Uniwersytet Wrocławski

Praca magisterska

Ocena jakości obrazów w sekwencji zdjęć

Autor:
Karol Kontny

Promotor:
Andrzej Łukaszewski

1 grudnia 2015

Streszczenie

Niniejsza praca prezentuje i porównuje różne algorytmy służące do oceny jakości obrazu, przede wszystkim skupiając się na ostrości. Proponuje też modyfikacje oraz szereg innych technik służących do oceny jakości obrazów.

W rozdziale drugim pracy są zaprezentowane techniki, które są często używane w algorytmach oceny jakości obrazów. Praca prezentuje kilka metod oceny ostrości obrazu (rozdział 3).

W kolejnym rozdziale opisane są popularne bazy danych zdjęć, często używane do testowania metod oceny jakości obrazów. Opisane są też zdjęcia, które zostały wykonane do tej pracy i na których były przeprowadzane w większości badania.

Rozdział piąty pracy to opis i porównanie metryk opisanych w rozdziale trzecim. W rozdziale szóstym proponujemy ulepszenia do algorytmów z rozdziału trzeciego oraz nowe algorytmy oraz opisujemy wyniki ich skuteczności.

Abstract

This paper presents and compares image quality assesment (IQA) algorithms, especially on sharpness assesment. We also propose modifications and another techniques for image quality assesment.

In second chapter of the paper we present popular techniques that are used in IQA algorithms. In third chapter we present few algorithms for sharpness measuement.

In next chapter there are popular image databases describes, which are often used to test IQA algorithms. Also in this chapter we described our own image database, which were taken for purpose of this paper. Most reaserch were done on that image database.

Fifth chapter is a description and comparison of algorithms described in chapter three. In sixth chapter we propose modifications for methods from chapter three and we propose new algorithms.

Spis treści

Rozdział 1

Wstęp

Celem pracy jest zaproponowanie algorytmicznej metody do porównywania jakości obrazów, zakładając, że dysponujemy sekwencją zdjęć przestawiającą ten sam obiekt. Wraz z rozwojem fotografii cyfrowej oraz powszechności urządzeń elektronicznych, które posiadają funkcję aparatu pojawiły się możliwości robienia bardzo wielu zdjęć, nie troszcząc się zbytnio o ich jakość. Wiele aparatów posiada funkcje robienia sekwencji zdjęć. Wśród tak wielu zdjęć często tylko kilka ujęć jest dobrych w sensie technicznym (ostrość, jasność, kontrast, nasycenie etc.), a także percepcyjnym (ustawienie obiektów na zdjęciu, dobry kadr). Przede wszystkim skupiamy się na ich ostrości, ale bierzemy pod uwagę także inne czynniki takie jak: szum, kontrast, jasność obrazu.

Od kilkunastu lat trwają badania nad algorytmami automatycznie oceniającymi obrazu pod wieloma kryteriami, powstało na ten temat wiele prac, została opracowana systematyka algorytmów do oceny jakości obrazu. Mimo tych wszystkich badań współczesne programy do obróbki zdjęć (jak np. Adobe Lightroom) nie posiadają funkcjonalności typu automatycznego wyboru dobrych zdjęć.

Określanie ostrości zdjęcia najczęściej odbywa się poprzez badanie krawędzi w obrazie, dlatego prezentujemy różne techniki pozyskiwania krawędzi, które są wykorzystywane w różnych algorytmach. Porównujemy kilka istniejących metryk służących do oceny jakości obrazu, które wykorzystują różne rozwiązania, aby przekonać się jakie są najskuteczniejsze, następnie wprowadzamy ulepszenia do nich. Proponujemy także inne algorytmy. Ponieważ algorytm powinien automatycznie określać jakość obrazu, celem jest uzyskanie ocen, które odpowiadają temu jak ludzie oceniają ten obraz. W tym celu przygotowaliśmy bazę zdjęć i porównujemy wyniki otrzymane przez algorytm z ocenami ludzi.

Rozdział 2

Metody pozyskiwania krawędzi

2.1 Operator Sobela

Jeżeli potraktujemy obraz jako sygnał $f(x)$, to krawędzie w obrazie można zdefiniować, jako miejsca gdzie są najbardziej dynamiczne zmiany sygnału, czyli jako miejsca gdzie pierwsza pochodna ma ekstrema[8]. W celu wykrycia krawędzi w obrazie, czyli sygnale dwuwymiarowym musimy policzyć jego pochodną, którą nazywamy gradientem. Zatem policzymy pochodne cząstkowe w obu kierunkach.

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} & \frac{\partial f(x)}{\partial x_2} \end{bmatrix} \tag{2.1}$$

Jednak obraz nie jest sygnałem ciągłym, ale dyskretnym, więc musimy w jakiś sposób oszacować pochodną obrazu. W tym celu stosuje się operatory, których nałożenie na obraz daje pewne przybliżenie pochodnej. Wśród wielu metod jest operator Sobela, który charakteryzuje się dość dobrymi wynikami, dodatkową zaletą jest separowalność filtra, co wpływa na możliwość jego efektywnej implementacji. Jest to operator, który należy do grupy operatorów do regulowanego wykrywania krawędzi (regularized edge detectors)[8]. Takie filtry wygładzają obraz, co czyni je bardziej odpornymi na przypadkowe i niepożądane wzmocnienie szumu w obrazie. Operator Sobela rozmywa obraz w kierunku prostopadłym do szukanej pochodnej. Maski operatora Sobela w obu kierunkach są widoczne poniżej.

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \tag{2.2}$$

$$M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \tag{2.3}$$

Gdy policzymy cząstkowe oszacowania pochodnej w kierunku poziomym i pionowym w danym punkcie, to możemy obliczyć wartość gradientu w nim ze wzoru:

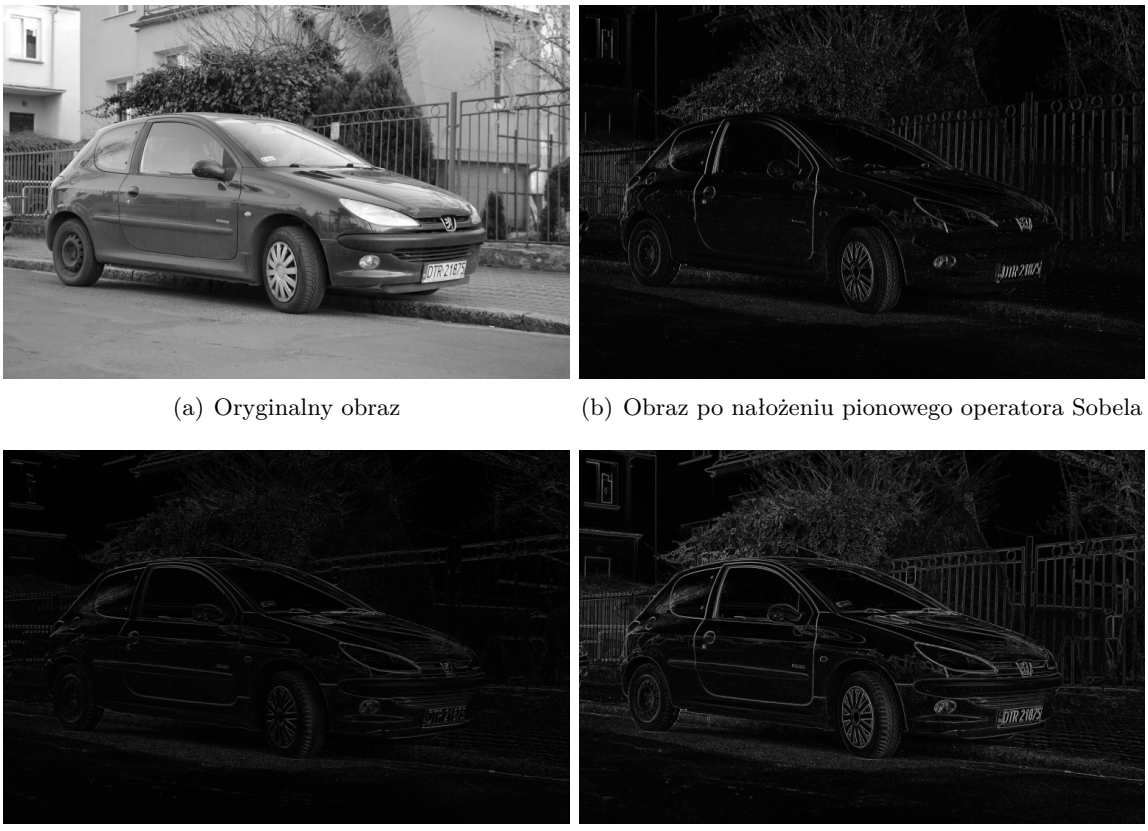
$$G = \sqrt{G_x^2 + G_y^2} \tag{2.4}$$

Gdzie G_x oraz G_y to wartości w danym punkcie obrazu po nałożeniu operatora, czyli po splocie obrazu z maską operatora. Czasami używa się bardziej uproszczonego szacowania ze względu na lepszą szybkość działania:

$$G = |G_x| + |G_y| \tag{2.5}$$

W wyniku tej operacji dostajemy wartości gradientu (długości wektorów). Musimy jeszcze znaleźć kierunek wektora w każdym punkcie. Kierunek otrzymujemy ze wzoru:

$$\theta = arctan\left(\frac{G_y}{G_x}\right) \tag{2.6}$$



(a) Oryginalny obraz

(b) Obraz po nałożeniu pionowego operatora Sobela

(c) Obraz po nałożeniu poziomego operatora Sobela

(d) Obraz po nałożeniu złożenia pionowego i poziomego operatora Sobela

Rysunek 2.1: Przykład działania operatora Sobela

2.2 Canny Edge Detector[1]

Canny scharakteryzował, jakie cechy powinien mieć dobry algorytm wykrywający krawędzie:

1. Prawdopodobieństwo pomyłki musi być niskie: algorytm musi wykrywać punkty należące do krawędzi z dużym prawdopodobieństwem oraz oznaczać punkty nienależące do krawędzi z małym prawdopodobieństwem.
2. Punkty zaznaczone jako krawędzie powinny być tak blisko, jak to możliwe prawdziwym środkom krawędzi.
3. Pojedyncza krawędź powinna mieć tylko jedną właściwą odpowiedź w wyniku.

Algorytm wykrywający krawędzie można podzielić na kilka kroków:

1. Wygładzanie
2. Obliczanie gradientu obrazu
3. Tłumienie (non-max suppression)
4. Obliczanie progów (thresholding)
5. Śledzenie krawędzi

Krok 1 służy wyeliminowaniu z obrazu szumu, który mógłby spowodować wykrycie krawędzi w miejscach, gdzie w rzeczywistości ich nie ma. Z drugiej strony zbytnie rozmycie obrazka może spowodować błędy w detekcji. W tym celu używa się filtra Gaussa na obrazku. W implementacji użyłem filtra o następującym jądrze:

$$M = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{2.7}$$

Krok 2 to obliczenie gradientu. Istnieje wiele operatorów, które z różną dokładnością aproksymują gradient obrazu. Istotne jest, żeby wybrać rozwiązanie, które będzie optymalne pod względem potrzebnej dokładności i szybkości działania. Dość powszechnie stosowany jest operator Sobela rozmiaru 3×3. Po tej operacji dostajemy wartość, jak i kierunek gradientu w każdym punkcie obrazu.

W kolejnym kroku chcemy wytłumić niektóre piksele na obrazku, aby spełnić założenia 2 i 3 o dobrym wykrywaczu krawędzi. W tym celu dla każdego piksela sprawdzamy, czy ma on największą wartość gradientu w swoim sąsiedztwie. Musimy również wziąć pod uwagę kierunek gradientu w tych obliczeniach: sprawdzamy kierunek i sąsiadujące piksele leżące prostopadle do kierunku i wygaszamy te, które nie mają wartości maksymalnej w porównaniu do sąsiadów. Standardowo krawędzie wykrywamy w czterech kierunkach — pionowym, poziomym oraz obu diagonalnych, zatem kierunek gradientu zaokrąglamy do najbliższej wielokrotności 45°. Możliwe jest wykrywanie krawędzi w mniejszej liczbie kierunków, jeżeli tylko taka jest potrzeba. Wynik działania tłumienia można zobaczyć poniżej.

(a) Przed tłumieniem (b) Po tłumieniu

Rysunek 2.2: Tłumienie

W kroku 4 ustalamy, powyżej jakich wartości uznajemy piksel za należący do krawędzi. Canny w swojej pracy zaproponował znalezienie dwóch progów. Pojedynczy próg powoduje, że krawędzie są poszarpane i brakuje w nich niektórych pikseli. Jeżeli punkt ma wartość powyżej górnego progu h, to automatycznie jest uznawany za krawędź. Jeżeli ma wartość powyżej dolnego progu l, to jest uznany za punkt należący do krawędzi i tylko wtedy, jeżeli łączy się z punktem, który został już zaklasyfikowany jako należący do krawędzi. Takie rozwiązanie powoduje, że jeżeli krawędzie w wyniku nie są poszarpane, nawet mimo wahających się wartości punktów. Canny zalecał, aby stosunek wartości h do l wynosił pomiędzy $2:1$ a $3:1$.

Ostatni krok to stworzenie ostatecznego wyniku na postawie wyliczonych wcześniej progów. Należy przejrzeć pośredni obraz wykonany w kroku 3 i wybrać piksele stanowiące krawędzie według kryterium opisanego w kroku 4. Na rysunku 2.3 widać wynik działania wykrywacza krawędzi:

(a) Oryginalny obraz (b) Obraz z wykrytymi krawędziami

Rysunek 2.3: Canny edge detector

2.3 Funkcje Haara i transformata Haara

Falki Haara (Haar wavelets) jest to baza, w której możemy przechowywać różne rodzaje sygnałów między innymi obrazy. W przeciwieństwie do np. bazy Fouriera w bazie Haara mamy zachowaną lokalność obiektów w transformacie obrazu. Dzieje się tak dlatego, że funkcje bazowe Haara są różne od zera tylko na jednym spójnym fragmencie funkcji. Funkcja matka falek Haara $\psi(t)$ jest określona wzorem[9][10]:

$$\psi(t) = \begin{cases} 1, & \text{dla } t \in [\frac{1}{2}), \\ -1, & \text{dla } t \in [\frac{1}{2}, 1) \\ 0, & \text{wpp} \end{cases} \tag{2.8}$$

Inne funkcje bazowe można otrzymać poprzez skalowanie i przesuwanie funkcji $\psi(t)$.

$$\psi_{j,k} = \psi(2^j t - k) \quad (k = 0, 1 \dots 2^j - 1) \tag{2.9}$$

Transformaty Haara używa się min. w przetwarzaniu obrazów do wykrywania krawędzi, a także bardziej ogólnie w przetwarzaniu sygnałów do ich kompresji.

(a) Oryginalny obraz (b) Obraz po transformacie Haara(3 poziomy)

Widać, że w lewej dolnej ćwiartce każdego poziomu „piramidy" są informacje o gradientach poziomych, w prawym górnym o pionowych, a w prawym dolnym o gradientach diagonalnych w każdej rozdzielczości.

Rozdział 3

Metryki ostrości obrazu

3.1 Wprowadzenie

Metryki oceniające jakość obrazu są dzielone na kilka różnych grup[11][7]:

- Subiektywne — metody oparte na subiektywnych ocenach ludzi. Najczęściej stosowana jest pięciostopniowa skala jakości, w której badana osoba ocenia pokazywane zdjęcie. Wśród tych metod można wyróżnić dwie główne grupy:
  - Metody pojedynczej stymulacji, gdzie pokazuje się oceniającemu jedno zdjęcie, na pewien czas, a po jego upływie prosi o ocenę.
  - Metody podwójnej stymulacji, w których oceniający widzi dwa zdjęcia np. zdjęcie referencyjne i oceniane lub prosi się o porównanie dwóch zdjęć i wybranie lepszego z nich.
- Obiektywne — metody algorytmiczne oceniające jakość zdjęć. Celem badań jest wynalezienie algorytmów, których wyniki odpowiadają subiektywnym ocenom ludzi.
  - Metryki pełnej referencji (full reference — FR) - takie, w których mamy dostęp do obrazu oryginalnego, niemającego wad i porównujemy jakiś obrazek z oryginalnym.
  - Metryki częściowej referencji (reduced reference — RR) - kiedy mamy tylko częściową informację o oryginalnym obrazie, na przykład o jego najważniejszych fragmentach.
  - Metryki bezreferencyjne (no reference — NR) - kiedy nie mamy żadnej informacji o obrazie idealnym, jest to przypadek, który najbardziej odpowiada założeniom postawionym w temacie pracy, gdyż nie mamy żadnej informacji o „właściwej" wersji obrazu, ale z drugiej strony zakładamy, że obrazy przedstawiają te same obiekty, dzięki czemu algorytm może posłużyć się technikami niedostępnymi w klasycznej wersji tego problemu.

W kolenych podrozdziałach opiszemy kilka różnych metod oceny ostrości obrazu typu no–reference.

3.2 Metryka Marziliano[2]

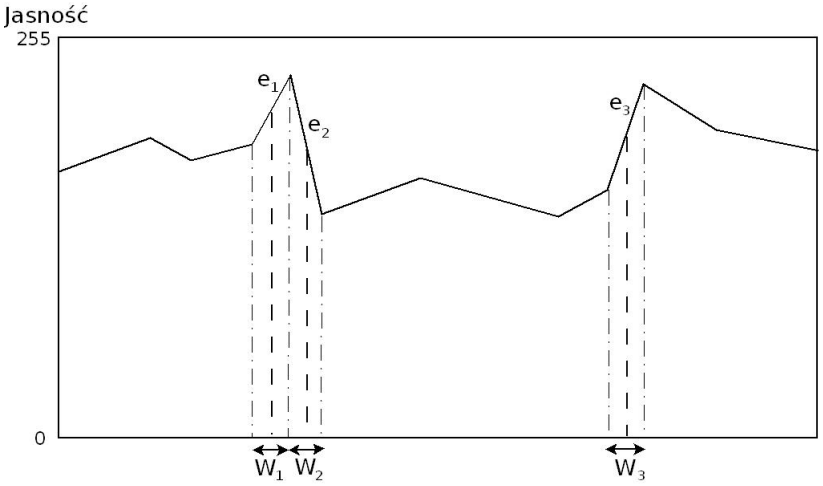
Jest to jedna z pierwszych i jednocześnie bardzo prosta metoda oceniania ostrości obrazów typu no–reference. Opiera się na detekcji krawędzi i sprawdzaniu ich szerokości. Zarys algorytmu można przedstawić w punktach:

1. Oblicz gradient obrazu.
2. Dla każdej krawędzi oblicz jej szerokość.
3. Oblicz wynik jako średnią szerokość krawędzi.

Najpierw używa się operatora Sobela na komponencie jasności (luminance) obrazu wejściowego $S = Sobel(I)$, aby znaleźć w nim krawędzie. Poprzez ustalenie progu t na obrazie wynikowym S eliminuje się szum i mało widoczne krawędzie. Dla polepszenia szybkości działania używa się operatora tylko w jednym kierunku — w pracy wybrano pionowy, autorzy twierdzą, że wybór obu kierunków nie wpływa na poprawę wyników. Następnie dla każdego punktu należącego do krawędzi e_i oblicza się pozycję początku krawędzi r_i i jej końca l_i. Jako początek i koniec rozumie się lokalne ekstrema w jasności obrazu I na prawo i lewo od punktu e_i. Szerokość krawędzi jest różnicą między pozycją końca krawędzi a jej początkiem czyli szerokość $W_i = r_i - l_i$. Wartość metryki dla całego obrazka to suma szerokości krawędzi przez ilość krawędzi, czyli ilość pikseli e_i. Metrykę można opisać wzorem:

$$B = \sum_{i=0}^{N} \frac{l_i + r_i}{N} \tag{3.1}$$

Im mniejsza jest szerokość krawędzi, tym ostrzejszy jest obraz, zatem mniejszy wynik oznacza bardziej ostry obraz.



Rysunek 3.1: Przekrój przez wiersz obrazu I, jasność zaznaczona ciągłą linią, przerywaną linią wystąpienia krawędzi, przerywano – kropkowaną linią lokalne minima i maksima dla krawędzi. W_1 i W_2 oznaczają szerokość krawędzi, e_1 i e_2

3.3 Just Noticable Blur[3]

Wśród wielu algorytmów mierzących ostrość obrazu, część z nich stworzona była do porównywania obrazów o tej samej zawartości. Autorzy chcieli wymyśleć algorytm, który dawałby podobne wyniki dla takiego samego obiektywnego rozmycia obrazu.

Algorytm opiera się na często stosowanej metodzie Just Noticable Difference (JND), której używa się w badaniu różnych czynników na ludzką percepcję. W tej metodzie sprawdza się, jak mała ilość badanego bodźca jest zauważalna przez człowieka. W tym przypadku przeprowadzono badania, kiedy człowiek jest w staniu zauważyć rozmycie obrazu przy zadanym kontraście. Taką najmniejszą wartość rozmycia autorzy nazywają Just Noticable Blur (JNB). Badania pozwoliły oszacować wartość parametru w_{JNB}, który jest wykorzystywany później w algorytmie. Przy szerokości krawędzi równej w_{JNB}, dla zadanego kontrastu prawdopodobieństwo wykrycia jej nieostrości jest równe 63%.

Schemat algorytmu:

1. Wykryj krawędzie w obrazie i oblicz jego gradient.
2. Podziel na bloki i odrzuć bloki gładkie.
3. Oblicz ocenę dla każdego niegładkiego bloku.
4. Oblicz ocenę dla całego obrazu na postawie ocen dla bloków.

Algorytm najpierw używa na wyjściowym obrazie I wykrywacza krawędzi Cannego na obrazie $C = Canny(I)$ oraz nakłada na obraz operator Sobela $S = Sobel(I)$. Następnie obraz jest dzielony na bloki o rozmiarze 64×64 pikseli. W każdym bloku obliczana jest ilość punktów krawędzi na podstawie C. Jeżeli jest mniejsza niż 0.2% pikseli w bloku, to blok jest uznany za gładki i nie jest dalej przetwarzany. W przeciwnym wypadku ocena rozmycia dla bloku R_b jest obliczana ze wzoru:

$$D_{R_b} = \left(\sum_{e_i \in R_b} \frac{w(e_i)}{w_{JNB}(e_i)}^{\beta} \right)^{\frac{1}{\beta}} \tag{3.2}$$

W powyższym wzorze e_i to piksele rozpoznane jako krawędzie na obrazie C, a w_{JNB} ma wartość zależną od kontrastu w bloku. Kontrast k jest obliczany jako różnica jasności między najjaśniejszym a najciemniejszym pikselem w bloku. w_{JNB} ma wartość 5 jeżeli $k \leq 50$ i wartość 3 jeżeli $k > 50$. Natomiast $w(e_i)$ to szerokość krawędzi obliczana tak samo, jak w metryce Marziliano[2] na postawie obrazu S. Parametr β ma wartość 3.6. Ilość wszystkich przetwarzanych bloków oznaczamy przez L.

Następnie, aby otrzymać wynik rozmycia dla całego obrazu używana jest metryka Minkowskiego:

$$D = \left(\sum_{R_b} |D_{R_b}|^{\beta} \right)^{\frac{1}{\beta}} \tag{3.3}$$

$$B = \frac{L}{D} \tag{3.4}$$

Wartość B jest zwracana jako wynik. ilość przetworzonych bloków L przez rozmycie D. Im wyższy jest wynik, tym obraz jest ostrzejszy.

3.4 Cumulative Probability of Blur Detection[4]

Algorytm jest ulepszeniem poprzedniego [3], stosowany jest inny model do oceny rozmycia obrazu, natomiast nadal stosowana jest ta sama koncepcja (JNB) jako podstawa do określenia czy dana krawędź jest rozmyta.

1. Wykryj krawędzie w obrazie i oblicz jego gradient.
2. Podziel na bloki i odrzuć bloki gładkie.
3. Dla każdej krawędzi w bloku oblicz prawdopodobieństwo jej uznania za rozmytą.
4. Znormalizuj wyniki i oblicz wynik.

Na obrazie I obliczany jest gradient za pomocą operatora Sobela $S = Sobel(I)$ i wykrywane są krawędzie za pomocą algorytmu Cannego $C = Canny(I)$. Obraz jest dzielony na bloki o rozmiarach 64×64 pikseli. Jeżeli blok jest uznany za niegładki (na takich samych zasadach jest algorytmie JNB) to dla każdego punktu krawędzi e_i obliczane jest prawdopodobieństwo jej uznania za rozmytą za pomocą wzoru:

$$P_{BLUR}(e_i) = 1 - exp\left(-\left|\frac{w(e_i)}{w_{JNB}(e_i)}\right|^{\beta}\right) \tag{3.5}$$

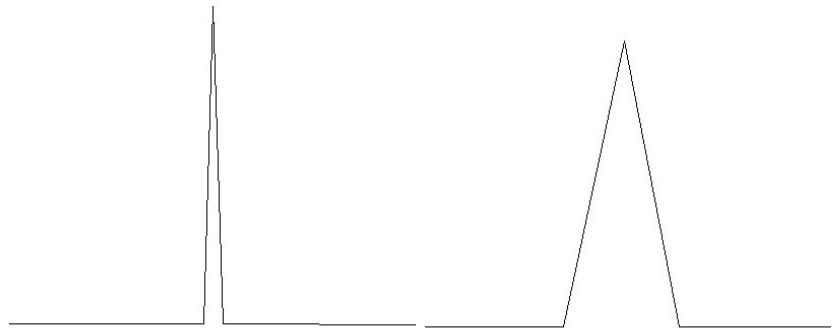
Gdzie $w(e_i)$ to szerokość krawędzi obliczana jak w pracy [2], a parametr $\beta = 3.6$. Wynik prawdopodobieństwa zaokrągla się do 0.01.

Po obliczeniu prawdopodobieństw dla wszystkich krawędzi oblicza się znormalizowany histogram prawdopodobieństw detekcji rozmycia, czyli P_{BLUR}/Liczba wszystkich krawędzi. Wynikiem jest suma gęstości prawdopodobieństw poniżej 63%. Im większy jest wynik, tym obraz jest ostrzejszy. Nieostre krawędzi mają dużą szerokość, zatem prawdopodobieństwo uznania ich za rozmyte jest duże. Wtedy suma gęstości prawdopodobieństw poniżej 63% będzie mała. Kiedy w obrazie jest dużo ostrych krawędzi, to suma małych prawdopodobieństw wykrycia nieostrości jest duża, więc wynik jest większy.

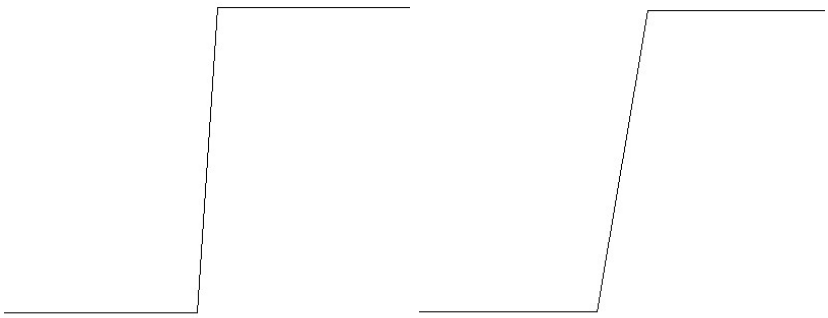
3.5 Metryka Tonga[5]

Ta metryka wykorzystuje przetwarzanie obrazów w dziedzinie falek (wavelets). Na obrazie wejściowym dokonywana jest transformata Harra. Krawędzie w obrazie można podzielić na cztery typy (3.5). Autorzy zauważają, ze w obrazach nieostrych dwa z tych typów nie występują, a pozostałe tracą swój kształt. Zadaniem algorytmu jest znalezienie, które piksele obrazu należą do krawędzi oraz do jakich typów krawędzi one należą. Schemat działania algorytmu można przedstawić następująco:

1. Oblicz transformatę Haara $H = haar(I)$ obrazu.
2. Skonstruuj mapę krawędzi $Emap_i$ dla każdego poziomu transformaty.
3. Skonstruuj mapę maksimów $Emap_i$ na postawie map krawędzi.
4. Oblicz ilość krawędzi poszczególnych typów w obrazie.



(a) Krawędź typu Dirac (b) Krawędź typu Roof



(c) Krawędź typu Astep (d) Krawędź typu Gstep

Rysunek 3.2: Ilustracja typów krawędzi

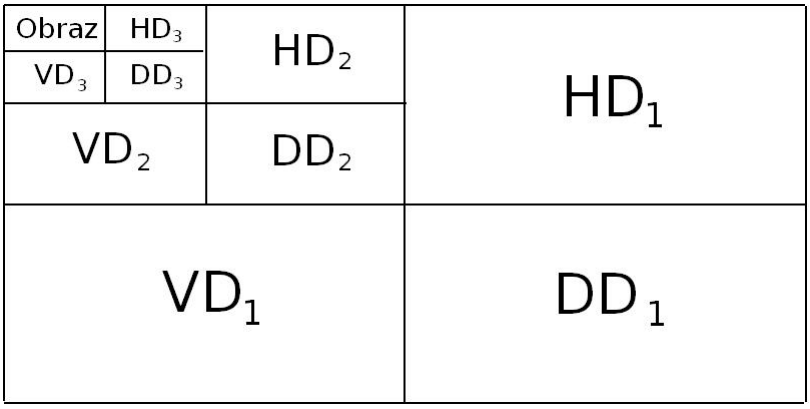
5. Oblicz wynik.

Po obliczeniu trzystopniowej transformaty Haara na obrazie I konstruujemy mapę krawędzi w każdej skali. Wartość każdego piksela otrzymujemy ze wzoru:

$$Emap_i(k,l) = \sqrt{VD_i^2(k,l) + HD_i^2(k,l) + DD_i^2(k,l)} \quad (i = 1,2,3) \tag{3.6}$$

Następnie musimy sprowadzić wszystkie mapy krawędzi do wspólnej skali, w tym celu dzielimy każdą z nich na okna rozmiaru $2^{4-i} \times 2^{4-i}$ i znajdujemy maksymalną wartość w takich oknach. Po tym kroku algorytmu otrzymujemy trzy mapy maksimów $Emax_i \quad (i = 1,2,3)$. Ponieważ transformata Haara wpływa rożnie na różne typy krawędzi, możemy teraz rozpoznać, które piksele należą do jakiego typu. Nie będziemy jednak rozważać pikseli, których wartość jest poniżej pewnego progu $t = 35$, gdyż są one mało widoczne dla człowieka. Do klasyfikowania krawędzi będziemy używać kilku reguł:

1. Jeżeli $Emax_1(k,l) > t$ lub $Emax_3(k,l) > t$ lub $Emax_3(k,l) > t$, to piksel należy do krawędzi, oznaczmy sumę takich punktów przez N_e.



Rysunek 3.3: Ilustracja piramidy z oznaczonymi poszczególnymi obszarami

2. Jeżeli piksel należy do krawędzi i $Emax_1(k,l) > Emax_2(k,l) > Emax_3(k,l)$, to piksel ma typ Dirac lub Astep, oznaczmy sumę, takich punktów przez N_{da}.
3. Jeżeli piksel należy do krawędzi i $Emax_1(k,l) < Emax_2(k,l) < Emax_3(k,l)$, to piksel ma typ Roof lub Gstep.
4. Jeżeli piksel należy do krawędzi i $Emax_1(k,l) < Emax_2(k,l) > Emax_3(k,l)$, to piksel ma typ Roof, oznaczmy sumę pikseli z tego warunku i poprzedniego przez N_{rg}.
5. Jeżeli piksel ma typ Gstep lub Roof i $Emax_1(k,l) < t$, to piksel należy do nieostrej krawędzi, sumę takich pikseli oznaczymy przez N_b.

Ostatnim krokiem algorytmu jest obliczenie wartości metryki, najpierw obliczamy stosunek $P = \frac{N_{da}}{N_e}$ czyli pikseli ostrych krawędzi do wszystkich krawędzi. Jeżeli $P > m$ to zdjęcie zostaje uznane za ostre i wynikiem jest 0, autorzy ustalili wartość parametru $m = 0.05$. W przeciwnym wypadku jako wynik algorytmu zwracane jest $B = \frac{N_b}{N_{rg}}$. Im większy wynik tym gorsza jakość zdjęcia.

3.6 Metryka Kerouh'a[6]

Ta metoda różni się znacznie od poprzedniego algorytmu, ale autorzy często odwołują się do pracy [5], która była punktem wyjściowym. Oto schemat algorytmu:

1. Oblicz transformatę falkową obrazu
2. Oblicz mapę krawędzi
3. Zidentyfikuj krawędzie i krawędzie rozmyte w transformacie
4. Oblicz wynik

Podobnie jak w algorytmie Tonga pierwszym krokiem jest użycie transformaty falkowej. Mapę krawędzi buduje się, nie korzystając z części diagonalnej:

$$Emap_i(k,l) = \sqrt{VD_i^2(k,l) + HD_i^2(k,l)} \quad (i = 1,2,3) \tag{3.7}$$

W algorytmie Tonga był stały próg, powyżej którego stwierdzało się wykrycie krawędzi, tutaj autorzy proponują wartość zależną od poziomu transformaty i oraz średniej.

$$t_i = 2^{i-1} \cdot mean(Emap_i(k,l)) \quad (i = 1,2,3) \tag{3.8}$$

Piksel $Emap_i(k,l) > t_i$, który należy do krawędzi, jest uznawany na nieostry, jeżeli różnica między nim, a jego sąsiadami (rozpatrujemy wszystkich ośmiu sąsiadów) jest mniejsza niż próg, którego wartość jest zależna od poziomu transformaty $\xi_i = 0.5 \cdot 2^{i-1} \quad (i = 1,2,3)$. Zliczamy wszystkie piksele stanowiące krawędzie NE_i i krawędzie nieostre NB_i dla każdego poziomu. Następnie obliczamy współczynnik rozmycia na każdym z poziomów:

$$Q_i = \frac{NB_i}{NE_i} \quad (i = 1,2,3) \tag{3.9}$$

Ostatecznie wynikiem algorytmu jest średnia ważona B. Poziomy, które mają większą rozdzielczość, są uważane za ważniejsze i mnożone przez większe wagi. Im większy wynik B tym lepsza jakość obrazu.

$$B = 1 - \frac{\sum_{i=1}^{3} 2^{3-i} Q_i}{\sum_{i=1}^{3} 2^{3-i}} \tag{3.10}$$

3.7 Metryka Choi[7]

Metryka służy do oceny jakości obrazu na podstawie rozmycia i szumu obecnego w obrazie. Algorytm jest podzielony na następujące fazy:

1. Ocena rozmycia obrazu
   (a) Detekcja krawędzi.
   (b) Szacowanie rozmycia obrazu.
2. Ocena szumu w obrazie.
3. Obliczenie wyniku.

3.7.1 Wykrywanie krawędzi i ocena rozmycia

Wykrywanie krawędzi odbywa się inaczej niż w poprzednich algorytmach, jest to metoda bardzo prosta w porównaniu do poprzednio opisanych. Krawędzie wykrywane są w kierunkach poziomym i pionowym oddzielnie. Opiszemy tu jedną metodę, do wykrywania poziomych krawędzi z indeksami $_h$, gdyż druga jest analogiczna i posiada indeksy $_v$. Najpierw obliczamy

gradient poziomy D_{hb} dla każdego piksela obrazu $I(i,j)$, a następnie średnią D_{hbm}. $H \cdot W$ to rozmiar obrazu (wysokość razy szerokość):

$$D_{hb}(i,j) = |I(i,j+1) - I(i,j-1)| \tag{3.11}$$

$$D_{hbm} = \frac{1}{H \cdot W} \sum_{i=1}^{H} \sum_{j=1}^{W} D_{hb}(i,j) \tag{3.12}$$

Następnie jest obliczana mapa pikseli *kandydujących* do bycia krawędziami $C_h(i,j)$.

$$C_h(i,j) = \begin{cases} D_{hb}(i,j) & \text{jeżeli } D_{hb}(i,j) > D_{hbm}, \\ 0 & \text{wpp} \end{cases} \tag{3.13}$$

Jeżeli piksel (i,j) ma wartość C_h większą niż dwóch sąsiadów, to zostaje uznane jako krawędź $E_h(i,j)$.

$$E_h(i,j) = \begin{cases} 1 & \text{jeżeli } C_h(i,j) > C_h(i,j+1) \text{ i } C_h(i,j) > C_h(i,j-1), \\ 0 & \text{wpp} \end{cases} \tag{3.14}$$

Gdy zostanie dokonana detekcja krawędzi, to kolejnym krokiem jest zdecydowanie które piksele są rozmyte. W tym celu oblicza się wskaźnik rozmycia w obu kierunkach: R_h oraz R_v i wybiera większy z nich i dla ustalonego progu $Th_b = 0.1$ dla wartości mniejszych uznaje się piksel za rozmyty. Wskaźnik rozmycia R_h obliczamy następująco:

$$A_h(i,j) = \frac{1}{2} D_{hb}(i,j) \tag{3.15}$$

$$R_h(i,j) = \frac{|I(i,j) - A_h(i,j)|}{A_h(i,j)} \tag{3.16}$$

R_v definiujemy dla kierunku wertykalnego analogicznie. Wtedy rozmyte piksele to:

$$B(i,j) = \begin{cases} 1 & \text{jeżeli } max(R_h(i,j), R_v(i,j)) < Th_b, \\ 0 & \text{wpp} \end{cases} \tag{3.17}$$

Ostatecznie do oceny rozmycia obrazu używamy dwie liczby: B_m oraz B_r:

$$B_m = \frac{Sum_{blur}}{Blur_{cnt}} \tag{3.18}$$

$$B_r = \frac{Blur_{cnt}}{Edge_{cnt}} \tag{3.19}$$

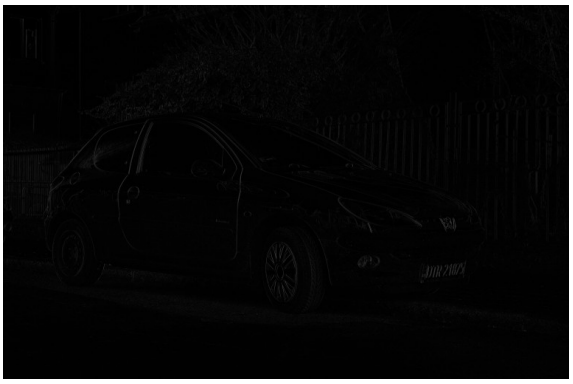
Gdzie $Edge_{cnt}$ to suma pikseli uznanych za krawędzie, $Blur_{cnt}$ to suma pikseli rozmytych, czyli takich których $B(i,j) = 1$, a Sum_{blur} opisuje się wzorem:

$$Sum_{blur} = \sum_{i=1}^{H} \sum_{j=1}^{W} max(R_h(i,j), R_v(i,j)) \tag{3.20}$$

(a) Obraz D_{hb} — gradient poziomy

(b) Obraz z wykrytymi krawędziami poziomymi

(c) Obraz D_{vb} — gradient pionowy



(d) Obraz z wykrytymi krawędziami pionowymi

Rysunek 3.4: Ilustracja wykrywania krawędzi w algorytmie Choi'a

3.7.2 Ocena szumu

Jako pierwszy krok wykrywania i oceny szumu wykonujemy wygładzanie obrazka filtrem „box" o rozmiarze 3×3 o masce:

$$M = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.21}$$

Na powstałym wyniku, podobnie jak w wypadku wykrywania krawędzi obliczamy gradient poziomy i pionowy D_{hn} oraz D_{vn}, a także wartości średnie D_{hnm} i D_{vnm}, prezentujemy tu wzory dla kierunku poziomego, wzory dla kierunku pionowego, indeksowane przez v są analogiczne:

$$D_{hn}(i,j) = |f(i,j+1) - f(i,j-1)| \tag{3.22}$$

$$D_{hnm} = \frac{1}{H \cdot W} \sum_{i=1}^{H} \sum_{j=1}^{W} D_{hn}(i,j) \tag{3.23}$$

Następnie obliczamy mapę pikseli *kandydujących* na szum oraz średnią mapy szumu N_{cm}:

(3.24)
$$N_c(i,j) = \begin{cases} max(D_{vn}(i,j), D_{hn}(i,j)) & \text{jeżeli } D_{hn}(x,y) \leq D_{hnm}(i,j) \text{ i } D_{vn}(i,j) \leq D_{vnm} \\ 0 & \text{wpp} \end{cases}$$

(3.25)
$$N_{cm} = \frac{1}{H \cdot W} \sum_{i=1}^{H} \sum_{j=1}^{W} N_c(i,j)$$

Ostatecznie podejmowana jest decyzja czy dany piksel jest szumem. Tworzona jest mapa szumu $N(i,j)$.

(3.26)
$$N(i,j) = \begin{cases} N_c(i,j) & \text{jeżeli} N_c(i,j) > N_{cm} \\ 0 & \text{wpp} \end{cases}$$

Podobnie jak w wypadku rozmycia obliczane są dwa wskaźniki N_m i N_r:

(3.27)
$$N_m = \frac{Sum_{noise}}{Noise_{cnt}}$$

(3.28)
$$N_r = \frac{Noise_{cnt}}{H \cdot W}$$

gdzie $Noise_{cnt}$ to ilość pikseli szumu, a Sum_{noise} to suma $N(i,j)$.

3.7.3 Obliczenie wyniku

Wynik metryki otrzymuje się poprzez ważenie wielkości opisujących rozmycie i szum w obrazie, wagi są równe $w_1 = 1.0$, $w_2 = 0.95$, $w_3 = 0.3$ i $w_4 = 0.75$. Wagi zostały dobrane poprzez eksperymenty, widać z nich, że szum jest mniej ważny od nieostrości obrazu w ocenie człowieka:

(3.29)
$$B = 1 - (w_1 B_m + w_2 B_r + w_3 N_m + w_4 N_r)$$

Rozdział 4

Zdjęcia

4.1 Wstęp

Do testowania algorytmów potrzebne są zdjęcia, najlepiej zróżnicowane pod względem zawartości — robione z różnych odległości, różnej ilości obiektów na zdjęciu, ilości planów etc. Zdjęcia powinny również zawierać różne błędy, które zdarzają się przy robieniu fotografii, aby sprawdzić, czy metryka prawidłowo ocenia zdjęcia poprawnie wykonane, jak i nieudane. Innym dobrym pomysłem jest wykorzystanie popularnych baz zdjęć, które są wykorzystywane przez autorów prac traktujących o jakości obrazu. W ten sposób możemy porównać nasze wyniki z wynikami różnych autorów na tych samych zdjęciach, co daje lepsze porównanie między algorytmami.

4.2 LIVE Database

Jedną z najbardziej znanych baz danych zdjęć, których używa się w testowaniu jakości obrazów jest LIVE Database [12][13][14]. Wśród zaimplementowanych algorytmów, tylko jeden nie był testowany przez autorów na tej bazie zdjęć. Baza składa się z 29 zdjęć, w rozdzielczości do 768×512 pikseli. Każde ze zdjęć jest dostępne w wersji referencyjnej oraz kilku wersjach zniekształconych poprzez różne algorytmy min: kompresję JPEG i JPEG2000, a także szum, rozmycie Gaussowskie. Jednak baza ma wiele wad, przez które używanie jej do testowania algorytmów może być niemiarodajne. W przypadku zadania postawionego w temacie pracy to nie jest zbyt dobra baza. Wszystkie zniekształcenia są zrobione w sposób syntetyczny, jednakowo na całych obrazach. Takie rozwiązanie sprawdza się w przypadku badania wpływu kompresji na jakość obrazu, ale nie są to błędy, które spotyka się podczas robienia zdjęć aparatem. W naturalnych warunkach zaburzenia nie zawsze obejmują całe zdjęcie, np. nieostrość na jednym z planów. W bazie również nie ma zdjęć zawierających bardziej naturalne błędy: poruszenia, niedoświetlenie, prześwietlenie, ustawienie ostrości na niewłaściwy plan. Ponadto baza LIVE zawiera zdjęcia w niskiej rozdzielczości (najlepsze mają 0.4 MPix, co w dzisiejszych czasach nie można uznać za miarodajne).

Rysunek 4.1: Przykłady zdjęć z bazy LIVE

4.3 Baza zdjęć

W związku z podanymi powyżej powodami, do badania algorytmów została stworzona baza zdjęć zawierająca zdjęcia z różnymi typowymi błędami, które mogą powstać podczas normalnego użytkowania aparatu. Wybraliśmy kilka zróżnicowanych scen pod względem zawartości zdjęcia, skomplikowania planów, odległości od obiektywu. Baza zawiera 95 zdjęć w rozdzielczości 25MPix zgrupowanych w 8 scen oraz 14 zdjęć w rozdzielczości 5MPix w dwóch scenach. Krótko omówimy charakterystykę różnych scen. Wszystkie zdjęcia zostały zrobione bez użycia statywu, dlatego minimalnie różnią się między sobą, co jest przypadkiem naturalnym, dodatkowo można było zbadać jeden z najczęściej występujących błędów, czyli poruszenie aparatu podczas robienia zdjęcia.

4.3.1 Drzewa

Scena z dużą głębią ostrości, z dużą ilością drobnych szczegółów (trawa, gałęzie). Wszystkie obiekty mają podobną ostrość na każdym zdjęciu. W bazie jest 6 zdjęć tej sceny.

Rysunek 4.2: Ujęcia sceny „Drzewa”

4.3.2 Hiacynt

Zdjęcie kwiatu z bliskiej odległości jest to dość trudne do oceny ujęcie, gdyż ze względu na bliską odległość do fotografowanego przedmiotu jest niewielka głębia ostrości. W niektórych zdjęciach ustawiono ostrość na plan za obiektem, co powoduje duże trudności dla algorytmów z rozpoznaniem najlepszego ujęcia. W bazie jest 12 zdjęć tej sceny.

Rysunek 4.3: Ujęcia sceny „Hiacynt”

4.3.3 Krzak

Zdjęcie robione ze średniej odległości (kilka metrów), jest na nim wiele drobnych szczegółów (liście, gałęzie, trawa). W każdym ujęciu pierwszy lub drugi plan jest co najmniej lekko nieostry, co powoduje problem z automatycznym ocenieniem jakości zdjęcia. W bazie jest 15 zdjęć w tej scenie.

Rysunek 4.4: Ujęcia sceny „Krzak”

4.3.4 Krzak 2

Scena podobna do poprzedniej, ale dodano jeszcze jeden plan — człowieka. W związku z tym jest to zdjęcie, które sprawia algorytmom kłopot, gdyż mała część obrazu na zdjęciu udanym jest ostra, reszta jest lekko lub bardziej nieostra. W bazie jest 12 zdjęć tej sceny.

Rysunek 4.5: Ujęcia sceny „Krzak 2”

4.3.5 Kubek

Zdjęcie podobnie jak „Hiacynt” robione z bardzo małej odległości, co skutkuje małą głębią ostrości. W odróżnieniu od tamtego nie ma ono szczegółów, po których można rozpoznać dobre ujęcie (powierzchnia obiektu jest biała). Jedyną pierwszoplanową informacją jest kontur obiektu. Zrobiono zdjęcia z ostrym pierwszym, a także drugim planem, który ma więcej szczegółów, czego algorytmy nie potrafią odróżnić. W bazie jest 12 zdjęć tej sceny.

Rysunek 4.6: Ujęcia sceny „Kubek”

4.3.6 Kwiaty

Zdjęcie zrobione ze średniej odległości, na zdjęciu jest wiele szczegółów, ale wszystkie są na jednym planie. W bazie jest 6 zdjęć w tej scenie.

Rysunek 4.7: Ujęcia sceny „Kwiaty”

4.3.7 Odra

Obraz z dużą głębią ostrości o średniej ilości szczegółów — na pierwszym planie trawa daje dużo krawędzi. Wszystkie obiekty na zdjęciu są jednakowo ostre, co czyni ujęcie dość łatwym dla algorytmów. W bazie jest 19 zdjęć tej sceny.

Rysunek 4.8: Ujęcia sceny „Odra”

4.3.8 Pokój

Zdjęcia robione w niższej rozdzielczości, cześć jest zrobiona przy użyciu lampy, a część bez — te są nieostre. W bazie są 4 zdjęcia tej sceny.

Rysunek 4.9: Ujęcia sceny „Pokój”

4.3.9 Salon

Zdjęcia podobnie jak scena „pokój” były zrobione gorszym sprzętem w niższej rozdzielczości (5Mpix). Zdjęcia różnią się ustawieniem ostrości i lampy. Warto, zauważyć, że scena jest bardzo prosta, ma mało obiektów, większość krawędzi jest pionowa lub pozioma. Dodatkowo środek obrazu jest pusty, większość obiektów znajduje się przy krawędziach zdjęcia. W bazie jest 10 zdjęć tej sceny.

Rysunek 4.10: Ujęcia sceny „Salon”

4.3.10 Samochód

Zdjęcie zrobione z niewielkiej odległości (kilka metrów), na pierwszym planie jest samochód, który nie posiada wielu szczegółów, drugi plan jest nieco bardziej skomplikowany. Zdjęcia gdzie drugi plan jest ostry, są problematyczne dla algorytmów, gdyż ma on więcej szczegółów, a pierwszy plan w takich przypadkach jest nieostry. W bazie jest 13 zdjęć tej sceny.

Rysunek 4.11: Ujęcia sceny „Samochód”

4.4 Ocena zdjęć

Do oceny zdjęć poproszono 9 osób. Miały one za zadanie ocenić każde ze zdjęć w skali od jeden (bardzo dobre) do pięć (bardzo nieudane). Ocenę przeprowadzono metodą pojedynczej stymulacji: pokazywano jedno zdjęcie na raz na czas maksymalnie 20 sekund, po czym proszono o ocenę. Następnie obliczono średnią ocen zdjęć (metoda MOS — Mean Opinion Score). Uszeregowano zdjęcia według średniej ocen — była to wzorcowa kolejność zdjęć według ocen respondentów. Wszystkie osoby oceniające zdjęcia oglądały je na tym samym monitorze – LG FLATRON L225WT, przy podobnym oświetleniu pomieszczenia.

Rozdział 5

Wyniki

5.1 Wprowadzenie

Wszystkie algorytmy wymienione w rozdziale trzecim zaimplementowano w języku C++. W przypadku metryk JNB oraz CPBD zrobiliśmy to na podstawie znalezionej implementacji autorów w MATLABie. Programy działają zarówno pod systemem Windows, jak i Linux. Algorytmy testowano na komputerze z procesorem INTEL Core 2 Duo E8200 2.66 Ghz i 2GB pamięci RAM.

Wyniki algorytmów porównywano z wcześniej obliczoną kolejnością wzorcową. Do oceny jakości algorytmu użyto dwóch metod: korelacji Pearsona oraz korelacji rang Spearmana. Są to najczęściej spotykane metody do oceny tego rodzaju algorytmów.

5.1.1 Korelacja Pearsona

Jest to najbardziej znana metoda obliczania korelacji zmiennych losowych.

$$\rho_{XY} = \frac{cov(X,Y)}{\sigma_Y \sigma_Y} \tag{5.1}$$

Gdzie $cov(X,Y)$ to kowariancja zmiennych losowych X i Y, a σ_X, σ_Y to odchylenia standardowe zmiennych losowych X, Y. Dla próby losowej wzór wygląda następująco:

$$\rho_{XY} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{5.2}$$

Aby ułatwić czytanie wyników algorytmów, zmieniliśmy znak wyników w przypadkach, kiedy wyższa ocena algorytmu oznacza lepszą jakość. W przeciwnym razie część wyników byłaby ze znakiem ujemnym oznaczającym dobrą korelację, a część z dodatnim. We wszystkich przypadkach wyższa korelacja oznacza lepszy wynik.

5.1.2 Korelacja rang Spearmana

Jest to tak zwana korelacja rang, nie porównujemy w niej wartości zmiennych losowych X oraz Y ale kolejność, która jest ustalona przez wartości X_i oraz Y_i. Najpierw ustalamy kolejność

rang: najlepsza obserwacja dostaje rangę 1, kolejna 2, ostatnia dostaje rangę n. Następnie ustalamy wartość współczynnika korelacji ze wzoru:

$$\rho = 1 - \frac{6 \sum_{i=1}^{n} d_i^2}{n(n^2 - 1)} \tag{5.3}$$

5.1.3 Kolejność wzorcowa

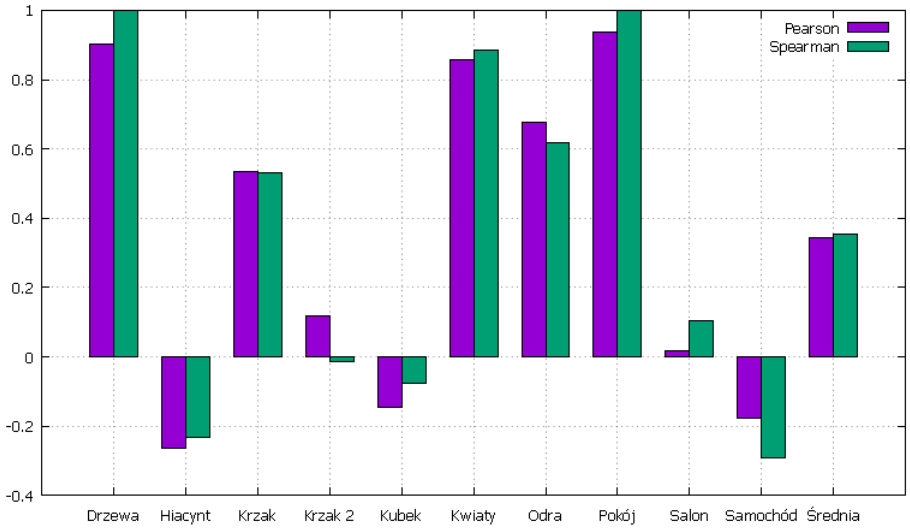
Dla każdego zdjęcia i obliczona średnią jego ocen a_i. W przypadku korelacji Persona te wartości a_i postawiamy jako watości zmienniej losowej X. W przypadku korelacji rang Spearmana na podstawie średnich a_i obliczamy rangi, które służa nam jako watości zmiennej losowej X.

Gdzie d_i oznacza różnicę między rangami zmiennych losowych X i Y, dla próby losowej o numerze i.

5.2 Oceny

Korelacja zmiennych losowych jest liczbą z zakresu $[-1, 1]$. Wartości ujemne oznaczają korelację wsteczną. Wyniki ujemne w naszych testach uznajemy za bardzo złe, algorytm który uzyskuje takie wyniki w jakiejkolwiek scenie nie może być uznany za dobry. Wartości korelacji mięcy 0 a 0.5 uznajemy za wynik słaby, który niewiele mówi o pożądanej przez nas kolejności wyników. Wyniki z przedziału 0.5 - 0.8 uznajemy za wyniki średnie, które pokazują mniej więcej oczekiwaną kolejność, ale posiada błędy. Wyniki powyżej 0.8 uznajemy za dobre, które z niewielkimi błędami pokazują właściwą kolejość.

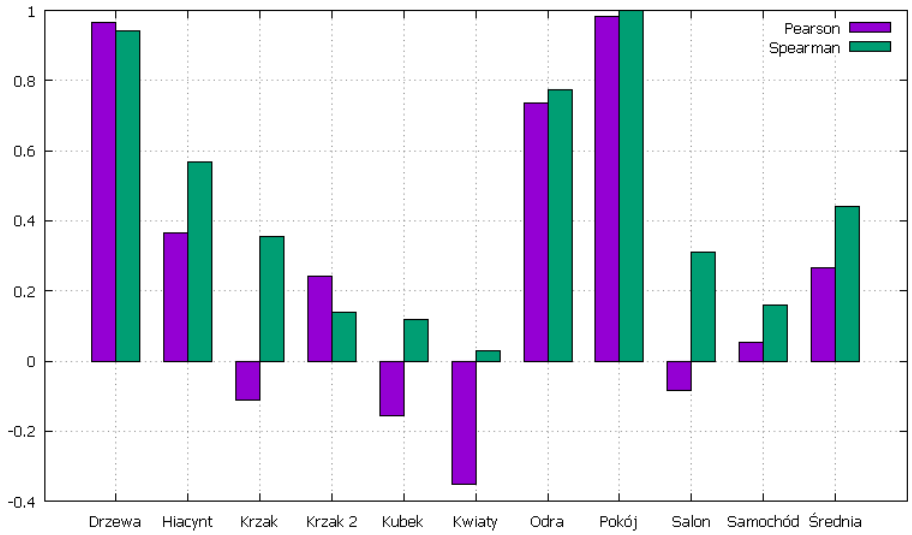
5.3 Marziliano



Tablica 5.1: Wyniki korelacji dla algorytmu Marziliano

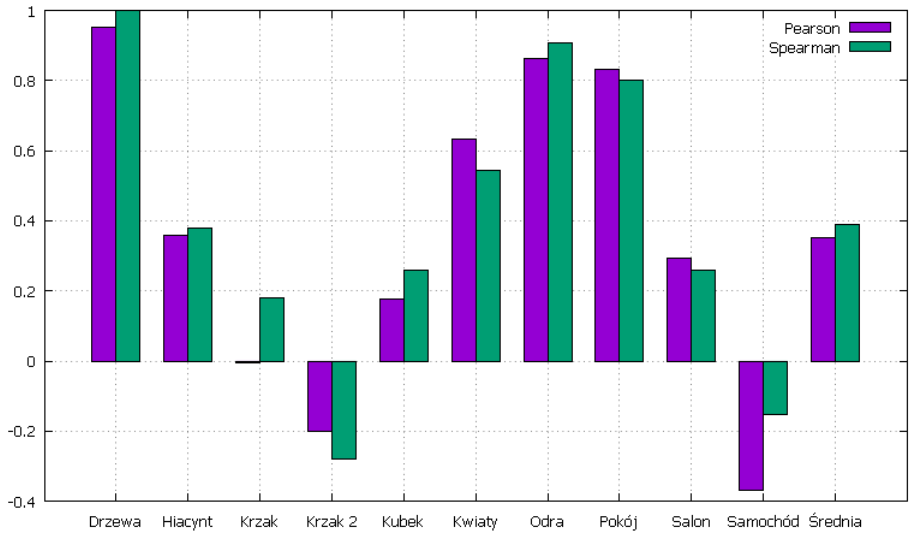
Metryka sprawdzała się na dość prostych scenach: „Drzewa”, „Kwiaty”, „Pokój”, także „Odra”, ale już zdecydowanie nie radziła sobie z bardziej skomplikowanymi: „Hiacynt”, „Kubek” gdzie zdjęcia były robione z bliskiej odległości, a także w scenach gdzie jest wiele planów. Nieco zaskakująca jest niska ocena w scenie „Salon”, gdzie nie ma wielu planów, a rozmycie powinno być łatwo wykrywalne.

5.4 JNB



Tablica 5.2: Wyniki korelacji dla algorytmu JNB

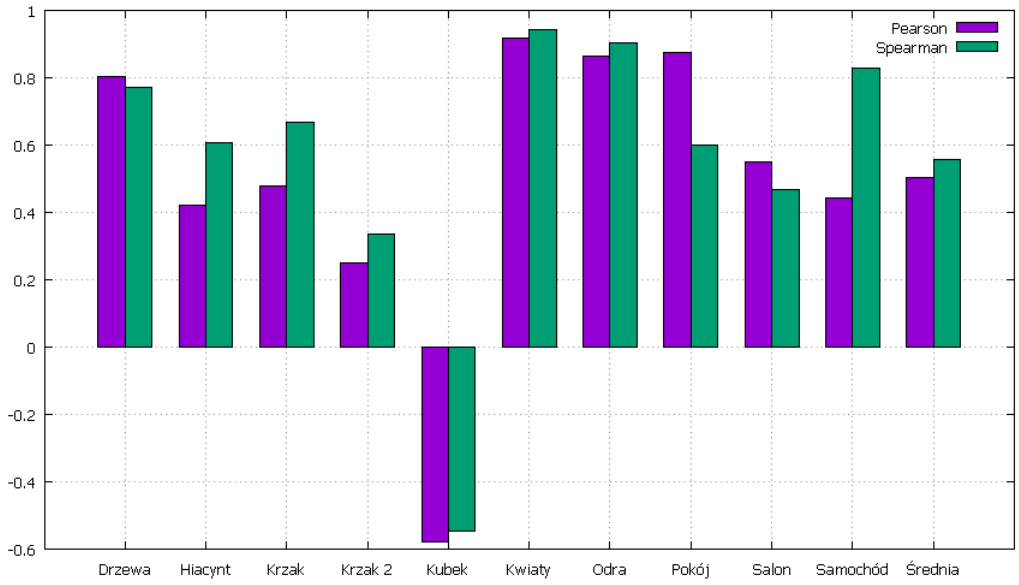
5.5 CPBD



Tablica 5.3: Wyniki korelacji dla algorytmu CPBD

Jest to metoda będąca ulepszeniem JNB. Niestety w przypadku zdjęć z bazy wynik korelacji Pearsona jest niewiele lepszy, a Spearmana nawet słabszy. Algorytm, podobnie jak poprzednik, radzi sobie ze scenami gdzie nie ma kilku planów.

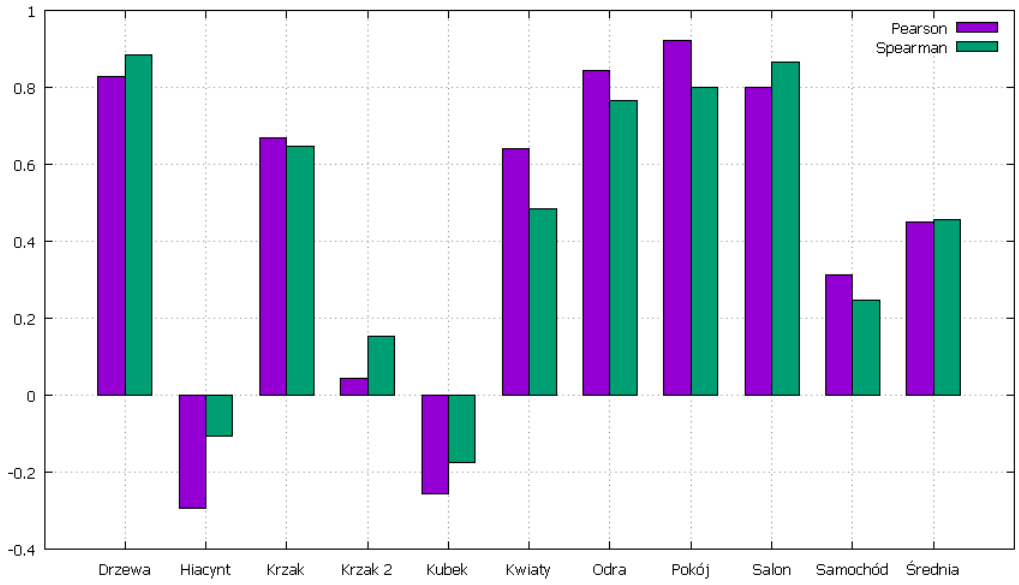
5.6 Tong



Tablica 5.4: Wyniki korelacji dla algorytmu Tonga

Algorytm daje znacznie lepsze wyniki od poprzednich: radzi sobie z łatwymi scenami, jak i z trudniejszymi: w scenach „Hiacynt", „Krzak" czy „Samochód" dawał znacznie lepsze wyniki niż poprzednie algorytmy, ale nadal nie można ich uznać za dobre. Z drugiej strony bardzo zła jest korelacja w scenie Kubek, gdzie poprzednie algorytmy dawały nieco lepsze wyniki, ale również słabe.

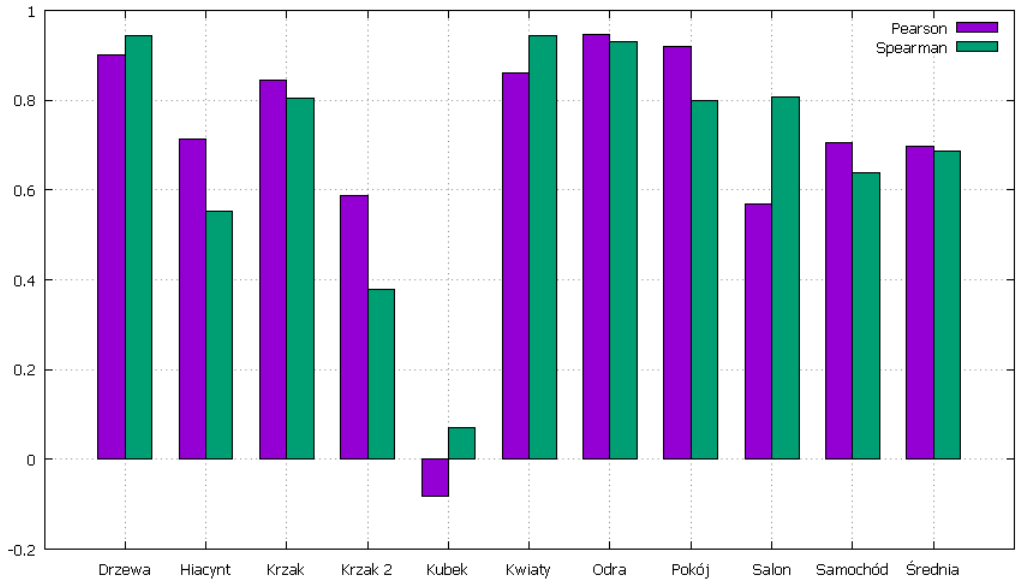
5.7 Kerough



Tablica 5.5: Wyniki korelacji dla algorytmu Kerougha

Metoda powstała przez ulepszenie poprzedniego algorytmu. Niestety wyniku są nieco gorsze od oryginalnej metryki. Jednak wartym zauważenia jest fakt, że algorytm dawał lepsze oceny na scenach „Krzak”.

5.8 Choi



Tablica 5.6: Wyniki korelacji dla algorytmu Choi

Ten algorytm dawał najlepsze wyniki ze wszystkich w tym rozdziale. Algorytm nie poradził sobie z właściwą oceną trudnej sceny: „Kubek”, wyniki dla scen „Krzak 2” i „Hiacynt” są lepsze niż w poprzednich algorytmach, ale nadal niezbyt dobre. Metoda działa zadowalająco w pozostałych przypadkach.

Rozdział 6

Ulepszenia i propozycje algorytmów

6.1 Podział obrazu

W algorytmach oceniających jakość obrazu wszystkie traktują całe zdjęcie w taki sam sposób. W większości praktycznych zastosowań obiekty będące blisko krawędzi obrazu mają mniejsze znaczenie niż te w środku. Ostrość jest przeważnie ustawiona na środek obrazu lub na jakieś ważny obiekt — jedna z zasad fotografii mówi, że ważne obiekty powinny być w 1/3 wysokości i szerokości zdjęcia. Ponieważ części obrazu będące blisko krawędzi są mniej ważne, to można je oceniać z mniejszą wagą. W tym celu zdjęcia dzielono na 64 części (8×8), w każdej obliczano wartość metryki dla każdej z części, a następnie obliczano ostateczną ocenę, sumując wyniki i ważąc je.

Algorytmy używające algorytmu wykrywania krawędzi Cannego były zmodyfikowane w taki sposób, żeby najpierw wykrywały krawędzie, a potem dzielono wynikowy obraz z krawędziami na fragmenty i za jego pomocą obliczano wartość metryki. Wykrywanie krawędzi w podzielonym obrazie mogłoby drastycznie wpłynąć na wynik, jako że progi algorytmu są obliczane na podstawie histogramu operatora Sobela. Zdjęcie po podzieleniu lokalnie może mieć zupełnie inny histogram niż globalnie. W pozostałych przypadkach nie stosowano modyfikacji.

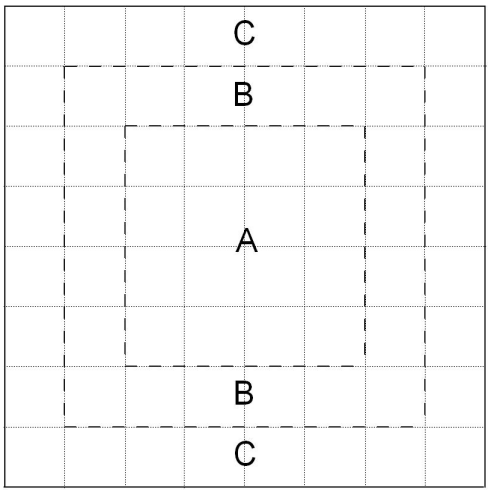
Na rysunku 6.1 widzimy podział zdjęcia na obszary. Obszar A w środku zdjęcia składający się z 16 fragmentów, które jest sumowany z wagą 0.6. Obszarowi C przy krawędzi (28 fragmentów, strefa zewnętrzna) nadano wagę 0.1, a obszarowi B pomiędzy nimi (20 fragmentów, strefa środkowa) wagę 0.3.

$$B_A = \sum_{j \in A} F(I_j) \tag{6.1}$$

$$B_B = \sum_{j \in B} F(I_j) \tag{6.2}$$

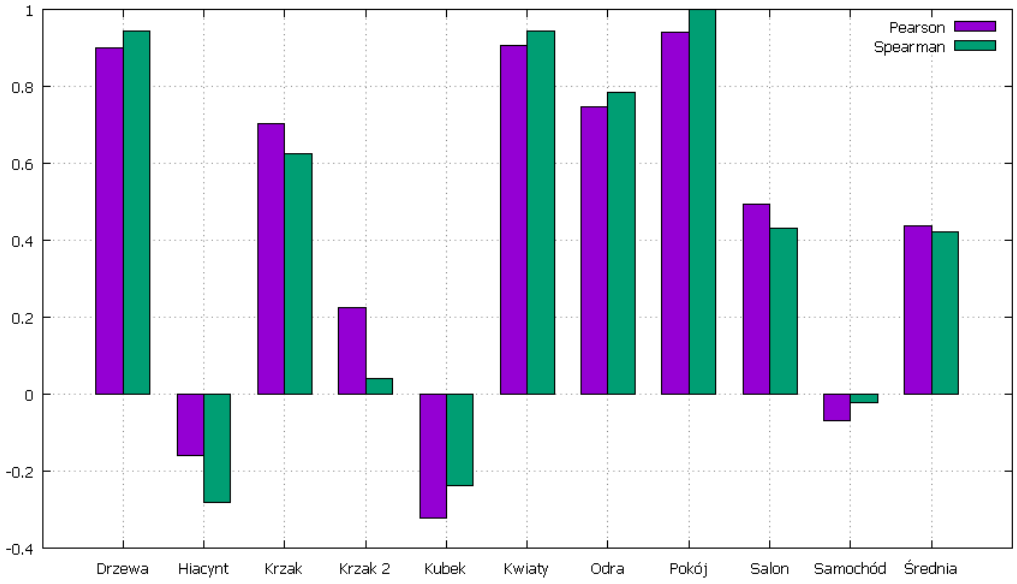
$$B_C = \sum_{j \in C} F(I_j) \tag{6.3}$$

$$B = 0.6B_A + 0.3B_B + 0.1B_C \tag{6.4}$$

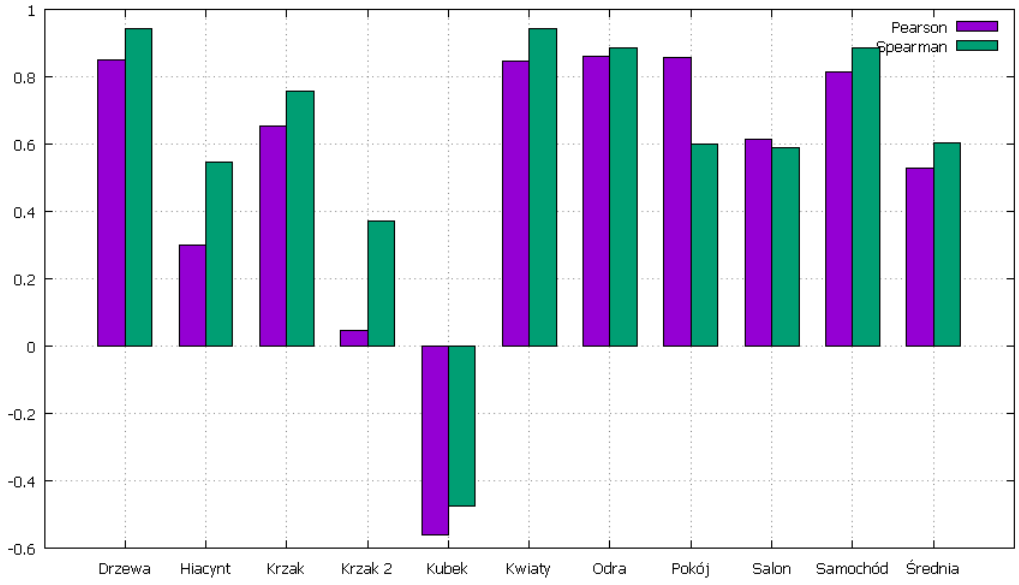


Rysunek 6.1: Ilustracja podziału obrazka na strefy wewnętrzną (A), środkową (B) i zewnętrzną (C).

W dwóch przypadkach zastosowanie tej metody dało pozytywne rezultaty: w przypadku algorytmów Marziliano oraz Tonga. W przypadku pozostałych algorytmów średnia ocena nie zmieniła się lub pogorszyła. W zmodyfikowanym algorytmie Marziliano poprawa w stosunku do oryginalnej metody nastąpiła w większości scen, oprócz sceny „Salon", gdzie obszar pośrodku zdjęcia jest pusty, natomiast większość obiektów znajduje się przy krawędziach obrazu. Nie jest to metoda, którą można stosować zawsze, choć w przypadku większości zdjęć powinna się sprawdzić, ale poprawa wyników jest niewielka.



Tablica 6.1: Wyniki korelacji dla algorytmu Marziliano z podziałem zdjęcia



Tablica 6.2: Wyniki korelacji dla algorymu Tonga z podziałem zdjęcia

W przypadku algorytmu Tonga widać poprawę w przypadku prawie wszystkich scen z wyjątkiem sceny „Krzak 2” gdzie nastąpiło znaczne pogorszenie wyniku, oraz sceny „Kubek” gdzie wynik jest tak samo zły, w obu przypadkach algorytm zupełnie zawiódł w tej scenie.

6.2 Długość krawędzi

Wiele z cytowanych prac w ocenie ostrości zdjęcia wykorzystuje badanie pośrednie([5][6]) lub bezpośrednie([2], [3], [4]) szerokości krawędzi. Do badania ostrości tej samej sceny można wykorzystać również długość wykrytych krawędzi na zdjęciu. Im dłuższe są kawałki, tym bardziej zdjęcie jest ostre. Niestety ta metoda nadaje się tylko do porównywania zdjęć tego samego obiektu, gdyż długość krawędzi może być zupełnie różna w zależności od fotografowanego obiektu. Z drugiej strony algorytm ten pośrednio wykorzystuje założenie o tym że zdjęcia przedstawiają te same obiekty.

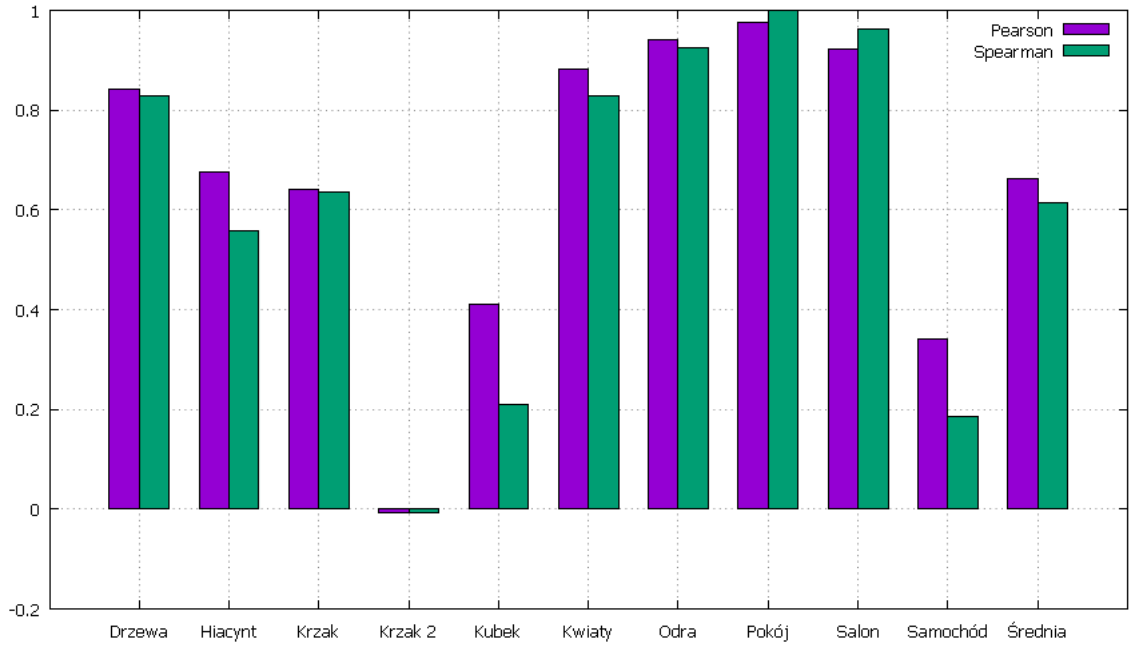
Na postawie tej własności skonstruowaliśmy następujący algorytm:

1. Dokonaj detekcji krawędzi w obrazie.
2. Policz długość każdej krawędzi.
3. Oblicz średnią długość krawędzi.

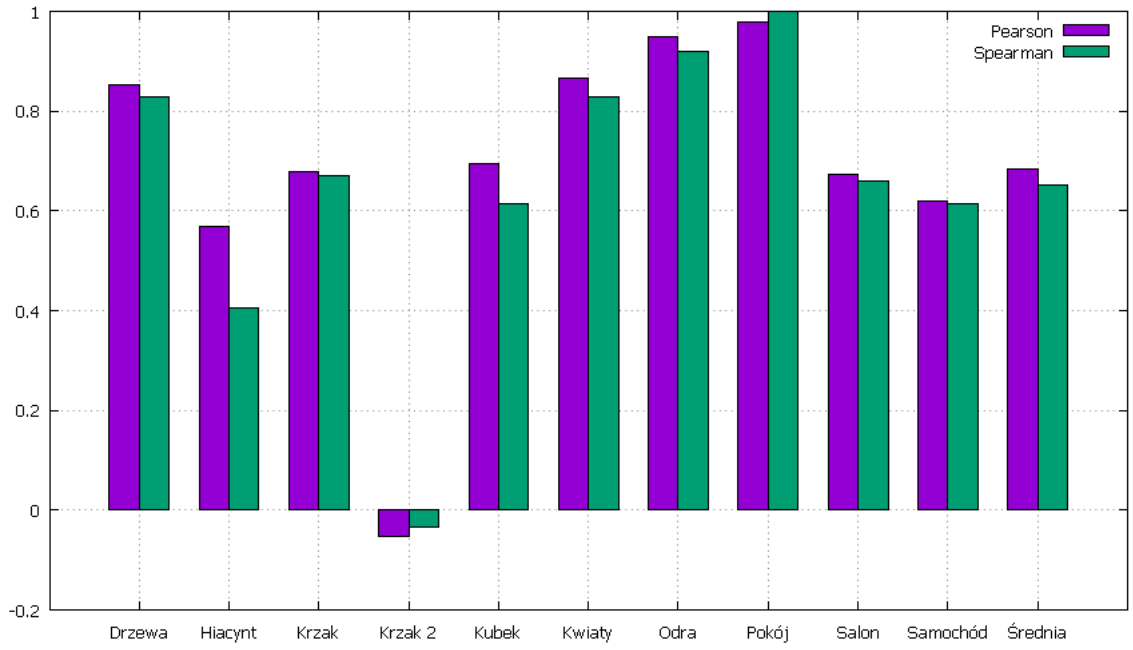
Na obrazie I wykrywamy krawędzie algorytmem Cannego: $C = Canny(I)$. Następnie znajdujemy na obrazie zapalone piksele oznaczające wystąpienie krawędzi. Od takiego piksela zaczynamy przeszukiwanie wszerz (BFS), aby znaleźć długość krawędzi e_i. W tej procedurze sprawdzamy wszystkie sąsiednie piksele (także diagonalne) i jeżeli były nieodwiedzone to dodajemy je do kolejki przeszukiwania, nie zajmujemy się kierunkiem krawędzi. Otrzymujemy długość i–tej krawędzi i oznaczmy ją przez l_i. Im większa jest policzona średnia z długości wszystkich krawędzi, tym lepsza jest ostrość zdjęcia.

$$B = \frac{\sum_{i=0}^{n} l_i}{n} \tag{6.5}$$

Wykonano też obliczenia dla wariantu tej metody z podziałem zdjęcia na fragmenty oraz z użyciem dwóch wersji algorytmu wykrywania krawędzi Cannego z podwójnym progiem, jak i pojedynczym progiem. Wyniki algorytmu są w większości lepsze od zaproponowanych metryk z rozdziału trzeciego. Dzięki wykorzystaniu założenia o sekwencji zdjęć w bardzo prosty sposób udało się skonstruować metodę bardzo prostą i często znacznie lepszą, niż bardziej zaawansowane algorytmy, które nie przyjmowały żadnych założeń.



Tablica 6.3: Wyniki korelacji dla metody długości krawędzi



Tablica 6.4: Wyniki korelacji dla metody długości krawędzi z podziałem zdjęcia

Ta metoda daje lepsze wyniki od poprzednich, w scenach gdzie nie ma wielu planów i jednocześnie wszystko może być ostre, wyniki są lepsze niż w poprzednio opisanych algorytmach. Nadal w scenach „Krzak 2" i „Kubek wyniki są błędne. Po podziale obrazka w większości scen wyniki się lekko poprawiły, jedynie w scenie „Salon" są wyraźnie gorsze, należy jednak zauważyć, że na tym zdjęciu większość obiektów jest przy krawędzi zdjęcia, a środek jest pusty.

6.3 Uwzględnianie histogramu zdjęcia

Wiele z powyższych algorytmów dobrze radzi sobie z rozpoznawaniem ostrych zdjęć, ale te mimo to nie są zbyt dobre. Algorytmy te nie uwzględniają ogólnej jasności obrazu, jako dobre uznają zdjęcia niedoświetlone lub prześwietlone. Aby temu przeciwdziałać, można do metryki dodać uwzględnianie histogramu zdjęcia, preferując zrównoważony. W kilku algorytmach dodaliśmy takie ocenianie, co zauważalnie poprawiało wyniki algorytmów. Dla każdego zdjęcia w serii (właściwie komponentu jasności) obliczano jego histogram, dzieląc go na 16 kubełków, w każdym znajdowały się wartości z przedziału $[16i, 16i + 15] \quad i = (0, 1, \ldots, 15)$. Następnie obliczano jego kwadrat odległości Q_k od histogramu wzorcowego – wyrównanego. Histogram wyrównany to taki, który w każdym kubełku $wz(i)$ ma tyle samo pikseli $wz(0) = wz(1) = \cdots = wz(15)$. Q_k oblicza się następującym wzorem:

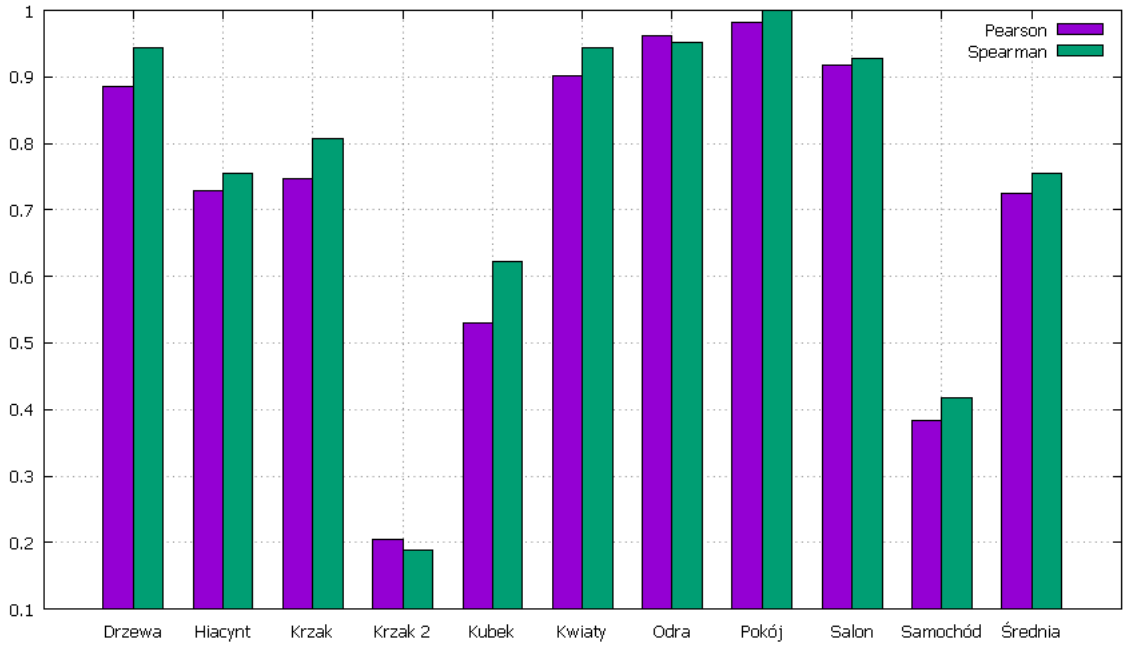
$$Q_k = \sum_{i=0}^{15} \left(\frac{hist(i) - wz(i)}{16} \right)^2 \tag{6.6}$$

Gdzie $hist(i)$ to ilość pikseli w i-tym kubełku histogramu, a k oznacza numer zdjęcia w danej serii. Następnie normalizujemy wszystkie względem najmniejszego wyniku i bierzemy

odwrotność dla mniejszej odległości otrzymać większy wynik:

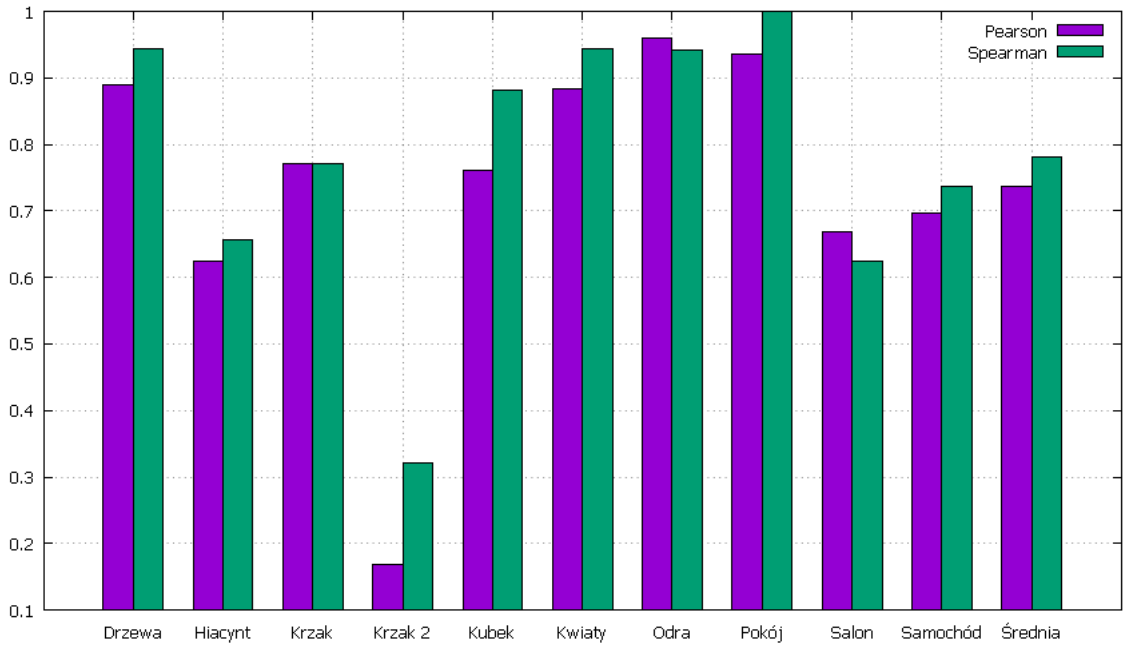
$$H_k = \frac{min(Q_1, Q_2, \ldots, Q_n)}{Q_k} \tag{6.7}$$

Jeżeli ocena histogramu byłą uwzględniana w obliczeniach jej byłe testowana w przedziale 0.2 – 0.25 od całego wyniku. Ostatecznie zdecydowaliśmy się na użycie wartości 0.25 we wszystkich algorytmach zaprezentowanych w pracy.



Tablica 6.5: Wyniki korelacji dla metody długości krawędzi z uwzględnieniem histogramu

Poprzednia metoda (długości krawędzi, sekcja 6.2, tablica 6.3) wybierała ostre zdjęcia, ale często źle oceniała zdjęcia prześwietlone lub niedoświetlone. Ocena histogramu pozwoliła wyeliminować w większości te błędy i obniżyć wartość oceny, gdzie histogram był niezbalansowany. Dodanie podziału zdjęcia ogólnie nie zmienia średnich wyników, ale ocena jest lepsza w przypadku sceny „Kubek", a gorsza w przypadku serii zdjęć „Salon". Warto zauważyć, że jest to pierwsza metoda, która właściwie oceniła scenę „Kubek".

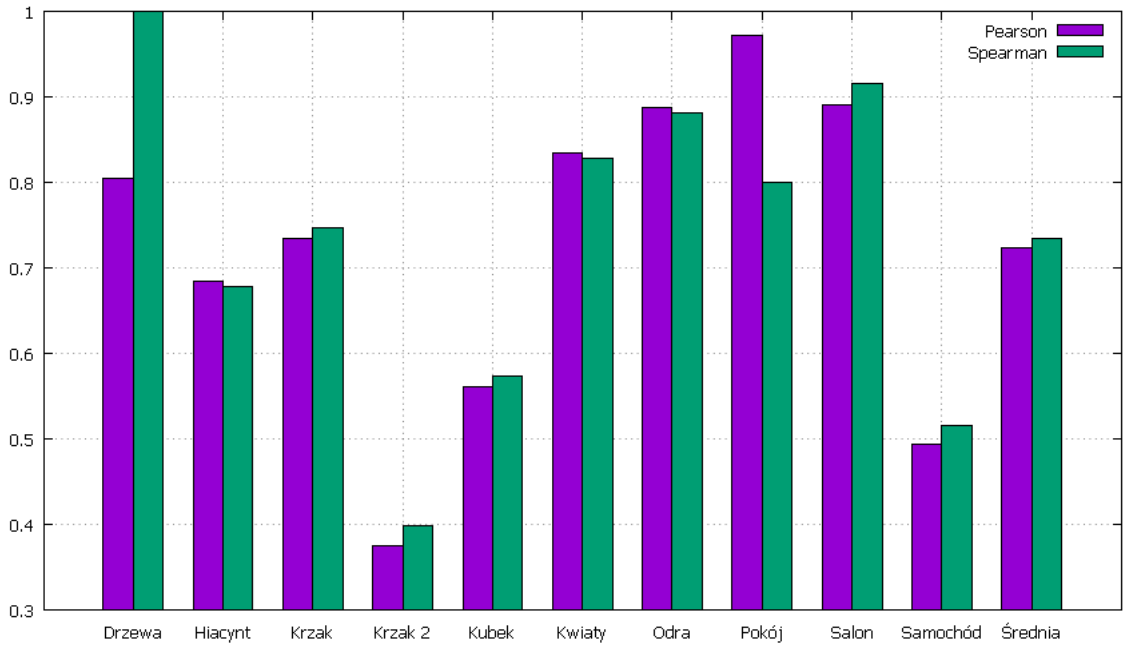


Tablica 6.6: Wyniki korelacji dla metody długości krawędzi z podziałem zdjęcia i uwzględnieniem histogramu

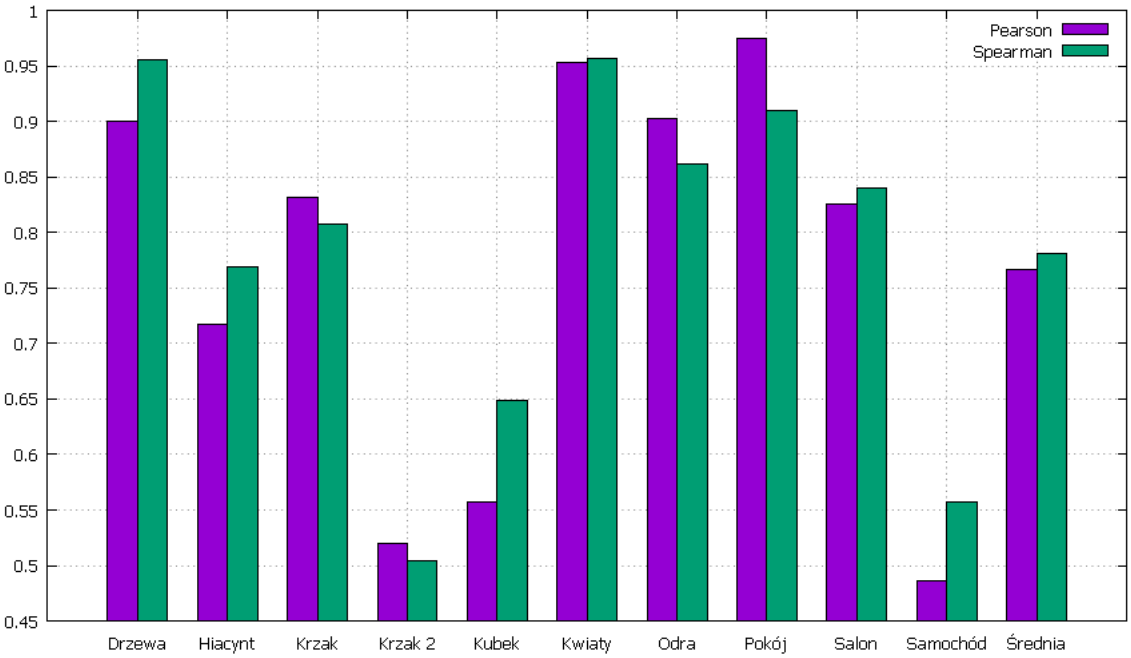
6.4 Pojedynczy próg

W algorytmie wykrywania krawędzi Cannego, który jest częścią wielu algorytmów, używany jest tak zwany podwójny próg (double thresholding). Aby piksel został zaliczony do krawędzi, musi mieć wartość powyżej górnego progu lub powyżej dolnego, ale wtedy musi mieć w sąsiedztwie piksel zaliczony do krawędzi. Powoduje to, że krawędzie są mniej „porwane” i lepiej widać kształt całego obiektu. Wyłączenie dolnego progu powoduje, że krawędzie są krótsze, ale część pikseli z lekko rozmytych krawędzi nie jest już do nich zaliczana. Przez to łatwiej odróżnić ostre zdjęcie od lekko nieostrego. Obliczyliśmy metodę długości krawędzi bez uwzględnienia histogramu, jak i z jego uwzględnieniem przy użyciu wykrywacza krawędzi Cannego z pojedynczym progiem.

W porównaniu do metody długości krawędzi z użyciem podwójnego progu (sekcja 6.2, tablica 6.3) wyniki są zauważalnie lepsze. Poprawiły się znacznie oceny scen „Kubek” oraz „Samochód”. W dalszym ciągu słabe wyniki otrzymano w przypadku sceny „Krzak 2”.



Tablica 6.7: Wyniki korelacji dla metody długości krawędzi z pojedynczym progiem



Tablica 6.8: Wyniki korelacji dla metody długości krawędzi z pojedynczym progiem i uwzględnieniem histogramu

Metoda długości krawędzi z pojedynczym progiem i uwzględnieniem histogramu lekko

poprawiła średnie wyniki. Duże zmiany są widoczne w trudnych scenach jak „Krzak" „Krzak 2" oraz „Kwiaty". W innej trudnej scenie „Samochód" ocena pozostała prawie bez zmian. W pewnych przypadkach (np. „Salon") ocena się pogorszyła. Daje to algorytm dający dość dobre wyniki.

6.5 Metoda histogramu gradientu

Innym pomysłem na rozpoznanie zdjęć ostrych jest wykorzystanie histogramu gradientu obrazu, który jest przybliżony za pomocą operatora Sobela. Spodziewamy się, że w ostrym obrazie będzie więcej pikseli o dużej wartości niż w zdjęciu nieostrym — tam krawędzie są rozmazane, szersze, a zatem pochodna nie osiąga dużych wartości. Na podstawie tych obserwacji i założeń skonstruowaliśmy następujący algorytm:

1. Oblicz operator Sobela w obrazie.
2. Oblicz histogram operatora.
3. Podziel histogram na kubełki.
4. Przemnóż ilość pikseli w każdym kubełku przez odpowiednią wagę.
5. Zsumuj wartości uzyskane dla każdego kubełka.

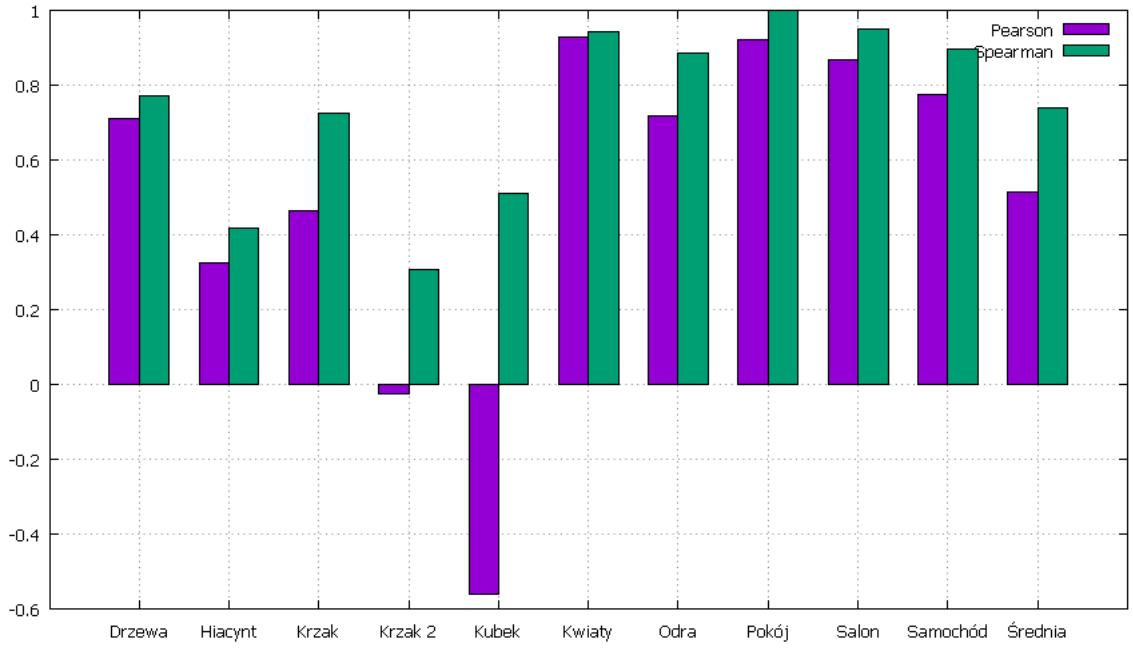
Wartości gradientu większe niż 255 zastępowaliśmy wartością 255, aby mieściły się w bajtowej zmiennej oraz ze względu na wygodę obliczeń, takich wartości było bardzo mało. W algorytmie histogram dzieliliśmy na 16 równych kubełków, w każdym były wartości z zakresu $[16i, 16i + 15] \quad i = (0, 1, \ldots, 15)$. Pierwsze dwa kubełki były pomijane ze względu na niewielkie wartości i wynikającą z tego małe zmiany w obrazie, które nie są istotne dla człowieka. Wagami dla poszczególnych kubełków h_i były kolejne wyrazy funkcji wykładniczej. Przetestowaliśmy algorytm dla podstawy 2 i 3. Wartość metryki możemy opisać wzorem:

$$B = \sum_{i=2}^{16} W(h_i) \times 2^{i-2} \tag{6.8}$$

gdzie h_i oznacza i–ty kubełek, a $W(h_i)$ ilość pikseli w i–tym kubełku.

Wyniki obu wariantów tej metody widzimy w tabelach 6.9 oraz 6.10. Są one do siebie zbliżone, ale metoda z podstawą 2 jest minimalnie lepsza w średnim przypadku. Obydwa warianty dobrze oceniały sceny, gdzie nie było wieku planów jak na przykład: „Drzewa", „Pokój", „Salon", „Kwiaty". Warto zwrócić uwagę na dobrą ocenę sceny „Samochód". Słabe wyniki otrzymano w przypadku scen trudnych jak: „Hiacynt", „Krzak 2" czy „Kubek".

Tablica 6.9: Wyniki korelacji dla metody histogramu gradientu z podstawą funkcji oceny 2



Tablica 6.10: Wyniki korelacji dla metody histogramu gradientu z podstawą funkcji oceny 3

6.6 Metoda wybierające ważne rejony

Algorytmy oparte na histogramie gradientu (sekcja 6.5) osiągają całkiem dobre wyniki w scenach o dużej głębi ostrości, gdzie wszystko jest jednakowo ostre lub nieostre, nie ma dużej ilości planów. Jednak w kilku scenach z bazy testowej (Hiacynt, Krzak 2, Kubek, Samochód) są ujęcia różnych planów, najczęściej któryś z nich jest nieostry. Również plan, na którym nam najbardziej zależy — właściwy jest często nieostry lub zajmuje niewielką część zdjęcia. W takich wypadkach ocena przez powyższe metryki jest błędna, gdyż najczęściej wybierają one zdjęcia, gdzie drugi lub trzeci plan jest ostry. Metryki te nie wykorzystują także w bezpośredni sposób założenia, o tym, że na wszystkich zdjęciach z sekwencji jest ten sam obiekt. Algorytmy te również oceniają całe zdjęcie jednakowo.

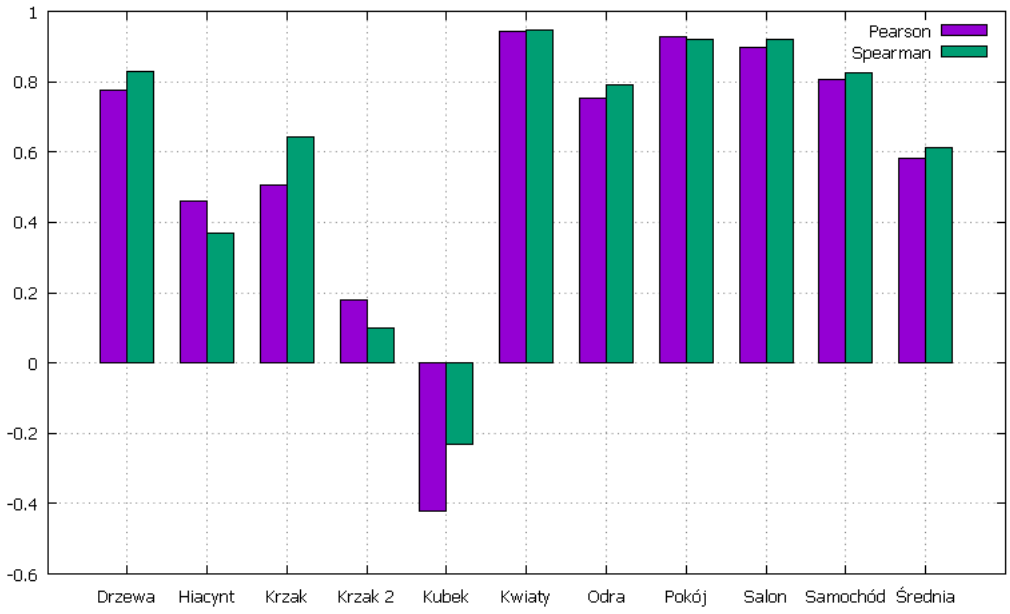
Ponieważ często tylko na części zdjęcia są ważne obiekty, to można nie oceniać całego zdjęcia, ale tylko fragmenty. Pozostaje sformułować kryterium, na podstawie którego można sądzić, że dany rejon jest ważny. Można wysunąć przypuszczenie, według którego, jeżeli dany rejon często pojawia się jako ostry, to jest to ważny obszar na zdjęciu. Do oceny ostrości ważnych fragmentów zdjęcia wykorzystaliśmy dwie metody.

6.6.1 Wybór ważnych fragmentów na podstawie metody histogramu gradientu

Jako ważne rejony zdjęcia można uznać te, które mają najlepszy histogram gradientu w myśl metryki z podrozdziału 6.6. Algorytm prezentuje się następująco:

1. Oblicz operator Sobela dla zdjęcia.
2. Podziel obraz po nałożeniu operatora na fragmenty.
3. Dla każdego fragmentu oblicz metrykę histogramu gradientu.
4. Wybierz najlepsze rejony.
5. Zsumuj wartości metryki histogramu dla najlepszych rejonów.

Obraz $S = Sobel(I)$, w naszej implementacji, jest dzielony na 64 (8×8) części, jest to wartość, która powoduje, że rzadko zdarzają się sytuacje, w których jedna część fragmentu jest ostra na jednym zdjęciu, a przeciwnie na innym. Dla każdego fragmentu S_i obliczamy metrykę histogramu gradientu z rozdziału 6.3, wynik dla każdego fragmentu oznaczamy przez B_i. Dla każdego zdjęcia tworzymy ranking, w którym fragmenty są posortowane względem ich ocen B_i, najlepszy fragment dostaje numer 1, kolejny 2, etc. Następnie dla każdego fragmentu $S_i (i = 0, 1, \ldots, 64)$ obliczamy, na jakiej średniej pozycji znajdował się w rankingach. W naszej implementacji wybieramy $t = 16$ najlepszych rejonów, zbiór najlepszych fragmentów oznaczmy przez W. Jest to 25% powierzchni całego zdjęcia, jeżeli jakiś obiekt na zdjęciu jest ustawiona ostrość, a reszta zdjęcia jest nieostra, to 25% jest wystarczającą częścią, aby wykryć ważny obiekt, ale jednocześnie nie uznać za istotne nieostrych fragmentów zdjęcia. Dla tych t najlepszych rejonów sumujemy ich wartości B_i i sumę tych liczb uznajemy za wynik. Należy pamiętać, że ta ocena istnieje w odniesieniu do tej konkretnej sekwencji zdjęć i poza nią nie mówi wiele o zdjęciu.

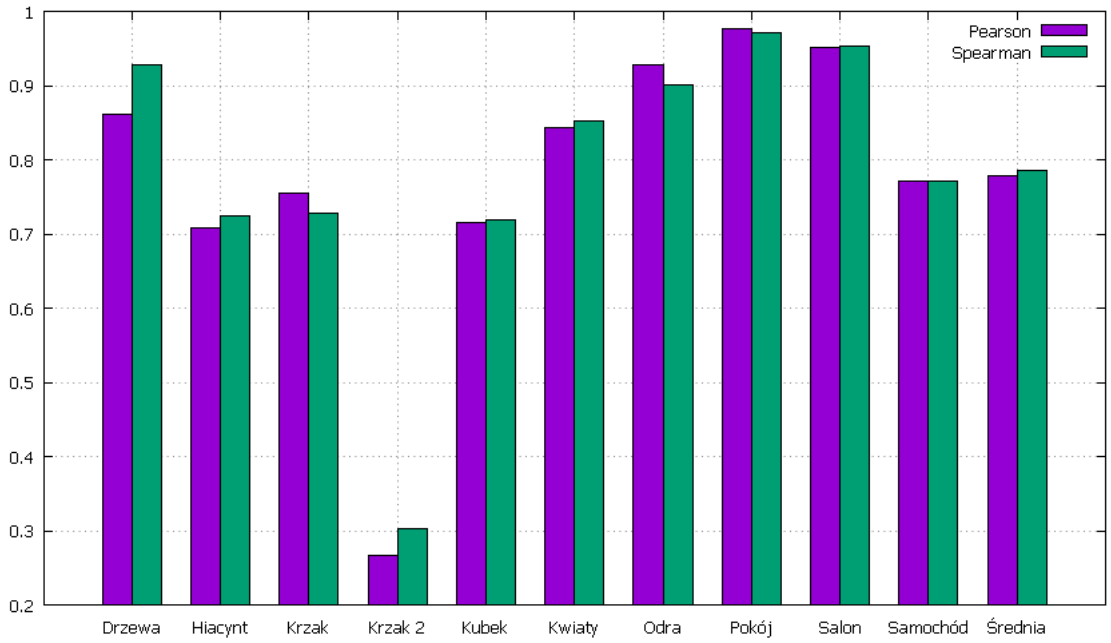


Tablica 6.11: Wyniki korelacji dla metody opisanej w rozdziale 6.6.1

Metoda nie poradziała sobie z właściwą oceną zdjęć, gdzie wyraźnie były różne plany: „Hiacynt", „Krzak 2" oraz „Kubek".j Ocena sceny „Krzak" jest nie jest bardzo zła, ale daleko jej do oceny dobrej. W związku z tym należało szukać innej metody z wyborem ważnych rejonów.

6.6.2 Wybór ważnych fragmentów na podstawie metody mieszanej

Ponieważ metoda z sekcji 6.6.1 miała wady, gdyż wybierała zdjęcia, które były nieostre, zwłaszcza w scenach gdzie było wiele planów, należało wprowadzić do niej poprawkę. Kroki 1 – 4 pozostają takie same jak w poprzednim algorytmie (podrozdział 6.6.1, metoda z podstawą 2), ale zmieniamy algorytm w punkcie 5. Oprócz metryki histogramu gradientu, dla każdej części zdjęcia, obliczamy także metrykę długości krawędzi (metoda opisana w rozdziale 6.2), wyniki te oznaczmy przez L_i. Sumujemy wartości L_i z fragmentów, które należą do zbioru W zamiast używać wartości B_i jak w podrozdziale 6.6.1.



Tablica 6.12: Wyniki korelacji dla metody opisanej w rozdziale 6.6.2

6.6.3 Wybór ważnych fragmentów na podstawie metody długości krawędzi

Kolejnym ulepszeniem poprzedniej metody było wykorzystanie metryki długości krawędzi w obu częściach algorytmu. W tej metodzie użyliśmy średniej długości krawędzi do wyznaczenia rejonów, które zostały uznane za najbardziej istotne (zbiór W). Punkt piąty algorytmu pozostaje taki sam jak w sekcji 6.6.2. Dodatkowo, aby zniwelować szum i nieistotne fragmenty zdjęcia dodaliśmy próg $th = 6$ – oznacza to, że krawędzie krótsze niż 6 pikseli są uznawane za nieistotne i pomijane w obliczaniu średniej długości krawędzi.

Poniżej zamieszczamy kilka przykładów (rysunki 6.2 – 6.5), jak obydwie metody (histogramu gradientu i długości krawędzi) wyznaczania istotnych obszarów na zdjęciu zadziałały na zdjęciach z naszej bazy. Rejony uznane za ważne są jaśniejsze.

(a) Metoda histogramu gradientu (b) Metoda długości krawędzi

Rysunek 6.2: Ważne rejony scena „Drzewa”

(a) Metoda histogramu gradientu (b) Metoda długości krawędzi

Rysunek 6.3: Ważne rejony scena „Hiacynt”

(a) Metoda histogramu gradientu (b) Metoda długości krawędzi

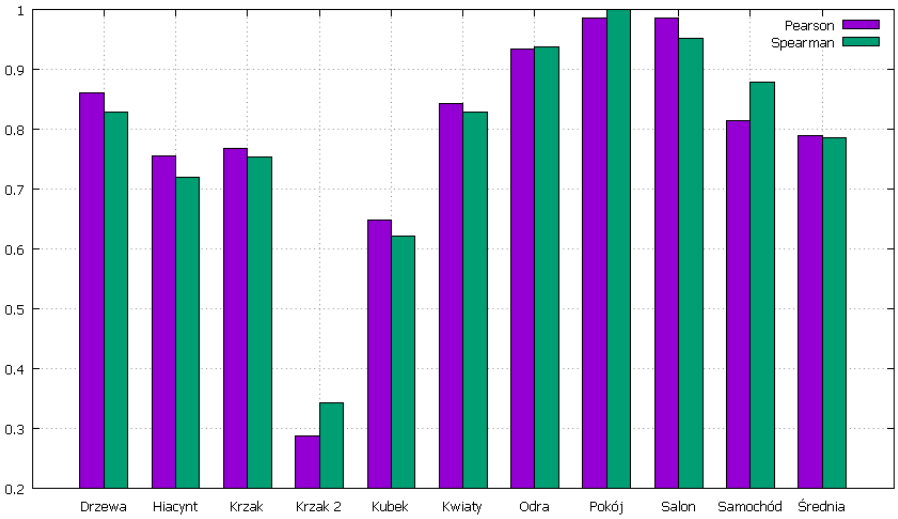
Rysunek 6.4: Ważne rejony scena „Krzak”

(a) Metoda histogramu gradientu (b) Metoda długości krawędzi

Rysunek 6.5: Ważne rejony scena „Salon”

Na zdjęciach widać, że metoda długości krawędzi lepiej radziła sobie z wyborem właściwych obszarów. Na przykład w scenie „Krzak” metoda wybrała spójny obszar, na którym faktycznie znajduje się pożądany obiekt. Również lepiej wybrała obszar, na którym znajduje się kwiat w scenie ”Hiacynt”. W scenie „Salon” mniej jest wybranych obszarów pustych, na których nie ma żadnych detali.

Jest to najlepsza metoda, którą udało się uzyskać, dająca dobre wyniki w przypadku prawie wszystkich scen, z wyjątkiem „Kubek” i „Krzak 2”, ale nawet tam wyniki są powyżej 0.5, co jest znacznym postępem w porównaniu do innych algorytmów. Ta metoda dość dobrze radzi sobie z rozpoznaniem co jest ważne na zdjęciu, pod warunkiem, że jest odpowiednia liczba zdjęć, które nie są zepsute. Z serii całkiem nieudanych zdjęć nie da się rozpoznać istotnych dla zdjęcia obszarów. Kiedy w serii przeważają zdjęcia być może nie idealne, ale takie, w których ostrość jest ustawiona na pożądany obiekt, to metoda zadziała. W tym przypadku odrzuciła zdjęcia z serii „Hiacynt” i „Kubek”, które miały ustawioną ostrość na drugi plan, podobnie w scenie „Samochód”.

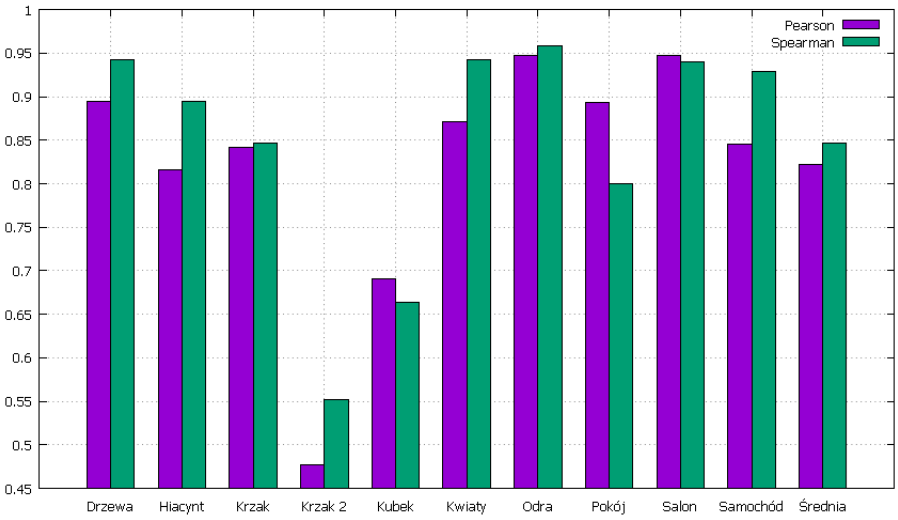


Tablica 6.13: Wyniki korelacji dla metody wybierającej ważne rejony według długości krawędzi z sekcji 6.6.3

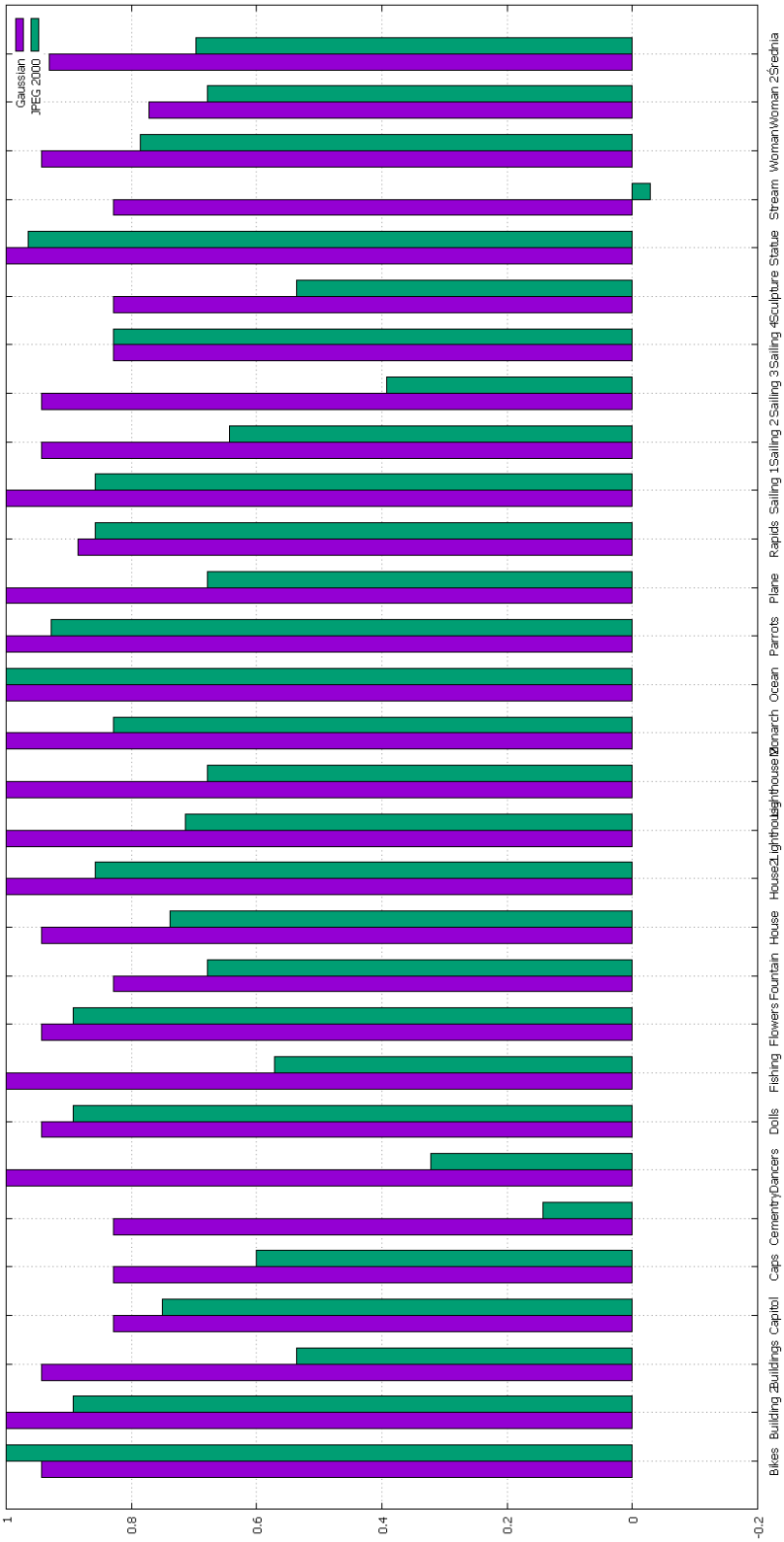
Metoda oceniająca także histogram (tablica 6.14) dała lepsze wyniki, gdyż gorzej oceniła zdjęcia wprawdzie ostre, ale zbyt jasne lub ciemne. Dzięki temu znacznie poprawiła się ocena w serii „Krzak 2” i mniej wyraźnie w scenach „Samochód”, „Kwiaty”, a także „Hiacynt”. W przypadku tych metod przetestowaliśmy je także na bazie zdjęć LIVE, jednak obliczając tylko korelację rang Spearmana ze względu na brak odpowiednich danych. Oceny były przeprowadzone dla zniekształceń obrazu poprzez rozmycie go filtrem Gaussa, jak i kompresją JPEG 2000.

Poniżej znajduje się wykres zbiorczy (tablica 6.17), na którym są pokazane średnie wyniki dla wszystkich zaprezentowanych metod. Algorytmy zaprezentowane w rozdziale szóstym są ponumerowane liczbami od 1 do 16, oto oznaczenia wszystkich metod:

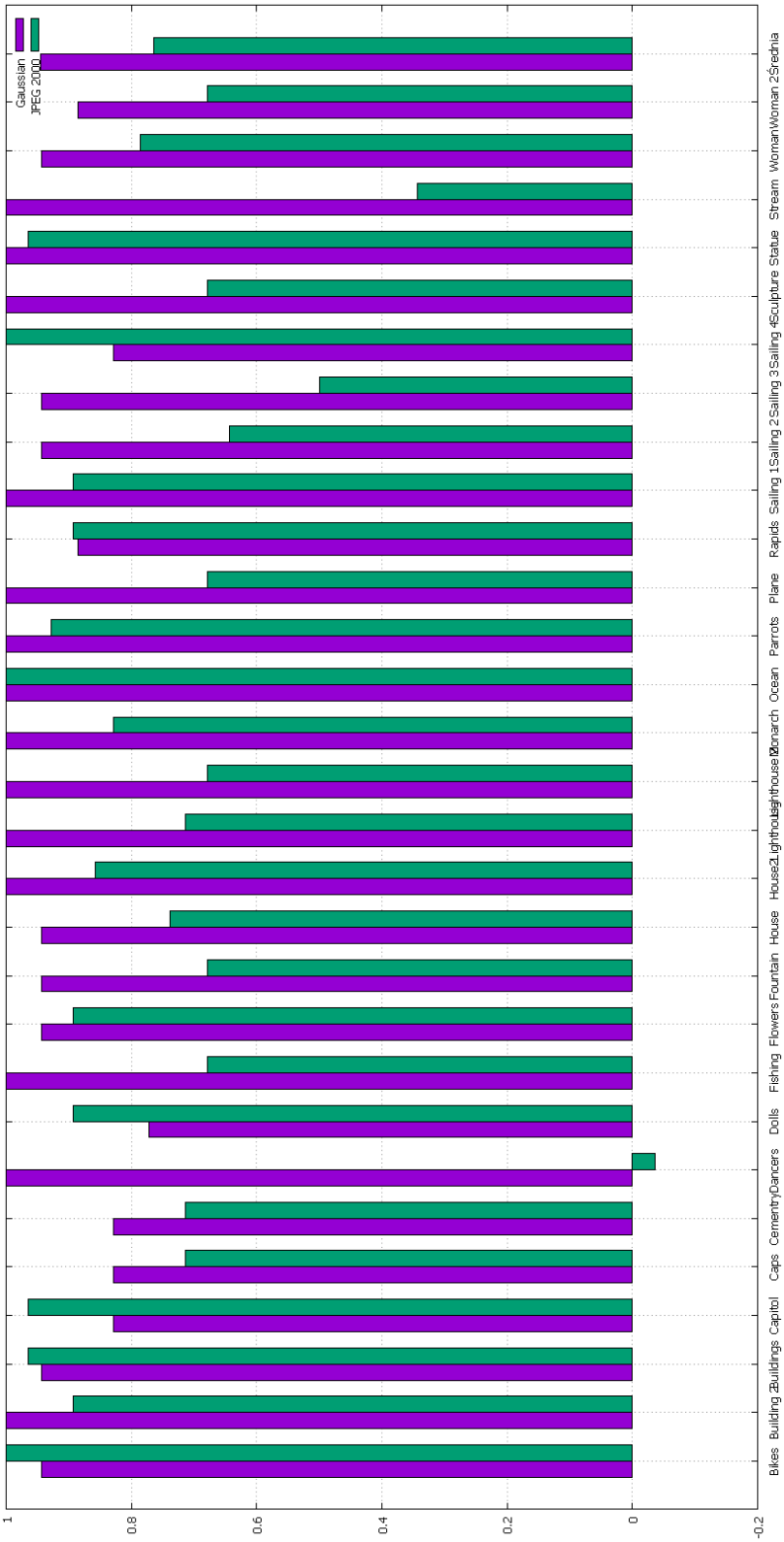
1. Metoda długości krawędzi (rozdział 6.2)
3. Metoda długości krawędzi z uwzględnieniem histogramu (rozdział 6.3)
6. Metoda długości krawędzi z pojedynczym progiem
8. Metoda długości krawędzi z pojedynczym progiem i uwzględnieniem histogramu
9. Metoda histogramu gradientu, podstawa 2 (rozdział 6.5)
10. Metoda histogramu gradientu, podstawa 3 (rozdział 6.5)
11. Metoda wybierająca ważne rejony na podstawie histogramu gradientu (rozdział 6.6.1)
12. Metoda wybierająca ważne rejony mieszana (rozdział 6.6.2)
15. Metoda wybierająca ważne rejony długości krawędzi (rozdział 6.6.3)
16. Metoda wybierająca ważne rejony długości krawędzi z uwzględnieniem histogramu (rozdział 6.6.3)



Tablica 6.14: Wyniki korelacji dla metody wybierającej ważne rejony według długości krawędzi z sekcji 6.6.3 z uwzględnieniem histogramu



Tablica 6.15: Wyniki korelacji dla metody wybierającej ważne rejony z sekcji 6.6.3 na bazie LIVE



Tablica 6.16: Wyniki korelacji dla metody wybierającej ważne rejony z sekcji 6.6.3 z uwzględnieniem histogramu na bazie LIVE

Tablica 6.17: Porównanie średnich wyników korelacji dla wszystkich metod dla własnej bazy zdjęć

Rozdział 7

Wnioski

W pracy przedstawiliśmy szereg metod oceny jakości obrazu, z głównym uwzględnieniem ostrości, jak i kilka ulepszeń i modyfikacji tych metod, jak i zupełnie inne algorytmy. Algorytmy służące do oceny jakości zdjęć często nie radzą sobie z właściwą oceną scen bardziej skomplikowanych, gdzie jest wiele planów, różnie ustawiona ostrość, różny poziom oświetlenia. Pokazaliśmy poprzez testy, że wiele popularnych i uznanych algorytmów nie ocenia zdjęć w sposób zgodny w z oceną zdjęć przez ludzi. Gdy mamy wybrać najlepsze zdjęcie z serii zdjęć tego samego obiektu, możemy użyć szerszej gamy technik, niż w zwykłej wersji tego problemu. Poprzez użycie takich metod najpierw udało nam się skonstruować algorytm, który mimo swojej prostoty działał dość dobrze w porównaniu do znacznie bardziej skomplikowanych algorytmów, które nie wykorzystywały założenia o podobieństwie zdjęć. Następnie przedstawiliśmy kilka pomysłów jak jeszcze można ten algorytm ulepszyć. Poprzez kolejne poprawki, uzyskaliśmy algorytm, który rzeczywiście wykrywał elementy istotne na zdjęciu, przez co radził sobie dobrze niemal we wszystkich przypadkach.

W pracy wykorzystaliśmy do testowania zarówno popularne zdjęcia używane w wielu badaniach, jak i własne, które bardziej odpowiadają rzeczywistemu robieniu zdjęć oraz różnorodności i lepiej testują jakość działania algorytmu.

Algorytmy przedstawione w pracy można dalej ulepszać aby działały jeszcze lepiej, można zmienić metodę wykrywania ważnych rejonów na taką, która nie musiałaby korzystać z założenia o dużym podobieństwie obiektów w scenie. Można użyć takiej metody jak korelacja krzyżowa do „kalibracji" zdjęć, co dawałoby lepsze wykrywanie istotnych obiektów. Innym pomysłem na rozwijanie algorytmów jest powiększenie puli kryteriów, na podstawie których określa się jakość obrazu.

Dodatek A

Programy

W ramach pracy powstały implementacje wszystkich wymienionych w niej programów. Wszystkie zostały napisane w języku C++ przy użyciu środowiska Microsoft Visual Studio 2010. Każda z metryk opisanych w rozdziale trzecim ma osobny program. Programu nie posiadają interfejsu graficznego, a jedynie konsolowy. Były napisane i testowane pod systemem Windows, ale działają również pod systemem Linux. Programy o nazwach „Marziliano", „JNB", „CPBD", „Tong", „Kerough", „Choi" są implementacjami odpowiednich prac opisanych w rozdziale trzecim. Program „Edge Detector" jest implementacją nowych metod.

A.1 Biblioteki

Programy używają biblioteki TurboJPEG w wersji 1.4.0 do wczytywania zdjęć w formacie JPG. Biblioteka ta musi zostać zlinkowana podczas kompilacji programu. Ponadto programy wykorzystują również bibliotekę CImg do wczytywania zdjęć w formacie BMP oraz do obliczenia transformaty Haara.

A.2 Plik konfiguracyjny

W kolejnych liniach pliku powinny znajdować się ścieżki względne lub bezwzględne do folderów ze zdjęciami. W takim folderze nie powinno być innych plików (podkatalogi są dopuszczalne). Wszystkie zdjęcia w folderze powinny należeć do jednej serii i powinny posiadać jedną rozdzielczość. Obsługiwane formaty zdjęć to JPG i BMP.

A.3 Uruchamianie

Programy „Marziliano", „JNB", „CPBD", „Tong", „Kerough", „Choi" potrzebują do prawidłowego działania dwóch argumentów. We wszystkich programach pierwszym parametrem do uruchomienia jest plik konfiguracyjny, a drugim nazwa pliku wynikowego. Program „Edge Detector" potrzebuje trzeciego argumentu, będącego liczbą z zakresu od 1 do 16, który determinuje która z metod z rozdziału 6 będzie uruchamiana. Poniżej znajduje się lista argumentów:

1. Metoda długości krawędzi (rozdział 6.2)

3. Metoda długości krawędzi z uwzględnieniem histogramu (rozdział 6.3)

6. Metoda długości krawędzi z pojednynczym progiem
8. Metoda długości krawędzi z pojednynczym progiem i uwzględnieniem histogramu
9. Metoda histogramu gradientu, podstawa 2 (rozdział 6.5)
10. Metoda histogramu gradientu, podstawa 3 (rozdział 6.5)
11. Metoda wybierająca ważne rejony na podstawie histogramu gradientu (rozdział 6.6.1)
12. Metoda wybierająca ważne rejony mieszana (rozdział 6.6.2)
15. Metoda wybierająca ważne rejony na podstawie długości krawędzi (rozdział 6.6.3)
16. Metoda wybierająca ważne rejony na podstawie długości krawędzi z uwzględnieniem histogramu (rozdział 6.6.3)

A.4 Plik wynikowy

Program w każdym folderze, który został wpisany do pliku konfiguracyjnego, zakłada podfolder wynikowy o nazwie „Wyniki", gdzie umieszcza pliki z rezultatami działania algorytmu. Program zakłada plik tekstowy, o nazwie podanej jako drugi argument przy uruchamianiu programu (rozszerzenie uzupełnia program, nie trzeba go dodawać). Dodatkowo programy, tam gdzie to możliwe, obliczają także daną metrykę w wersji z podziałem zdjęcia i wyniki również umieszczają w pliku o nazwie takiej jak drugi parametr wejścia, ale z sufiksem „_splitted". W kolejnych liniach pliku wynikowego są wypisane nazwy plików wejściowych (zdjęć) i liczby – wyniki algorytmu. Linie są posortowane w porządku od najlepszego zdjęcia do najgorszego w ocenie danego algorytmu.

Rozdział 8

Bibliografia

[1] John Canny. A computational approach to edge detection. *IEEE Transactions On Pattern Analysis and Machine Intelligence*, PAMI-8(6), 1986.

[2] Pina Marziliano, Frederic Dufaux, Stefan Winkler, Touradj Ebrahimi. Perceptual blur and ringing metrics: Application to jpeg 2000. *Signal Processing: Image Communication*, 19, 2004.

[3] Rony Ferzli, Lina Karam. A no-reference objective image sharpness metric based on the notion of just noticable blur (jnb). *IEEE Transactions On Image Processing*, 18(4), 2009.

[4] Lina Karam, Niranjan Narvekar. A no-reference image blur metric based on the cumulative probability of blur detection (cpbd). *IEEE Transactions On Image Processing*, 20(9), 2011.

[5] Hanghang Tong, Mingjing Li, Hongjiand Zhang, Changshui Zhang. Blur detection for digital images using wavelet transform. *In Proceedings of IEEE International Conference on Multimedia & Expo*, strony 17–20, 2004.

[6] F. Kerouh, A. Serir. A no reference quality metric for measuring image blur in wavelet domain. *International Journal of Digital Information and Wireless Communications (IJDIWC)*, 2011.

[7] Min Goo Choi, Jung Hoon Jung, Jae Wook Jeon. No-reference image quality assesment using blur and noise. *World Academy of Science, Engeneering and Technology*, wolumen 3, 2009.

[8] Bernd Jahne. *Digital Image Processing*. Springer, wydanie 6, 2005.

[9] Alfred Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematishe Annalen*, 1910.

[10] Piotr Porwik, Agnieszka Lisowska. The haar-wavelet transform in digital image processing: Its status and achievments. *Machine Graphics & Vision*, 13:79–98, 2004.

[11] Kim-Han Thung, Paramesran Raveendran. A survey of image quality metrics. *Technical Postgraduates (TECHPOS), 209 International Conference for*, 2009.

[12] H.R Sheikh, Z.Wang, L.Cormack, A.C.Bovik. Live image quality assesment database release 2. http://live.ece.utexas.edu/research/quality.

[13] H.R Sheikh, M.F. Sabir, A.C.Bovik. A statistical evaluation of refent full reference image quality assessment algorithms. *IEEE Transactions on Image Processing*, 15(11):3440–3451, 2006.

[14] H.R Sheikh, Z.Wang, A.C.Bovik, E.P Simoncelli. Image quality assesment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.