



POLITECHNIKA POZNAŃSKA

WYDZIAŁ INFORMATYKI I TELEKOMUNIKACJI

Instytut Informatyki

Praca dyplomowa magisterska

**WPŁYW DANYCH WEJŚCIOWYCH NA EFEKTYWNOŚĆ
DOSTĘPNYCH ALGORYTMÓW ROZPOZNAWANIA
OBRAZU W PROCESIE UKŁADANIA PUZZLI**

inż. Patryk Jasiak, 136394

Promotor
dr inż. Ariel Antonowicz

POZNAŃ 2024

Spis treści

| | |
|--|-----------|
| Streszczenie | 1 |
| 1 Wstęp | 3 |
| 1.1 Cel i zakres pracy | 4 |
| 1.2 Plan pracy | 4 |
| 2 Charakterystyka wizji komputerowej | 5 |
| 2.1 Rys historyczny | 5 |
| 2.2 Trendy rozwojowe | 9 |
| 2.2.1 Generative adversarial networks (GAN) | 9 |
| 2.2.2 Wyjaśnialna sztuczna inteligencja | 10 |
| 2.2.3 Segmentacja semantyczna | 10 |
| 2.2.4 Uczenie transferowe | 11 |
| 2.2.5 Rozwój konwolucyjnych sieci neuronowych | 12 |
| 2.2.6 Przetwarzanie brzegowe oraz w czasie rzeczywistym | 13 |
| 2.2.7 Przetwarzanie w trzech wymiarach | 13 |
| 3 Opis wybranych algorytmów analizy obrazu | 15 |
| 3.1 Oddzielenie puzzli od tła | 15 |
| 3.1.1 Maska kolorów | 15 |
| 3.1.2 Scale-invariant feature transform (SIFT) | 16 |
| 3.1.3 Oriented FAST and Rotated BRIEF (ORB) | 16 |
| 3.1.4 Segmentacja semantyczna | 17 |
| 3.2 Szukanie konturów w obrazie | 17 |
| 3.3 Skalowanie elementów | 18 |
| 3.4 Lokalizacja wierzchołków i krawędzi puzzli | 18 |
| 3.4.1 Wstępne odrzucenie części czworokątów | 19 |
| 3.4.2 Sposób klasyfikacji potencjalnych narożników | 20 |
| 3.5 Korekcja perspektywy i rotacji | 21 |
| 4 Zbiór danych testowych | 23 |
| 4.1 Kryteria doboru obrazów | 23 |
| 4.2 Szczegółowa analiza porównawcza wybranych danych wejściowych | 24 |
| 4.3 Sposób oceny efektywności algorytmów | 26 |
| 5 Sposoby klasyfikacji elementów | 29 |

| | |
|---|-----------|
| 6 Metody oceny dopasowania elementów | 33 |
| 6.1 Porównanie elementów między sobą | 34 |
| 6.2 Wstępne układanie elementów | 34 |
| 6.3 Ocena całkowitego dopasowania | 37 |
| 7 Proces umieszczania puzzli na obrazie wynikowym | 39 |
| 7.1 Punkty łączenia ze sobą elementów | 39 |
| 7.2 Przykładowe rozwiązania | 41 |
| 8 Podsumowanie i analiza wyników | 43 |
| 8.1 Zestawienie algorytmów zastosowanych na różnych etapach przetwarzania | 43 |
| 8.2 Prezentacja wyników działania algorytmów | 44 |
| 8.2.1 Oddzielanie elementów od tła | 44 |
| 8.2.2 Lokalizacja narożników | 48 |
| 8.2.3 Dopasowanie krawędzi | 48 |
| 9 Wnioski | 51 |
| Literatura | 53 |

Streszczenie

Proces układania puzzli jest złożony i wymaga zastosowania różnych algorytmów na poszczególnych etapach. W pracy został opisany wpływ, jaki na wybrane algorytmy analizy obrazu mają dane wejściowe. Zbiór testowy zawierał obrazy zawierające zestawy puzzli z różną liczbą elementów sfotografowane w różnych warunkach oświetleniowych oraz na zmiennym tle. Z przeprowadzonych badań wynika, że do oddzielania puzzli od tła najlepiej nadaje się segmentacja semantyczna z użyciem zdjęć zestawów puzzli jako dane uczące. Następnie należy zastosować algorytm analizujący kształt konturów, aby zlokalizować wierzchołki. Do analizy krawędzi powinny zostać użyte zarówno jej dane geometryczne, jak i kolorystyczne. Najlepsze łączenie elementów w całość zapewnia algorytm układający obramowanie, a następnie wypełniający je. Na jakość otrzymanych rozwiązań najbardziej wpływa liczba elementów układanki. Puzzle dwudziesto elementowe udaje się ułożyć poprawnie w większości przypadków. Natomiast trzydziestopięcio elementowe są już znacznie bardziej problematyczne. W przypadku większej liczby elementów trudność sprawia nawet znalezienie obramowania. Jest to spowodowane tym, że wystarczy jedna źle rozpoznana krawędź, aby ułożenie nie było możliwe, a większa ilość elementów zwiększa ilość krawędzi i w konsekwencji szanse na pojawienie się błędów.

Abstract

The process of putting together a jigsaw puzzle is a compound problem. It is necessary to use different algorithms on each step of it. This work focuses on how input data influences selected algorithms. The test set consisted of images of puzzle sets with different number of pieces, photographed in variable light conditions and on various backgrounds. Results of conducted experiments suggests that to separate puzzles from background, it is the best to use semantic segmentation with images of puzzle sets as training data. Most accurate corner detection can be achieved with algorithm that analyzes shape of contours. Shape and color data should be used to compare edges to each other and find the best matches. The most successful algorithm to fit pieces together was to start by creating a border and filling it in the next step. The number of puzzle pieces in an image is a parameter that influences the result the most. Puzzle sets with twenty or fewer elements can be solved in most cases, but sets with thirty-five elements or more start to become a lot more problematic. In case of larger puzzle sets, it is difficult to find a border. The cause of this is that one wrongly classified edge can make finding a solution impossible, and the more elements in a set, the greater the chance of an error.

Rozdział 1

Wstęp

Wizja komputerowa jest kluczowym elementem wielu nowoczesnych technologii, takich jak: samochodów autonomicznych, robotyki czy biometrii. Bez systemów rozpoznawania i analizy widzianych obiektów, obraz z kamer nie jest wystarczający, aby na jego podstawie wydobyć z niego wszystkie zawarte w nim informacje. Algorytmy używane w celu rozpoznawania obiektów są wrażliwe na zmiany warunków otoczenia takich jak oświetlenie, kolor tła, zmiana perspektywy czy orientacji. W szczególności kiedy analizowane obiekty różnią się od siebie jedynie drobnymi szczegółami. Z tego powodu dane wejściowe mają znaczący wpływ na efektywność procesu rozpoznawania obrazu. Analiza charakterystycznych cech obiektów pozwala znaczco usprawnić takie procesy jak kontrola jakości lub rozpoznawanie twarzy.

W przypadku układanek takich, jakie puzzle, które są przykładem zbioru elementów o niskim zróżnicowaniu, niewielkie różnice pomiędzy fragmentami mają znaczący wpływ na nich wzajemne oddziaływanie, a w konsekwencji proces rozwiązywania układanki. Z tego powodu wykorzystanie zbioru danych, który zawiera te same zestawy puzzli w różnym otoczeniu, pozwala sprawdzić szerokie spektrum potencjalnie problematycznych warunków.

Wybór puzzli jako zbioru testowego, pozwala na testowanie algorytmów analizy obrazu, stosujących różne podejścia. Do odnajdywania elementów układanki nadają się algorytmy analizujące krawędzie, ale również takie, które skupiają się na dopasowaniu kolorów, możliwe jest również użycie sztucznej inteligencji. Każdy rodzaj algorytmów sprawdza się inaczej w różnych warunkach.

Algorytmy dopasowujące kolory, działają lepiej, gdy tło obrazu jest w kolorze innym niż elementy układanki. Natomiast analiza konturów nie jest zbyt dobrym wyborem, gdy krawędzie elementów są rozmyte lub w tle obrazu widoczne są dodatkowe kontury. Zastosowanie sieci neuronowych wymaga stworzenia odpowiedniego modelu, dobranie reprezentatywnych elementów do trenowania go jest trudne i wymaga wielu przykładów uczących. Dodatkowo, jeśli dane wejściowe będą mocno odbiegać jakością od zbioru treningowego. Model nie będzie osiągać zadowalających rezultatów. Poprzez sprawdzenie różnych rodzaju algorytmów, możliwe jest dobranie kilku z nich, w taki sposób, aby na kolejnych etapach klasyfikacji elementów układanki wykorzystać pożądane cechy każdego z nich i stworzenia podejścia pozwalającego na zminimalizowanie wpływu jakości danych wejściowych na działanie systemu.

Poprawne odnajdywanie i charakteryzowanie elementów układanki, nie jest jedynym istotnym aspektem podczas procesu układania. Ważne jest również, jakie rozwiązanie zostanie zaproponowane, w sytuacji kiedy nie wszystkie puzzle zostały zawarte na zdjęciu. Estymowanie poprawnego rozwiązania bez posiadania wszystkich danych jest potrzebne, aby systemy wizji komputerowej mogły być używane w nieprzewidywalnym środowisku, jest to zagadnienie szczególnie istotne w przypadku samochodów autonomicznych lub robotów kroczących.

W aplikacjach wymagających szybkości działania niektóre rozwiązania mogą okazać się zbyt czasochłonne, mimo wysokiej dokładności, dlatego czas uzyskania rozwiązania jest kolejnym istotnym elementem wpływającym na ocenę efektywności algorytmów.

1.1 Cel i zakres pracy

Celem pracy jest przeprowadzenie analizy algorytmów rozpoznawania obrazu, które mogą zostać użyte do rozpoznania puzzli, w celu oceny wpływu danych wejściowych na efektywność procesu automatycznego rozwiązywania układanki. Zakres pracy obejmuje:

1. Przegląd ogólnie dostępnych rozwiązań dotyczących rozpoznawania obrazu w kontekście układania puzzli.
2. Opracowanie zestawu danych testowych różnych pod względem jakości obrazu, liczby puzzli, sposobie rozmieszczenia, tła, itp.
3. Implementacja wybranych algorytmów rozpoznawania obrazu i porównanie ich w oparciu o zrealizowany zestaw testowy.
4. Analiza wpływu poszczególnych składowych na skuteczność analizowanego algorytmu.
5. Integracja wybranych algorytmów przetwarzania obrazu w celu wskazania prawdopodobieństwa poprawnego dopasowania.
6. Opisanie wyników badań i opracowanie wniosków z przeprowadzonej analizy.

Oprócz przedstawionych celów pracy, jej zadaniem jest również odpowiedzenie na poniższe pytania badawcze:

- Jakie algorytmy analizy obrazu dają najlepsze efekty w różnych warunkach?
- Jaki sposób analizy elementów układanki dostarcza najwięcej informacji?
- Jakie są efektywne podejścia do automatycznego układania puzzli?

1.2 Plan pracy

Praca została podzielna na 9 rozdziałów:

- Rozdział 1 - opisuje cel, zakres pracy oraz metodologię.
- Rozdział 2 - zawiera historyczne i aktualne trendy w dziedzinie wizji komputerowej.
- Rozdział 3 - charakteryzuje wybrane algorytmy analizy obrazu oraz ich rolę w procesie klasyfikacji elementów.
- Rozdział 4 - wyjaśnia sposób konstruowania zbioru testowego.
- Rozdział 5 - przedstawia sposoby klasyfikacji elementów układanki.
- Rozdział 6 - skupia się na metodach oceny dopasowania puzzli do siebie.
- Rozdział 7 - opisuje sposób automatycznego rozwiązywania puzzli i umieszczanie ich w obrazie wynikowym.
- Rozdział 8 - zawiera porównanie wyników wybranych algorytmów analizy obrazu.
- Rozdział 9 - zawiera wnioski oraz możliwe usprawnienia przygotowanego programu.

Rozdział 2

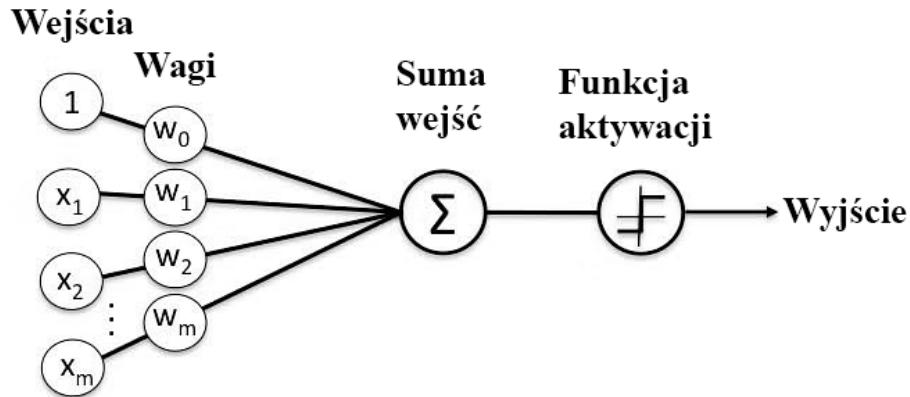
Charakterystyka wizji komputerowej

Wizja komputerowa jest interdyscyplinarną dziedziną informatyki zajmującą się tym, w jaki sposób komputery mogą wydobyć informację z cyfrowych obrazów i materiałów wideo, które następnie mogą być używane w procesach decyzyjnych. Rodzaj uzyskanych danych ma duże znaczenie i jego wybór jest kluczowy, aby informacje były użyteczne. Zadania, którymi najczęściej zajmuje się wizja komputerowa to między innymi [25]:

- **klasyfikacja obrazów** pozwala trafnie określić, do jakiej grupy należą. Znajduje na zdjĘciu najbardziej istotny obiekt, na podstawie którego następuje grupowanie. Może być stosowana np. w klasyfikacji zdjęć dodawanych przez użytkowników do serwisów agregujących je i dzielących tematycznie;
- **detekcja obiektów** umożliwia rozpoznawanie i lokalizowanie w obrazie obiektów określonego typu. Zebrane w ten sposób dane mogą zostać następnie porównane ze wzorcem, co może okazać się użyteczne np. do wykrywania wadliwych elementów na linii produkcyjnej;
- **przetwarzanie obrazu płaskiego na scenę 3D** wizja komputerowa znajduje również zastosowanie w rekonstrukcji scen 3D z obrazów 2D. Zastosowanie tej technologii pozwala np. na tworzenie trójwymiarowych obiektów z serii zdjęć pod różnymi kątami, innym przykładem może być, odtworzenie geometrii pokoju na podstawie zdjęcia.

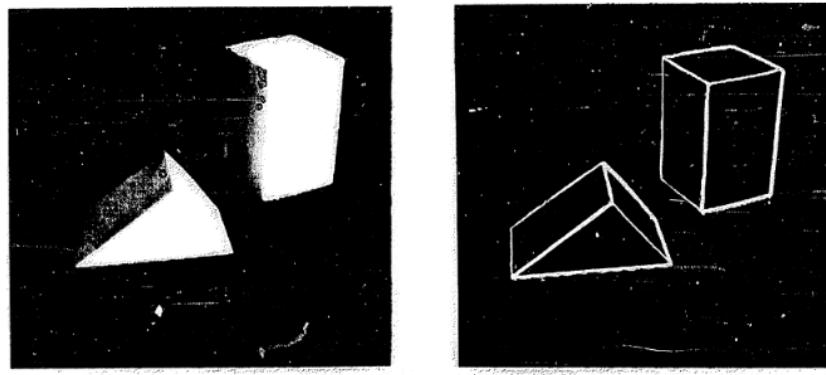
2.1 Rys historyczny

Wizja komputerowa ma swoje początki w dziedzinie sztucznej inteligencji, w latach 50 ubiegłego wieku. W okresie tym powstało wiele koncepcji, którymi wizja komputerowa zajmuje się do tej pory. Jedną z pierwszych prób zastosowania sieci neuronowych, do analizy obrazu był zaproponowany przez Franka Rosenblatta perceptron, którego schemat widoczny jest na rysunku 2.1. Zaprezentowany przez niego program potrafił po uprzednim nauczeniu, za pomocą danych uczących, rozpoznawać różnicę pomiędzy dwoma kształtami: kołem i kwadratem [4]. Była to również próba naśladowania sposobu, w jaki ludzie rozpoznają obiekty w obrazie.



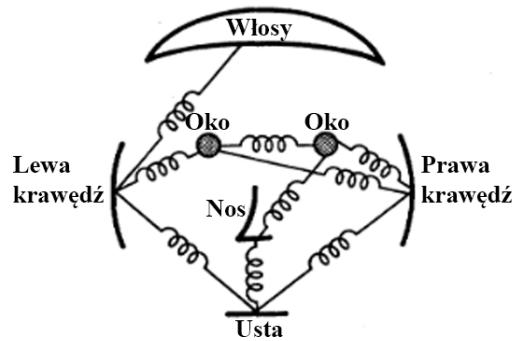
RYSUNEK 2.1: Schemat perceptronu [4]

Kolejnym krokiem w rozwoju rozpoznawania obrazu były między innymi algorytmy detekcji krawędzi. W 1963 roku Lawrence Roberts stworzył program, który był pierwszą próbą interpretowania płaskich obrazów, jako trójwymiarowych scen. Bazując głównie na percepceji głębi oraz detekcji krawędzi program ten rekonstruował obiekty 3D z obrazu, a następnie umieszczał je na scenie w odpowiednich miejscach [11]. Elementy, z których składały się sceny, były prostymi bryłami geometrycznymi, jednak połączenie w analizie różnych cech fotografowanych obiektów było znaczącym krokiem w rozwoju rozpoznawania obrazu. Rysunek 2.2 zawiera przykładową scenę oraz jej odtworzenie przez program.



RYSUNEK 2.2: Przykład działania programu stworzonego przez Roberts'a, z lewej strony znajduje się obraz bazowy, a po prawej odtworzony przez program [11].

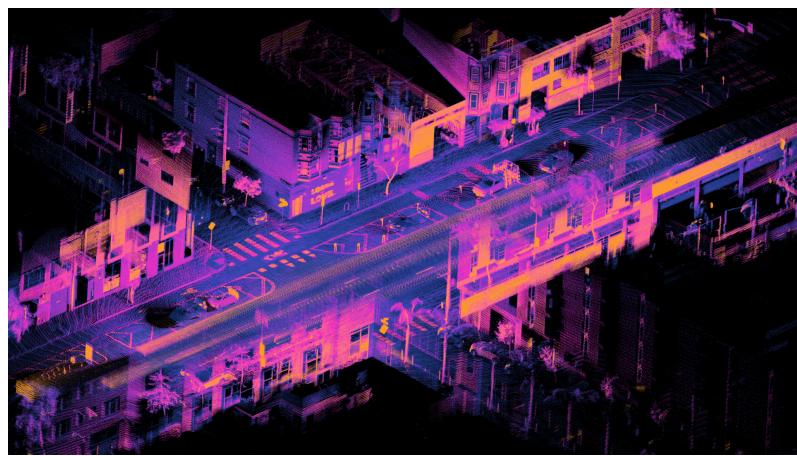
Znaczącym punktem w rozwoju wizji komputerowej było modelowanie oraz rozpoznawanie obiektów bazując na ich elementach oraz relacjach między nimi [1]. Przedstawiony przez M. Fischlera algorytm miał za zadanie identyfikować poszczególne części twarzy, bazując na relacjach pomiędzy nimi. Przedstawione rozwiązanie potrafiło z dużą dokładnością rozpoznawać włosy, oczy oraz obrus twarzy, natomiast miało trudności z umiejscowieniem ust i nosa. Podejście bazujące na pictorial structures pozwoliło na powstanie wielu bardziej zaawansowanych algorytmów rozpoznawania obiektów, które skupiają się na znajdowaniu cech szczególnych i ich wzajemnym rozmieszczeniu. Rozmieszczenie elementów twarzy wraz z ich wzajemnymi relacjami widoczne jest na rysunku 2.3.



RYSUNEK 2.3: Relacje pomiędzy częściami twarzy zastosowane w algorytmie M. Fischlera [1]

Inną rozwijającą się w tym samym czasie gałęzią wizji komputerowej było optyczne rozpoznawanie znaków. Dziedzina ta zajmuje się automatyczną konwersją tekstów pisanych ręcznie lub drukowanych na format komputerowy. W latach siedemdziesiątych Ray Kurzweil stworzył maszynę, pozwalającą na skanowanie drukowanego tekstu, a następnie czytanie go na głos, dzięki syntezatorowi mowy. Produkt ten miał za zadanie pomóc osobom niewidomym [22].

Mimo powstawania wielu rozwiązań dedykowanych dla problemów wizji komputerowej dziedzina ta nigdy nie przestała być ściśle związaną ze sztuczną inteligencją. Wraz ze wzrostem popularności uczenia maszynowego na początku XXI wieku możliwości wizji komputerowej zostały znacznie poszerzone. Zmiany te były spowodowane również ciągłym zwiększeniem się mocy obliczeniowej komputerów. Rozbudowane algorytmy rozpoznawania obiektów wymagają wykonywania wielu złożonych operacji matematycznych na obrazie. Przyspieszenie obliczeń pozwala pracować na obrazach o większej rozdzielcości i w konsekwencji poprawić jakość oraz użyteczność wyników przetwarzania. W skutek czego znacznie efektywniejsze stało się również stosowanie wizji komputerowej dla materiałów wideo, co do tej pory stanowiło wyzwanie z uwagi na znacznie większą ilość danych niż w przypadku pojedynczego obrazu. Skutkiem tych zmian była możliwość użycia wizji komputerowej w aplikacjach wymagających działania w czasie rzeczywistym. Technologia ta znalazła zastosowanie między innymi w robotyce. SLAM, czyli simultaneous localization and mapping, to technika używana do tworzenia i aktualizowania map nieznanego otoczenia, jednocześnie pozwalająca na lokalizację obiektu tworzącego mapę w nowo odkrytej przestrzeni [24]. Rysunek 2.4 przedstawia wizualizację chmury punktów uzyskanej przy użyciu lidaru.



RYSUNEK 2.4: Chmura punktów stworzona przy pomocy SLAM, z wykorzystaniem lidaru [24]

Rozwój sztucznej inteligencji w kolejnych latach nie zwalniał, konwolucyjne sieci neuronowe oraz uczenie głębokie, które rozwijały się równolegle z uczeniem maszynowym, stały się najczę-

niej rozwijaną technologią z tej dziedziny. Jedną z możliwych przyczyn wzrostu zainteresowania nimi jest to, że znajdują zastosowanie w wielu odrębnych dziedzinach. Odpowiednio skonstruowana sieć neuronowa, posiadająca wystarczającą ilość danych uczących jest w stanie podejmować decyzje z dużą dokładnością. W 2016 roku AlphaGo program komputerowy stworzony do gry w gry planszowe, pokonał w meczu jednego z czołowych graczy go Lee Sedola, wygrywając cztery z pięciu rozegranych gier [19]. Uznawane jest to za kluczowy moment w rozwoju sztucznej inteligencji, z uwagi na dużą liczbę możliwych ruchów w grze. Toczy się ona na planszy 19 na 19 pól, dodatkowo ocena pozycji jest skomplikowana, gdyż zależy od rozmieszczenia kamieni na polu gry. Dlatego klasyczne podejście, opierające się na drzewach decyzyjnych, już po kilku ruchach wymagają ogromnej mocy obliczeniowej.

Automatyczne generowanie obrazów jest dziedziną pokrewną do wizji komputerowej, jednak skupia się ona na tworzeniu nowych grafik z uwzględnieniem parametrów podawanych przez użytkownika. Na przestrzeni ostatnich kilku lat zagadnienie znacząco się rozwinęło. Projekty takie jak: Stable Diffusion, DALL-E lub ChatGPT umożliwiają tworzenie złożonych obrazów na podstawie podanego tekstu opisu sceny, która ma zostać wygenerowana. Technologia ta bazuje na uczeniu głębokim, wytwarzane przez nią obrazy bazują na dostarczonych wcześniej danych uczących. Po dostarczeniu odpowiedniego zestawu danych, zawierającego obrazy stworzone przez konkretnego artystę, sztuczna inteligencja potrafi bardzo dobrze odtwarzać jego styl [18]. Przykładem takiego zastosowania generatora obrazu jest rysunek 2.5, ukazujący grafikę w stylu Vincenta van Gogh.



RYSUNEK 2.5: Grafika stworzona przez Stable Diffusion, inspirowana obrazem "Gwiaździsta noc" Vincenta van Gogh [opracowanie własne]

Głębokie sieci neuronowe są również wykorzystywane do detekcji obiektów, przykładem takiego zastosowania tej technologii jest architektura YOLO (You Only Look Once). Jej nazwa opisuje sposób, w jaki przebiega detekcja poszczególnych obiektów, wszystkie z nich identyfikowane są w jednej iteracji. Takie podejście pozwala zmniejszyć ilość obliczeń oraz zwiększa efektywność uczenia się sieci. Rozwiązanie to stosowane jest w wielu dziedzinach, przykładem zastosowania może być wykrywanie i odróżnianie od siebie nut na pięciolinii [9]. Jest to proces podobny do rozpoznawania puzzli, jednak skupia się na wydobyciu z obrazu danych innego rodzaju, w przypadku nut, najważ-

niejszą informacją jest to, jaki dźwięk oznacza. Natomiast w przypadku wykrywania puzzli ważne jest dobre odwzorowanie kształtu jego krawędzi.

Problem automatycznego układania puzzli ze względu na liczbę kroków w nim występujących pozwala zastosować bardzo dużo różnych sposobów jego rozwiązania. Jeden z przykładowych sposobów na jego przeprowadzenie został opisany w publikacji Travisa V. Allena [2]. Przedstawiony proces można podzielić na 5 etapów:

- wyodrębnienia elementów układanki od tła obrazu;
- pozyskanie informacji dotyczących kształtu elementu;
- skategoryzowanie krawędzi elementu na podstawie kształtu do jednej z trzech kategorii: płaska, wypukła, wklęsła;
- lokalne dopasowanie elementów-znalezienie najlepszego dopasowanie dwóch krawędzi;
- globalne dopasowanie-ułożenie układanki.

Rozwiążanie to będzie punktem odniesienia podczas tworzenia systemu automatycznego rozwiązywania układanki.

2.2 Trendy rozwojowe

Najważniejsze trendy rozwojowe w dziedzinie wizji komputerowej to:

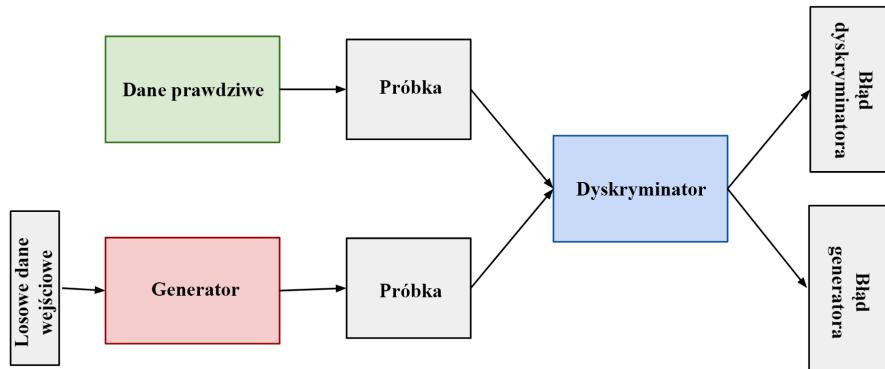
- zastosowanie generative adversarial networks;
- wyjaśnalna sztuczna inteligencja;
- segmentacja semantyczna;
- uczenie transferowe;
- dalszy rozwój konwolucyjnych sieci neuronowych i uczenia głębokiego;
- przetwarzanie brzegowe oraz w czasie rzeczywistym;
- przetwarzanie danych trojwymiarowych.

2.2.1 Generative adversarial networks (GAN)

Sieci GAN to kolejny przełom w dziedzinie sztucznej inteligencji ostatnich 10 lat. Składają się one z dwóch części [23]:

- generator jest odpowiedzialny za tworzenie nowych danych, starając się, aby były jak najbardziej zbliżone do danych uczących. Efekty jego pracy są następnie przekazywane do dyskryminatora;
- dyskryminator rozróżnia między danymi prawdziwymi, a wygenerowanymi przez drugą część programu. Generator jest następnie karany, za stworzenie danych rozróżnialnych od prawdziwych. Dyskryminator jest szkolony, aby coraz lepiej rozpoznawać dane generowane.

W trakcie uczenia się taka sieć jest w stanie sama się doskonalić, gdyż efekty pracy każdej z jej części wpływają na ocenę efektywności drugiej, co prowadzi do rywalizacji między nimi. Dzięki zastosowaniu tej technologii można znaczco zmniejszyć zapotrzebowanie na dane uczące, ponieważ nowe dane treningowe powstają w procesie uczenia się sieci. Dane produkowane przez sieci typu GAN są trudne do rozróżnienia od rzeczywistych, w konsekwencji czego sieci te znajdują zastosowanie w wielu dziedzinach, takich jak sztuka generatywna, synteza obrazów, czy generowanie realistycznych tekstów. Jednocześnie mogą one stanowić źródło danych uczących dla innych typów sieci. Rysunek 2.6 przedstawia schemat podstawowej sieci GAN oraz kolejność przetwarzania danych w procesie trenowania sieci.



RYSUNEK 2.6: Schemat podstawowej wersji sieci GAN [23].

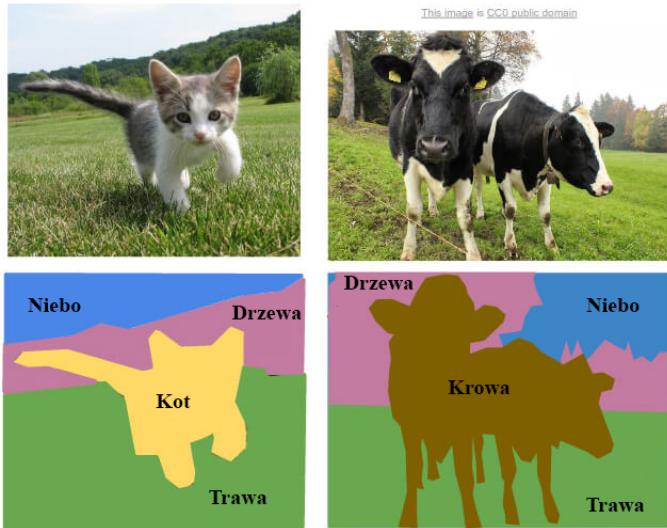
2.2.2 Wyjaśnialna sztuczna inteligencja

Jednym z problemów wynikających z rozwoju sztucznej inteligencji jest to, że sposoby podejmowania decyzji przez tego typu systemy stają się coraz mniej zrozumiałe dla człowieka, wynika to głównie ze zwiększenia liczby warstw w sieciach neuronowych oraz zastosowaniu coraz bardziej złożonych algorytmów przetwarzania. W konsekwencji prowadzi to do postrzegania takich systemów jako czarne skrzynki, a wyniki ich działania stają się bardzo trudne do przewidzenia i wyjaśnienia. Ma to szczególne znaczenie w przypadku systemów, od których zależy ludzkie życie. W takim przypadku zastosowanie niezrozumiałej sieci neuronowej, może prowadzić do nieufności i strachu przed wprowadzeniem rozwiązania opierającego się o nią do użytku. Jednym z możliwych rozwiązań jest rozwój wyjaśnialnej sztucznej inteligencji. Technologia ta skupia się na tworzeniu systemów, w których dokładność może być dobrze estymowana, bazując na danych wejściowych. Dodatkowo ważne jest, aby proces decyzyjny można było prześledzić, na przykład wizualizując, które neurony zostały aktywowane na poszczególnych warstwach. Ostatnim istotnym elementem jest to, aby decyzje podejmowane przez system były zrozumiałe dla człowieka, w tym celu wymagana jest edukacja osób pracujących ze sztuczną inteligencją, w sposób podejmowania przez nią decyzji [26].

2.2.3 Segmentacja semantyczna

Najczęściej stosowane podejście w zagadnienniu rozpoznawania obiektów, polega na znajdowaniu i rozpoznaniu poszczególnych obiektów danej klasy w obrazie. Jednym z odmiennych sposobów na rozwiązanie tego problemu jest segmentacja semantyczna, której zadaniem jest przypisanie każdemu pikselowi obrazu klasy [14]. W tym celu stosowane są odpowiednio skonstruowane sieci konwolucyjne, w których każda warstwa ma za zadanie rozpoznanie jednego typu obiektów. Podejście to jest przydatne, kiedy granice między obiektemi mają skomplikowany kształt, nato-

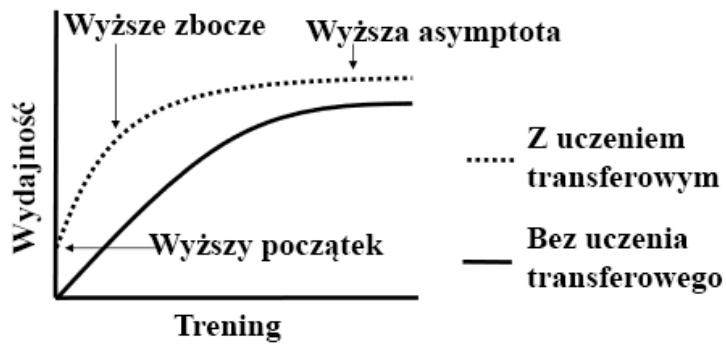
miast możliwą wadą, jest fakt, że w podstawowej wersji algorytm ten nie jest w stanie rozróżniać między instancjami tej samej klasy. Problem ten obrazuje rysunek 2.7, gdzie dwie krowy zostały skategoryzowane jako jedna instancja klasy krowa. Dlatego w przypadku kiedy ważne jest, aby zidentyfikować konkrety obiekt danej klasy rozwiązań to nie będzie odpowiednie.



RYSUNEK 2.7: Wynik działania segmentacji semantycznej [10].

2.2.4 Uczenie transferowe

Ludzie oraz inne istoty rozumne posiadają umiejętność używania wiedzy zdobytej w trakcie wykonywania jednego zadania, aby rozwiązać nowy, podobny problem. Natomiast powszechnie stosowane algorytmy uczenia maszynowego nie posiadają tej cechy. Każdy nowy problem wymaga uczenia się od nowa. Uczenie transferowe, to metoda pozwalająca na generalizację i przenoszenie wiedzy między różnymi problemami. Pozwala to na przyspieszenie uczenia się algorytmów oraz na uzyskanie przez nie wyższej maksymalnej trafności [13], co przedstawiono na rysunku 2.8.



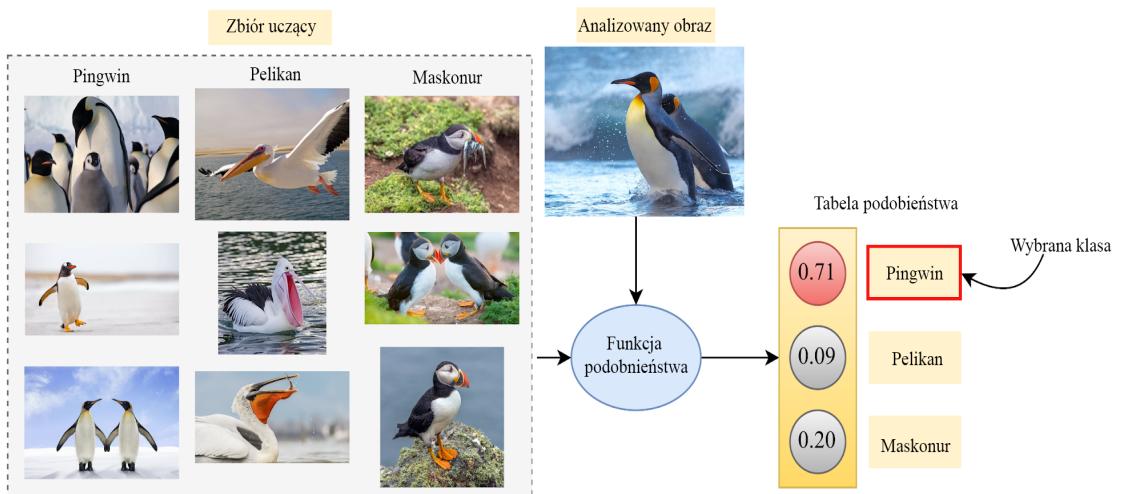
RYSUNEK 2.8: Trzy sposoby, w które uczenie transferowe może poprawić uczenie się sieci neuronowej [13].

Istotnym problemem w uczeniu transferowym jest unikanie przekazywania informacji, które mogą negatywnie wpływać na skuteczność sieci. Doskonałe uczenie transferowe pozwalałoby wybierać kiedy zadanie, z którego odbywa się transfer, jest wystarczająco pokrewne do nowego zadania, jednak w praktyce jest to trudne do stwierdzenia. Jeśli zadania będą za bardzo od siebie odbiegać, transfer może pogorszyć wyniki przetwarzania.

2.2.5 Rozwój konwolucyjnych sieci neuronowych

Pomimo wspomnianych wcześniej przełomów w dziedzinie uczenia głębokiego, rozwój sieci konwolucyjnych nie zwalnia. Dlatego można spekulować, że w przyszłości pojawią się kolejne algorytmy wykorzystujące tę technologię [6]:

- **self-attention mechanisms** - uczenie nadzorowane, które jest obecnie bardzo popularnym sposobem tworzenia modeli, wymaga dużej ilości odpowiednio przygotowanych danych. Wymagane jest skategoryzowanie danych wejściowych, aby na ich podstawie ocenić efektywność sieci i trenować model. Uczenie nienadzorowane nie wymaga kategoryzowanych danych, natomiast podejście takie sprawdza się zwykle gorzej niż uczenie nadzorowane. Podejście łączącym zalety obu wyżej wymienionych, może być zastosowanie mechanizmu self-attention. Pozwala on modelowi na samodzielna identyfikację ważnych cech wejściowych. Self-attention znalazło już zastosowanie w przetwarzaniu języka naturalnego, to dzięki zastosowaniu tego mechanizmu modele takie jak BERT [7] są zdolne do rozumienia kontekstu poszczególnych słów;
- **few-shot learning** - nowatorskie podejście do konstruowania sieci. Model uczony jest na niewielkiej liczbie obiektów dla każdej klasy. Ten sposób uczenia pozwala na stworzenie modelów, które potrafią skutecznie generalizować i klasyfikować nowe, niewidziane wcześniej dane wejściowe [8]. Modele takie znajdują zastosowanie w procesie klasyfikacji danych wejściowych, nie wymagając jednocześnie dużych ilości już skategoryzowanych danych. Przykład procesu klasyfikacji obrazu ptaka przedstawiono na rysunku 2.9;
- **uczenie transferowe i wyjaśnalna sztuczna inteligencja** - to dwa ze wspomnianych wcześniej rozwiązań, które mogą w najbliższym czasie znaleźć zastosowanie w sieciach konwolucyjnych. Uczenie transferowe pozwoliłoby modelom na osiągnięcie lepszych rezultatów. Natomiast wyjaśnalna sztuczna inteligencja jest niezbędna, by skomplikowane, wielowarstwowe modele mogły być zrozumiałe przez człowieka, co jest niezbędne, aby zwiększyć zaufanie społeczeństwa do nowych rewolucyjnych technologii.



RYSUNEK 2.9: Przykładowe działanie modelu wykorzystującego few-shot learning do skategoryzowania niewidzianego wcześniej obrazu ptaka [17].

2.2.6 Przetwarzanie brzegowe oraz w czasie rzeczywistym

Modele stosowane w uczeniu maszynowym często wymagają dużych zasobów pamięciowych, aby rozpocząć przetwarzanie. Wielkość modelu wpływa również na liczbę operacji, które muszą zostać wykonane, żeby otrzymać rezultaty, w wyniku czego większość obecnie stosowanych rozwiązań wykorzystuje komputery wyspecjalizowane w przetwarzaniu tego rodzaju danych. Jest to rozwiązanie, które zapewnia możliwość otrzymywania wyników w krótkim czasie, natomiast w aplikacjach, gdzie urządzeń korzystających z modelu jest wiele, scentralizowana struktura wymaga do działania możliwości przekazywania dużych ilości danych do systemu je przetwarzającego. Jest to szczególnie problematyczne w przypadku systemów związanych z Internetem Przedmiotów, gdzie tysiące urządzeń może komunikować się między sobą, w konsekwencji czego ilość przesyłanych danych rośnie jeszcze bardziej, a komputery zajmujące się przetwarzaniem modeli wymagają coraz więcej zasobów, aby generować wyniki w tym samym czasie. Możliwym rozwiązaniem tego problemu jest przetwarzanie brzegowe z wykorzystaniem uczenia maszynowego. Jeśli każde z połączonych urządzeń wykona nawet początkową fazę przetwarzania, znaczco odciąży to cały system. Wraz ze zwiększaniem się mocy obliczeniowej mikrokontrolerów, możliwości implementowania w nich modeli sztucznej inteligencji są coraz większe, w konsekwencji czego coraz większa ilość przetwarzania może odbywać się w urządzeniu brzegowym, bez konieczności komunikacji z systemem. Wzrastający trend w stosowaniu sztucznej inteligencji w mikrokontrolerach obrazuje na przykład rosnącą popularność TinyML, biblioteki pozwalającej na implementację modeli uczenia maszynowego w mikrokontrolerach [16].

Problem długiego oczekiwania na wynik przetwarzania jest jeszcze bardziej zauważalny, w sytuacjach gdy potrzebne jest przetwarzanie w czasie rzeczywistym. W systemach czasu rzeczywistego nawet niewielkie opóźnienia mogą powodować poważne konsekwencje. Dlatego, aby zastosować w nich sztuczną inteligencję, wymagane jest, żeby obliczenia były wykonywane w przewidywalnym czasie, co nie zawsze jest możliwe, gdy wykorzystywane jest połączenie internetowe.

2.2.7 Przetwarzanie w trzech wymiarach

Obecnie stosowanie wizji komputerowej ograniczone jest głównie do przetwarzania obrazów 2D, jednak takie dane nie zawsze są wystarczające. Analiza wolumetrycznych danych jest szczególnie potrzebna np. w medycynie. Obrazy powstające w wyniku przeprowadzenia tomografii komputerowej analizowane osobno, dostarczają znacznie mniej informacji, niż można byłoby uzyskać podczas przetwarzania w trzech wymiarach. Autonomiczne roboty oraz pojazdy to inne zastosowanie, w którym przetwarzanie obrazów 2D nie jest niewystarczające. Trzeci wymiar jest wymagany, aby pojazd mógł prawidłowo określić swoją lokalizację w przestrzeni. Dodanie głębi do systemu wizji komputerowej jest jednak skomplikowane. Obraz tylko z jednej kamery nie pozwala prawidłowo określić perspektywy i odległości. Dlatego wymagane jest zastosowanie większej liczby sensorów oraz bardziej złożone przetwarzanie pozwalające na wspólne przetwarzanie obrazów z kilku źródeł [3]. W przypadku odpowiednio przygotowanych danych wejściowych istnieje ryzyko błędnego określenia perspektywy. Przykładem tego może być pokój Amesa przedstawiony na rysunku 2.10.



RYSUNEK 2.10: Pokój Amesa, złudzenie optyczne obrazujące jak w odpowiednich warunkach perspektywa obrazu może być myląca. Dzięki odpowiedniemu układowi ścian osoba stojąca po prawej stronie wydaje się znaczaco mniejsza niż osoba po lewej [20].

Rozdział 3

Opis wybranych algorytmów analizy obrazu

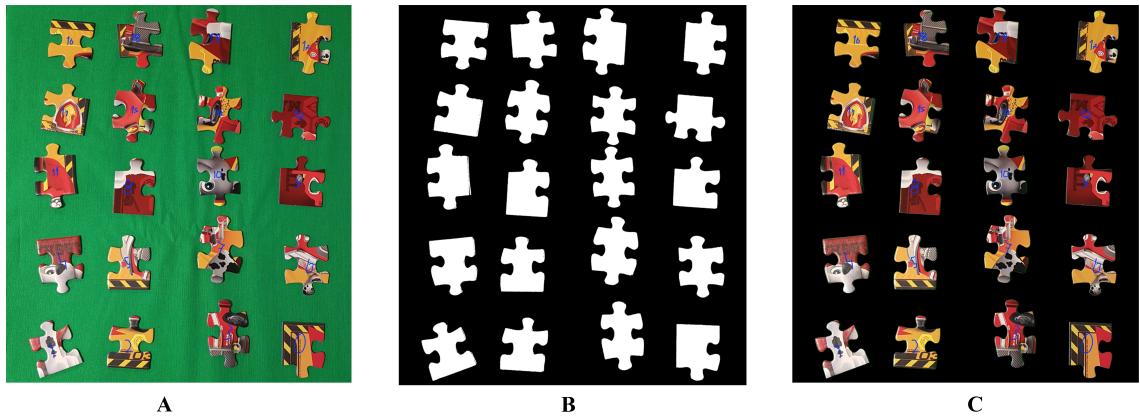
Proces analizy obrazu jest zadaniem wieloetapowym, dlatego ważne jest, aby dobrać odpowiednie algorytmy do każdego z kroków przetwarzania. Efekty algorytmów stosowanych do podobnych celów mogą znaczco różnić się od siebie. Kluczowe jest, aby podczas dobierania algorytmów mieć na uwadze to, żeby wyniki ich działania były kompatybilne między sobą.

3.1 Oddzielenie puzzli od tła

Pierwszym krokiem w procesie automatycznego rozwiązywania puzzli jest odseparowanie ich od tła, jest to też krok, w którym jakość danych wejściowych ma największe znaczenie, ponieważ jeśli puzzle zostaną niepoprawne odizolowane, dalsze kroki nie będą możliwe. Dlatego kluczowe jest, aby algorytmy oddzielające puzzle działały poprawnie na obrazach różnej jakości. Detekcja obiektów oraz oddzielanie ich od tła może być wykonana na wiele sposobów.

3.1.1 Maska kolorów

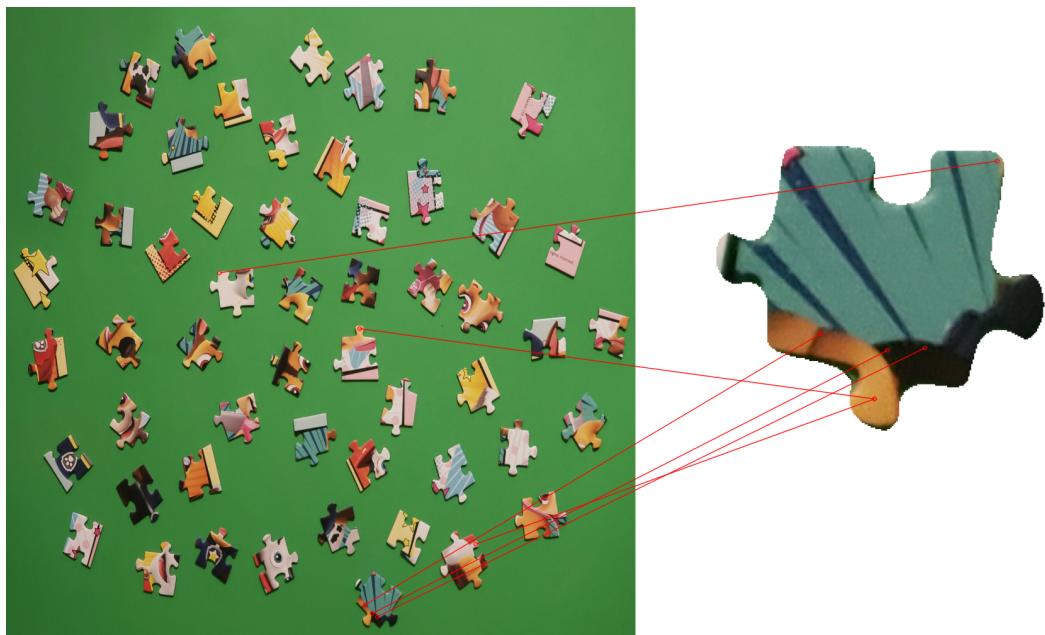
To jeden z najprostszych sposobów wykrywania istotnych elementów w obrazie. Algorytm usuwa z obrazu wejściowego wszystkie piksele, których kolor znajduje się w wybranym przedziale. Pozostałe piksele stanowią efekt działania algorytmu. Bardzo istotną wadą takiego podejścia jest to, że kolor tła nie może pojawiać się w obiektach, które mają zostać wyodrębnione. W przeciwnym wypadku fragmenty istotnych obiektów mogą zostać uznane za tło i zostaną usunięte. Innym problemem jest fakt, że bez zastosowania dodatkowego filtrowania, krawędzie wyodrębnianych obiektów mogą być postrzępione, ponieważ kolor tła miesza się w pewnym stopniu z krawędziami kontrastujących z nim obiektów. Podobny efekt ma rzucanie przez obiekty cienia. Dlatego zalecane jest rozmycie obrazu wejściowego, przed zastosowaniem maski kolorów. Rysunek 3.1 przedstawia efekt działania maski kolorów na zdjęciu puzzli na jednokolorowym tle. Obraz z lewej strony (A) to dane wejściowe, z których następnie zostaje stworzona maska (B), efektem działania algorytmu jest nałożenie maski na obraz wejściowy, co zostało pokazane na obrazie po prawej (C).



RYSUNEK 3.1: Efekt działania algorytmu maski kolorów [opracowanie własne].

3.1.2 Scale-invariant feature transform (SIFT)

Bardziej złożonym algorytmem detekcji obiektów jest SIFT. Algorytm ten został stworzony przez Davida Lowe w 1999 roku. Działanie algorytmu polega na ekstrakcji punktów kluczowych tzw. key points, ze zbioru obrazów referencyjnych. Następnie podczas analizy nowego obrazu punkty kluczowe są porównywane z wynikami przetwarzania obrazów referencyjnych [5]. W podstawowej formie algorytm ten sprawdza się najlepiej w odnajdywaniu ze zbioru uczącego, w kiedy skala, bądź rotacja obiektu jest zmieniona. Rysunek 3.2 obrazuje odnalezione punkty kluczowe puzzla znajdującego się w prawym górnym rogu. Można zauważyć, że algorytm jest bardziej czuły na punkty kluczowe wewnętrzne puzzla, niż jego kształt.



RYSUNEK 3.2: Odnajdowanie puzzla w obrazie przy pomocy algorytmu SIFT [opracowanie własne].

3.1.3 Oriented FAST and Rotated BRIEF (ORB)

ORB to alternatywa dla algorytmów takich jak SIFT, ale w przeciwieństwie do niego nie jest on opatentowany, dlatego można używać go bezpłatnie w produktach komercyjnych. Zasada działania bazuje na użyciu FAST (Features from Accelerated Segment Test), aby znaleźć punkty kluczowe w obrazie. Kolejnym krokiem jest poprawienie lokalizacji punktów kluczowych za po-

mocą algorytmu detekcji wierzchołków (Harris corner detection). Punkty kluczowe mają następnie przypisaną orientację oraz obliczane są ich binarne deskryptory. Służy do tego Algorytm BRIEF(Binary Robust Independent Elementary Features). Do późniejszego porównywania punktów kluczowych w obrazie docelowym z obrazem bazowym wykorzystuje się najczęściej odległość Hamminga.

3.1.4 Segmentacja semantyczna

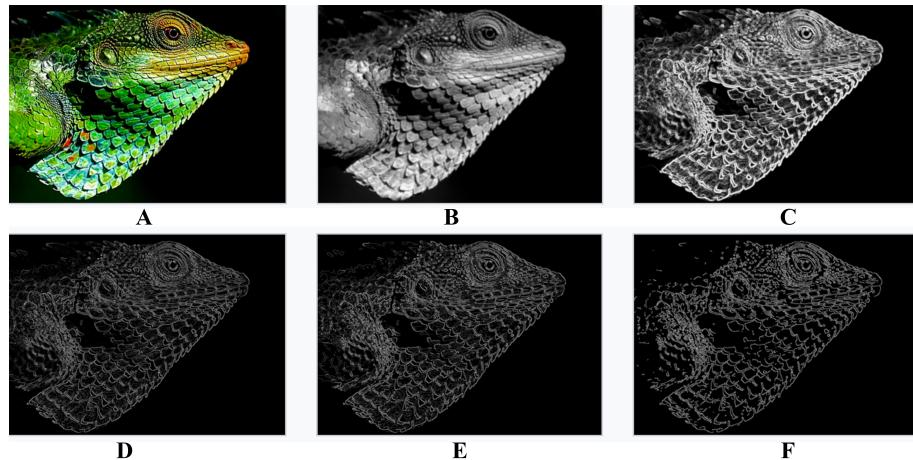
Wspomniana wcześniej segmentacja semantyczna nadaje się dobrze do detekcji puzzli w przygotowanym zbiorze testowym, gdyż elementy układanki nie powinny nachodzić na siebie, co sprawia, że brak rozróżnienia między instancjami szukanej klasy obiektów (elementów układanki) nie stanowi problemu. Natomiast skomplikowany kształt krawędzi puzzli powinien zostać dobrze zachowany.

3.2 Szukanie konturów w obrazie

Kontury są istotną cechą analizowanych fragmentów układanki, ponieważ jednym z głównych aspektów oceny dopasowania puzzli do siebie jest to, czy ich kształt do siebie pasuje. Aby przeanalizować topologię krawędzi, należy wydobyć z obrazu kontur przedstawionego na nim elementu układanki. Jednym z najczęściej stosowanych algorytmów służących do odnajdowania konturów jest filtr Canny'ego. Jego działanie można opisać w 6 krokach:

1. **Redukcja kolorów obrazu do skali szarości.**
2. **Rozmycie Gaussa** ma na celu wygładzenie obrazu i pozbycie się szumu.
3. **Odnalezienie gradientu intensywności obrazu** - algorytm używa czterech filtrów, aby odnaleźć pionowe, poziome i skośne krawędzie.
4. **Próg wielkości gradientu** jest to etap, w którym znajdowane są miejsca z największymi zmianami intensywności, krawędzie z niższą intensywnością są tłumione.
5. **Podwójny próg** jest stosowany, aby usunąć piksele błędnie rozpoznane jako krawędzie, w tym celu stosowane są dwa progi intensywności, próg wysoki oraz próg niski. Piksel z intensywnością powyżej progu wysokiego zostaje sklasyfikowany jako mocny, jeśli intensywność jest poniżej wysokiego progu, a powyżej niskiego, jest on uznawany jako słaby, natomiast piksele poniżej niskiego progu są odrzucone.
6. **Śledzenie krawędzi** istnieje ryzyko, że nie wszystkie słabe piksele należą do rzeczywistych krawędzi, dlatego aby odrzucić niepoprawne, należy sprawdzić, czy sąsiadują one z silnym pikselem, jeśli nie, zostają odrzucone.

Na rysunku 3.3 przedstawiono efekty kolejnych etapów działania algorytmu Canny'ego, obraz A to dane wejściowe, natomiast wynikiem działania algorytmu jest obraz F.



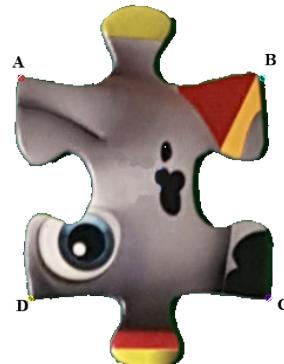
RYSUNEK 3.3: Kolejne etapy przetwarzania obrazu w algorytmie detekcji krawędzi [21].

3.3 Skalowanie elementów

Opcjonalnym krokiem poprzedzającym analizę puzzli jest przeskalowanie ich do wybranej wielkości. Etap ten pozwala zminimalizować wpływy rozdzielczości obrazu wejściowego na dalsze etapy automatycznego układania. Aby zachować wzajemne proporcje elementów, skalowanie musi odbywać się z zastosowaniem stałej skali, a nie stałej wielkości obrazu wynikowego. W zastosowanym algorytmie dobór skali elementów, dokonywany jest, bazując na wielkości pierwszego analizowanego elementu, pozwala to na obliczenie skali, którą należy zastosować, aby elementy miały ostatecznie zadaną wielkość, niezależnie od tego, jaka była rozdzielczość danych wejściowych. Rozwiązań to ułatwia porównywanie działanie kolejnych kroków procesu automatycznego układania puzzli dla danych wejściowych o różnym rozmiarze.

3.4 Lokalizacja wierzchołków i krawędzi puzzli

Podczas łączenia ze sobą elementów układanki ważne jest znalezienie cech, które są wspólne dla każdego z nich. Do grupy tej należą rogi oraz krawędzie. Każdy puzzel posiada 4 wierzchołki, które służą jako punkty separujące krawędzie od siebie, rysunek 3.4 obrazuje znalezione narożniki elementu. Szukanie rogów jest procesem złożonym, ze względu na możliwą dużą różnorodność w kształcie poszczególnych elementów układanki. Dlatego skuteczność algorytmów stosowanych w tym celu zależy w dużej mierze od tego, jak bardzo elementy układanki różnią się od siebie nawzajem.



RYSUNEK 3.4: Puzzel ze zidentyfikowanymi rogami, oznaczonymi kolejno A, B, C, D [opracowanie własne].

Do dalszych badań wybrano dwa algorytmy:

- **opierający się na algorytmie detekcji wierzchołków Harrisa** - polega na użyciu algorytmu detekcji narożników Harrisa z rosnącą wielkością bloku;
- **analizujący kształt konturu** - danymi wejściowymi dla algorytmu jest zbiór punktów stanowiących kontur badanego elementu, uszeregowany tak, aby zachować ciągłość konturu, piksele sąsiadujące ze sobą są kolejnymi elementami zbioru. Algorytm przegląda zbiór punktów, obliczając kąt pomiędzy trzema kolejnymi, jeśli kąt znacząco odbiega od 180° , punkt środkowy zostaje skategoryzowany jako możliwy róg.

Efektem przetwarzania obu z tych podejść jest zbiór punktów, skategoryzowanymi jako potencjalne narożniki układanki, w dalszej części rozdziału nazywany zbiorem kandydatów. W kolejnym etapie z punktów wybierany jest zbiór czterech, które tworzą najlepsze rogi.

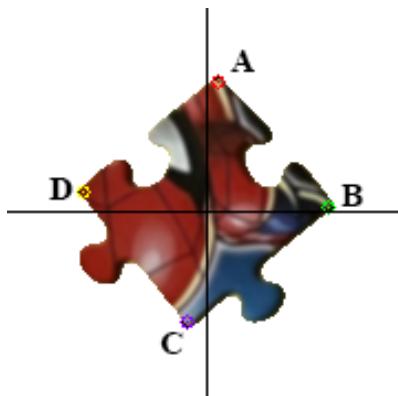
3.4.1 Wstępne odrzucenie części czworokątów

Ze względu na możliwą liczbę punktów w zbiorze kandydatów sprawdzanie każdej możliwej kombinacji punktów szybko staje się trudne obliczeniowo. Dlatego, aby przyspieszyć ten proces, zastosowano algorytm mający na celu odrzucenie większości mniej prawdopodobnych punktów, kroki działania algorytmu opisano poniżej. Algorytm zostaje wykonany tyle razy, ile wynosi liczebność zbioru kandydatów, za każdym razem z innym punktem początkowym. Efektem działania tego kroku jest otrzymanie zbioru potencjalnych czworokątów stanowiących wierzchołki puzzla.

Działanie algorytmu składa się z następujących kroków:

1. Wybranie punktu początkowego ze zbioru kandydatów, nazywanego n ;
2. Obliczenie i zapisanie kąta α pomiędzy odcinkiem z punktem początkowym w punkcie $(n-1)$ i końcowym w punkcie n . A prostą $f(x) = 0$;
3. Porównanie obliczonego kąta z poprzednim kątem, jeśli różnica jest odpowiednio blisko 90° punkt n zostaje zapisany jako potencjalny narożnik;
4. Sprawdzenie, czy zostały odnalezione cztery narożniki, jeśli tak algorytm zwraca wynikowy zbiór punktów. W przeciwnym wypadku następuje wybranie kolejnego punktu $(n+1)$ i przejście do kroku 2.

Proszym rozwiązaniem, może być podział zbioru kandydatów na cztery podzbiory pod względem ich położenia w obrazie, na te znajdujące się najbliżej lewego górnego, prawego górnego oraz dolnych rogów obrazu, a następnie tworzenie czworokątów wybierając po jednym punkcie z każdego z nich. To rozwiązanie nie sprawdza się jednak w przypadku kiedy puzzel w obrazie ustawiony jest w specyficzny sposób. Rysunek 3.5 przedstawia ulożenie puzzla, w którym wierzchołki A i B znajdują się w najbliższej tego samego narożnika obrazu.



RYSUNEK 3.5: Przykład puzzla, w którym dwa wierzchołki znajdują się w tym samym podzbiorze [opracowanie własne].

3.4.2 Sposób klasyfikacji potencjalnych narożników

Każdy z otrzymanych w poprzednim kroku wierzchołków jest poddawany klasyfikacji w celu znalezienia najlepszej kombinacji punktów. Cechami poddawanymi analizie są:

- stosunek pola powierzchni czworokąta do jego obwodu — znajduje czworokąty o najbardziej kwadratowym kształcie;
- stosunek pola powierzchni czworokąta do pola powierzchni całego obrazu — odpowiada za odrzucenie zbyt małych czworokątów;
- stosunek długości przeciwnieległych krawędzi — pozwala na wybranie bardziej prostokątnych czworokątów;
- stosunek długości przekątnych — przekątne powinny mieć jak najbardziej zbliżone długości;
- kąt pomiędzy krawędziami — umożliwia odrzucenie czworokątów, w których kąty pomiędzy krawędziami najbardziej odbiegają od 90° .

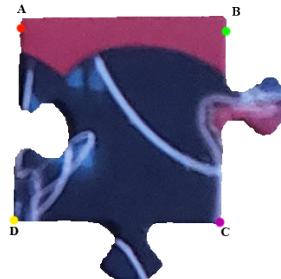
Krawędzie służą jako punkty łączące puzzle ze sobą, dlatego ich analiza dostarcza najwięcej informacji przydatnych w procesie automatycznego układania. W przypadku najczęściej występujących prostokątnych puzzli, w których każdy element łączy się maksymalnie z czterema sąsiadującymi, istnieją trzy możliwe typy krawędzi:

- **wypukłe** zawierające wypustkę pasującą do krawędzi wklęszych;
- **wklęsłe** zawierające dziurę, w którą pasuje wypustka wypukłej krawędzi;
- **płaskie** tworzące krawędź układanki.

Średnia wysokość krawędzi

Decyzja o tym, jaki typ przypisać analizowanej krawędzi, podejmowana jest, bazując na odległości punktów tworzących krawędź w stosunku do odcinka, który tworzą jej wierzchołki. Aby uprościć obliczenia, krawędź przed przystąpieniem do tego etapu jest obracana w taki sposób, aby jej wierzchołki leżały na prostej $f(x) = 0$. Następnie sumowana jest odległość w osi y dla każdego punktu krawędzi. Jeśli odległość ta jest blisko zera, oznacza to, że krawędź jest płaska, jeśli odległość jest dodatnia, krawędź jest wypukła, natomiast kiedy odległość jest ujemna, krawędź zostaje oznaczona jako wklęsła.

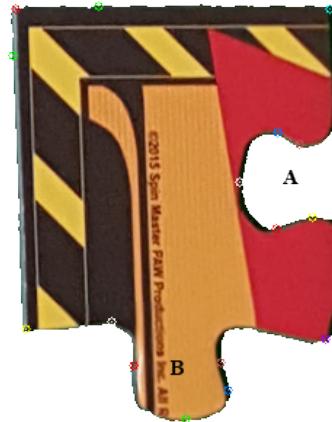
Rozwiążanie to jest proste obliczeniowo, natomiast skuteczność jego działania zależy w dużym stopniu od tego, jak dobrze została zidentyfikowana pozycja wierzchołków. Rysunek 3.6 przedstawia puzzel, w którym rogi A i B zostało niewłaściwie oznaczone, w konsekwencji czego suma odległości dla górnej krawędzi jest dodatnia i została ona niewłaściwie sklasyfikowana jako wypukła.



RYSUNEK 3.6: Element układanki z niepoprawnie oznaczonymi wierzchołkami A i B [opracowanie własne].

Znalezienie wypustki lub dziury

Rozwiążaniem bardziej odpornym na drobne błędy w oznaczaniu wierzchołków jest odziszenie wypustki lub dziury na krawędzi. Początkowy etap działania algorytmu jest taki sam jak w przypadku poprzedniego rozwiązania, natomiast, zamiast obliczać sumę odległości wszystkich punktów, wyszukiwane jest pięć punktów, które tworzą wypustkę lub dziurę w krawędzi. Proces klasyfikacji bazuje na ocenie odległości najwyższego lub najniższego punktu w zależności od tego, czy znaleziono wypustkę, czy dziurę. Rysunek 3.7 przedstawia element z oznaczonymi pięcioma punktami tworzącymi dziurę (A) i wypustkę (B). Punkty najniższy i najwyższy zostały oznaczone odpowiednio kolorami białym i zielonym.

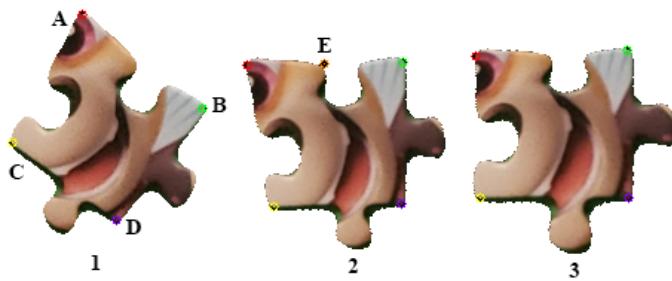


RYSUNEK 3.7: Przykład zastosowania algorytmu wykrywania punktów tworzących wypustki i dziury [opracowanie własne].

3.5 Korekcja perspektywy i rotacji

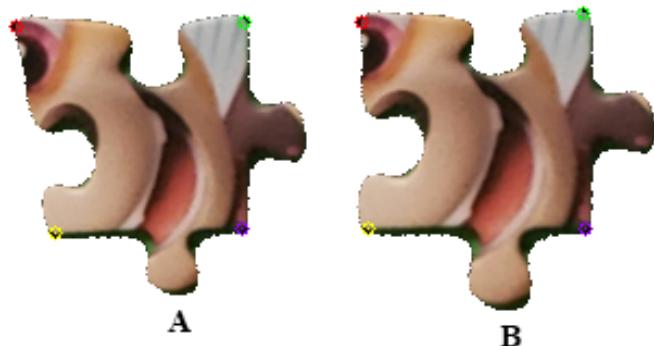
Wydobyte z obrazu puzzle różnią się między sobą rotacją, aby ułatwić analizę kształtu krawędzi oraz umieszczanie elementów układanki w obrazie wynikowym. Zostały one obrócone tak, aby górna krawędź elementu była równoległa do prostej $f(x) = 0$. W tym kroku zostały wybrane dwie metody rotacji i korekcji perspektywy:

- rotacja względem środka górnej krawędzi - prostsza z dwóch zastosowanych metod rotacji elementu, wadą jest brak możliwości korekcji perspektywy. Rysunek 3.8 obrazuje puzzel przed (1) oraz po rotacji (2), punkt E stanowi środek rotacji;
- transformacja czteropunktowa (four point perspective transform) - bardziej złożona metoda transformacji obrazu, rogi puzzla zostają ustawione w taki sposób, aby stworzyć prostokąt o bokach długości równie odległości od rogu A do B oraz od rogu A do C. Rysunek 3.8 przedstawia element układanki przed (1) i po (3) zastosowaniu transformacji.



RYSUNEK 3.8: Przykład zastosowania obu wyżej wymienionych sposobów transformacji puzzla [opracowanie własne].

Z przeprowadzonych badań wynika, że zastosowanie transformacji czteropunktowej nie jest dobrym rozwiązańiem w przypadku analizy puzzli, ponieważ korekcja perspektywy może znacząco zmienić kształt elementów i uniemożliwić ich późniejsze połączenie ze sobą. Jest to problem najbardziej zauważalny w przypadku elementów o kształcie odbiegającym od prostokąta. Rysunek 3.9 przedstawia element układanki, którego kształt został znacząco zmieniony w wyniku przeprowadzenia transformacji. Obraz A to puzzel przed korekcją perspektywy, natomiast B po korekcji.



RYSUNEK 3.9: Przykład puzzla, w którym transformacja czteropunktowa spowodowała znaczące zmiany kształtu [opracowanie własne].

Rozdział 4

Zbiór danych testowych

Algorytmy przetwarzania obrazu różnią się skutecznością w zależności od danych wejściowych. Dlatego do uzyskania miarodajnych wyników ważne jest, aby zbiór danych posiadał możliwie jak najbardziej zróżnicowane obrazy.

Rysunek 4.1 zawiera przykładowe zdjęcia ze zbioru testowego, każdy zestaw puzzli został sfotografowany na kilku tłaach. Dodatkowo zdjęcia zostały wykonane w różnych warunkach oświetleniowych oraz ze zmienią perspektywą.



RYSUNEK 4.1: Przykładowe obrazy wchodzące w skład zbioru testowego [opracowanie własne].

4.1 Kryteria doboru obrazów

Zbiór testowy zawiera zdjęcia siedmiu zestawów puzzli, całkowita liczba obrazów w zbiorze testowym wynosi 108. Każdy zestaw został sfotografowany w zmiennych warunkach, w celu znalezienia czynników, które mają największy wpływ na skuteczność algorytmów wykrywania obiektów.

Do wybranych czynników należą:

- rozdzielcość obrazu — większa rozdzielcość pozwala ukazać w obrazie więcej szczegółów, jednak powoduje również znaczące zwiększenie ilości danych do przetworzenia, przez co wpływa na prędkość działania algorytmów;
- jasność — zdjęcia wykonane przy słabym oświetleniu są mniej wyraźne, wpływa to również na zmniejszenie się kontrastu między sfotografowanymi obiektymi. Jednak zbyt duża ilość światła ma podobny efekt;
- kontrast — poza naświetleniem na kontrast wpływa również różnorodność kolorów ukazanych w obrazie, jeśli puzzle są w kolorze podobnym do tła, na którym się znajdują, trudniej jest określić ich kształt;
- kąt położenia elementów względem obiektywu — perspektywa wpływa na kształt obiektów ukazanych na zdjęciu. Jest to szczególnie zauważalne w przypadku fotografowania płaskich obiektów, takich jak puzzle. Najłatwiejsze do identyfikacji są obiekty, których powierzchnia jest zwrócona prostopadle do obiektywu;
- kształt rozpoznawanych obiektów — obiekty o mniej skomplikowanym kształcie są łatwiejsze do oddzielenia od tła.

Oprócz wyżej wymienionych cech obrazu istnieją również takie, które występują tylko w przypadku puzzli. Elementy mogą być odwrócone awersem w dół, co powoduje brak możliwości ich późniejszego dopasowania do pozostałych elementów układanki. Istnieje również możliwość obrócenia wszystkich elementów rewersem do góry, w takim przypadku wszystkie elementy układanki będą w tym samym kolorze, a ocena dokładności ułożenia będzie możliwa jedynie na podstawie kształtu połączonych ze sobą elementów. W przypadku zestawów puzzli, w których elementy mają do siebie zbliżony kształt, istnieje szansa, że wiele ułożen będzie tak samo poprawnych.

Ponadto puzzle mogą nachodzić na siebie nawzajem, lub być już częściowo ułożone. W przypadku nachodzenia na siebie oryginalny kształt elementów może być niemożliwy do stwierdzenia. Dlatego puzzle takie powinny być odrzucone na etapie oceny kształtu elementów. Istnieje kilka sposobów wykrycia nachodzących na siebie puzzli. Najprostszy z nich to ocena relatywnej wielkości danego elementu w stosunku do pozostałych. Jeśli jest znacznie większy, można z dużym prawdopodobieństwem przyjąć, że jest on niepoprawny.

Dostęp do zbioru testowego możliwy jest na stronie github.com [15].

4.2 Szczegółowa analiza porównawcza wybranych danych wejściowych

Porównywanie ze sobą wyników działania przedstawionego programu może być utrudnione przez fakt, że każdy z etapów działania algorytmu wpływa na etapy następujące po nim. Ze względu na to, poza oceną całościową, przeprowadzono bardziej wnikliwą analizę dla części danych ze zbioru testowego. Kryterium wyboru obrazów było:

- tło — zostały wybrane obrazy wykonane na różnych tłaach;
- elementy układanki — aby ograniczyć liczbę czynników wpływających na efektywność algorytmów, wybrano obrazy zawierające te same zbiory elementów;
- jakość obrazów — wybrane zostały zdjęcia wykonane w wysokiej jakości;
- oświetlenie — ograniczono zbiór do zdjęć wykonanych w dobrych warunkach oświetleniowych.

Ograniczenie wielkości zbioru poddawanego dokładniejszej analizie było spowodowane ilością pracy potrzebnej na przygotowanie danych porównawczych dla każdego obrazu. Poniższa lista zawiera przygotowane dodatkowe dane porównawcze:

- maska oddzielająca elementy od tła — przygotowana maska binarna, która używana jest jako baza do przeprowadzenia analizy działania algorytmów separacji elementów układanki od tła;
- oznaczenie prawidłowych wierzchołków elementów — narożniki każdego puzzla zostały oznacone w celu porównania jakości automatycznego znajdowania rogów;
- tablica zawierająca poprawne ułożenie elementów — dostęp do poprawnego ułożenia pozwala na sprawdzenie jakości całego przebiegu automatycznego układania oraz na zlokalizowanie, jakie puzzle najczęściej są źle ustawiane.

W analizie końcowej nie był brany pod uwagę obraz wynikowy, ponieważ jego analiza byłaby bardziej skomplikowana i wymagałaby stworzenia obrazu zawierającego prawidłowe ułożenie elementów. Dodatkowo nie jest oczywistym, jakie kryterium powinno być uwzględniane podczas tej analizy.

Zbiór obrazów z przygotowanymi danymi do szczegółowej analizy zawiera dwadzieścia obrazów. W jego skład wchodzi pięć zestawów puzzli, dla każdego zestawu przygotowano po cztery zdjęcia na różnorodnych tłaach.

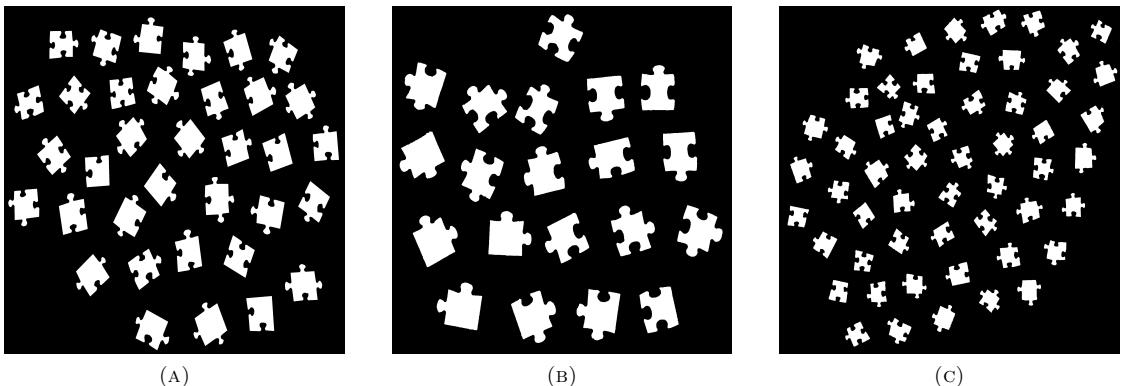
Poniższy rysunek zawiera wycinek zbioru danych wejściowych, dla których stworzone zostały dodatkowe dane porównawcze:



RYSUNEK 4.2: Przykładowe obrazy stanowiące część szczegółowo opisanego zbioru testowego [opracowanie własne].

4.3 Sposób oceny efektywności algorytmów

Aby zbadać wpływ różnych czynników na efektywność procesu układania, wymagane jest stworzenie dla każdego obrazu ze zbioru testowego maski binarnej. Zawiera ona dokładnie odwzorowaną pozycję puzzli w obrazie i pozwala w obiektywny sposób ocenić efektywność algorytmów oddzielania puzzli od tła. Wadą tego rozwiązania jest czasochłonność wynikająca z konieczności ręcznego poprawiania maski. Rysunek 4.3 prezentuje maski binarne stworzone dla wybranych zdjęć ze zbioru testowego.



RYSUNEK 4.3: Maski binarne trzech różnych zdjęć ze zbioru testowego, kolorem białym oznaczono obecne na obrazie puzzle [opracowanie własne].

Ocena skuteczności działania algorytmów wyodrębniających puzzle z obrazu polega na porównaniu maski stworzonej przez algorytm, do maski wzorcowej, stworzonej ręcznie. Obliczenie poprawności wyniku opisuje równanie 4.1:

$$Tm = \frac{P - N}{K} \quad (4.1)$$

gdzie:

- Tm — trafność odwzorowania;
- P — liczba pikseli poprawnie sklasyfikowanych jako puzzle;
- N — liczba niepoprawnie sklasyfikowanych pikseli;
- K — liczba pikseli zawierających puzzle w masce wzorcowej.

Z uwagi na złożoność procesu automatycznego układania puzzli istnieje kilka kluczowych kroków, które cechują się tym, że niepoprawny wynik jednego z nich może całkowicie uniemożliwić, lub znaczco utrudnić kolejne etapy automatycznego układania. Do etapów tych należą:

- oddzielanie puzzli od tła;
- wyznaczenie wierzchołków elementów;
- klasyfikacja krawędzi puzzli.

Są to pierwsze trzy etapy przetwarzania danych wejściowych, dlatego w celu wyodrębnienia wpływu różnych cech obrazu na każdy z nich. Po zakończeniu jednego z tych kroków wyniki jego przetwarzania zostają zapisane, a następnie poddane ocenie podobnie jak w przypadku oddzielania puzzli od tła. Różnicą pomiędzy nimi są równania pozwalające ocenić dokładność działania algorytmów.

Do oceny efektywności wyznaczania wierzchołków pojedynczego elementu użyte zostało równanie 4.2:

$$Tw = 1 - \frac{|AA'| + |BB'| + |CC'| + |DD'|}{L(A, B, C, D)} \quad (4.2)$$

gdzie:

- Tw — trafność odnalezienia wierzchołków;
- A, B, C, D — poprawne położenie wierzchołków;
- A', B', C', D' — testowane położenie wierzchołków;
- L (A, B, C, D) — obwód czworokąta stworzonego przez prawidłowe wierzchołki.

Natomiast poprawność klasyfikacji krawędzi wyliczana jest ze wzoru 4.3:

$$Tk = \frac{Pk}{K} \quad (4.3)$$

gdzie:

- Tk — trafność klasyfikacji krawędzi;
- Pk — poprawnie sklasyfikowane krawędzie;
- K — liczba krawędzi elementu, w przypadku badanych zestawów puzzli zawsze wynosi ona 4.

Rozdział 5

Sposoby klasyfikacji elementów

Elementy układanki można podzielić na cztery rodzaje, w zależności od liczby płaskich krawędzi:

- **narożniki** — elementy posiadające dwie płaskie krawędzie, stanowią rogły układanki;
- **krawędzie** — wraz z narożnikami tworzą obramowanie układanki, zawierają jedną płaską krawędź;
- **wewnętrzne** — nie posiadają płaskich krawędzi, stanowią wypełnienie układanki;
- **błędnie sklasyfikowane** — puzzle posiadające więcej niż dwie płaskie krawędzie.

Elementy należące do czwartej kategorii, tzn. błędnie sklasyfikowane, nie są używane podczas dalszych etapów układania.

Analiza krawędzi

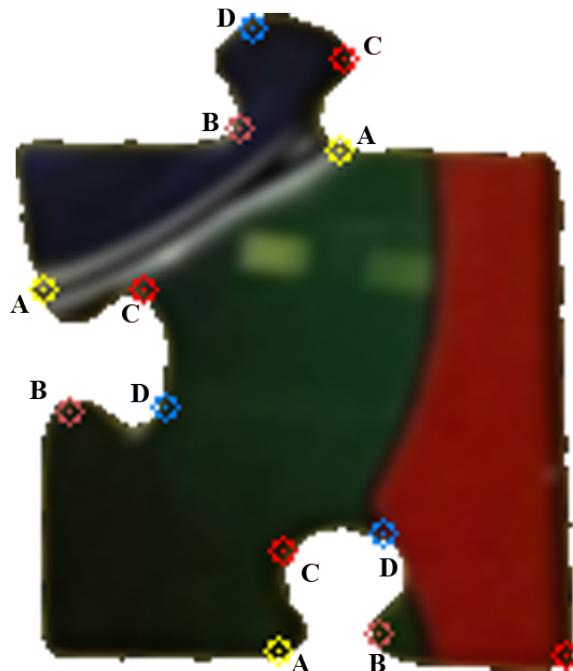
W celu znalezienia najlepszego dopasowania, dla każdej krawędzi puzzla obliczany jest szereg wskaźników, mających ograniczyć liczbę potencjalnych kandydatów. Wszystkie parametry obliczane są po obróceniu krawędzi w taki sposób, aby jej wierzchołki leżały na prostej $f(x) = 0$. Krawędzie na tym etapie analizowane są niezależnie od siebie nawzajem. Zbiór wskaźników można podzielić na te, które bazują na kształcie krawędzi oraz biorące pod uwagę jej kolor.

Do grupy parametrów związanych z kształtem krawędzi należą:

- **średnia wysokość krawędzi** — sposób obliczania opisany został w punkcie 3.4.2, poza rolą pełnonią podczas klasyfikacji krawędzi może służyć również do określenia, które krawędzie pasują do siebie;
- **najwyższy i najniższy punkt** — ze wszystkich punktów wybierane są dwa położone najdalej od prostej $f(x) = 0$ o zwrocie dodatnim, zapisywany jako najwyższy punkt i ujemny oznaczony jako punkt najniższy;
- **odległość pomiędzy wierzchołkami tworzącymi krawędź** — może być obliczana na dwa sposoby: długość krawędzi w osi x lub jako liczba punktów wchodzących w skład krawędzi;
- **położenie punktów tworzących dziurę lub wypustkę** — punkty opisane w rozdziale 3.4.2 mogą również posłużyć do znajdowania podobieństw pomiędzy krawędziami. Poza ich lokalizacją na krawędzi, analizie zostają poddane następujące odległości:
 - pomiędzy punktami u podstawy dziury lub wypustki;

– pomiędzy punktami w najszerzej części dziury lub wypustki.

Rysunek 5.1 obrazuje, które pary punktów zostały poddane analizie. A i B to punkty u podstawy dziury i wypustki, natomiast punkty C i D znajdują się w jej najszerzej części.



RYSUNEK 5.1: Puzzel z wyszczególnionymi punktami tworzącymi dziury i wypustki [opracowanie własne].

Kolor krawędzi

Kolor wzdłuż krawędzi pozwala na ocenę dopasowania krawędzi do siebie. Warto zaznaczenia jest, że proces oddzielania elementów od tła, przeprowadzony wcześniej, może doprowadzić do sytuacji, w której krawędzie elementu nie zostaną idealnie wyznaczone. W wyniku czego, podczas analizy kolorystycznej krawędzi istnieje ryzyko, otrzymania niepoprawnych danych wejściowych. Ponadto nawet w sytuacji bardzo dobrego oddzielenia puzzli od tła, w zależności od jakości zdjęcia część pikseli puzzla będzie zawierała kolor tła. Rysunek 5.2 obrazuje oba przypadki wyżej wymienionych problemów, krawędź B została niepoprawnie oddzielona od tła, natomiast kolor krawędzi A został zmieniony pomimo poprawnego odseparowania.



RYSUNEK 5.2: Dwa problemy pojawiające się podczas analizy kolorów krawędzi elementów [opracowanie własne].

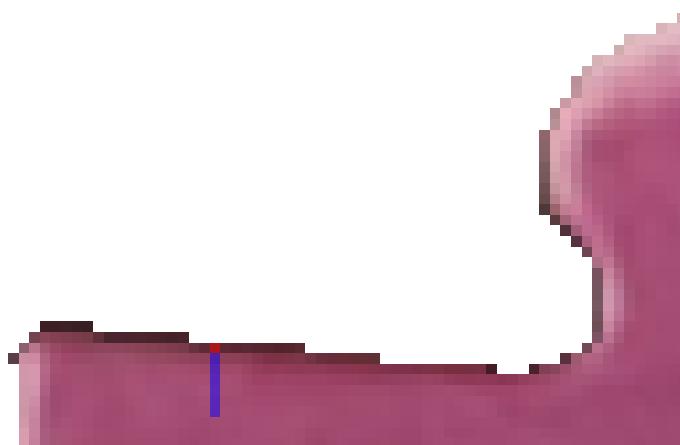
W celu zwiększenia skuteczności analizy krawędzi oraz, aby uodpornić proces ten na wyżej wymienione problemy, analizie poddawane nie są jedynie piksele wchodzące w skład krawędzi, ale również te znajdujące się w głębi elementu. Rysunek 5.3 przedstawia obszar puzzla poddawany analizie, został on oznaczony kolorem niebieskim.



RYSUNEK 5.3: Element układanki z oznaczonym na niebiesko obszarem krawędzi poddawanym analizie kolorów [opracowanie własne].

Do cech kolorystycznych wykorzystywanych w procesie analizy krawędzi należą:

- **dominujący kolor** — kolor, który występuje najczęściej wzdłuż danej krawędzi;
- **średnia kolorów** — średnia obliczana z kolorów wszystkich analizowanych pikseli;
- **cząstkowe dominujące kolory** — najczęściej występujące kolory, ale obliczane dla każdego punktu na krawędzi. Zbiór, z którego obliczany jest dominujący kolor, to punkty znajdujące się poniżej aktualnie przetwarzanego piksela krawędzi. Rysunek 5.4 przedstawia, które punkty, oznaczone kolorem niebieskim, zostały uwzględnione podczas analizy jednego z punktów krawędzi, oznaczonego kolorem czerwonym.



RYSUNEK 5.4: Fragment puzzla obrazujący sposób obliczania cząstkowych kolorów dominujących [opracowanie własne].

Cząstkowe dominujące kolory obliczane są, aby zmniejszyć ilość przetwarzania w późniejszym etapie automatycznego układania, którym jest porównywanie krawędzi ze sobą.

Rozdział 6

Metody oceny dopasowania elementów

Przed przystąpieniem do umieszczania puzzli w obrazie wynikowym, należy porównać elementy ze sobą, w celu wybrania najlepszego ich dopasowania. Proces ten podzielony jest na dwa etapy. Po dokonaniu analizy krawędzi każdego z elementów osobno zgromadzone w ten sposób dane należy porównać. Aby przyspieszyć ten proces, porównywane są między sobą jedynie elementy o odpowiedniej orientacji i typie.

- narożniki — porównywane są jedynie z krawędziami;
- elementy krawędziowe — mogą łączyć się zarówno z narożnikami, elementami krawędziowymi, jak i wewnętrznymi, jednak nie każda krawędź puzzla pasuje do każdego z wymienionych rodzajów elementów. Krawędź przeciwległa do płaskiej krawędzi łączy się z elementami wewnętrznymi, natomiast dwie pozostałe łączą się z narożnikami lub krawędziami. Rysunek 6.1 obrazuje, które krawędzie mogą zostać połączone z narożnikami lub elementami krawędziowymi, oznaczone na czerwono. Krawędź oznaczona na niebiesko łączy się z elementem wewnętrznym;
- elementy wewnętrzne — łączą się z innymi elementami wewnętrznymi oraz z krawędziowymi.



RYSUNEK 6.1: Element układanki z wyszczególnionymi krawędziami pasującymi do elementów krawędziowych, oznaczone na czerwono oraz elementów wewnętrznych, oznaczone na niebiesko [opracowanie własne].

Poza rodzajem puzzla podczas porównawczej dwóch krawędzi znaczenie ma również ich rodzaj. Krawędzie wypukłe porównywane są jedynie z wklęsłymi i odwrotnie. Dodatkowo część połączeń pomiędzy elementami krawędziowymi lub narożnikami można odrzucić, mając na uwadze położenie płaskiej krawędzi. Aby elementy można było do siebie dopasować, płaska krawędź po

połączeniu ich ze sobą musi się łączyć. Rysunek 6.2 przedstawia dwa elementy, które nie pasują do siebie ze względu na położenie płaskiej krawędzi.



RYSUNEK 6.2: Schemat blokowy opisujący działanie algorytmu tworzenia obramowania układanki [opracowanie własne].

6.1 Porównanie elementów między sobą

Po dokonaniu wstępniego odrzucenia niepasujących elementów, każda z krawędzi puzzla poddawana jest ocenie dopasowania do innych pasujących krawędzi. Proces ten polega na porównaniu cech krawędzi obliczonych w punkcie 5. Z przeprowadzonych badań wynika, że cechą, pozwalającą szybko odrzucić dużą liczbę potencjalnych dopasowań jest długość krawędzi. W przypadku dużej różnicy długości pomiędzy krawędziami, połączenie takie nie jest dalej analizowane. Poza porównywaniem obliczonych wcześniej współczynników, dla każdego połączenia krawędzi ze sobą, zostaje wyliczona różnica wysokości poszczególnych punktów oraz podobieństwo kolorów kolejnych punktów krawędzi. Wartości współczynników są następnie sumowane, a otrzymany wynik jest uznawany za trafność dopasowania krawędzi.

Efektem działania tego etapu jest wygenerowanie dla każdej krawędzi listy potencjalnie pasujących krawędzi, która używana jest w późniejszych etapach układania, do oceny trafności ułożenia.

6.2 Wstępne układanie elementów

Po dokonaniu analizy porównawczej krawędzi możliwe jest przystąpienie do procedury dopasowania elementów do siebie. Możliwe jest zastosowanie kilku różnych sposobów łączenia elementów.

Łączenie najlepiej pasujących elementów

Proces ten polega na znalezieniu dwóch elementów na liście, które najlepiej do siebie pasują i połączenie ich ze sobą, następnie czynność jest powtarzana. Po połączeniu wszystkich pojedynczych elementów efektem działania będzie lista zawierająca większe elementy składające się z kilku puzzli. Następnym krokiem będzie znalezienie najlepszego dopasowania na liście nowopowstałych elementów. Proces ten jest powtarzany do momentu, aż dalsze łączenie elementów nie będzie możliwe. Nie został on jednak zaimplementowany ze względu na kilka problemów:

- złożoność algorytmów potrzebnych do analizy coraz bardziej skomplikowanych kształtów, które powstają podczas łączenia ze sobą mniejszych elementów;
- trudność zapisu i przetwarzania już przeanalizowanych połączeń elementów — problem ten wynika z niedoskonałości działania algorytmu w poprzednim punkcie tzn. otrzymanej listy pasujących krawędzi. Istnieje szansa, że najlepiej pasująca krawędź nie jest tą, która powinna zostać połączona w celu uzyskania poprawnego rozwiązania układanki. W takim przypadku może dojść do sytuacji, gdzie otrzymane elementy nie będą do siebie pasować.

Możliwym rozwiązaniem jest cofnięcie ostatniego połączenia i próba wybrania innego. Jednak nie istnieje sposób na stwierdzenie, w którym dokładnie momencie doszło do niepoprawnego połączenia. W najgorszym wypadku, kiedy do niepoprawnego połączenia doszło na samym początku, algorytm musiałby przeszukać wszystkie możliwe kombinacje. Jednocześnie to właśnie na początku dopasowania istnieje największa szansa na niepoprawne dopasowanie elementów.

Łączenie sąsiednich elementów

W pierwszym kroku zostaje wybrany jeden z wierzchołkowych elementów układanki, do którego w następnym etapie zostają dodane dwa najlepiej pasujące elementy. Następnie zostają dodane elementy do już dodanych. Proces kończy się w sytuacji, kiedy wszystkie elementy zostaną połączone, lub kiedy pozostałe z nich nie łączą się ze sobą. Rozwiążanie to jest prostsze w implementacji, niż poprzednio wymienione, natomiast równie trudno jest znaleźć niepoprawnie umieszczony element, bez wyczerpującego łączenia elementów ze sobą.

Układanie obramowania, następnie wypełnianie

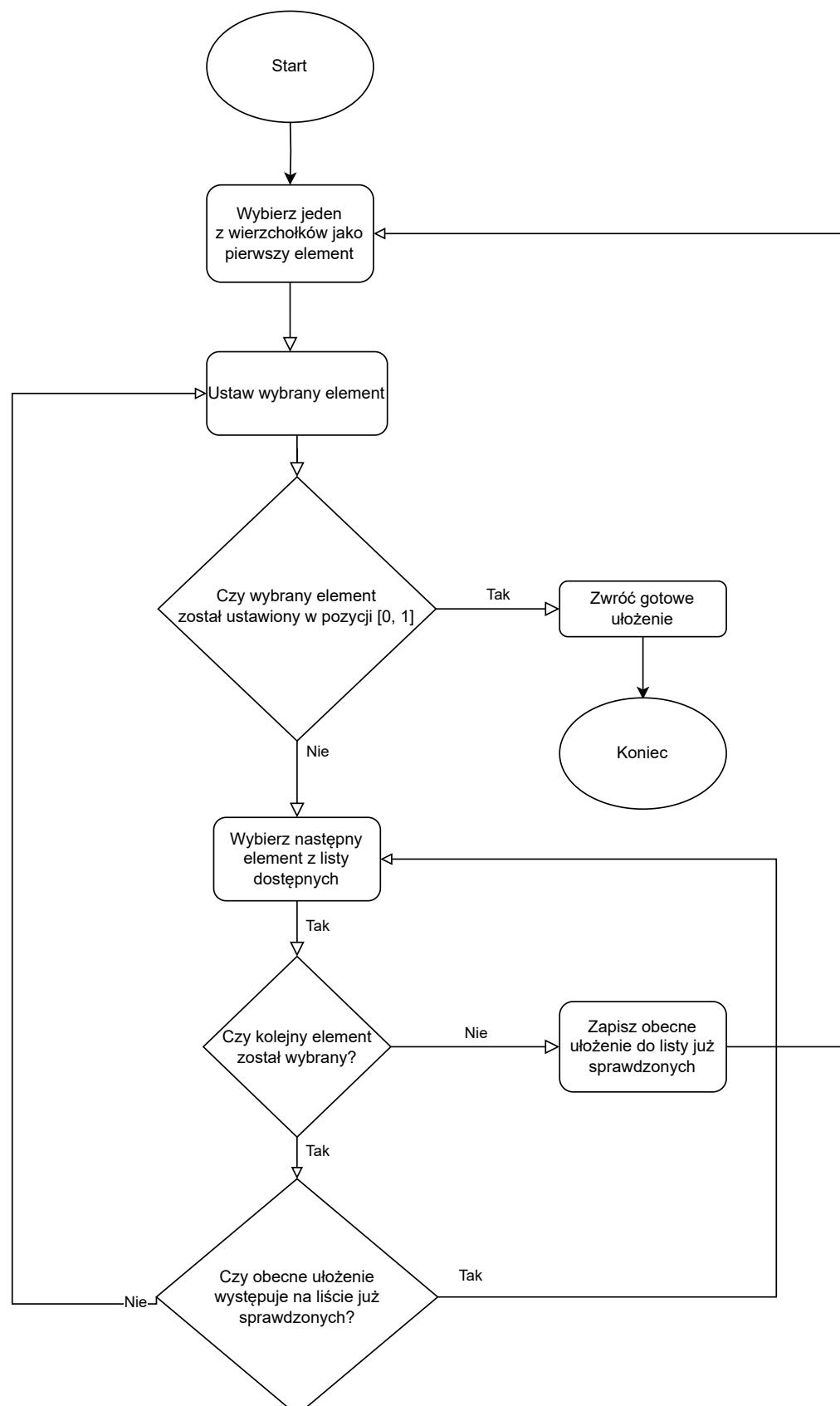
Proces najbardziej przypomina sposób, w jaki ludzie układają puzzle. Jak w poprzednim algorytmie na początku wybierany jest jeden z wierzchołków jako element startowy, następnie dopasowywany jest do niego tylko jeden z najlepiej pasujących elementów.

W celu bardziej czytelnego zapisu położenia elementów są one umieszczane w dwu wymiarowej tablicy, która odzwierciedla wzajemną lokalizację puzzli oraz połączenia między nimi. Pierwszy element zostaje umieszczony w komórce $(0, 0)$, pierwsza liczba oznacza położenie w osi X , druga natomiast w osi Y .

W kolejnym kroku do ostatnio dopasowanego elementu, dołączany jest następny element krawędziowy lub wierzchołkowy, w zależności od tego, który lepiej pasuje. Krok ten jest powtarzany do momentu, aż zabraknie pasujących elementów lub stworzone zostanie całe obramowanie, co można rozpoznać po tym, że ostatni elementłączony jest nie tylko z poprzednio położonym, ale również z elementem startowym. Zostanie on umieszczony w komórce $(0, 1)$ tablicy.

Jeśli stworzenie obramowania się nie uda, lista dopasowanych elementów zostaje zapisana, a w kolejnej iteracji algorytm wybierze inne dopasowanie.

Rysunek 6.3 przedstawia opisany algorytm w postaci schematu blokowego.



RYSUNEK 6.3: Schemat blokowy opisujący działanie algorytmu tworzenia obramowania układanki [opracowanie własne].

Dodatkowo możliwa jest optymalizacja etapu tworzenia obramowania, gdy znana jest liczba puzzli krawędziowych oraz wewnętrznych, istnieje możliwość określenia kształtu układanki. Układ równań 6.1 pozwala na obliczenie dwóch możliwych kształtów obrazu wynikowego.

$$\begin{cases} 2A + 2B = L \\ A \times B = P \end{cases} \quad (6.1)$$

gdzie:

- A, B — wymiary układanki;
- L — liczba krawędzi i wierzchołków;
- P — całkowita liczba elementów układanki.

Znajomość wymiarów układanki na etapie dopasowywania do siebie elementów pozwala wybrać, czy następny dopasowany element powinien być krawędzią lub wierzchołkiem, co umożliwia zmniejszenie liczby nieudanych dopasowań i przyspieszenie obliczeń.

Po stworzeniu obramowania następuje proces wypełniania, można w tym celu zastosować rozwiązanie z punktu 6.2, jednak istnieje ryzyko pojawienia się problemów, które ta metoda powoduje. Dlatego proces wypełniania przeprowadzony został podobnie do procesu tworzenia obramowania. Na początku zostaje wybrany element najlepiej pasujący do elementów w lewym górnym rogu obramowania, następnie zostaje do niego dołączony element z prawej strony. Proces ten jest powtarzany do momentu ułożenia pełnej warstwy. Aktualnie dołączony element zostaje połączony z elementem startowym. Etap jest powtarzany do momentu połączenia wszystkich puzzli lub gdy dołączenie następnego nie jest możliwe. Po ułożeniu warstwy rozpoczyna się układanie następnej, aż wszystkie puzzle zostaną wykorzystane. Podczas układania wewnętrznych warstw ocena jakości dopasowania nowego elementu jest bardziej skomplikowana, gdyż należy uwzględnić wartości dopasowania krawędzi do każdego z sąsiadujących elementów.

Zastosowanie tego rozwiązania pozwala w łatwiejszy sposób kontrolować to, kiedy nastąpi połączenie niewłaściwego elementu, ponieważ cały proces układania podzielony jest na mniejsze etapy, których wynik można w łatwy sposób ocenić. A wynikiem działania jest lista kolejno ustawionych elementów, co ułatwia proces zapisywania i sprawdzania już sprawdzonych ułożeń.

Wadą takiego sposobu zapisywania ułożeń jest możliwość pominięcia niektórych konfiguracji, kiedy kilka krawędzi wybranego elementu mogłyby zostać połączone z tą samą krawędzią poprzedniego puzzla, z uwagi na zapisywanie połączeń między elementami, a nie krawędziami. Pozostałe z możliwych krawędzi zostaną pominięte podczas następnej iteracji.

6.3 Ocena całkowitego dopasowania

Po zakończeniu procesu dopasowywania elementów następuje obliczenie całkowitej wartości dopasowania. Jest to suma wartości dopasowania wszystkich połączonych elementów. W przypadku, kiedy nie udało się dopasować wszystkich elementów, zostaje naliczona kara za każdy pomity element. Po czym następuje kolejna iteracja algorytmu z punktu 6.2. W ten sposób możliwe jest wybranie najlepszego rozwiązania. Liczba iteracji jest stała i wybierana na początku działania programu.

Taka ocena dopasowania nie gwarantuje otrzymywania poprawnego dopasowania, gdyż ocenianym kryterium nie jest poprawność ułożenia, a ocena tego jak dobrze połączone krawędzie pasują do siebie.

W celu sprawdzenia jakości otrzymanego rozwiązania wybrane najlepsze dopasowanie jest porównywane z poprawnym. Proces ten polega na sprawdzeniu, czy wszystkie elementy znajdują się w odpowiednich komórkach tabeli wynikowej oraz, czy zostały właściwie obrócone. Ze względu na to, że stworzenie poprawnych rozwiązań możliwe jest jedynie ręcznie i fakt, że będą one inne dla każdego obrazu wejściowego. Takie rozwiązania zostały stworzone tylko dla wybranej grupy danych ze zbioru testowego.

Rozdział 7

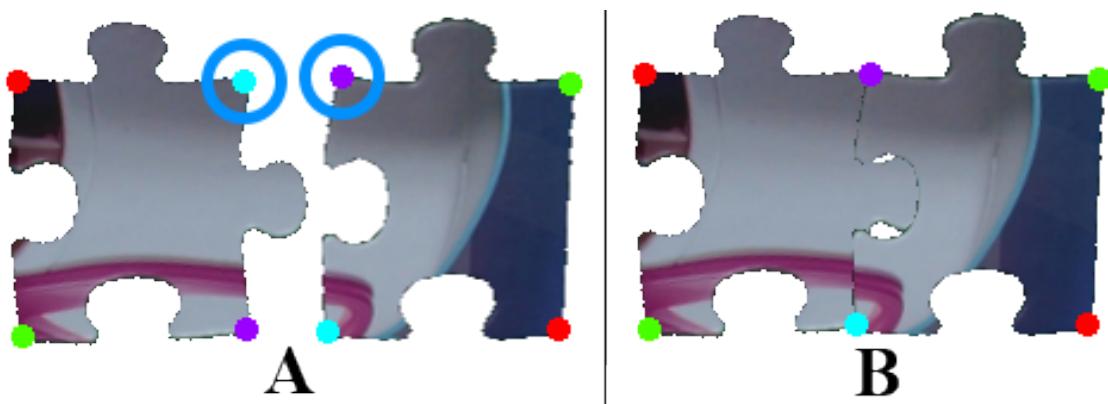
Proces umieszczania puzzli na obrazie wynikowym

Ostatnim etapem w procesie automatycznego układania jest prezentacja otrzymanego ułożenia. Danymi wejściowymi tego procesu jest otrzymana w poprzednim rozdziale dwuwymiarowa tablica elementów.

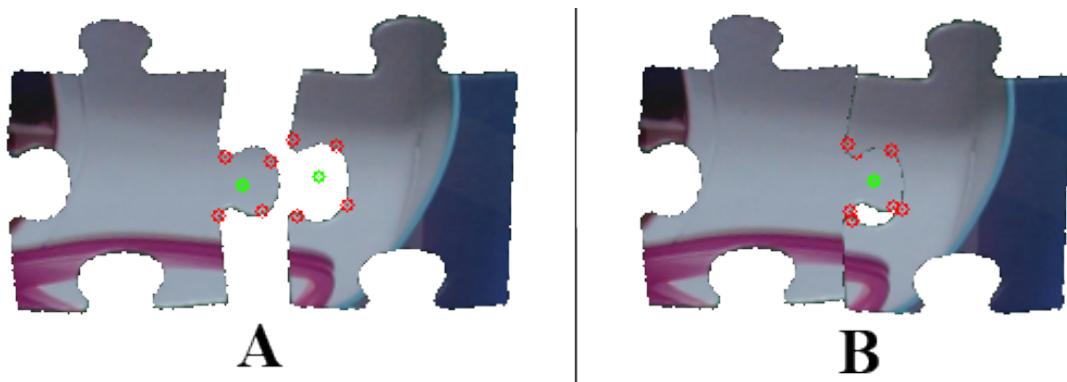
7.1 Punkty łączenia ze sobą elementów

Elementy można łączyć ze sobą na kilka sposobów:

- łączenie wierzchołkowe — nowy element zostaje umieszczony tak, aby lewy górny wierzchołek krawędzi łączonej, znajdował się w tym samym miejscu, co prawy górny wierzchołek krawędzi, do której jest on dołączany. Na rysunku 7.1 w części A zostały zaznaczone na niebiesko wierzchołki wybrane w procesie łączenia. Efekt połączenia elementów ukazano w części B;
- łączenie dziury i wypustki — punktami łączącymi obie krawędzie są geometryczne środki, znalezionych wcześniej dziur lub wypustek. Na rysunku 7.2 w części A oznaczono kolorem czerwonym punkty stanowiące zidentyfikowaną dziurę oraz wypustkę. Kolorem zielonym oznaczono punkt stanowiący geometryczny środek, będący miejscem łączenia dla innego elementu. Część B zawiera wizualizację połączenia ze sobą elementów.



RYSUNEK 7.1: Przykład zastosowania wierzchołków elementów jako punktów łączenia ich ze sobą [opracowanie własne].

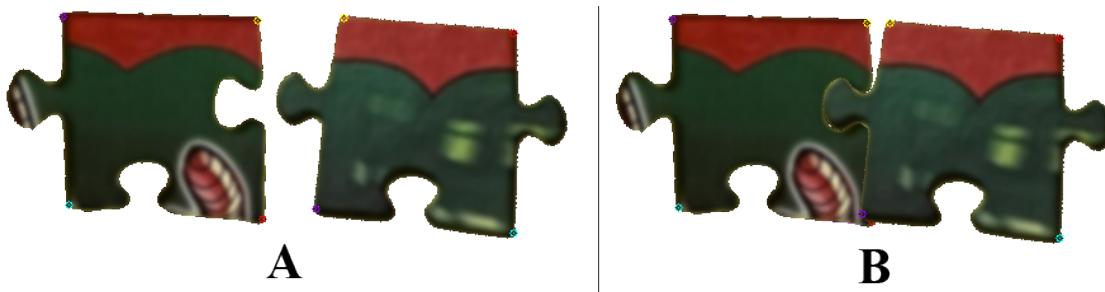


RYSUNEK 7.2: Zastosowanie geometrycznego środka dziury oraz wypustki jako punktów połączenia ze sobą elementów [opracowanie własne].

Każde z wymienionych rozwiązań jest czułe na niedoskonałość dostarczonych danych. W przypadku łączenia wierzchołkowego, jeśli narożniki elementów zostaną nieprawidłowo oznaczone, będzie on umieszczony w niewłaściwym miejscu. Z uwagi na to, że kolejne elementy umieszczane są tak, aby dopasować je do już dodanych, drobne błędy powstające podczas dopasowywania kumulują się i w niektórych przypadkach mogą znaczaco wpływać na obraz wynikowy. Łączenie dziur i wypustek wrażliwe jest na błędne określenie położenia dziury lub wypustki. Problem kumulowania się drobnych błędów dotyczy również tego algorytmu.

Możliwym sposobem łączenia, którego problem dodawania się do siebie błędów dotyczy w mniejszym stopniu, jest układanie bazujące na umieszczaniu elementów w osobnych komórkach o stałym rozmiarze. Pomiędzy elementami będą w takim przypadku znaczące przerwy, jednak niepoprawne ustawnienie jednego elementu nie wpłynie na pozostałe. Rozwiązanie to nie zostało zastosowane, ze względu na to, że obrazy wynikowe nie są poddawane ocenie.

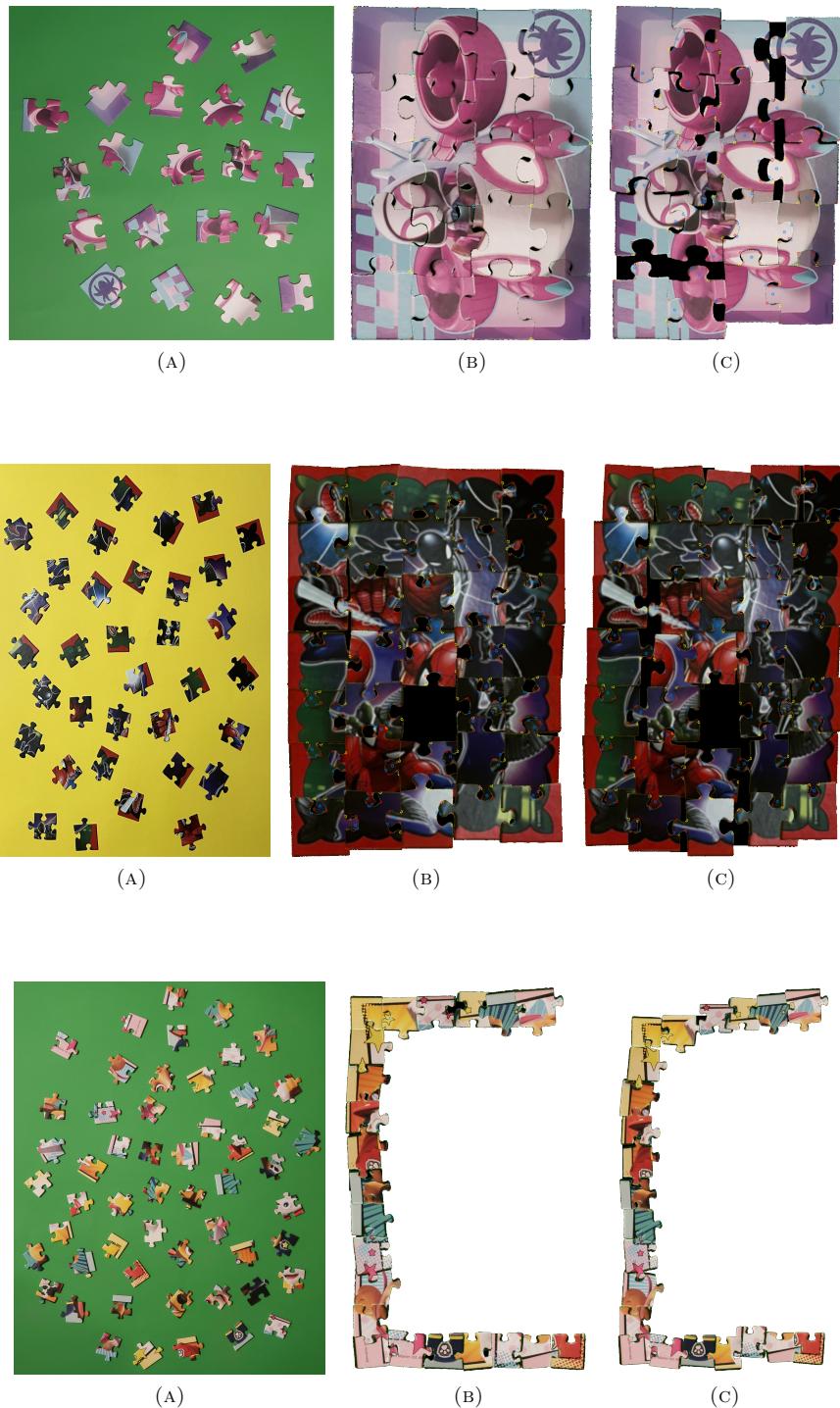
Inny problem, który może pojawić się podczas łączenia elementów, związany jest z ich wzajemną rotacją. W zastosowanym algorytmie rotacja elementu odbywa się na etapie korekcji perspektywy i rotacji. A wszystkie późniejsze obroty wykonywane są o wielokrotność kąta prostego względem środka elementu. Rozwiązanie to pozwala na łatwiejsze obliczanie pozycji istotnych punktów, takich jak narożniki, czy położenie dziur i wypustek. Natomiast jego wadą jest brak możliwości idealnego dopasowania elementów, których kształt wymaga łączenia pod innym kątem. Rysunek 7.3 przedstawia elementy, które do dokładnego dopasowania wymagają rotacji jednego z elementów. Fragment A przedstawia elementy przed połączeniem, natomiast fragment B po połączeniu ze sobą metodą łączenia dziury i wypustki.



RYSUNEK 7.3: Łączenie elementów układanki, w którym do dokładnego dopasowania jeden z elementów wymaga dodatkowej rotacji [opracowanie własne].

7.2 Przykładowe rozwiązania

Celem działania algorytmu automatycznego układania puzzli jest stworzenie obrazu zawierającego rozwiązanie układanki. Na rysunku 7.4 przedstawione zostały wybrane rozwiązania z zastosowaniem obu wyżej wymienionych sposobów łączenia elementów. Obraz A to dane wejściowe, obraz B rozwiązańe z zastosowaniem łączenia wierzchołkowego, w obrazie C zastosowano łączenie dziury i wypustki.



RYSUNEK 7.4: Przykładowe ułożenie puzzli o różnej ilości elementów [opracowanie własne].

Z przedstawionych na rysunku 7.4 wyników działania programu można zauważyć, że algorytm

dostarcza lepszych rozwiązań dla układanek o mniejszej liczbie elementów. Jest to spowodowane tym, że wraz ze zwiększeniem się liczby elementów wzrasta możliwość pomyłki podczas analizy któregoś z nich. A jeden błędnie oznaczony element może spowodować brak możliwości ułożenia całości układanki. Dodatkowo wpływ na pogorszenie się wyników ma wielkość poszczególnych elementów, w przypadku, gdy elementów jest więcej, a zdjęcia wykonywane są przy pomocy tego samego urządzenia. Każdemu elementowi zostanie przypisana mniejsza część obrazu, co powoduje zmniejszenie ilości detali oraz utrudnienie w odseparowaniu ich od tła.

Lepszy efekt uzyskano z zastosowaniem algorytmu łączącego do siebie puzzle przy użyciu narożników. Spowodowane jest to tym, że poprawne odnalezienie wierzchołków elementu może w niektórych przypadkach być łatwiejsze niż zlokalizowanie centralnego punktu dziury lub wypustki. Dodatkowo błąd w lokalizacji wierzchołków jest zwykle mniejszy niż w przypadku dziury, czy wypustki.

Rozdział 8

Podsumowanie i analiza wyników

8.1 Zestawienie algorytmów zastosowanych na różnych etapach przetwarzania

Złożoność problemu automatycznego układania puzzli powoduje, że proces ten został podzielony na kilka pod problemów. Wiąże się to z możliwością przetestowania różnych algorytmów dedykowanych do poszczególnych etapów. Poniższa lista zawiera informacje, jakie algorytmy zostały wybrane:

1. Oddzielanie elementów od tła:
 - a) maska kolorów z wyborem koloru dominującego — zakres kolorów oznaczanych jako tło, jest dobierany, bazując na najczęściej występującym kolorze w obrazie wejściowym;
 - b) segmentacja semantyczna uczona rozpoznawać pojedyncze elementy — sieć neuronowa, której jako dane uczące podano zbiór pojedynczych elementów układanki na różnych tłaach;
 - c) segmentacja semantyczna uczona rozpoznawać grupy elementów — jako dane uczące podano zbiór obrazów zawierających zbiory puzzli na różnych tłaach.

Sieć neuronowa wykorzystywana w algorytmie segmentacji semantycznej to DeepLabv3 [12].

2. Lokalizacja narożników:
 - a) Harris corner detection — zastosowanie algorytmu do wyznaczenia zbioru potencjalnych wierzchołków, które następnie poddawane są selekcji w celu wybrania najlepszego czworokąta;
 - b) analiza kształtu konturu — zasada działania algorytmu została opisana w rozdziale 3.4.
3. Dopasowanie krawędzi — znalezienie najbardziej pasujących do siebie krawędzi, można przeprowadzić, bazując na:
 - a) geometrii krawędzi — porównanie krawędzi między sobą uwzględniając jedynie ich kształt;
 - b) kolor krawędzi — podczas analizy brane pod uwagę są jedynie kolory krawędzi;
 - c) kolor i geometria — dopasowanie bazujące na kolorze oraz geometrii krawędzi.
4. Łączenie ze sobą elementów — znalezienie najlepszego rozwiązania wykorzystuje tylko jeden algorytm, opisany w rozdziale 6.2.

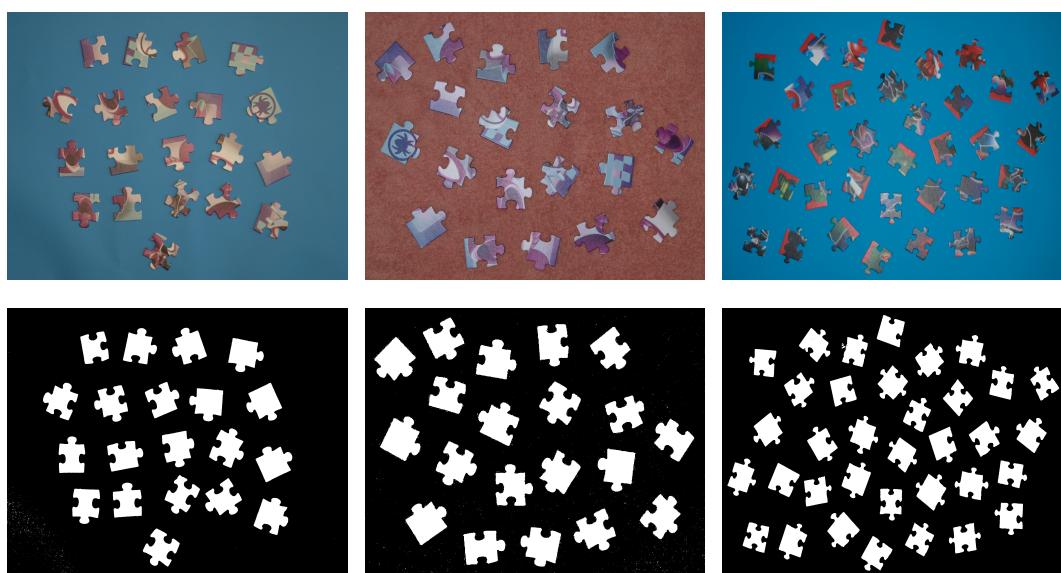
5. Umieszczanie elementów w obrazie wynikowym — etap ten nie jest oceniany, z uwagi na trudność w automatycznej ocenie jakości obrazu wynikowego. Wykorzystywany jest tylko algorytm łączenia wierzchołkowego przedstawiony w rozdziale 7.1.

8.2 Prezentacja wyników działania algorytmów

8.2.1 Oddzielanie elementów od tła

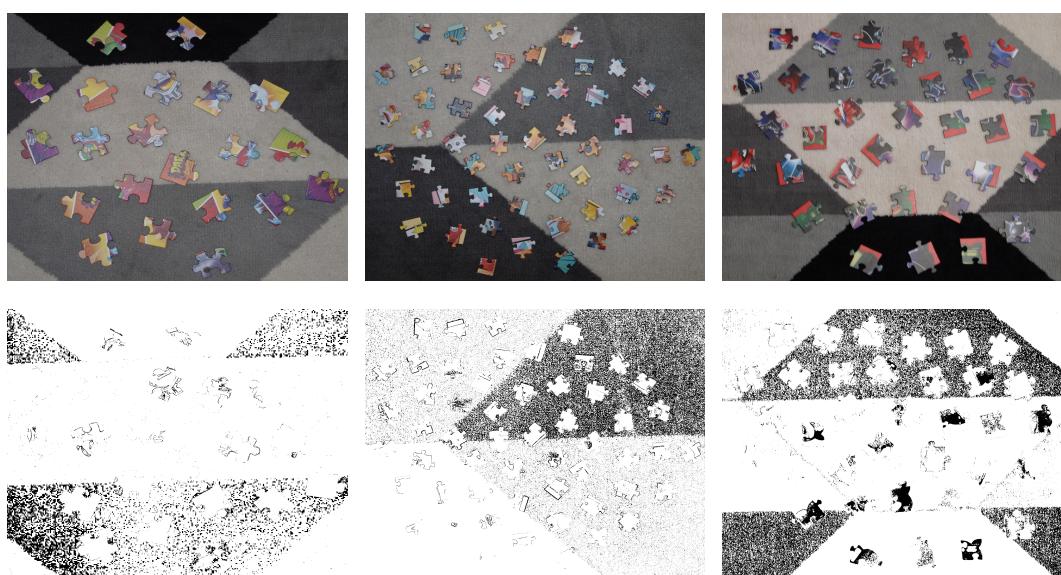
Maska kolorów

Rysunek 8.1 zawiera obrazy, dla których maska kolorów osiągnęła najlepsze efekty.



RYSUNEK 8.1: Najlepsze efekty działania maski kolorów [opracowanie własne].

Rysunek 8.2 przedstawia dane wejściowe, dla których algorytm osiągnął najgorsze rezultaty.



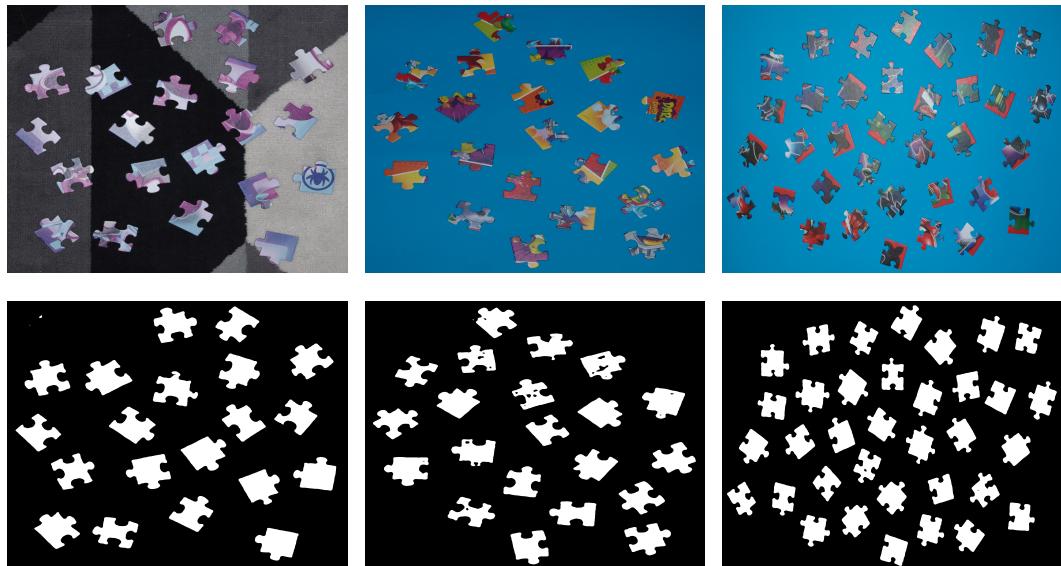
RYSUNEK 8.2: Najgorsze efekty działania maski kolorów [opracowanie własne].

Z uwagi na ograniczenie się do maskowania kolorów z danego zakresu. Zdjęcia, w których tło

zawiera niejednorodne kolory, były dla tego podejścia znacznie trudniejsze, podobnie jak sytuacje, w których elementy mają w sobie kolor podobny do tła.

Segmentacja semantyczna uczona rozpoznawać pojedyncze elementy

Rysunek 8.3 zawiera obrazy, dla których algorytm osiągnął najlepsze efekty.



RYSUNEK 8.3: Najlepsze efekty działania algorytmu segmentacji semantycznej uczona rozpoznawać pojedyncze elementy [opracowanie własne].

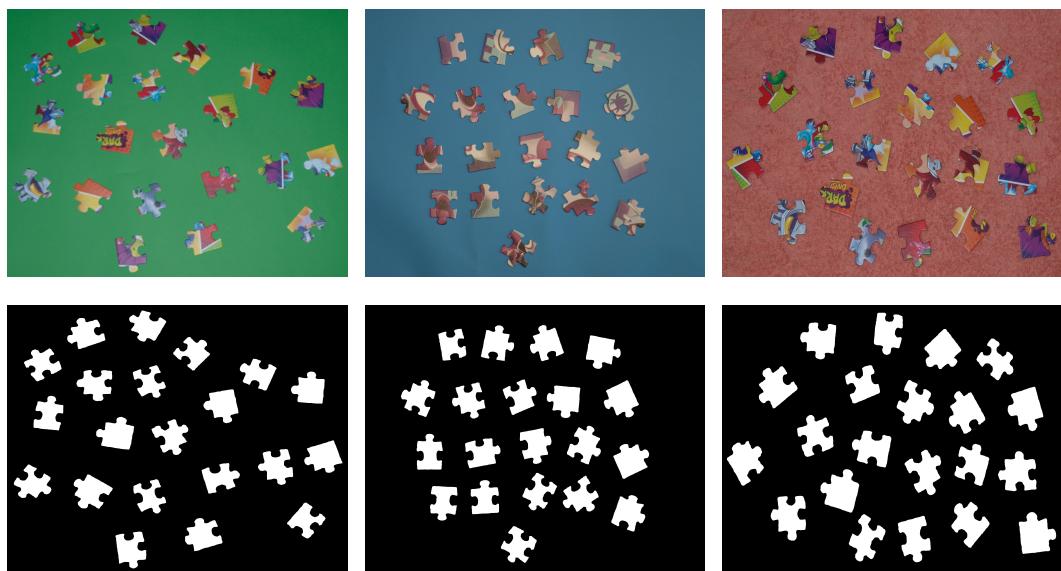
Rysunek 8.4 przedstawia dane wejściowe, dla których algorytm osiągnął najgorsze rezultaty.



RYSUNEK 8.4: Najgorsze efekty działania algorytmu segmentacji semantycznej uczona rozpoznawać pojedyncze elementy [opracowanie własne].

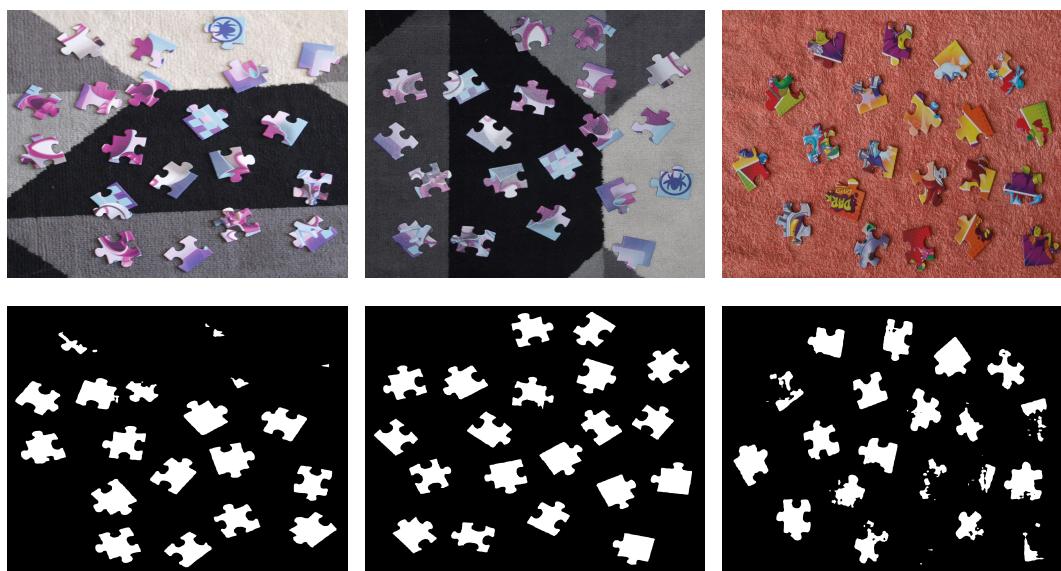
Segmentacja semantyczna uczona rozpoznawać grupy elementów

Rysunek 8.5 zawiera obrazy, dla których algorytm osiągnął najlepsze efekty.



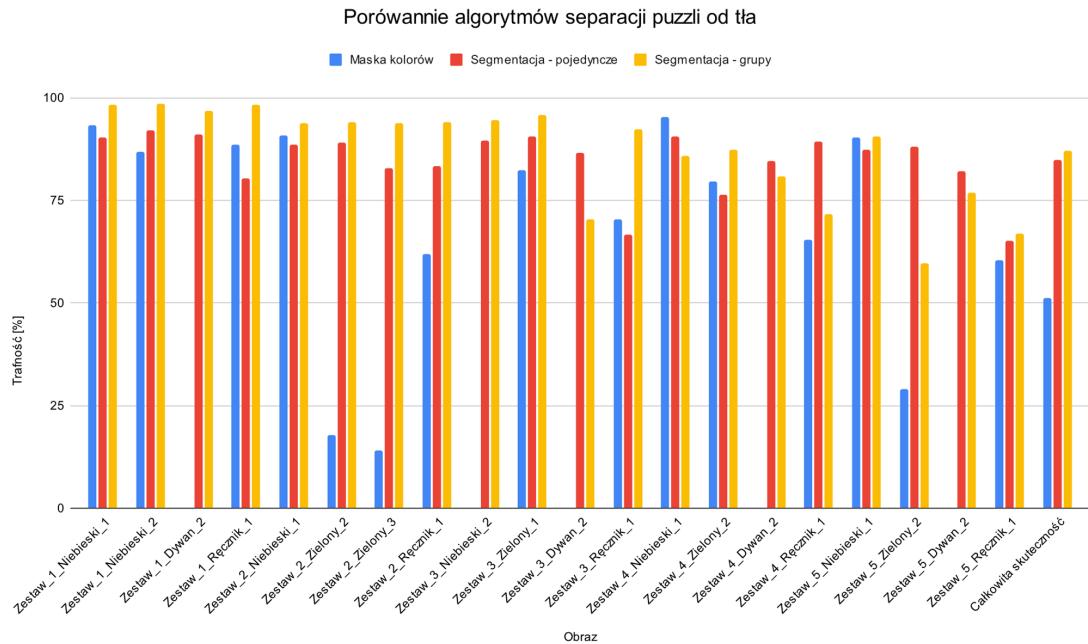
RYSUNEK 8.5: Najlepsze efekty działania algorytmu segmentacji semantycznej uczonej rozpoznawać grupy elementów [opracowanie własne].

Rysunek 8.6 przedstawia dane wejściowe, dla których algorytm osiągnął najgorsze rezultaty.



RYSUNEK 8.6: Najgorsze efekty działania algorytmu segmentacji semantycznej uczonej rozpoznawać grupy elementów [opracowanie własne].

Rysunek 8.7 przedstawia trafność powyższych algorytmów oddzielania puzzli od tła, zestawy 1, 2 i 3 to zestawy puzzli 20 elementowych, zestaw 4 zawiera 35 elementów, natomiast zestaw 5 - 54 elementy.



RYSUNEK 8.7: Porównanie efektów działania algorytmów wykrywania puzzli na różnych tłaach [opracowanie własne].

Sieci neuronowe uczone do przeprowadzania segmentacji semantycznej, szkolone były na części obrazów ze zbioru testowego, co może wpływać na osiągane przez nie rezultaty. Do danych uczących zostały dodane również obrazy zawierające tła bez puzzli.

Z przeprowadzonych badań wynika, że średni najlepszy efekt uzyskała segmentacja semantyczna wykorzystująca jako dane uczące zestawy puzzli. Może być to spowodowane tym, że dane testowe, to również zdjęcia zestawów puzzli. Jednak przygotowanie zbioru uczącego dla tej metody, jest bardziej czasochłonne, gdyż wymaga dużej liczby zdjęć z manualnie przygotowaną poprawną separacją elementów. Natomiast segmentacja wykorzystująca pojedyncze elementy wymaga mniej pracy podczas przygotowania zbioru uczącego. Na potrzeby przeprowadzonych eksperymentów zbiór uczący zawierał 57 obrazów w tym:

- 13 zdjęć tła bez elementów;
- 10 zdjęć elementów na niebieskim tle;
- 10 zdjęć elementów na zielonym tle;
- 10 zdjęć elementów na tle ręcznika;
- 10 zdjęć elementów na tle dywanu.

Zbiór danych uczących dla segmentacji z użyciem grup elementów zawierał 23 obrazy, w których skład wchodzi:

- 13 zdjęć tła bez elementów;
- 5 obrazów na tle niebieskim;
- 2 obrazy na tle zielonym;
- 2 obrazy na tle ręcznika;
- 1 obraz na tle dywanu.

Pomimo mniejszej liczby obrazów każdy z nich zawierał 20 oznaczonych elementów, w sumie 200 elementów.

Maska kolorów pomimo prostoty implementacji wygenerowała najlepsze rezultaty dla kilku obrazów, głównie na niebieskim tle, jednak nie nadaje się ona w sytuacji, kiedy elementy umieszczone są na różnobarwnych tłaach.

8.2.2 Lokalizacja narożników

Lokalizacja narożników metodą wykorzystującą algorytm Harris corner detection uzyskała następującą trafność:

- puzzle 20 elementowe — 60.2%;
- puzzle 35 elementowe — 60.4%;
- puzzle 54 elementowe — 64.0%.

Wykorzystanie do detekcji rogów algorytmu bazujące na analizie kształtu konturu elementu uzyskało trafności przedstawione poniżej:

- puzzle 20 elementowe — 99.9%;
- puzzle 35 elementowe — 98.1%;
- puzzle 54 elementowe — 98.6%.

Algorytm opierający się na analizie kształtu konturu osiągnął bardzo dużą skuteczność w przeciwieństwie do algorytmu używającego Harris corner detection. Powód tak dużej rozbieżności tych dwóch rozwiązań, to lepsze dopasowanie pierwszego podejścia do zbioru danych, jakimi są puzzle. Analizując kontur, potencjalne wierzchołki znajdują się kolejno po sobie, co pozwala ograniczyć zbiór kandydatów. Natomiast Harris corner detection nie gwarantuje, że potencjalne wierzchołki są uszeregowane w konkretny sposób, powoduje to konieczność sprawdzania wszystkich możliwych kombinacji w celu znalezienia najlepszych kandydatów. Z uwagi na to, jeśli potencjalnych kandydatów zostanie wygenerowane zbyt wiele, algorytm zajmie znacznie więcej czasu niż w przypadku podejścia bardziej dopasowanego do zbioru danych wejściowych.

8.2.3 Dopasowanie krawędzi

Dopasowanie krawędzi może być przeprowadzone na wiele sposobów, z uwagi na ilość informacji, jakie mogą być użyte podczas ich opisywania. Do oceny działania tego etapu posłużyono się jedynie zestawami zawierającymi 20 elementów, gdyż dla zestawów zawierających więcej elementów żadne z przedstawionych rozwiązań nie pozwoliło uzyskać jednoznacznego wyniku.

Dopasowanie geometryczne

Podczas geometrycznego dopasowywania elementów najbardziej przydatnymi parametrami były:

- długość krawędzi;
- porównanie wysokości kolejnych punktów krawędzi;
- odległość między punktami opisującymi dziurę lub wypustkę.

Natomiast najmniej przydatne to:

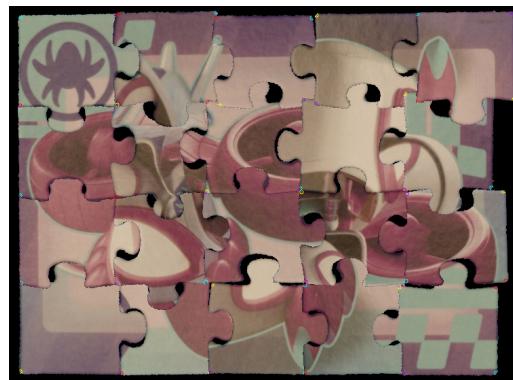
- średnia wysokość krawędzi;
- najwyższa i najniższa wysokość.

Rysunek 8.8 w części A przedstawia najlepsze uzyskane rozwiązania z użyciem dopasowania geometrycznego. Część B zawiera najgorsze ułożenie.



RYSUNEK 8.8: Najlepsze i najgorsze dopasowanie elementów przy użyciu kształtu krawędzi [opracowanie własne].

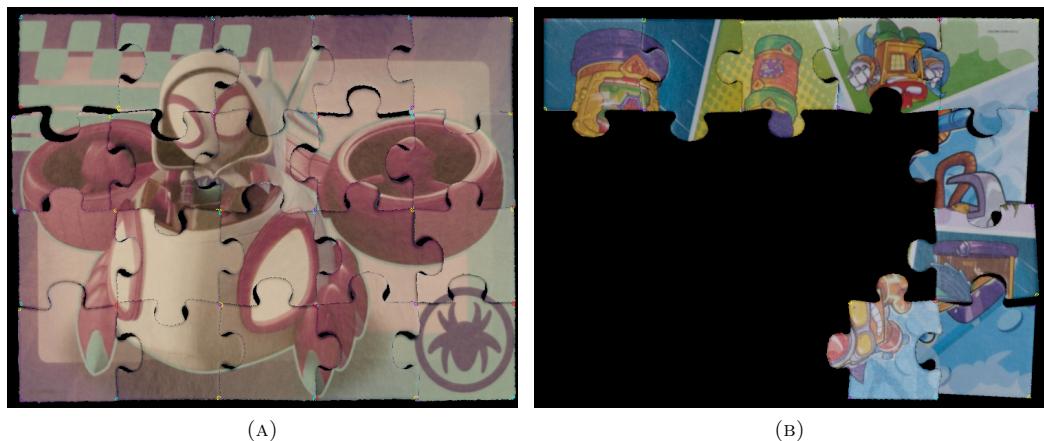
W dopasowaniu geometrycznym najbardziej znaczącą rolę odgrywa porównanie długości krawędzi. Poniższa grafika przedstawia, w jaki sposób zmieniło się najlepsze rozwiązanie, kiedy ten parametr nie był brany pod uwagę.



RYSUNEK 8.9: Najgorsze dopasowanie elementów przy użyciu kształtu krawędzi [opracowanie własne].

Dopasowanie kolorystyczne

W przypadku dopasowania kolorystycznego jedynym parametrem poprawiającym jakość ułożenia okazało się porównanie kolorystyczne kolejnych punktów krawędzi. Na rysunku 8.10 w części A przedstawiono najlepsze efekty użycia tego parametru, natomiast najgorsze w części B.



RYSUNEK 8.10: Najlepsze i najgorsze dopasowanie elementów przy użyciu kolorów krawędzi [opracowanie własne].

Dopasowanie kolorystyczne może przynieść bardzo dobre efekty, w przypadku najlepszego rozwiązania, wszystkie elementy zostały dopasowane poprawnie. Natomiast gdy kolorystyka jest bardziej zróżnicowana, jak w przypadku najgorszego rozwiązania uzyskane efekty są znacznie gorsze niż w przypadku podejścia geometrycznego.

Dopasowanie geometryczne i kolorystyczne

Łącząc oba wyżej wymienione sposoby oceny dopasowania, dla danych wejściowych użytych do stworzenia najgorszego rozwiązania w poprzednich algorytmach, uzyskano efekt przedstawiony na rysunku 8.11.



RYSUNEK 8.11: Efekt uzyskany po połączeniu dopasowania geometrycznego i kolorystycznego [opracowanie własne].

Na podstawie przeprowadzonych obserwacji można zauważyć, że najlepsze rezultaty przyniosło podejście łączące geometryczne i kolorystyczne dane krawędzi. Dopasowanie kolorystyczne przydatne jest do oceny szczegółów, natomiast geometryczne pozwala na lepszą ocenę, które elementy można ze sobą połączyć.

Rozdział 9

Wnioski

Z przeprowadzonych badań można określić wpływ jakości danych wejściowych na skuteczność działania poszczególnych algorytmów. W pierwszym etapie procesu układania tzn. oddzielania elementów od tła najbardziej wszechstronnym podejściem okazało się zastosowanie segmentacji semantycznej uczonej na zdjęciach zestawów puzzli. Jest to jednak metoda wymagająca najwięcej czasu na przygotowanie danych uczących. Dlatego użycie w procesie uczenia sieci, zdjęć pojedynczych elementów układanki ma również swoje zastosowanie, kiedy zbiór danych uczących jest mniejszy. Zaskakujące okazało się, że maska kolorów, będąca dużo prostszym algorytmem, sprawdzała się lepiej w kilku przypadkach. Natomiast jej istotną wadą jest to, że działa ona dobrze tylko w przypadku zdjęć z jednokolorowym tłem i równomiernym oświetleniem. Głównymi cechami obrazów, które wpływały na działanie algorytmów na tym etapie były:

- tło — obrazy z jednorodnym tłem ułatwiają odseparowanie od niego elementów układanki;
- oświetlenie — równomierne oświetlenie podobnie jak tło pomaga w rozróżnianiu elementów;
- kontrast — zauważalnie lepsze rezultaty udało się uzyskać, kiedy kolor tła kontrastował z kolorem elementów.

Ważnym etapem w procesie analizowania elementów układanki jest znalezienie wierzchołków, w tym kroku znacząco lepsze rezultaty uzyskał algorytm analizujący kształt konturów. Może być to spowodowane tym, że jest on wyspecjalizowany w analizowaniu geometrii puzzli, która pomimo tego, że jest dość skomplikowana, nie różni się w znaczącym stopniu między elementami. Kolejną ważną cechą tego rozwiązania jest szybkość działania, która była zdecydowanie wyższa niż w przypadku algorytmu opierającego się na Harris corner detection. Cechy obrazów, które miały największy wpływ na działanie tych algorytmów to:

- odległość pomiędzy elementami — jeśli w obrazie elementy na siebie nachodzą lub są bardzo blisko siebie, mogą one zostać uznane za jeden element, co powoduje błędna detekcję wierzchołków;
- rotacja elementów — w szczególnych przypadkach, kiedy element ustawiony jest diagonalnie, wybrane algorytmy znajdowania narożników mogą mieć trudność z prawidłowym odnajdowaniem rogów;
- kąt, pod jakim zrobione zostało zdjęcie — najlepsze rezultaty uzyskano, kiedy kąt ustawienia puzzli do obiektywu był bliski 180° . Jest to spowodowane tym, że taka perspektywa najlepiej zachowuje kształt elementów. Późniejsza korekcja perspektywy może powodować zmiany w geometrii elementów i utrudniać ich poprawną klasyfikację.

Z analizy poszczególnych etapów działania programu można wywnioskować, że najlepszy zestaw algorytmów jest następujący:

1. Znajdowanie elementów w obrazie przy pomocy — segmentacji semantycznej uczonej na zdjęciach zbiorów puzzli.
2. Detekcja narożników przy użyciu algorytmu analizującego kształt konturów.
3. Analiza porównawcza krawędzi z zastosowaniem danych geometrycznych oraz kolorystycznych.
4. Łączenie elementów ze sobą metodą tworzenia obramowania, a następnie warstwowego wypełniania w głąb.
5. Stworzenie obrazu wynikowego łącząc elementy ze sobą wierzchołkami.

W sytuacji gdy analizowane są jedynie obrazy na jednokolorowym tle, dobrą alternatywą dla segmentacji semantycznej może również okazać się maska kolorów, która wymaga znacznie mniej przetwarzania.

Przygotowany program potrafi poprawnie układać puzzle zawierające do 20 elementów, bardziej liczne zestawy są trudniejsze, głównie ze względu na to, że do uzyskania prawidłowego rozwiązania wymagane jest, aby wszystkie puzzle zostały poprawnie rozpoznane. Pomimo dużej efektywności poszczególnych algorytmów wystarczy jedna źle rozpoznana krawędź, aby ułożenie było niemożliwe. Dodatkowo jeśli źle zostanie rozpoznany element wchodzący w skład obramowania, wewnętrzne elementy nigdy nie zostaną ułożone. Dlatego najbardziej problematycznym krokiem w procesie automatycznego układania puzzli z użyciem wybranych algorytmów wydaje się być kategoryzacja elementów.

Możliwe usprawnienia

Krokiem, którego udoskonalenie wpłynęłoby najbardziej na jakość otrzymywanych rozwiązań, jest znajdowanie wierzchołków elementów, ponieważ jeden źle rozpoznany narożnik może zupełnie zmienić kształt krawędzi puzzla.

Innym wartym usprawnienia aspektem, może być sposób łączenia elementów. Pomimo iż układanie warstwami zapewnia najlepsze rezultaty, jest bardziej czułe na błędy w klasyfikacji elementów tworzących obramowanie układanki. Możliwym rozwiązaniem tego problemu jest kontynuowanie układania kolejnych warstw nawet jeśli w poprzednich warstwach nie udało się stworzyć pełnego obramowania.

Literatura

- [1] Fischler M. A. and Elschlager R. A. The representation and matching of pictorial structures. in *IEEE Transactions on Computers*, vol. C-22, no. 1, pp. 67-92, Jan. 1973, doi: 10.1109/T-C.1973.223602, 1973.
- [2] Travis V. Allen. Using computer vision to solve jigsaw puzzles <https://api.semanticscholar.org/CorpusID:43959392>, 2016.
- [3] Cyganek Boguslaw and Siebert J Paul. *An introduction to 3D computer vision techniques and algorithms*. John Wiley & Sons, 2011.
- [4] Rosenblatt Frank. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychol Rev.* 1958 Nov;65(6):386-408. doi: 10.1037/h0042519. PMID: 13602029., 1957.
- [5] Lowe David G. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.
- [6] Lindsay W. Grace. Convolutional Neural Networks as a Model of the Visual System: Past, Present, and Future. *Journal of Cognitive Neuroscience*, 33(10):2017–2031, 09 2021.
- [7] Devlin J., Chang M.W., Lee K., and Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.
- [8] Snell Jake, Swersky Kevin, and Zemel Richard. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.
- [9] Kalamoniak Mateusz and Piniarska Zuzanna and Subocz Marek and Łukasik Ewa. *Internetowy system rozpoznawania zapisu natowego wykorzystujący głębokie sieci neuronowe*, pages 16–21. Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie, 2021.
- [10] Fei-Fei L, Johnson Justin, and Yeung Serena. Detection and segmentation. [on-line, dostęp: 02.01.2024] http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf.
- [11] Roberts G. Lawrence. Machine perception of three-dimensional solids. Diss. Massachusetts Institute of Technology, 1963.
- [12] Chen Liang-Chieh, Papandreou George, Schroff Florian, and Adam Hartwig. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.
- [13] Torrey Lisa and Shavlik Jude. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.
- [14] Wang Panqu, Chen Pengfei, Yuan Ye, Liu Ding, Huang Zehua, Hou Xiaodi, and Cottrell Garrison. Understanding convolution for semantic segmentation. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 1451–1460. Ieee, 2018.
- [15] Jasiak Patryk. Repozytorium zawierające zbiór danych testowych. [on-line, dostęp: 05.05.2024] <https://github.com/PatrykJasiak/PuzzleImages>, 2024.
- [16] Warden Pete and Situnayake Daniel. *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly Media, 2019.

- [17] Kundu Rohit. Everything you need to know about few-shot learning. [on-line, dostęp: 02.01.2024] <https://blog.paperspace.com/few-shot-learning/>.
- [18] Lorenz Dominik Esser Patrick Ommer Björn Rombach Robin, Blattmann Andreas. High-resolution image synthesis with latent diffusion models. [on-line, dostęp: 06.01.2024] <https://doi.org/10.48550/arXiv.2112.10752>, 2022.
- [19] Źródło internetowe. Alphago — wikipedia, the free encyclopedia. [on-line, dostęp: 13.03.2024] <https://en.wikipedia.org/wiki/AlphaGo>.
- [20] Źródło internetowe. Ames room — wikipedia, the free encyclopedia. [on-line, dostęp: 06.01.2024] https://en.wikipedia.org/wiki/Ames_room.
- [21] Źródło internetowe. Canny edge detector — wikipedia, the free encyclopedia. [on-line, dostęp: 06.01.2024] https://en.wikipedia.org/wiki/Canny_edge_detector.
- [22] Źródło internetowe. Optical character recognition — wikipedia, the free encyclopedia. [on-line, dostęp: 02.01.2024] https://en.wikipedia.org/wiki/Optical_character_recognition.
- [23] Źródło internetowe. Overview of gan structure. [on-line, dostęp: 06.01.2024] https://developers.google.com/machine-learning/gan/gan_structure.
- [24] Źródło internetowe. Simultaneous localization and mapping — wikipedia, the free encyclopedia. [on-line, dostęp: 06.01.2024] https://en.wikipedia.org/wiki/Simultaneous_localization_and_mapping.
- [25] Źródło internetowe. What is computer vision? [on-line, dostęp: 02.01.2024] <https://www.ibm.com/topics/computer-vision>.
- [26] Źródło internetowe. What is explainable ai? [on-line, dostęp: 09.01.2024] <https://www.ibm.com/topics/explainable-ai>.



© 2024 inż. Patryk Jasiak

Instytut Informatyki, Wydział Informatyki i Telekomunikacji
Politechnika Poznańska

Skład przy użyciu systemu L^AT_EX na platformie Overleaf.