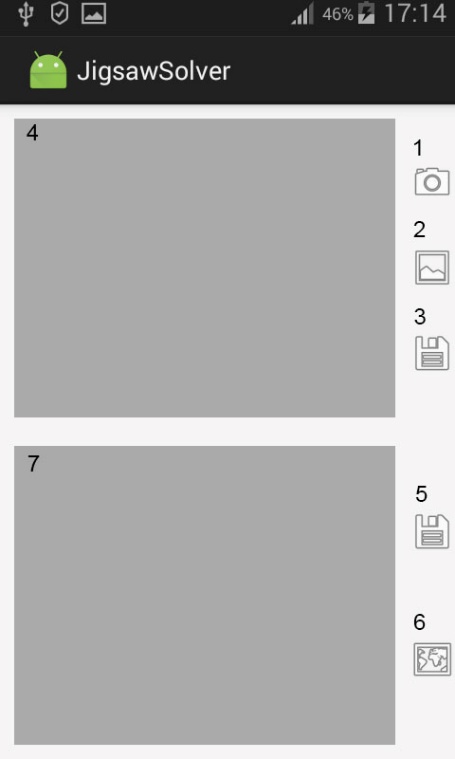
1. Widok zdjęcia elementu lub grupy elementów, które chcemy dopasować.
2. Przycisk aparatu - funkcjonalność jw.
3. Przycisk Galerii - funkcjonalność jw.
4. Przycisk zapisu - funkcjonalność jw.
5. Widok pełnego obrazu, którego element chcemy dopasować.

Sposób obsługi:

Najpierw należy wybrać zdjęcie pełnego obrazu z galerii lub wykonać takie zdjęcie aparatem. Następnie należy najechać podglądem kamery na element/elementy rewersem do dołu, który/e chcemy dopasować i kliknąć w dowolnym miejscu podglądu, aby wykonać zdjęcie tego/tych elementu/elementów. Ważne jest aby element/y zajmował/y większą część zdjęcia oraz aby zdjęcie nie było prześwietlone lub pod zbyt ostrym kątem względem powierzchni, na które znajdują się puzzle. takie działanie utrudnia dopasowanie. Ważne także, aby na zdjęciu znajdował się cały puzzel lub wszystkie puzzle, oraz aby żaden nie był ucięty. Następnie po chwili na rysunku pełnego obrazu zostanie narysowana rama czworokąta wskazująca umiejscowienie elementu/elementów w całościowym widoku. Obraz wraz z ramą można zapisać na pamięci urządzenia.

* 1. Ekran układania puzzli

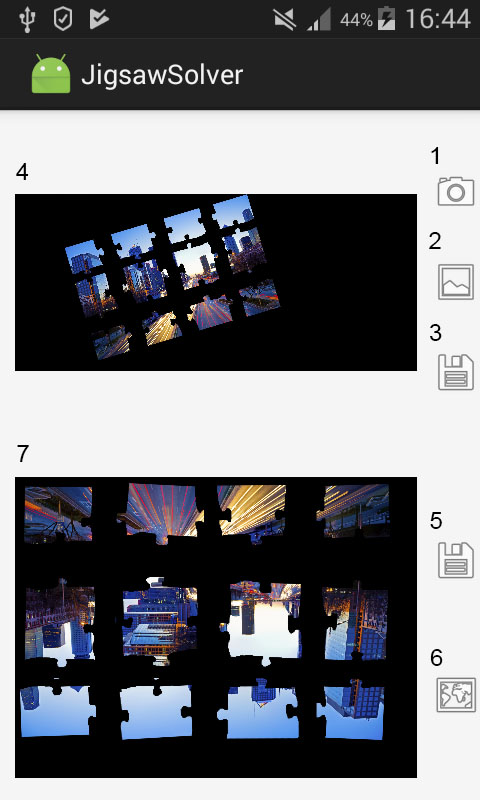
W tym widoku można zrobić zdjęcie rozłożonych rewersem do dołu puzzlom, a następnie wyświetlony zostanie rysunek z puzzlami ułożonymi w prawidłowym porządku. Ekran układania puzzli bez obrazków przedstawiony został na rys 13, a po ułożeniu puzzli na rys 14 .



Rys. 13: ekran układania puzzli z niewypełnionymi widokami.

Opis rysunku :

1. Przycisk aparatu, otwiera okno aparatu, gdzie można wykonać zdjęcie, które wstawione zostanie do pola nr 4.
2. Przycisk galerii, umożliwia wybór zdjęcia do pola nr 4 z pamięci urządzenia.
3. Przycisk zapisu elementów. Umożliwia zapisanie  obrazu z pola 4 do pamięci telefonu. Działa tak samo jak przycisk zapisu w widoku dopasowania elementu do obrazka.
4. Widok przeznaczony na zdjęcie rozłożonych rewersem do góry kompletnych puzzli, które chcemy ułożyć. Aby wstawić zdjęcie należy wybrać przycisk nr 1 lub nr 2, a następnie wybrać lub wykonać zdjęcie.
5. Przycisk zapisu obrazu ułożonych puzzli. Umożliwia zapisanie  obrazu z pola 7 do pamięci telefonu. Działa tak samo jak przycisk zapisu elementów, z tą różnicą, że zapisuje do pamięci widok z pola nr 7.
6. Przycisk układania. Po wybraniu tego przycisku aplikacja przetworzy obraz z pola nr 4, a następnie wyświetli ułożone puzzle w polu nr 7.
7. Widok przeznaczony na wynikowy obraz ułożonych puzzli z widoku 4.



Rys. 14: ekran układania puzzli po ułożeniu.

1. Przycisk aparatu - funkcjonalność jw.

2 .Przycisk galerii - funkcjonalność jw.

3. Przycisk zapisu elementów - funkcjonalność jw.

4. Widok przeznaczony na zdjęcie rozłożonych rewersem do góry kompletnych puzzli, które chcemy ułożyć. Aby wstawić zdjęcie należy wybrać przycisk nr 1 lub nr 2, a następnie wybrać lub wykonać zdjęcie.

5. Przycisk zapisu obrazu ułożonych puzzli - funkcjonalność jw.

6. Przycisk układania. Po wybraniu tego przycisku aplikacja przetworzy obraz z pola nr 4, a następnie wyświetli ułożone puzzle w polu nr 7.

7. Widok przeznaczony na wynikowy obraz ułożonych puzzli z widoku 4.

Sposób obsługi:

Na początku należy wykonać zdjęcie kompletu puzzli na idealnie czarnym tle, lub wybrać zdjęcie z tłem zamienionym na  tło czarne. Puzzle muszą być rewersem do dołu oraz nie mogą na siebie nachodzić ani się stykać. Zdjęcie musi być ostre oraz najlepiej wykonane pod kątem 90° względem powierzchni, na której znajdują się puzzle. Najlepiej aby puzzle wypełniały jak największą część zdjęcia, ale aby żaden nie był przycięty lub pominięty. Kluczowe jest także aby zdjęcie nie było prześwietlone, a puzzle nie były zniszczone. Następnie można zapisać wykonane zdjęcie w pamięci telefonu lub od razu kliknąć przycisk układania, aby po chwili w polu nr 7 wyświetlone zostały puzzle ułożone tak, że aby ułożyć  puzzle należy każdy puzzel połączyć z sąsiadującymi puzzlami krawędzi, po której stronie znajduje się sąsiadujący puzzel. Układ nie zawsze ma orientację zgodną z obrazkiem z opakowania, co nie wpływa na poprawność ułożenia puzzli.

1. Podsumowanie
   1. Zrealizowane założenia

- Zbudowanie czytelnego interfejsu użytkownika

- Zaimplementowanie funkcji obsługi elementów sprzętowych (kamery, wczytywanie z pliku)

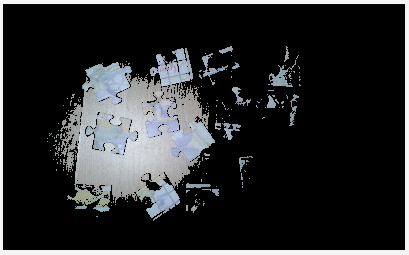
- Opracowanie i zaimplementowanie metody układania elementów.

- Opracowanie i zaimplementowanie metody dopasowywania obrazu jako części innego obrazu oraz określenie położenia tej części w pełnym  obrazie.

* 1. Niezrealizowane założenia

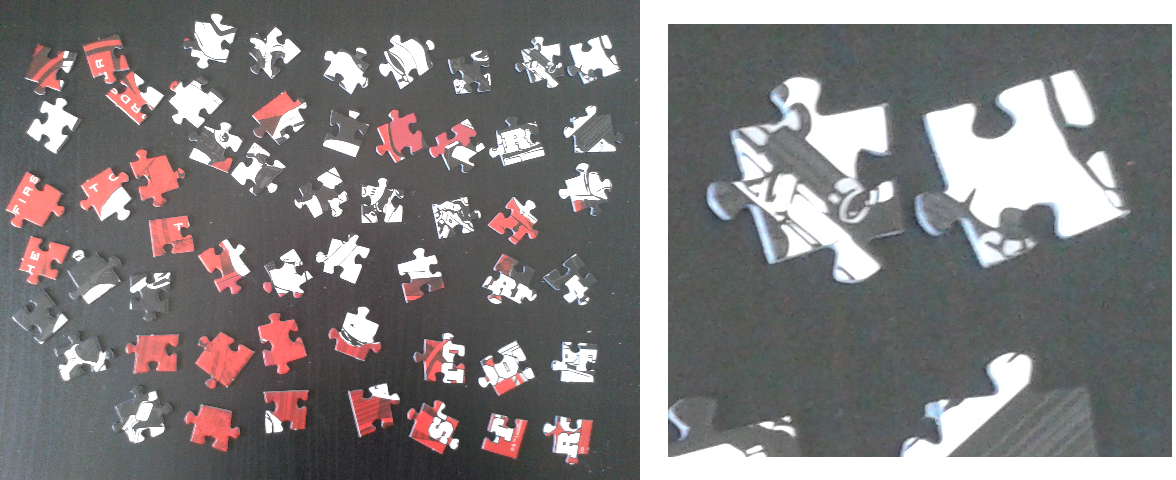
1. wydzielanie puzzli z obrazu

Wydzielanie rzeczywistych elementów z obrazu wiąże się z wieloma problemami. Jednym z nich jest konieczność wykonania zdjęcia dobrej jakości, co nie zawsze jest możliwe na danym urządzeniu. Do tego nie istnieje uniwersalny algorytm usuwania tła z obrazu, który działałby dokładnie i pewnie.



Rys. 15: przykładowy efekt działania przy ściśle określonych parametrach progowania.

1. Dopasowywanie elementów (częściowo)



Rys. 16: niedoskonałości zdjęć.

Rzeczywiste zdjęcia puzzli są zniekształcone, między innymi przez błąd perspektywy, rozmazanie spowodowane drganiem ręki, czy też przez niedoskonałość samego sprzętu. Dopasowywanie elementów jest przez to utrudnione, bo odpowiednie pary krawędzi na zdjęciu mogą mieć znacznie różny kształt. Problem pogłębia się wraz ze wzrostem liczby elementów układanki. Liczba potencjalnych par rośnie wykładniczo razem z ilością elementów. Dodatkowo rzeczywiste puzzle nie zawsze są dokładnie wycięte i często awers nie jest idealnie płaski. Konsekwencją jest obecność na zdjęciach szarego wypełnienia fizycznego kawałka, jeśli jest nachylony względem kamery. Ekstrakcja rzeczywistego konturu jest wtedy utrudniona. Pewnym rozwiązaniem wydaje się uwzględnienie kolorów przy określaniu stopnia dopasowania, jednak takie rozwiązanie nie daje spodziewanych efektów. Kolor elementu na zdjęciu jest zależny od jego oświetlenia, więc od jego położenia. Dodatkowo elementy pasujące do siebie mogą mieć skrajnie różne kolory. Jest to szczególnie widoczne w układankach przedstawiających krajobraz z płaską linią horyzontu, która często przebiega niedaleko krawędzi.