AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Wydział Elektrotechniki, Automatyki, Informatyki i Inżynierii Biomedycznej

Projekt dyplomowy

Zastosowanie transformaty Fouriera-Mellina do dopasowywania wzorca na obrazach

Template matching in digital images with use of the Fourier-Mellin transform

Autor: *Dorota Maria Wejdman*
Kierunek studiów: *Automatyka i Robotyka*
Opiekun pracy: *dr inż. Piotr Pawlik*

Kraków, 2020

Serdeczne podziękowania składam

Dr inż. Piotrowi Pawlikowi za wsparcie
udzielone przy przeprowadzonych badaniach
i pisaniu pracy

Spis treści

1. Wprowadzenie

1.1. Opis pracy

Celem pracy jest zbadanie zastosowania transformaty Fouriera-Mellina w celu dopasowania wzorca na obrazach. Poszukiwanie wzorca jest techniką w dziedzinie przetwarzania obrazów, która polega na lokalizacji małych fragmentów obrazu odpowiadających obrazowi referencyjnemu. Zagadnienie znajduje zastosowanie w różnych dziedzinach, takich jak kontrola jakości czy nawigacja robota.

Podstawowym zadaniem jest implementacja transformaty oraz przeprowadzenie szeregu różnych eksperymentów poprawiających jej wyniki, tak aby ostatecznie polepszyć działanie algorytmu. Rozważane zostaną przypadki, w których wzorzec jest obrócony, przeskalowany oraz przesunięty.

Zaimplementowany algorytm poszukiwania wzorca opiera się na 2 etapach, pierwszym z nich jest znalezienie obrotu i skali wzorca, a drugim znalezienie położenia, już odpowiednio obróconego i przeskalowanego wzorca, na obrazie referencyjnym. W celu odszukania parametrów obrazu wzorcowego niezbędne jest przeprowadzenie dyskretnej transformaty Fouriera obu obrazów wejściowych, następnie transformacji obrazu amplitudy do układu log-polar oraz ponownym przekształceniu Fouriera. Wykonanie korelacji fazowej obu obrazów, w celu znalezienia parametrów (kąta i skali), umożliwia doprowadzenie wzorca do tej samej orientacji i skali, co obraz referencyjny. Dzięki temu w kolejnym kroku możliwe jest odszukanie za pomocą korelacji fazowej, lokalizacji poszukiwanego wzorca.

W praktyce badania mogą zostać wykorzystane, m. in. aby wyszukać obiekt na zdjęciu lotniczym, a w rozwinięciu implementacji nawet do rozpoznawania wzorca na nagraniu. Wszelkie eksperymenty i testy zostaną więc przeprowadzone na tego typu obrazach.

Możliwe jest również zastosowanie algorytmu w celu rozpoznawania etykiet towarów na taśmie produkcyjnej, rozpoznawania elementów elektronicznych na złożonych schematach czy znajdywaniu lokalizacji na mapach poziomicowych [1].

W następnych rozdziałach zostaną przedstawione: teoretyczny opis zagadnień związanych z prezentowanym algorytmem, dokładny opis algorytmu, zastosowane biblioteki i funkcje oraz przeprowadzone eksperymenty wraz z testami ich wpływu na działanie algorytmu.

1.2. Prace innych autorów dotyczące Transformaty Fouriera Mellina

Praca pt. “Novel Template Matching Method With Sub-Pixel Accuracy Based on Correlation and Fourier-Mellin Transform” [1] dotyczy, tak jak niniejsza praca, zastosowania transformaty Fouriera Mellina w poszukiwaniu wzorca na obrazie. Jednak przedstawiono w niej przykłady wykorzystania algorytmu szukania obrazu wzorcowego na mapie poziomic i układach elektronicznych obróconych w zakresie [-20°,20°]. Autorzy skupili się także na akceleracji algorytmu. Wyniki pracy wykazały, że metoda jest odporna na biały szum oraz badaną rotację, i daje wysokie wyniki poprawności i precyzji działania.

Autorzy G. S. Cox and G. de Jager w publikacji „Invariance in Template Matching“ [9] przedstawili inwariantność skali, obrotu, przysłonięcia oraz średniego poziomu intensywności obrazu wraz z przykładami. Praca jest teoretyczna, tłumaczy w jaki sposób transformacja Fouriera Mellina może służyć do rozpoznawania wzorców pomimo obrotu i skalowania, co w praktyce jest zastosowane w obecnej pracy.

Artykuł „An Application of Fourier-Mellin Transform in Image Registration” traktuje o ulepszeniu algorytmu wykorzystującego transformację Fouriera Mellina analogicznie do badań przeprowadzonych w niniejszej pracy, jednak poprzez inne eksperymenty, m.in. usuniecie konwersji z układu kartezjańskiego na log polar [10]. Zagadnienie transformacji jest rozpatrywane w odróżnieniu od obecnej pracy, w celu dopasowania różnych obrazów przedstawiających tą samą scenerię. (ang. Image registration). Praca dowodzi także poprawności działania algorytmu niezależnie od białego szumu.

Autorzy rozprawy “Experiments on pattern recognition using invariant Fourier–Mellin descriptors “ badają, nieanalizowany w obecnej pracy, wpływ szumów na transformację Fouriera Mellina [11].

Praca "Template Matching Based Object Recognition with Unknown Geometric Parameters" dotyczy poszukiwanie wzorca z nieznanymi obrotem i rozmiarem, zamiast transformacji Fouriera Mellina, stosowane jest przybliżenie funkcją delta oraz aproksymacja wzorców (coarse-to-fine) z użyciem równania dyfuzyjnego [12].

“Fingerprint Matching Using Phase-Only Correlation and Fourier-Mellin Transforms” to artykuł przedstawiający zastosowanie transformacji Fouriera Mellina i korelacji fazowej – BLPOC (Band-limited Phase-Only Correlation) w celu rozpoznawania odcisków palców, niezależnie od skali i obrotu. Zagadnienie różni się tym od rozpoznawania wzorca w niniejszej pracy, iż autorzy rozpatrują wzorzec jako obraz tego samego rozmiaru co obraz referencyjny, a nie jako jego fragment. Dodatkowo do algorytmu wprowadzono sprawdzenie 3 różnych kątów, zamiast jednego, co znacznie poprawiło rezultaty [13].

2. Wybrane zagadnienia z przetwarzania obrazu

2.1. Transformata Fouriera

Transformacja Fouriera jest transformacją z dziedziny czasu w dziedzinę częstotliwości. Przekształcenie można wyrazić następującym wzorem [2]:

$$\hat{g}(\omega) = \int_{-\infty}^{\infty} g(t) e^{-i\omega t}\, dt \tag{1}$$

Gdzie:

$\omega = 2\pi f$ – pulsacja (f-częstotliwość)

g(t) – funkcja w dziedzinie czasu

g(f) - funkcja w dziedzinie częstotliwości

Odwrotną transformatę Fouriera, którą można wyznaczyć za pomocą wzoru:

$$g(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \hat{g}(\omega) e^{-i\omega t}\, df \tag{2}$$

W praktyce często przetwarzane są dane dyskretne, wtedy stosowana jest dyskretna transformata Fouriera (DFT). Ze względu na dużą złożoność obliczeniową (N^2) powstał szybszy algorytm realizujący to zadanie, nazwany FFT (Fast Fourier Transform) o złożoności $(N * log_2 N)$, który będzie wykorzystywany w obecnej pracy.

W celu wykonania dwuwymiarowej transformacji Fouriera wyznacza się kolejno transformacje wierszy oraz kolumn. Dwuwymiarową, dyskretną transformację Fouriera, wykorzystywaną w analizie obrazów cyfrowych, można przeprowadzić za pomocą wzoru [3]:

$$F(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x,y) e^{-\left(j\frac{2\pi ux}{N} + j\frac{2\pi vy}{M}\right)} \tag{3}$$

dla $u = 0,1,\ldots,N-1,\ v = 0,1,\ldots,M-1$

Gdzie:

F – obraz wyjściowy (F-obraz)

f – obraz wejściowy

(x, y) – współrzędne obrazu wejściowego

(u, v) - współrzędne obrazu wyjściowego

(M, N) – rozmiar obrazu wejściowego i wyjściowego

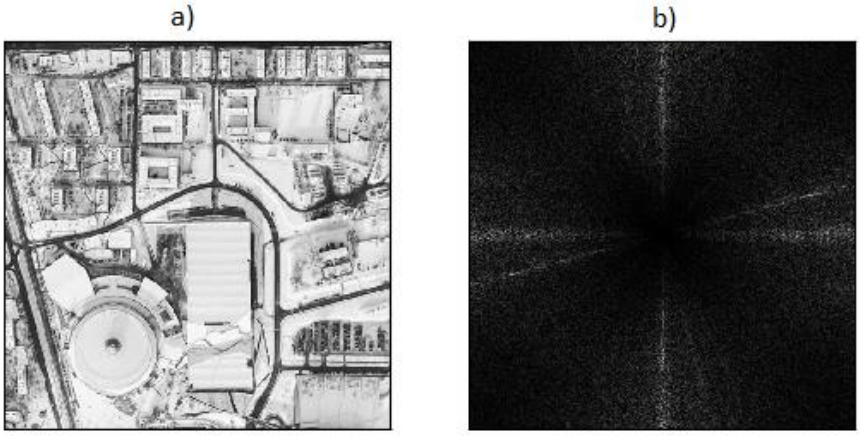
Analogicznie odwrotną dyskretną dwuwymiarową transformatę można zapisać jako:

$$f(x,y) = \frac{1}{MN}\sum_{x=0}^{N-1}\sum_{y=0}^{M-1} F(u,v)e^{\left(j\frac{2\pi ux}{N}+j\frac{2\pi vy}{M}\right)} \tag{4}$$

dla $x = 0,1,\ldots,N-1,\ v\ =\ 0\ ,1,\ldots,M-1$

Wynik przekształcenia Fouriera reprezentowany jest przez liczby zespolone, w związku z tym nie można go w pełni przedstawić na obrazie. Możliwe jest pokazanie jedynie jego składowych: części rzeczywistej i urojonej lub amplitudy i fazy. W celu wizualizacji rezultatu w skali szarości, niezbędne jest przeskalowanie wyniku do zakresu 0-255, tak aby każdemu pikselowi odpowiadał poziom jasności.

Przykład realizacji transformaty Fouriera obrazu przedstawiono na rysunku 2.1.



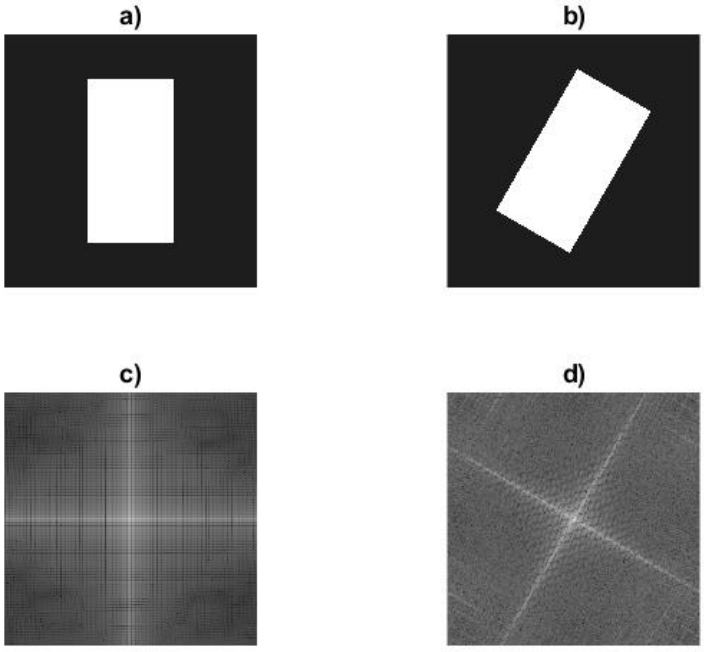
Rys. 2.1. Przykład Transformaty Fouriera
a) obraz wejściowy b) logarytm modułu transformaty Fouriera obrazu wejściowego

W analizie obrazu wysokie częstotliwości wskazują na występowanie krawędzi, a więc wyraźnych zmian koloru lub poziomu jasności, niskie natomiast odpowiadają płynnym przejściom koloru i jednolitym fragmentom.

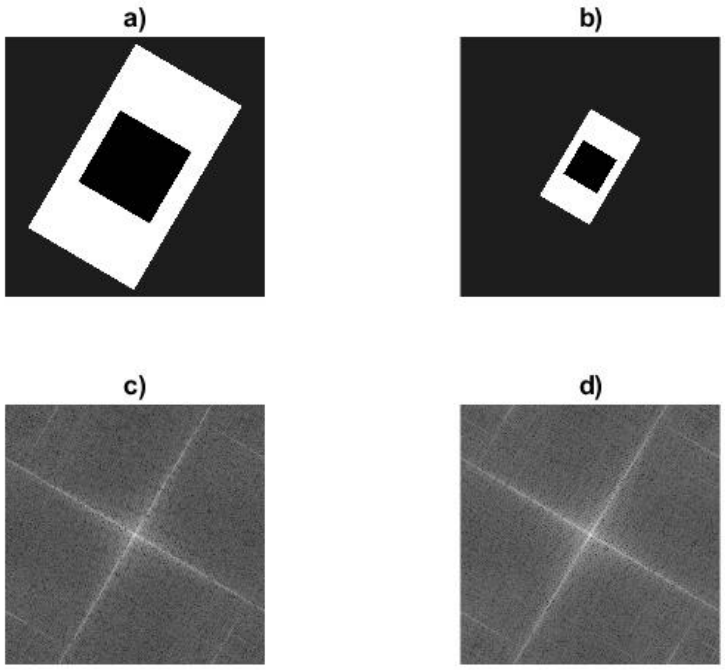
Przekształcenie Fouriera wyróżnia się szczególnymi właściwościami, wykorzystywanymi w obecnej pracy. Są to inwariantność

- skali
- obrotu.

Obrót zostaje odzwierciedlony takim samym obrotem obrazu transformaty, natomiast pomniejszenie spowoduje, że obraz będzie odpowiednio powiększony. Analogicznie reprezentowane jest powiększenie. Właściwości te zostały przedstawione na kolejnych rysunkach 2.2 i 2.3.



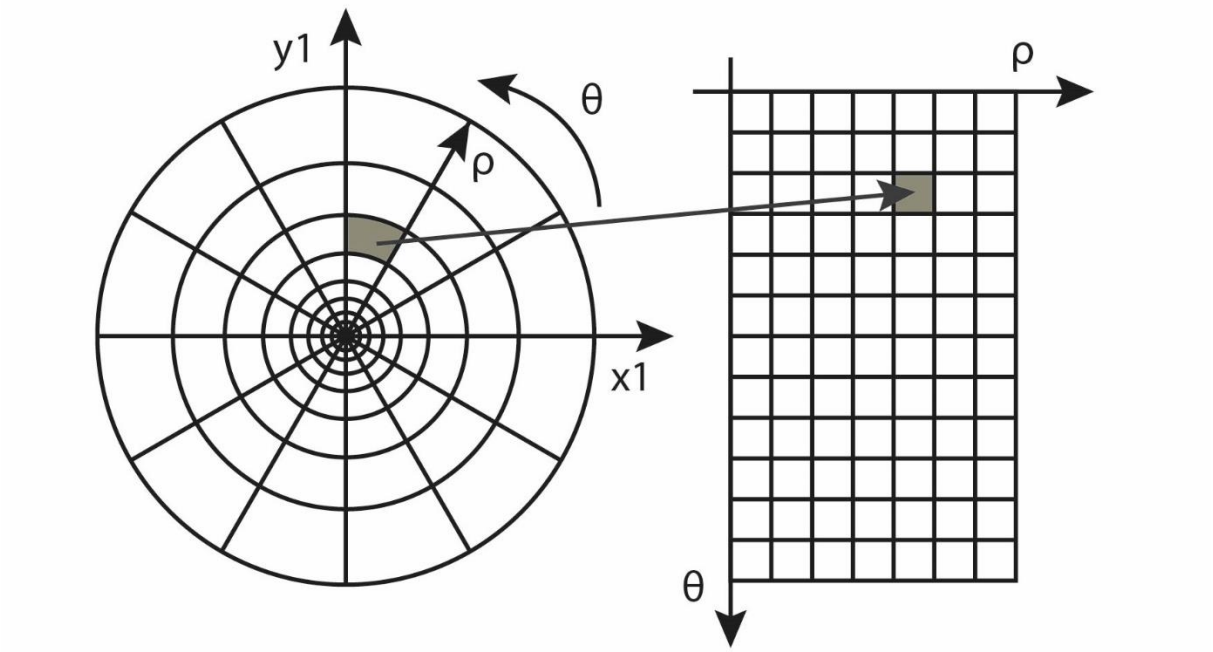
Rys. 2.2. Przykład Transformaty Fouriera obróconego obrazu
a) obraz wejściowy, b) obraz wejściowy obrócony o 30 stopni, c) logarytm modułu transformaty Fouriera obrazu a, d) logarytm modułu transformaty Fouriera obrazu b



Rys. 2.3. Przykład Transformaty Fouriera przeskalowanego obrazu
a) obraz wejściowy, b) obraz pomniejszony, c) logarytm modułu transformaty Fouriera obrazu a, d) logarytm modułu transformaty Fouriera obrazu b

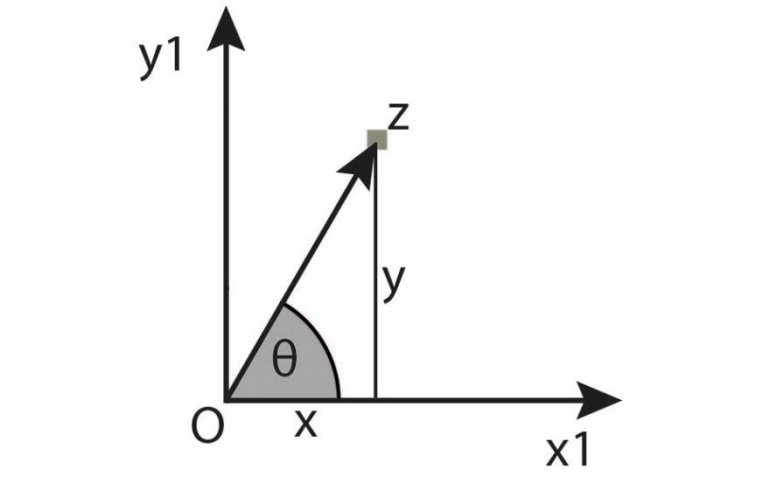
2.2. Współrzędne logarytmiczno-biegunowe

Współrzędne log-polar (logarytmiczno-biegunowe) są dwuwymiarowym systemem współrzędnych. Pierwsza współrzędna oznacza logarytm z odległości punktu od środka układu, a druga – kąt pomiędzy osią x, a linią przechodzącą przez dany punkt i punkt środkowy. Układ przypomina rozmieszczenie fotoreceptorów w ludzkim oku. Zmiana reprezentacji obrazu ze współrzędnych kartezjańskich na współrzędne log-polar, daje możliwości w odczytywaniu obrotu, oraz skali jako przesunięć wzdłuż odpowiednich osi, niezbędne do wyszukiwania wzorca na obrazie. Ponadto przekształcony obraz posiada dużo szczegółów blisko swojego centrum, a w dalszym otoczeniu traci dokładność, ponieważ pola są znacznie większe. Przykładową ‘siatkę’, prezentującą rozmieszczenie pikseli w układzie log polar oraz zależność położenia punktu w tym układzie i we współrzędnych kartezjańskich przedstawiono na rysunku 2.4.



Rys. 2.4. Transformacja log-polar

W celu wyznaczenia współrzędnych log-polar na podstawie współrzędnych kartezjańskich zaprezentowano rysunku 2.5, na podstawie którego wskazano odpowiednie wzory.



Rys. 2.5. Zależność pary współrzędnych logarytmiczno-biegunowych (ρ, θ) od (x,y)

Z twierdzenia Pitagorasa odległość punktu Z od środka układu można określić jako:

$$|\mathrm{OZ}| = \sqrt{x^2 + y^2} \tag{5}$$

aby uzyskać logarytmiczną wartość współrzędnej ρ należy wziąć logarytm z odlgłości $|\mathrm{OZ}|$:

$$\rho = \log\sqrt{x^2 + y^2} \tag{6}$$

Współrzędną kątową można wyznaczyć z zależności boków trójkąta:

$$\frac{y}{x} = \mathrm{tg}\,\theta \tag{7}$$

$$\theta = \arctan\frac{y}{x} \text{ dla } x > 0 \tag{8}$$

Wzory na odwrotną transformację współrzędnych:

$$\cos\theta = \frac{x}{\sqrt{x^2 + y^2}} \tag{9}$$

$$x = \sqrt{x^2 + y^2}\cos\theta \tag{10}$$

$$\sqrt{x^2 + y^2} = e^{\log\sqrt{x^2+y^2}} = e^{\rho} \tag{11}$$

Podstawiając wzór (11) do (10) otrzymujemy:

$$x = e^{\rho}\cos\theta \tag{12}$$

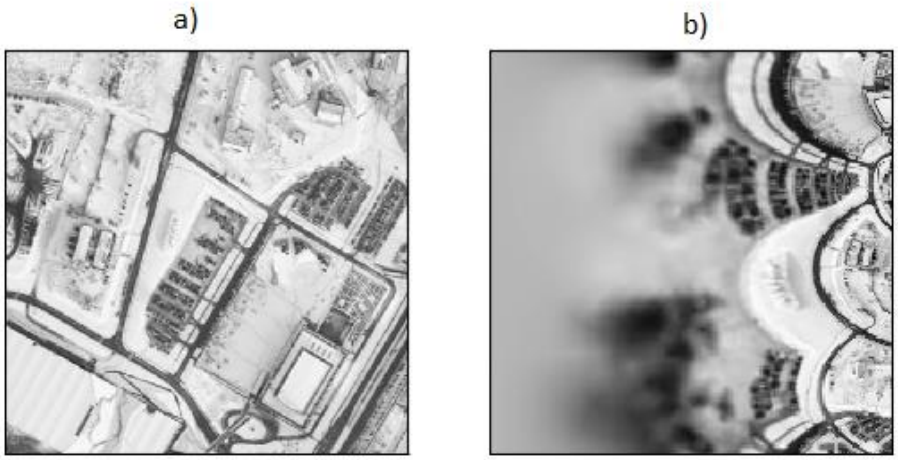
Analogicznie:

$$y = \sqrt{x^2 + y^2}\sin\theta \tag{13}$$

$$y = e^{\rho}\sin\theta \tag{14}$$

Na to jak obraz będzie reprezentowany w układzie log polar ma wpływ maksymalny promień, a także liczba i wielkość pól. Jeżeli maksymalny promień będzie równy połowie szerokości obrazu, to nastąpi utrata informacji znajdujących się w rogach obrazu. Ustawienie tej wartości jako połowę szerokości przekątnej obrazu spowoduje z kolei wygenerowanie obszarów niezawierających informacji o obrazie.

Przykład obrazu przekształconego do układu log polar, gdzie parametr MaxRadius, czyli maksymalny promień ustawiono na połowę szerokości obrazu, przedstawiono na rysunku 2.6.



Rys. 2.6. Przykład Transformaty Log-polar
a) obraz wejściowy b) transformata log-polar

2.3. Transformacja Mellina (Fouriera-Mellina)

Dwuwymiarowa transformacja Mellina dana jest wzorem [5]:

$$FM(u,v) = \int_0^\infty \int_0^\infty K_\theta(x,y,u,v)\, dxdy \tag{15}$$

$$K_\theta(x,y,u,v) = x^{\frac{2\pi iu}{sin\theta}-1} y^{\frac{2\pi iv}{sin\theta}-1} e^{\frac{\pi i}{tg\theta}[u^2+v^2+log\, x^2+logy^2]}, 0 < \theta \leq \frac{\pi}{2} \tag{16}$$

Operacja transformacji Fouriera-Mellina jest ściśle związana z teorią szeregu Dirichleta, znajduje zastosowanie w wielu dziedzinach matematycznych. Możliwe jest przekształcenie transformaty Fouriera oraz transformaty Laplace'a do postaci transformaty Mellina i odwrotnie.

W przetwarzaniu obrazów, transformację praktycznie można wykonać jako dwuwymiarową transformację Fouriera, a następnie konwersję do układu współrzędnych logarytmiczno-biegunowych.

2.4. Korelacja fazowa

Korelacja w dziedzinie faz umożliwia określenie przesunięcia jednego obrazu względem drugiego. Opiera się na reprezentacji obrazu w dziedzinie częstotliwości. Można ją przeprowadzić poprzez wykonanie kolejnych operacji [6]:

- Zastosowania okna Hanninga na obu obrazach (w celu osłabienia efektu krawędzi)
- Obliczenie dyskretnej transformaty Fouriera obrazów
- Obliczenie (Cross-Power Spectrum) widma mocy korzystając ze wzoru:

$$R = \frac{D * \overline{W}}{|D * \overline{W}|} \tag{17}$$

Gdzie

* - operacja mnożenia wykonana osobno na każdym elemencie macierzy

D, W – obrazy wejściowe

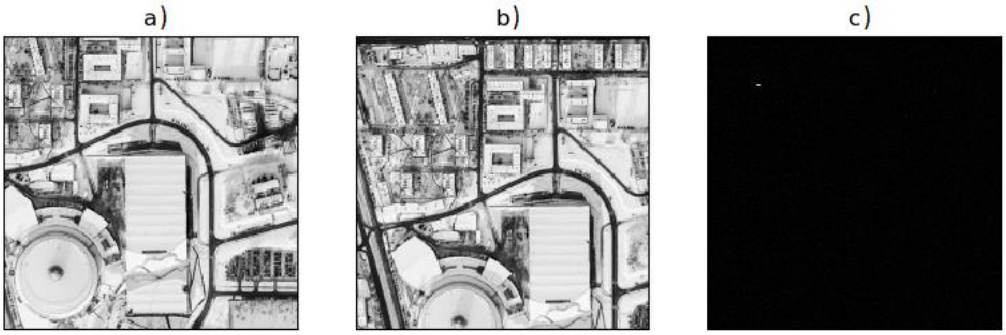
R – widmo mocy

- Zastosowanie odwrotnej transformaty Fouriera
- Wyznaczenie współrzędnych maximum

$$(x, y) = \arg\max(r) \tag{18}$$

r – wynik po zastosowaniu odwrotnej transformacji Fouriera

Rysunek 2.7 przedstawia przykładowe zastosowanie korelacji fazowej w celu znalezienia przesunięcia obrazów względem siebie. Biały punkt na obrazie c wskazuje współrzędne przesunięcia, względem osi x i y.



Rys. 2.7. Przykład korelacji fazowej obrazów.
a) obraz wejściowy b) przesunięty obraz wejściowy c) korelacja fazowa obrazów a i b

2.5. Okno Hanninga (Hanna)

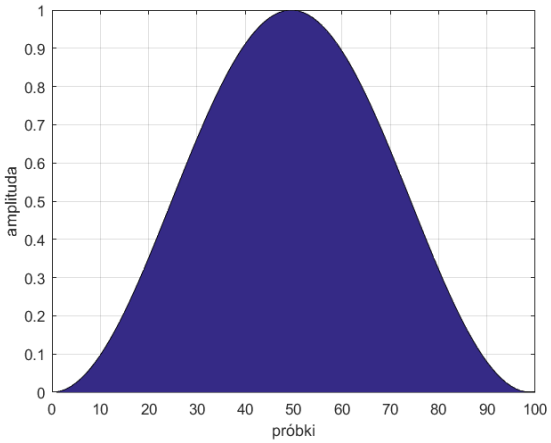
Okno nazwane od meteorologa Julius'a von Hann, stosowane w celu wygładzenia obrazu. Jego zastosowanie polega na pomnożeniu funkcji sygnału przez funkcję okna, zdefiniowaną jako [7]:

$$w(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{M-1}\right) \tag{19}$$

dla $n \in < 0, M-1 >$

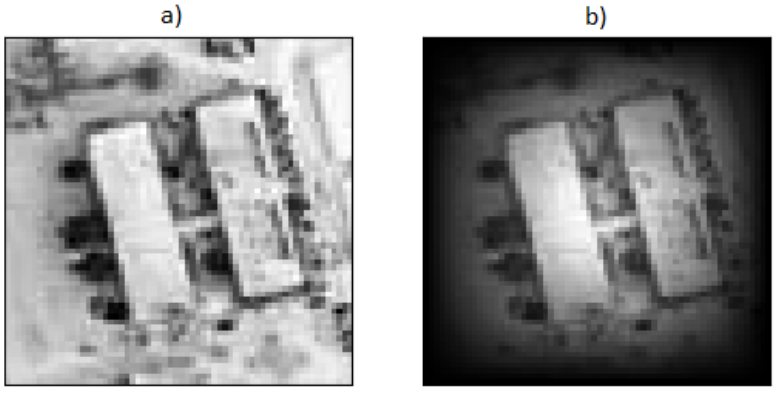
M – liczba punktów wyjściowego sygnału

n – numer próbki sygnału



Rys. 2.8. Funkcja okna Hanninga

W przetwarzaniu obrazu niezbędne jest użycie dwuwymiarowej funkcji okna. Jedną z możliwości jest wykonania operacji jest iloczyn diadyczny, czyli pomnożenie dwóch wektorów (kolumnowego i wierszowgo) tego samego rozmiaru. Wspomniane wektory są iloczynami danych wierszy (kolumn) i funkcji okna. Przykład zastosowania okna przedstawiono na rys 2.9.



Rys. 2.9. Przykład zastosowania okna Hanninga na obrazie a) obraz wejściowy b) obraz po zastosowaniu okna

2.6. Interpolacja w grafice komputerowej

Interpolacja to proces polegający na ustaleniu nowej wartości piksela na podstawie jego otoczenia, tak aby wynik operacji jak najlepiej odwzorowywał obraz wejściowy. Istnieje wiele metod interpolacji, m.in.: interpolacja liniowa, dwuliniowa, dwusześcienna, na trójkącie, algorytmem Lanczosa, Mitchela, Hermite'a, najbliższego sąsiada czy funkcjami sklejanymi. Operacja może wykorzystywać różne liczby punktów sąsiadujących, co wpływa na jej dokładność [8] (Źródło zostało wykorzystane w całym rozdziale 2.6).

Interpolacja może służyć zarówno do powielania pikseli jak i ich redukowania, a więc powiększania i pomniejszania obrazu. W obecnej pracy zastosowano zmianę rozdzielczości obrazu za pomocą różnych metod interpolacji oraz zbadano ich wpływ na działanie algorytmu. Rozważono interpolację:

- dwuliniową,

- dwusześcienną (uwzględniającej sąsiadujące punkty w kwadratach 4×4),

- algorytmem Lanczosa (uwzględniającej sąsiadujące punkty w kwadratach 8x8).

2.6.1. Interpolacja dwuliniowa

Interpolacja dwuliniowa na obrazie 2D wykorzystuje sąsiedztwo 4 punktów, bezpośrednio stykających się ze środkowym pikselem. Funkcja jądra interpolacji jest linią prostą. Operacja nazwana jest dwuliniową dlatego że jest złożeniem interpolacji liniowych w dwóch kierunkach. Na zamieszczonym poniżej przykładzie można zauważyć, że obraz powiększony tą metodą jest widocznie rozmyty.

Rys. 2.10. Przykład interpolacji dwuliniowej
a) obraz przed operacją b) obraz powiększony dwukrotnie metodą dwuliniową

2.6.2. Interpolacja dwusześcienna

Interpolacja dwusześcienna jest metodą, która równoważy efekt rozmycia. Funkcją jądra interpolacji jest wielomian 3 stopnia, a wykorzystywane sąsiedztwo jest czteropunktowe. Jest to jedna z najczęściej używanych metod w różnych dziedzinach. Istnieją także metody sześcienne z sąsiedztwem dwu, sześcio- a nawet ośmio-punktowym.

Rys. 2.11. Przykład interpolacji dwusześciennej
a) obraz przed operacją b) obraz powiększony dwukrotnie metodą dwusześcienną

2.6.3. Interpolacja algorytmem Lanczosa

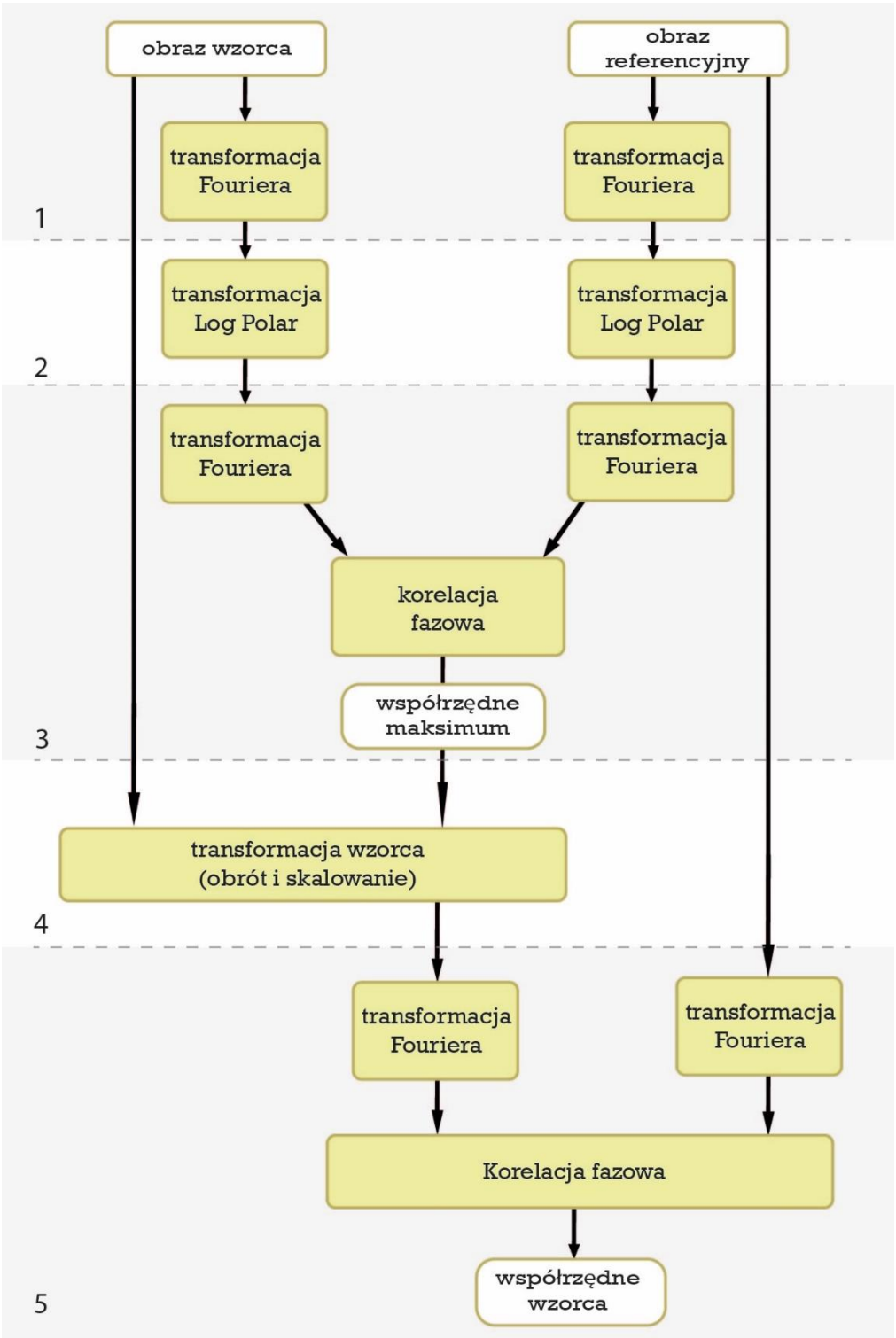
Interpolacja algorytmem Lanczosa jest modyfikacją metody opartej na funkcji sinc. Modyfikacja polega na zastosowaniu okna Lanczos w celu ograniczenia sąsiedztwa, które w przypadku rozważanej funkcji jest nieskończone. Liczba punktów sąsiedztwa w tym algorytmie może być dowolnie dobrana. Metoda zmniejsza zjawisko postrzępionych krawędzi, jednocześnie powodując niewielkie rozmycie, w stosunku do poprzednio wymienionych metod, co widać na rys 2.12.

Rys. 2.12. Przykład interpolacji metodą Lanczosza
a) obraz przed operacją b) obraz powiększony dwukrotnie metodą Lanczosa

3. Praktyczna realizacja transformaty Fouriera Mellina

3.1. Opis algorytmu

Na rysunku 3.1 przedstawiono blokowy schemat algorytmu zastosowanego w obecnej pracy wraz z zaznaczeniem poszczególnych etapów, które będą kolejno opisane.



Rys. 3.1. Schemat algorytmu wykorzystującego transformatę Fouriera Mellina do poszukiwania wzorca na obrazie

W celu przygotowania danych wejściowych do kolejnych operacji, należy przekształcić obrazy tak, aby miały ten sam rozmiar. Jest to niezbędne m. in. po to, aby wykonać ich korelację fazową. W związku z tym wzorzec, którego wielkość w analizowanych przypadkach stanowi ok. 0.5% powierzchni całego obrazu, umieszczono na środku czarnego tła, o rozmiarze obrazu referencyjnego.

W pierwszym etapie, po wczytaniu odpowiednich obrazów wejściowych, o takich samych rozmiarach, wykonano transformację Fouriera każdego z nich. Ta operacja umożliwia rozmieszczenie informacji o obrazie względem jego środka. Jest to istotne w późniejszej analizie transformaty log-polar, w której również występuje rozkład informacji względem punktu centralnego. Algorytm wzbogacono o filtrację górnoprzepustową, której zadaniem było stłumienie niskoczęstotliwościowych elementów obrazu oraz wydobycie wysokich częstotliwości, zawierających informację o szczegółach.

Kolejne kroki mają za zadanie wyznaczenie skali oraz obrotu transformaty Fouriera wzorca względem transformaty obrazu referencyjnego. W tym celu w etapie drugim, na obu F-obrazach (obrazach w dziedzinie częstotliwości), wykonuje się operację transformacji logarytmiczno-biegunowej. Dla obrazów w dziedzinie częstotliwości zmiana obrotu obrazu wejściowego powoduje taki sam obrót F-obrazu, a powiększenie odzwierciedlone jest odpowiednim pomniejszeniem (i odwrotnie). Tak więc znajdując szukane paramtery F-obrazów, można zidentyfikować również te parametry dla obrazów wejściowych.

We współrzędnych log-polar, obrót obrazu objawia się przesunięciem względem osi pionowej, a zmiana skali - przesunięciem względem osi poziomej. Dzięki temu, w etapie trzecim, wyznaczenie korelacji fazowej obrazów (będących wynikiem poprzednich operacji) pozwala na znalezienie przesunięcia jednego względem drugiego i tym samym szukanych parametrów obrotu i skali. W tym celu należy przeprowadzić kolejną transformację Fouriera, a następnie korelację fazową uzyskanych F-obrazów.

Współrzędne maksimum tej korelacji wskazują skalę oraz rotację F-obrazów w układzie log-polar względem siebie, a odpowiednie przeliczenie wartości umożliwia określenie tych parametrów dla obrazów w układzie kartezjańskim. Wykonując te operacje należy wziąć pod uwagę, że przesunięcie w przeciwne strony w kierunku pionowym odpowiada za pomniejszenie lub powiększenie, a w kierunku poziomym – rotację w różne strony. W czwartym etapie, wejściowy obraz wzorca jest transformowany, zgodnie z parametrami, a po ukończeniu tych operacji ma taką samą orientację i skalę jak obraz referencyjny.

W ostatnim kroku wykonywane są: transformacja Fouriera i korelacja fazowa już przetransformowanego wzorca oraz obrazu referencyjnego. Przy odpowiednim wyznaczeniu skali i obrotu, maksimum korelacji określa współrzędne wzorca na obrazie referencyjnym. Na końcu obraz wzorca jest transformowany zgodnie z wyznaczonymi współrzędnymi do położenia, w którym powinien się znajdować.

3.2. Narzędzia pracy

Algorytm został zaimplementowany w języku Python, z wykorzystaniem funkcji wykonujących operacje na obrazie, z bibliotek NumPy oraz OpenCv.

3.2.1. Funkcje biblioteki OpenCV.

Dwie funkcje biblioteki OpenCV: LogPolar i WarpPolar przekształcają obraz do przestrzeni logarytmiczno-biegunowej. W niniejszej pracy zastosowano WarpPolar [14].

Dokumentacja funkcji:

```
void cv :: logPolar (InputArray src, OutputArray dst, Point2f center, double M, int flags)

Python:
dst = cv.logPolar (src, center, M, flags[, dst])


void cv :: warpPolar (InputArray src, OutputArray dst, Size dsize, Point2f center,
double maxRadius, int flags )

Python:
dst = cv.warpPolar (src, dsize, center, maxRadius, flags[, dst]) [1]
```

Najważniejszym, modyfikowalnym parametrem funkcji jest maxRadius, czyli maksymalny promień w układzie log-polar.

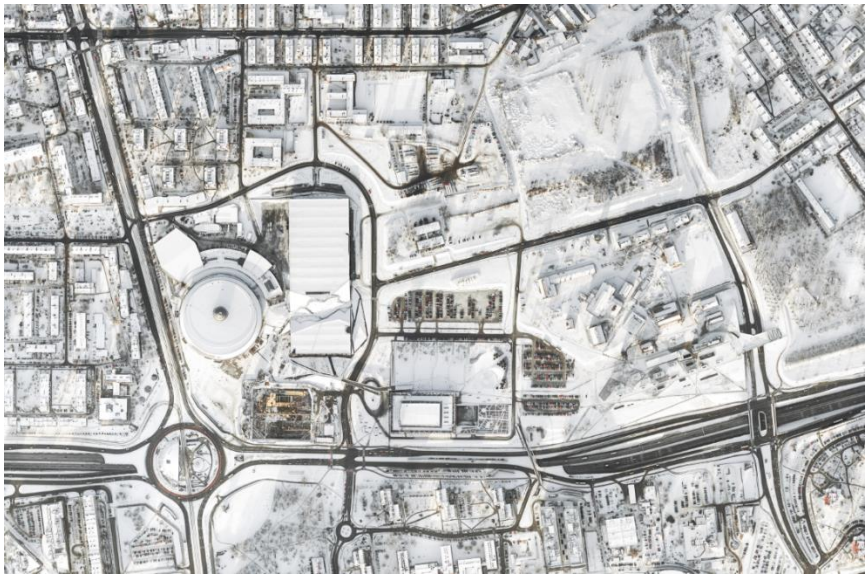
3.2.2. Pozostałe funkcje

W celu zastosowania transformaty Fouriera oraz odwrotnej transformaty Fouriera na obrazie wykorzystano funkcję fft2 i ifft2 z biblioteki numpy. Z tej samej biblioteki wykorzystano funkcję hanning, aby zastosować na obrazie okno Hanninga

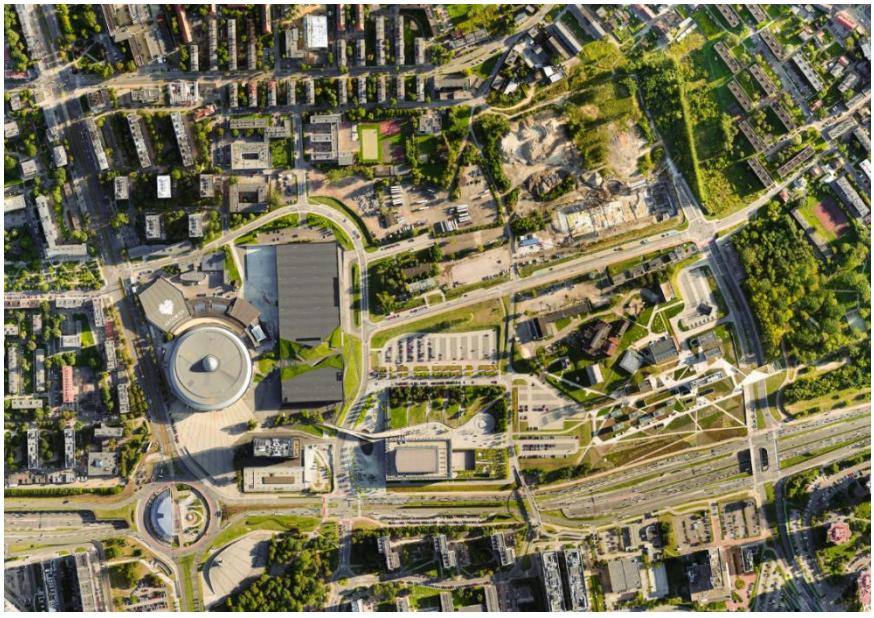
```
numpy.fft.fft2(a, s=None, axes=(-2, -1), norm=None)

numpy.fft.ifft2(a, s=None, axes=(-2, -1), norm=None)

numpy.hanning(M)
```

3.3. Metodyka testów

Testy zostały przeprowadzone na obrazach przedstawiających miasto z lotu ptaka, zrobionych dronem, a więc takich jakie mogą mieć zastosowanie w praktyce. Dwa zdjęcia obrazują tą samą scenerię latem i zimą, dlatego możliwe jest zbadanie wpływu warunków atmosferycznych, występujących w trakcie akwizycji danych, na działanie programu.



Rys. 3.2. Oryginalny obraz „zima.jpg”

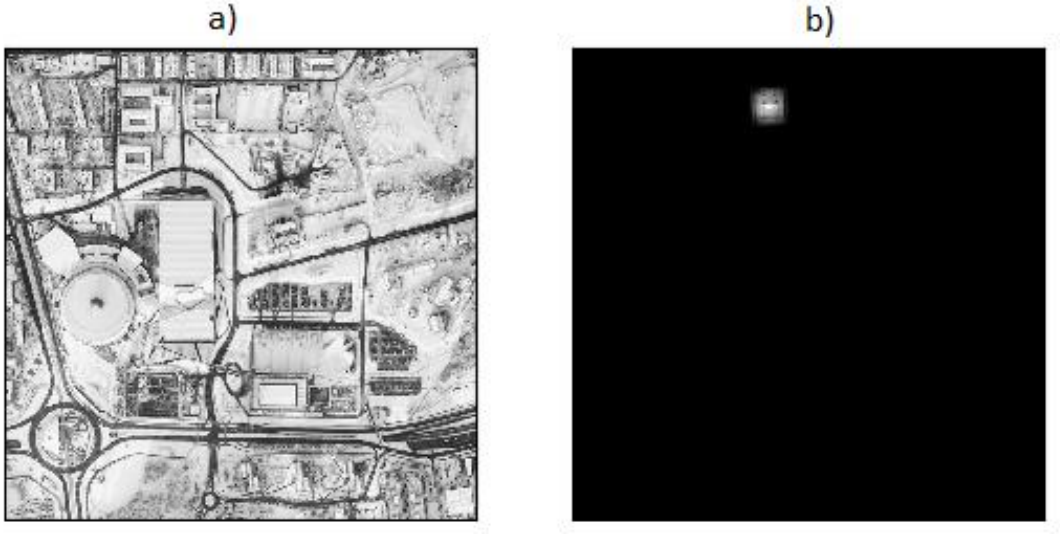


Rys. 3.3. Oryginalny Obraz „lato.jpg”

W implementacji, obrazy zostały przeformatowane do postaci plików bmp i reprezentowane w skali szarości. Na ich podstawie wygenerowano bazę wzorców, mniejszych, kwadratowych fragmentów, obejmujących konkretne budynki lub fragmenty zabudowania. Dla przykładu kilka z nich przedstawiono na rysunku 3.4.

Rys. 3.4. Przykładowe obrazy wzorcowe

Wynikiem programu było umieszczenie wzorca w odpowiednim miejscu na czarnym tle o rozmiarach obrazu referencyjnego, co pozwalało na wizualną interpretację rezultatów.



Rys. 3.5. Przykładowy wynik programu
a) wejściowy obraz referencyjny b) obraz wyjściowy wraz ze zlokalizowanym wzorcem

3.4. Działanie podstawowego algorytmu

Przedstawiony wcześniej algorytm został przetestowany na różnych obrazach wejściowych o różnych rozmiarach i odpowiednich bazach wzorców.

Pierwszą serię testów poszukiwania wzorca przeprowadzono dla obrazów referencyjnych w małych rozmiarach i wzorców o stałym rozmiarze 64 px. Obraz referencyjny był w takiej samej skali i orientacji jak wzorce, dlatego jedynym zadaniem programu w tym przypadku było odnalezienie właściwego położenia. Wyniki pierwszych testów dla obrazów referencyjnych o rozmiarach 200 – 722 px zestawiono tabeli 3.1.

Tabela 3.1. Testy podstawowego algorytmu dla małych rozmiarów obrazu

Rozdzielczość [px]	Liczba dobrych wyników	Liczba testów	Wynik [%]
722	6	12	50%
500	8	15	53%
400	5	6	83%
200	2	2	100%
SUMA	21	35	60%

Ogólna poprawność działania została wyliczona jako 60%, jednak można zauważyć, że mniejszy obraz wejściowy (przy stałym rozmiarze wzorca) dawał dużo lepsze rezultaty. W przypadku rozdzielczości powyżej 500px poprawność wynosi ok 50%. Ta zależność zostanie przeanalizowana oraz wykorzystana w celu poprawy wyników w dalszej części pracy.

W tabeli 2 zestawiono wyniki programu bazowego, na wejściowym obrazie rozdzielczości 4000px o różnych zadanych obrotach i skalach (kolumny 2 i 3, etykiety obrót i skala). Łącznie sprawdzono aż 546 przypadków testowych. Poprawność działania została wyznaczona jako stosunek dobrze zlokalizowanych wzorców do liczby przeprowadzonych testów, wartość podano w procentach. Określono ją wizualnie, w kolejnych tabelach mogą znaleźć się „połówkowe" wyniki, wartość 0.5 wpisano w sytuacjach, gdy pomimo dużego błędu kąta lub skali, wzorzec i tak został dobrze umiejscowiony na obrazie referencyjnym. Kolejna kolumna - średni błąd skali została obliczona na podstawie różnic między właściwą skalą, a tą wyznaczoną w każdym przypadku testowym. Średni błąd kąta reprezentuje procentową wartość stosunku źle wyznaczonych kątów do wszystkich próbek.

Tabela 3.2. Testy podstawowego algorytmu dla dużych obrazów

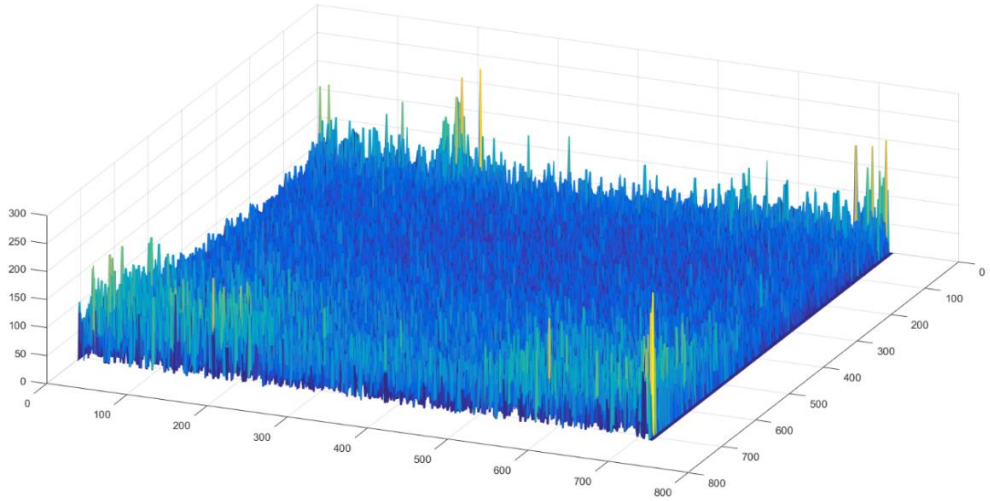
Rozdzielczość	Skala	Kąt [°]	Liczba dobrych wyników	Liczba testów	Średni błąd skali	Błędne kąty	Wynik [%]
4000	-	-	17	70	0,028	24%	25%
4000	-	-90	3	70	0,026	85%	0%
4000	-	-30	0	49	0,075	98%	0%
4000	-	-135	0	43	0,010	94%	0%
4000	0,75	-	0	40	0,261	85%	0%
4000	0,80	-	0	70	0,173	72%	0%
4000	0,90	-	5	70	0,105	63%	7%
4000	1,10	-	3	70	0,116	48%	4%
4000	1,20	-	2	29	0,193	18%	7%
4000	1,25	-	0	40	0,248	78%	0%
Suma/średnia			30	546	0,120	48%	5%

Prawidłowe rozwiązania zdarzają się w przypadku braku obrotu i przeskalowania i wynoszą jedynie 25 %, w innych przypadkach wynoszą zaledwie kilka procent lub są bliskie 0. Średnia wartość poprawności rezultatów działania programu bazowego na dużych rozmiarach wynosi 5%, co nie jest zadowalającym wynikiem. Przy braku obrotu i skali już pojawia się duża liczba źle wyznaczonych kątów. Ogółem średnio połowa kątów została źle wyliczona, a błąd skali wynosi aż 0,12 czyli 12%.

3.5. Opis i wyniki eksperymentów

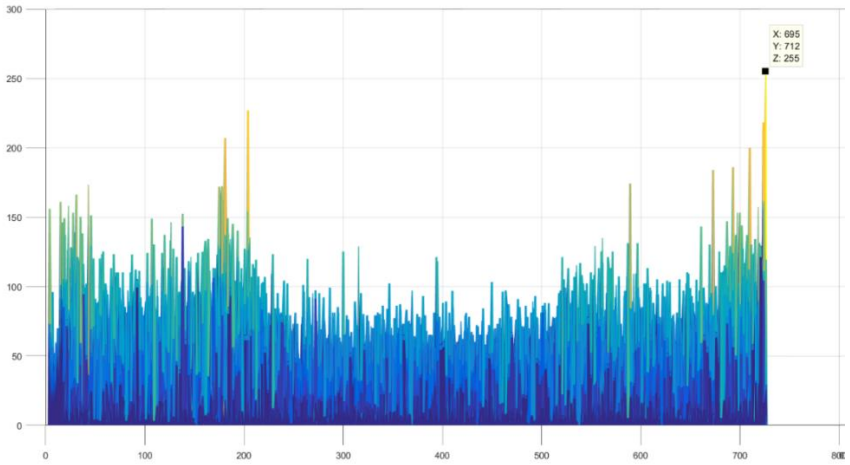
3.5.1. Maximum korelacji fazowej

Pierwsze badania algorytmu pokazują niezadowalające rezultaty znajdywania poszukiwanego wzorca. Nieprawidłowości pojawiały się już w badaniach wzorców, o takiej samej skali i orientacji jak obraz referencyjny. Prowadzi to do szukania błędów w samej lokalizacji wycinka na większym obrazie. Korelacja fazowa, która miała podać przesunięcie obrazów transformat Fouriera w układzie log-polar, a więc określić skalę i rotację wzorca, wskazywała nieprawidłowe współrzędne. Odpowiedni punkt nie był największą wartością wyniku korelacji. Obraz korelacji fazowej przedstawiono na rysunku 3.6 w 3D, tak aby zwizualizować zjawisko występowania kilka grup lokalnych maksimów (największym wartościom obrazu odpowiadają najwyższe słupki w kolorze żółtym).



Rys. 3.6. Obraz przedstawiający korelację fazową w 3D

Na przykładowym obrazie można dostrzec 3 skupiska maksimów, rysunek 3.7 przedstawia widok od przedniej strony, umożliwiający porównanie wysokości żółtych słupków.



Rys. 3.7. Obraz korelacji fazowej z perspektywy

W przedstawionym przykładzie maksimum, które wskazywało na prawidłowy obrót i skalę, było czwartym z kolei.

Nasuwającym się rozwiązaniem tego problemu było zbadanie kilku lokalnych maksimów. Dla każdego z nich przeprowadzono transformację wzorca i kolejną korelację z obrazem referencyjnym. Wybrano ten punkt (wskazujący obrót i skalę), który dawał później najwyższy współczynnik w kolejnej korelacji, tj. korelacji obrazu referencyjnego z odpowiednio obróconym i przeskalowanym wzorcem. Istotnym zadaniem był właściwy dobór ilości sprawdzanych maksimów, tak aby poprawić działanie algorytmu, ale zarazem nie wykonywać wielu powtarzających się operacji, które znacznie wydłużały czas działania programu.

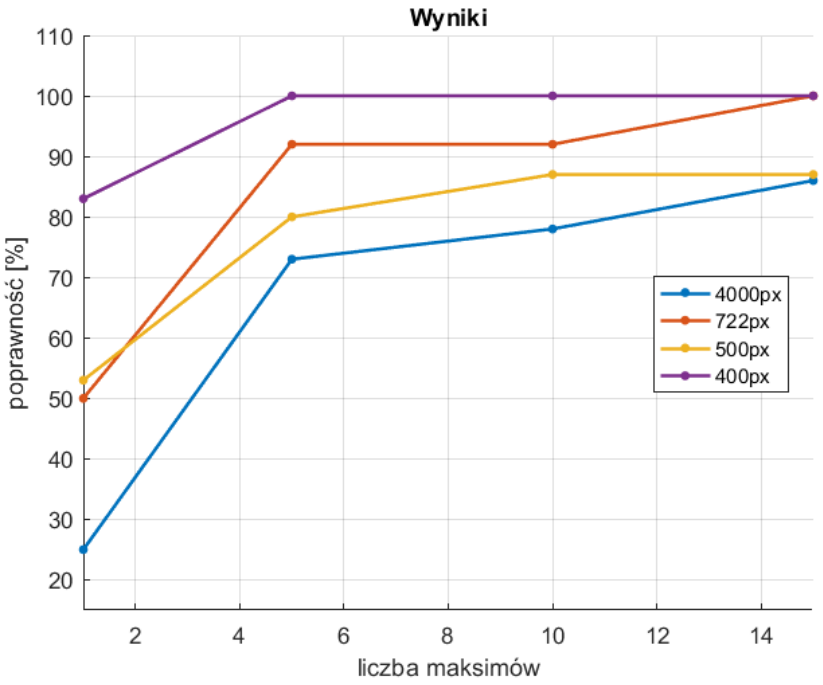
Efekty przeprowadzonych testów zamieszczono w kolejnych tabelach, przedstawiono rezultaty zależnie od dobranej liczby sprawdzanych maksimów, dla obrazów prostych, obróconych oraz powiększonych. Początkowo badano obrazy wejściowe o mniejszych rozdzielczościach (400 – 722 px), natomiast docelowe testy zostały przeprowadzone na obrazach o rozdzielczości prawie 4k.

Tabela 3.3. Badania zależności działania algorytmu od ilości sprawdzanych maksimów korelacji fazowej dla obrazu bez obrotu i przeskalowania

Rozdzielczość [px]	1 maksimum	5 maksimów	10 maksimów	15 maksimów
722	50%	92%	92%	100%
500	53%	80%	87%	87%
400	83%	100%	100%	100%
4000	20%	71%	76%	85%
Średnia	52%	86%	89%	93%

Tabela 3.3 przedstawia poprawność wyników w zależności od rozmiaru obrazu referencyjnego (etykieta ‘Rozdzielczość’) oraz ilości sprawdzanych maksimów (1-15). Wyniki podano w procentach, wyrażają one stosunek liczby dobrze znalezionych wzorców do liczby przeprowadzonych testów. Na podstawie tabeli 2 wygenerowano wykres, przedstawiający wzrost poprawności wyników od zwiększającej się liczby badanych maksimów (rysunek 3.8).

Największy wzrost jakości wyników wynika ze zmiany z 1 do 5 maksimów i zachodzi to w każdym przypadku, w dalszych etapach poprawa także występuje, ale jest znacznie mniejsza.



Rys. 3.8. Zależność poprawności działania algorytmu od ilości sprawdzanych maksimów.

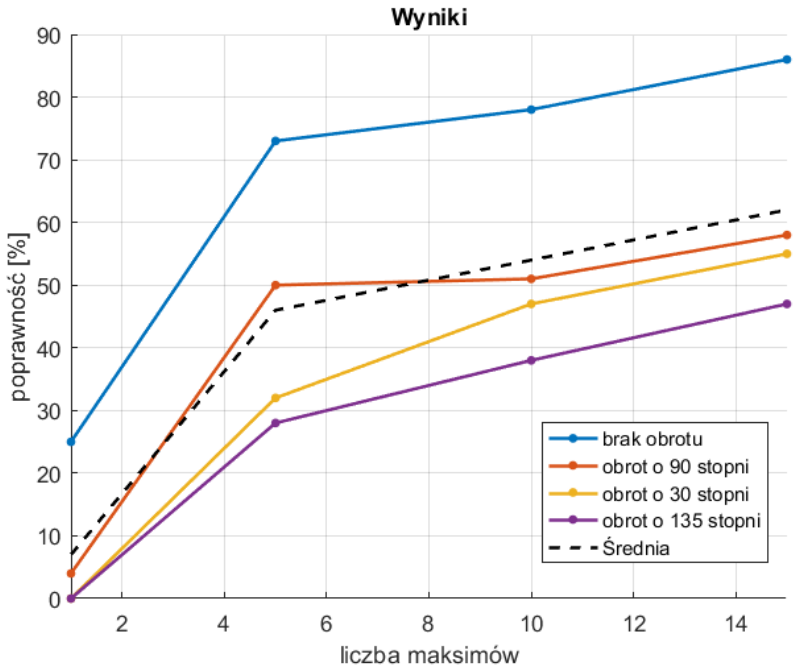
W tabeli 3.4 przedstawiono wyniki badane na obrazie o rozdzielczości 4000 dla różnej liczby maksimów, obrotów oraz skali.

Tabela 3.4. Badania zależności działania algorytmu od ilości sprawdzanych maksimów korelacji fazowej dla obrazu w rozdzielczości 4000 px

Rozdzielczość [px]	Skala	Kąt [°]	1 maksimum	5 maksimów	10 maksimów	15 maksimów
4000	-	-	25%	73%	78%	86%
4000	-	-90	4%	50%	51%	58%
4000	-	-30	0%	32%	47%	55%
4000	-	-135	0%	28%	38%	47%
Średnia			7%	46%	54%	62%
4000	0,75	-	0%	1%	1%	1%
4000	0,80	-	0%	3%	4%	4%
4000	0,90	-	7%	13%	14%	15%
4000	1,10	-	4%	15%	25%	23%
4000	1,20	-	7%	3%	10%	10%
4000	1,25	-	0%	4%	6%	8%
Średnia			3%	7%	10%	10%

Analizując rezultaty, uzyskane na tym etapie, można zauważyć znaczny wpływ liczby badanych maksimów na wyniki, zwłaszcza w przypadku obróconych obrazów, gdzie występuje poprawa o ponad 50%.

Wyniki dotyczące obrazów w rozdzielczości 4000 px również przedstawiono na wykresie (rysunek 3.9).



Rys. 3.9. Zależność poprawności działania algorytmu od ilości sprawdzanych maksimów dla różnych obrotów obrazu (4000 px)

W celu zweryfikowania poprawy rozwiązań, zebrano informacje o poprawnie wyznaczonych kątach i skalach, te wskaźniki poprawy przedstawiono w tabelach 4 i 5. Pierwsza z nich prezentuje procentową wartość błędu kąta. Średnio wyniki poprawiły się o 10% pod tym względem, a w szczególnych przypadkach, związanych z obrotem, aż 24% i 32%.

Tabela 3.5. Wskaźnik poprawy kąta

Rozdzielczość [px]	Skala	Kąt [°]	5 maksimów	10 maksimów	15 maksimów
4000	-	-	24,8%	21,0%	14,0%
4000	-	-90	50,0%	58,8%	42,3%
4000	-	-30	67,4%	51,0%	43,0%
4000	-	-135	74,9%	61,1%	42,5%
4000	0,75	-	85,0%	80,0%	80,0%
4000	0,80	-	82,5%	81,5%	78,5%
4000	0,90	-	74,0%	76,5%	80,0%
4000	1,10	-	66,0%	64,0%	57,5%
4000	1,20	-	77,0%	76,0%	77,0%
4000	1,25	-	85,0%	82,5%	80,0%
Średnia			69%	65%	59%

W kwestii skali znaczny wpływ na dobre wyznaczenie parametru miała zmiana na sprawdzanie 10 maksimów. Średnie wartości różnią się aż o 0.1 a więc 10%, co przy badanych skalowaniach jest dużą poprawą.

Tabela 3.6. Wskaźnik poprawy skali

Rozdzielczość [px]	Skala	Kąt [°]	5 maksimów	10 maksimów	15 maksimów
4000	-	-	0,01	0,01	0,01
4000	-	-90	0,01	0,02	0,02
4000	-	-30	0,03	0,03	0,02
4000	-	-135	0,08	0,10	0,06
4000	0,75	-	0,27	0,28	0,27
4000	0,80	-	0,12	0,21	0,22
4000	0,90	-	0,10	0,11	0,11
4000	1,10	-	0,11	0,10	0,10
4000	1,20	-	0,22	0,19	0,19
4000	1,25	-	1,27	0,23	0,23
Średnia			0,22	0,13	0,12

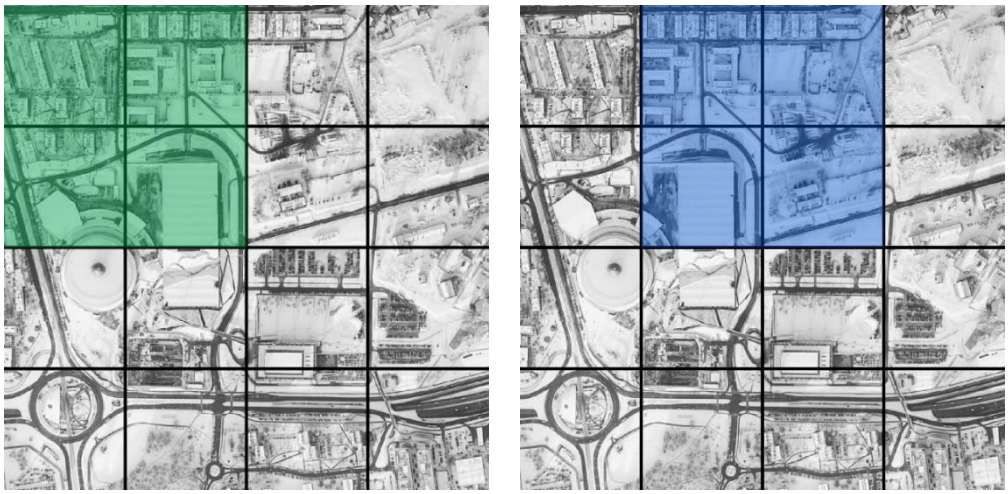
Nawet po zbadaniu kilku przypadków widać, że tylko niewielka zmiana skali, pomniejszenie o maksymalnie 10%, daje poprawne wyniki, co w efekcie jest mało praktyczne. Dodatkowo sprawdzono także, jak algorytm radzi sobie ze znalezieniem skali wzorców o większym rozmiarze – 200 px, a więc czterokrotnie mniejszym od obrazu referencyjnego. Taki wzorzec zawiera znacznie więcej informacji nie obejmuje konkretnego budynku, ale większy fragment mapy. W tym przypadku algorytm poprawnie szacował skalę nawet przy pomniejszeniu do 50%. Ta obserwacja prowadzi do dalszych eksperymentów z rozmiarem wzorca i obrazu referencyjnego, opisanych w kolejnym podrozdziale.

3.5.2. Fragmentacja obrazu wejściowego

Z poprzednich przeprowadzonych testów można wnioskować, iż algorytm gorzej działa dla obrazów referencyjnych znacznie większych od wzorca. Znajduję się na nich bardzo dużo informacji nie dotyczących poszukiwanego fragmentu, a te właściwe zostają utracone przy wykonywaniu kolejnych operacji.

Właściwym podejściem wydaje się podzielenie obrazu na mniejsze części i przeprowadzenia poszukiwania wzorca na każdej z nich osobno. Te obszary muszą się na siebie nakładać, aby uwzględnić sytuację, w której wzorzec znajduje się na granicy dwóch lub czterech obszarów, ponieważ informacje o ½ lub ¼ obiektu mogłyby nie być wystarczające, aby go poprawnie zlokalizować. Przy odpowiedniej fragmentacji obrazu, zyskuje się dodatkową zaletę, wynikającą z tego, iż dany najmniejszy obszar będzie kilkakrotnie sprawdzany. Takie podejście zwiększa prawdopodobieństwo odczytania obrotu i skali, a co za tym idzie - ich prawidłowej lokalizacji.

W niniejszej pracy zastosowano fragmentację, przedstawioną na rysunku 3.11.

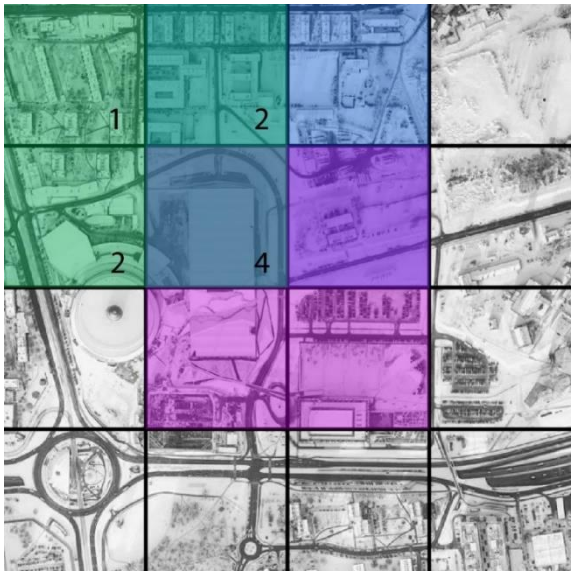


Rys. 3.10. Obraz referencyjny z wydzielonymi obszarami

Obraz wejściowy został podzielony na 16 jednakowych kwadratowych obszarów, bok każdego z nich jest równy ¼ rozmiaru całego obrazu. Analizie zostały poddane kwadratowe obszary złożone z 4 takich bloków, zaznaczone na rysunku 3.11. zielonym i niebieskim kolorem. Istnieje 9 takich fragmentów, wszystkie operacje będą przeprowadzane na każdym z nich.

Na kolejnym rysunku przedstawiono efekt nakładania się na siebie analizowanych obszarów. Na obrazie zaznaczono, ile razy będzie analizowany dany najmniejszy zaznaczony fragment. Może się wydawać, że to zjawisko powoduje dodatkowy nakład niepotrzebnych obliczeń, jednak warto zauważyć, że jest to dodatkowa zaleta. Szukanie obiektu na kilku zbiorach danych wejściowych zwiększa szanse na poprawne zlokalizowanie wzorca.

Elementy znajdujące się w strefie najbliższej centrum całego obrazu referencyjnego będą sprawdzane czterokrotnie, czyli 4 razy więcej niż te znajdujące się w rogach obrazu, to z kolei spowoduje prawdopodobieństwo wykrycia obiektu znajdującego się blisko centrum obrazu będzie większe.



Rys. 3.11. Efekt nakładania się fragmentów obrazu

Po dodaniu do algorytmu fragmentacji obrazu wejściowego, należało sprawdzić wartości maksimów korelacji każdego fragmentu ze wzorcem i wybrać największą z nich oraz odpowiadający jej fragment. Dla małych obrazów wejściowych przetestowano różne obroty, wyniki zestawiono w tabeli 3.7.

Tabela 3.7. Badania fragmentacji obrazu referencyjnego o różnych obrotach dla małych rozmiarów obrazu

Rozdzielczość [px]	Kąt [°]	Liczba dobrych wyników	Liczba testów	Wynik [%]
722	90	8	11	73%
722	45	5	7	71%
504	140	6	7	86%
Suma/Średnia		19	25	76%

Przy małych rozmiarach (504-722 px) obrazu, algorytm wzbogacony o fragmentację, daje dosyć wysokie rezultaty, średnio 76% poprawności. Dokładniejsze badania przeprowadzone dla rozmiarów 4000 px przedstawia tabela 3.8. W tabeli dodatkowo zaprezentowano porównanie rezultatów z poprzednią wersją algorytmu uwzględniającą 10 maksimów, zestawiono obok siebie wartości wyników oraz obliczono różnicę między nimi, podaną w ostatniej kolumnie, która waha się od 3% do aż 53%.

Tabela 3.8. Badania fragmentacji obrazu referencyjnego dla dużych rozmiarów obrazu

Rozdzielczość [px]	Skala	Kąt [°]	Liczba dobrych wyników	Liczba testów	10 maksimów	Fragmentacja	Różnica [%]
4000	-	-	70	70	78%	100%	22%
4000	-	-90	56	70	51%	80%	29%
4000	-	-30	45,5	47	47%	97%	50%
4000	-	-135	41	45	38%	91%	53%
4000	0,75	-	1,5	40	1%	4%	3%
4000	0,80	-	6,5	70	4%	9%	5%
4000	0,90	-	25,5	70	14%	36%	23%
4000	1,10	-	49,5	70	25%	71%	46%
4000	1,20	-	8	30	10%	27%	17%
4000	1,25	-	5	40	6%	13%	6%
Suma/Średnia			308,5	552	27%	53%	25%

W przypadku obrotów poprawność działania jest prawie stuprocentowa. W odpowiednim znalezieniu skali nadal nie uzyskano zadowalających rezultatów, jedynie przy zmniejszeniu lub zwiększeniu obrazu referencyjnego od 10-20 % wyniki są obiecujące, przy większych różnicach poprawność oscyluje wokół zaledwie 10%. Warto zwrócić uwagę, że w przypadku zmiany skali zostaje utracona część informacji obrazie, co nie występuję przy samym obrocie. W przypadku niewielkich zmian skali obrazu (do 20%) ilość tych informacji nie jest aż tak duża

więc możliwe jest poprawne wyznaczenie parametrów i lokalizacja wzorca, dodatkowo wprowadzona fragmentacja poprawia wyniki, nawet o 46% (skala 1.1).

W tabeli 3.9 zestawiono miary błędu skali i kąta, 2 ostatnich wersji programu (10 maksimów, fragmentacja), aby móc porównać prawidłowość wyznaczonych parametrów.

Tabela 3.9. Badania fragmentacji obrazu referencyjnego dla dużych rozmiarów obrazu

Rozdzielczość [px]	Skala	Kąt [°]	Maksima		Fragmentacja		Poprawa skali	Poprawa kąta
			Średni błąd skali	Błędne kąty	Średni błąd skali	Błędne kąty		
4000	-	-	0,007	21,0%	0,000	0%	0,007	21%
4000	-	-90	0,017	58,8%	0,007	17%	0,010	42%
4000	-	-30	0,029	51,0%	0,005	3%	0,024	49%
4000	-	-135	0,097	61,1%	0,000	6%	0,097	55%
4000	0,75	-	0,278	80,0%	0,276	68%	0,002	13%
4000	0,80	-	0,212	81,5%	0,226	79%	-0,014	3%
4000	0,90	-	0,105	76,5%	0,111	57%	-0,006	20%
4000	1,10	-	0,097	64,0%	0,046	17%	0,051	47%
4000	1,20	-	0,192	76,0%	0,144	63%	0,048	13%
4000	1,25	-	0,228	82,5%	0,202	78%	0,026	5%
Suma/Średnia			0,126	65,2%	0,102	38,6%	0,025	26,7%

Porównanie błędów w wyznaczeniu skali i obrotu ukazuje, że wprowadzenie fragmentacji miało pozytywny wpływ przede wszystkim na obliczenie kąta. Przy braku obrotu i skalowania wszystkie kąty były dobrze wyznaczone. W kwestii obróconych obrazów nastąpiła poprawa o 42-55%, a w kolejnych przypadkach mniejsza, ale nadal istotna o 3-47%, średnio 39% kątów było błędnych, a to o 27% mniej niż przed fragmentacją.

Średnia poprawa wyliczenia skali nie jest duża, natomiast w poszczególnych przypadkach ma znaczenie, np. w wersie 4 (obrót o 135 stopni) średni błąd wynoszący 0.1 (10%) został całkowicie zniwelowany. Znaczne różnice dotyczą także skali 1,2 oraz 1,1, gdzie poprawa wynosi 0.05 (5%). Sytuacje, gdy błąd różni się o +/- 1% nie mają dużego wpływu na działanie algorytmu.

3.5.3. Powiększenie wzorca

Początkowo obraz wzorca był reprezentowany jako mniejszy element umieszczony na czarnym tle. To podejście mogło powodować pewne „zakłamanie” podczas kolejnych transformacji obrazu, dlatego następnym eksperymentem było powiększenie wzorca do rozmiarów obrazu referencyjnego. W związku z tą różnicą, niezbędna okazała się odpowiednia zmiana wyliczenia skali, która została uzależniona od wstępnego powiększenia wzorca.

Operację przeprowadzono poprzez użycie funkcji resize z biblioteki openCv, która zawiera opcjonalny parametr interpolation, decydujący o metodzie interpolacji. Spośród możliwych opcji, sprawdzono zastosowanie interpolacji:

- dwuliniowej,

- dwusześciennej (uwzględniającej sąsiedztwo 4×4),

- algorytmem Lanczosa (uwzględniającej sąsiedztwo 8x8).

Wyniki poprawności działania w zależności od dobranej metody interpolacji zebrano w tabeli 3.10. Nazwy dodanych kolumn odpowiadają wybranemu parametrowi, decydującemu o metodzie interpolacji. W zależności od wybranej metody przedstawiono procentową wartość poprawności rezultatów. Obroty 30/135 testowano na 49/43 próbkach, pozostałe przypadki - na 70 próbkach.

Tabela 3.10. Badania algorytmu, z powiększeniem wzorca, w zależności od metody interpolacji

Rozdzielczość [px]	Skala	Kąt [°]	Linear	Cubic	Lanczos
4000	-	-	55,8%	6,3%	43,3%
4000	-	-90	50,2%	2,9%	43,3%
4000	-	-30	45,1%	8,3%	13,1%
4000	-	-135	39,6%	8,9%	1,3%
4000	0,75	-	33,5%	17,1%	10,0%
4000	0,80	-	47,5%	10,0%	31,7%
4000	0,90	-	33,8%	14,0%	7,9%
4000	1,10	-	26,7%	1,9%	12,1%
4000	1,25	-	16,7%	4,2%	37,9%
Średnia			38,8%	8,2%	22,3%

Badania wykazały, że najlepszą spośród sprawdzanych metod jest interpolacja liniowa. Prawie w każdym przypadku daje znacznie lepsze wyniki od obu pozostałych interpolacji, pomimo iż jest najmniej złożoną metodą. Średnia poprawa względem algorytmu Lanczosa wynosi ok. 16%, a względem metody dwukubicznej aż 30%. Jedynie przy skali 1,25 metoda Lanczosa daje ponad dwukrotnie lepszy wynik.

W celu porównania metod wyznaczono także procentowy błąd kąta oraz błąd skali. W tym przypadku różnice w skali są znacznie mniejsze, ponieważ obliczona skala zawierała też pomniejszenie wzorca odwrotne do powiększenia funkcją resize, dlatego w odróżnieniu od wcześniejszych przypadków poprawna skala jest dziesięciokrotnie mniejsza. Nie ma więc sensu porównywać tych danych z poprzednimi, ale można wykazać znaczne różnice pomiędzy metodami interpolacji. Tabela 3.11 przedstawia błąd skali, podany w formacie zmiennoprzecinkowym, natomiast tabela 3.12 pokazuje procentowy błąd kąta.

Tabela 3.11. Średni błąd skali dla algorytmu z powiększeniem wzorca

Rozdzielczość [px]	Skala	Kąt [°]	Linear	Cubic	Lanczos
4000	-	-	0,030	0,082	0,047
4000	-	-90	0,041	0,085	0,050
4000	-	-30	0,046	0,040	0,044
4000	-	-135	0,056	0,030	0,038
4000	0,75	-	0,032	0,041	0,050
4000	0,80	-	0,032	0,044	0,036
4000	0,90	-	0,045	0,060	0,070
4000	1,10	-	0,053	0,089	0,084
4000	1,25	-	0,068	0,096	0,035
Średnia			0,045	0,063	0,050

Przedstawione średnie błędy kąta potwierdzają, że najlepszą metodą interpolacji jest interpolacja dwuliniowa, w jej przypadku błąd jest najmniejszy, na drugim miejscu, z niewielką różnicą, plasuje się interpolacja metodą Lanczosa, a na trzecim – dwukubiczna.

Tabela 3.12. Błąd kąta dla algorytmu z powiększeniem wzorca w zależności od interpolacji

Rozdzielczość [px]	Skala	Kąt [°]	Linear	Cubic	Lanczos
4000	-	-	20,4%	78,0%	48,4%
4000	-	-90	50,9%	97,5%	54,6%
4000	-	-30	52,4%	86,8%	97,4%
4000	-	-135	50,8%	88,3%	97,6%
4000	0,75	-	55,0%	60,0%	82,5%
4000	0,80	-	45,9%	77,5%	53,8%
4000	0,90	-	63,8%	74,2%	77,8%
4000	1,10	-	48,8%	88,5%	76,3%
4000	1,25	-	61,3%	75,9%	52,1%
Średnia			49,9%	80,7%	71,1%

Interpolacja dwuliniowa równie dobrze sprawdza się przy wyznaczaniu kąta w porównaniu do pozostałych metod, średnio poprawnie wyznacza 50% kątów, czyli o 21% więcej niż interpolacja Lanczosa i 30% więcej niż metoda dwukubiczna.

Reasumując, każde ze sprawdzonych kryteriów potwierdza przewagę interpolacji liniowej nad pozostałymi oraz to, że interpolacja dwukubiczna daje najgorsze rezultaty.

W celu zweryfikowania wpływu zastosowania powiększenia wzorca na działanie programu, wybrano metodę interpolacji dającą najlepsze wyniki i porównano ją z poprzednią wersją, uwzględniającą fragmentację. Rezultaty pokazano w tabeli 3.13.

Tabela 3.13. Porównanie metod bez i z powiększeniem wzorca

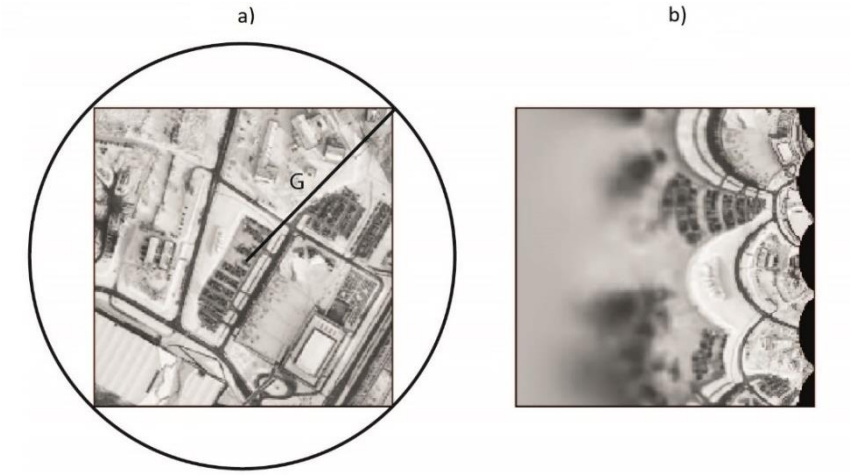
Rozdzielczość [px]	Skala	Kąt [°]	Fragmentacja	Resize (Linear)	Różnica
4000	-	-	100,0%	50,0%	-50,0%
4000	-	-90	80,0%	43,4%	-36,6%
4000	-	-30	96,8%	46,2%	-50,6%
4000	-	-135	91,1%	47,0%	-44,1%
4000	0,75	-	3,8%	38,4%	34,7%
4000	0,80	-	9,3%	48,8%	39,5%
4000	0,90	-	36,4%	26,9%	-9,6%
4000	1,10	-	70,7%	27,5%	-43,2%
4000	1,20	-	27,0%	13,3%	-13,7%
4000	1,25	-	12,5%	16,7%	4,2%
Średnia			52,8%	35,8%	-16,9%

Wyniki porównania wprowadzenia operacji zmiany rozmiaru wzorca są niejednoznaczne. Rezultaty poprawiają się jedynie w przypadku zmiany skali, a znacznie pogarszają w badaniach obrotu. Różnice pomiędzy wynikami z poprzedniego etapu rozwijania algorytmu wahają się od -50% do 40%. Można zaobserwować zależność wyników od przypadków testowych. Znaczna poprawa występuje przy zmniejszaniu obrazu wejściowego. Tak więc zastosowanie tej modyfikacji nie jest jednoznacznym ulepszeniem metody realizacji algorytmu.

3.5.4. Zmiana parametru funkcji WarpPolar

Wspomniana wcześniej funkcja WarpPolar wykorzystana do transformacji do układu współrzędnych logarytmiczno-biegunowych jest możliwa do sparametryzowania pod względem maksymalnej długości promienia (parametr MaxRadius). To od niego zależy, jak zostaną reprezentowane wybrane części obrazu w nowym układzie współrzędnych. Transformacja do nowego układu powoduje, że pewnie fragmenty są dokładniej przedstawione od innych. Modyfikując wspomniany parametr można to zmieniać. Maksymalny promień transformacji log-polar można ustawić jako połowę szerokości obrazu wejściowego (zmienną nazwano – R) lub jako połowę jego przekątnej (zmienna – G). Na rysunkach 3.13 i 3.14 pokazano, jak wygląda przykładowa transformacja log-polar dla różnych wartości parametru MaxRadius.

Rys. 3.12. Transformacja log polar, dla parametru MaxRadius = R
a) obraz wejściowy, z zaznaczonym maksymalnym promieniem i obszarem który wyznacza
b) obraz po transformacji



Rys. 3.13. Transformacja log polar, dla parametru MaxRadius = G
a) obraz wejściowy, z zaznaczonym maksymalnym promieniem i obszarem który wyznacza
b) obraz po transformacji

Testy opisanej w niniejszym rozdziale modyfikacji przedstawiono w tabeli 3.14.

Tabela 3.14. Badania parametryzacji funkcji WarpPolar dla dużych obrazów wejściowych.

Rozdzielczość [px]	Skala	Kąt[°]	Liczba dobrych wyników	Liczba testów	Średni błąd skali	Błędne kąty	MaxRadius = G	MaxRadius = R	Różnica [%]
4000	-	-	70	70	0,000	0,0%	100%	50%	50%
4000	-	-90	66	70	0,000	3,3%	94%	43%	51%
4000	-	-30	8	49	0,035	85,5%	16%	46%	-30%
4000	-	-135	14,5	43	0,038	66,5%	34%	47%	-13%
4000	0,75	-	4,5	70	0,050	66,5%	6%	38%	-32%
4000	1,25	-	5,5	70	0,034	57,5%	8%	49%	-41%
4000	0,80	-	7	70	0,037	59,8%	10%	27%	-17%
4000	0,90	-	25,5	70	0,019	45,3%	36%	28%	9%
4000	1,10	-	22	70	0,013	49,0%	31%	17%	15%
Suma/średnia			223	582	0,025	48,1%	37,4%	38,3%	-0,9%

Wyniki przeprowadzenia tego eksperymentu nie są jednoznaczne co widać na zaprezentowanych przykładach. W przypadku braku obrotu, obrotu o 90° oraz skali 0.9 i 1.1 wyniki są lepsze niż dotychczas. Niestety w reszcie przypadków następuje spadek jakości rozwiązań. Średnia wyliczona wartość poprawności jest bliska zeru, a więc nie wskazuje jednoznacznie czy rozwiązanie jest lepsze, czy gorsze. Pomimo iż wyniki się pogarszają, w kwestii błędu kąta i skali następuje mała poprawa, o odpowiednio 2% i 0.013.

W celu zweryfikowania działania metody ze zmienionym parametrem, przetestowano także mniejsze obrazy wejściowe, o mniejszych skalach. Przedstawiają one zależność poprawności rezultatów od zadanej skali i przyjętego maksymalnego promienia (R lub G) oraz wyliczoną różnicę między wynikami, której znak świadczy o pogorszeniu lub polepszeniu rezultatów.

Tabela 3.15. Badania parametryzacji funkcji WarpPolar dla obrazu o rozdzielczości 722 w różnych skalach.

Skala	0,7	0,6	0,5	0,4
R	100%	100%	85%	0%
G	15%	85%	100%	23%
Poprawa	85%	15%	-15%	-23%

Można tu zauważyć ciekawą zależność, ponieważ w przypadku niedużych zmian skali lepsze efekty daje ustawienie maksymalnego promienia log-polara jako połowy szerokości obrazu. Natomiast dla większych zmian skali, tj. ponad dwukrotnych pomniejszeń, lepsze rezultaty pojawiają się przy zastosowania połowy przekątnej obrazu. Może to wynikać z faktu, iż w drugim przypadku uzyskany obraz będzie pomniejszony, a więc jego transformata Fouriera będzie powiększona, co z kolei spowoduje, że na środku obrazu widocznych będzie więcej szczegółów. Taka sama bezpośrednia zależność dotyczy błędu skali, który dopiero w przeskalowaniu obrazu wejściowego do 50%, jest mniejszy dla parametru MaxRadius równego G. Te wyniki przedstawiono w tabeli 3.16.

Tabela 3.16. Średni błąd skali przy parametryzacji funkcji WarpPolar dla obrazu o rozdzielczości 722 w różnych skalach.

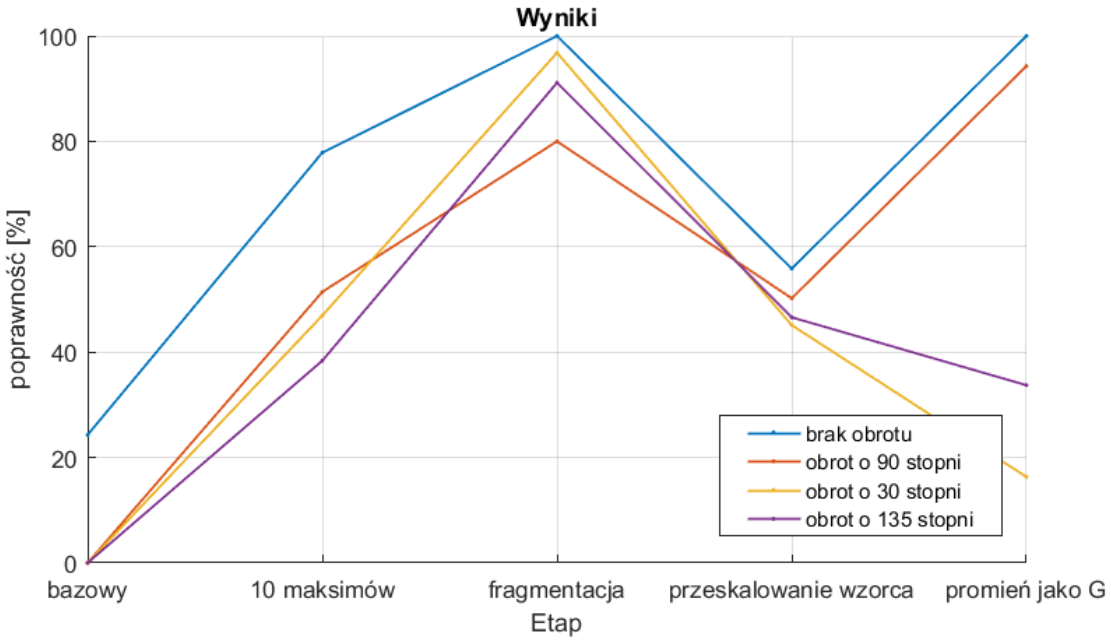
Skala	0,8	0,7	0,6	0,5
R	0,843	0,085	0,047	0,162
G	1,008	0,737	0,159	0,111
Poprawa	-0,165	-0,652	-0,112	0,051

4. Podsumowanie

4.1. Zebranie wyników

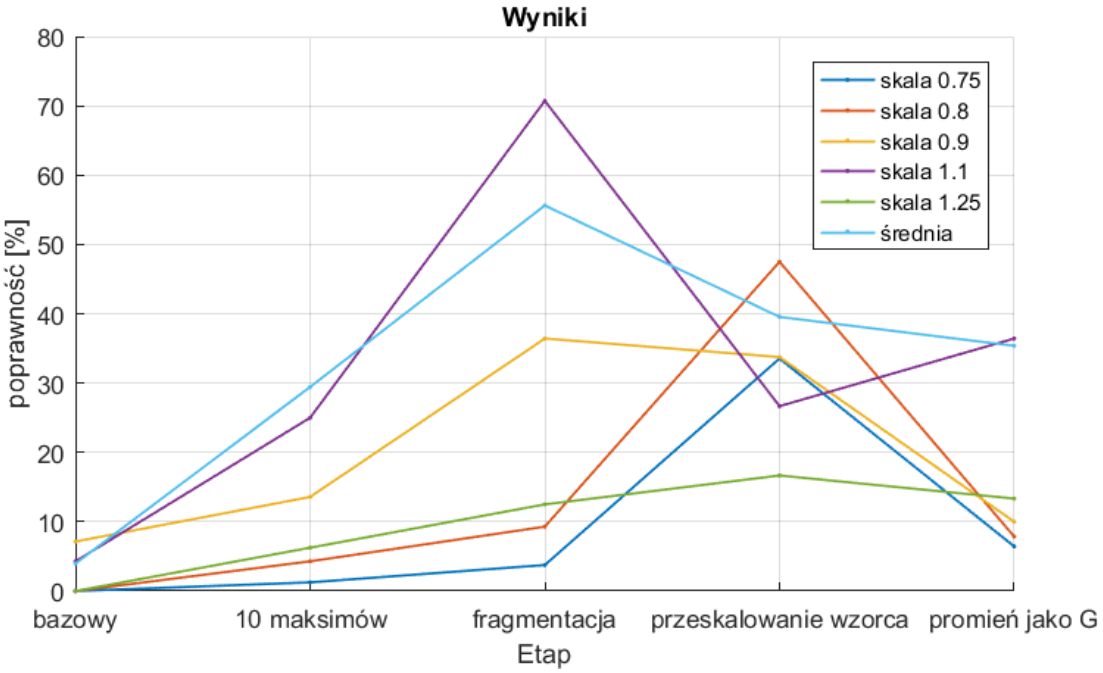
Najlepszym podsumowaniem przeprowadzonych badań jest zbiorowy wykres przedstawiający progres działania algorytmu po dokonaniu kolejnych modyfikacji. W zależności od przypadku testowego przedstawia przebieg zmieniającej się poprawności lokalizacji wzorca na obrazie referencyjnym.

Na rysunku 4.1 zaprezentowano postęp w przypadkach braku i z obrotami. Na osi y znajduje się procentowa poprawność działania algorytmu, a na osi x poszczególne etapy, zawierające kolejne eksperymenty. Są to algorytm bazowy bez wprowadzenia ulepszeń, następnie algorytm wzbogacony o sprawdzanie 10 maksimów, dodanie fragmentacji obrazu wejściowego, zmiana rozmiaru wzorca do rozmiaru obrazu referencyjnego i na końcu, zastosowanie połowy przekątnej jako maksymalnego promienia log-polara.



Rys. 4.1. Zależność poprawności wyników, na różnych etapach rozwoju algorytmu dla obrotów

W pierwszych eksperymentach jakość rezultatów znacznie rosła, niemal jednostajnie. W testach dotyczących obrotów, modyfikacje związane z fragmentacją i maksimami korelacji fazowej doprowadziły do uzyskania 80-100% skuteczności. Niewiele pozostało do ulepszenia. Dalsze eksperymenty miały za zadanie pomóc w poprawnym wyznaczeniu skali, jednak w połowie testów negatywnie wpłynęły na wskazywanie prawidłowych obrotów. Jedynie w przypadku obrotu o 90° uzyskany wynik był najlepszy po dodaniu wszystkich modyfikacji. Analizując przebieg dla braku obrotu można wnioskować, że dalsze eksperymenty poprawiające rozpoznanie skali, ostatecznie nie wpłynęły na wyniki utrzymujące się na poziomie 100%.



Rys. 4.2. Zależność poprawności wyników, na poszczególnych etapach rozwoju algorytmu dla zmiany skali

Obserwując przebiegi dotyczące skalowania obrazu, na pierwszych 3 etapach można dojść do takich samych wniosków. W niektórych przypadkach testowych poprawa była znacznie większa od innych. Po dodaniu do algorytmu zmiany rozmiaru wzorca (resize) 3 z 5 wyników poprawiają się, największy wzrost można zaobserwować przy pomniejszeniu (tj. skali 0.75 oraz 0.8). Zmiana parametru promienia transformacji log-polar z połowy szerokości obrazu na połowę przekątnej, nie przyniosła obiecujących rezultatów, zwiększa liczbę poprawnie wyszukanych wzorców tylko w jednym przypadku.

4.2. Wnioski

W niniejszej pracy wykazano, że transformata Fouriera Mellina ma zastosowanie w poszukiwaniu wzorca na obrazie, na przykładzie obiektów na zdjęciach lotniczych. Zaprezentowane modyfikacje miały znaczący, pozytywny wpływ na otrzymywane wyniki.

Już pierwszy eksperyment polegający na sprawdzaniu kilku lokalnych maksimów korelacji fazowej przyniósł znaczącą poprawę, zwłaszcza przy wyznaczaniu odpowiedniego kąta, a także w niewielkich różnicach skali. W tym drugim przypadku fragmentacja obrazu wejściowego miała również duże znaczenie, podnosząc zarazem poprawność pozostałych wyników. Odpowiedzią na duże błędy w określaniu skali, było zastosowanie funkcji resize w celu powiększenia wzorca do rozmiaru obrazu referencyjnego. Wprowadzenie tej operacji znacznie poprawiło wyniki w rozpoznawaniu skali, jednak w niektórych przypadkach pogorszyło wyznaczanie właściwego kąta. Należy się zastanowić nad tym czy wprowadzać to rozwiązanie, zależnie od potrzeb wykorzystania algorytmu, a więc tego czy większe różnice pomiędzy wzorcem, a obrazem referencyjnym będą dotyczyć skali czy obrotu. To podejście można także

wykorzystać w dalszych eksperymentach i próbach, mających na celu redukcję błędów dotyczących obrotu. W kwestii doboru parametrów transformacji log-polar, wyniki również nie są jednoznaczne, ponieważ wybór innego promienia wpływa różnie na poprawność działania, zależnie od skali. W dalszym rozwoju rozwiązania można dodać sprawdzanie kilku możliwości i wybór tej, która daję największe maksimum korelacji obrazu referencyjnego z przetransformowanym wzorcem.

4.3. Perspektywy kontynuacji badań

Istnieje wiele dalszych możliwości rozwoju badań nad zastosowaniem metody wykorzystującej transformatę Fouriera Mellina, zarówno w poszukiwaniu wzorca jak i pozostałych dziedzinach przetwarzania obrazów, np. „image registration". Warte sprawdzenia są także inne zastosowania algorytmu, np. kontrola jakości, gdzie rozpoznawane elementy mogą być mniej szczegółowe.

W badanym zastosowaniu również można wskazać wiele sposobów dalszego rozwijania metody. Warte uwagi jest zbadanie działania algorytmu w zależności od filtracji i jej parametrów, co nie było wykonywane w obecnej pracy. Innym polem do rozwoju jest implementacja funkcji transformującej do układu log-polar od początku i tym samym, jej dokładna parametryzacja oraz zbadanie wpływu dobranych parametrów na wyniki.

Bazując na przeprowadzonych badaniach, należałoby zastanowić się nad nowymi eksperymentami, mającym na celu poprawienie wyznaczania skali. Wyszukiwanie tego parametru poprawiło się w efekcie wprowadzania nowych modyfikacji, jednak wyniki nadal nie są tak zadowalające jak w przypadku obrotu.

Istnieje także potrzeba akceleracji algorytmu, który obecnie wykonuje się stosunkowo długo dla dużych rozmiarów obrazów wejściowych.

Bibliografia

[1] Zhang, G., Lei, M., and Liu, X., 2009, “Novel Template Matching Method With Sub-Pixel Accuracy Based on Correlation and Fourier-Mellin Transform”, Opt. Eng., 48(5), p. 057001.

[2] David W. Kammler, “A First Course in Fourier Analysis”, Upper Saddle River, NJ: Prentice Hall, 2000,

[3] Tomasz P. Zieliński „Cyfrowe przetwarzanie sygnałów”, 2005.

[4] Weiman, Chaikin, “Logarithmic Spiral Grids for Image Processing and Display, Computer Graphics and Image Processing”, 1979.

[5] V.D.Sharma, “Applications of Two Dimensional Fractional Mellin Transform, International Journal of Scientific and Innovative Mathematical Research,” Vol. 2, Issue 9, September 2014.

[6] Wikipedia contributors. "Phase correlation." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 28 Jan. 2019. Web. 12 Oct. 2019.

[7] https://docs.scipy.org/doc/numpy/reference/generated/numpy.hanning.html

[8] Przemysław Wójtowicz, „Cyfrowe metody powiększania obrazów rastrowych”, 2004.

[9] G. S. Cox and G. de Jager (Member IEEE), “Invariance in Template Matching”, Department of Electrical Engineering University of Cape Town South Africa.

[10] Guo, Xiaoxin & Xu, Zhiwen & lu, Yinan & Pang, Yunjie. “An Application of Fourier-Mellin Transform in Image Registration”. Proceedings - Fifth International Conference on Computer and Information Technology, CIT 2005. 2005. 619 - 623. 10.1109/CIT.2005.62.

[11] Yunlong Sheng and Henri H. Arsenault, “Experiments on pattern recognition using invariant Fourier–Mellin descriptors”.

[12] R. M. Dufour, E. L. Miller, and N. P. Galatsanos, "Template Matching Based Object Recognition with Unknown Geometric Parameters," IEEE Transactions on Image Processing, Vol. 11, No. 12, pp. 1385-1396, 2002.

[13] Jianxin Zhang, Zongying Ou, Honglei Wei, “Fingerprint Matching Using Phase-Only Correlation and Fourier-Mellin Transforms”. Publisher: IEEE.

[14] https://docs.opencv.org/trunk/da/d54/group__imgproc__transform.html 12.10.2019

[15] „Optyka Fourierowska. Filtracja przestrzenna". Politechnika Warszawska, WF.

[16] D. Casasent and D. Psaltis, “Scale invariant optical correlation using Mellin transforms," Optical Communications, vol. 17, pp. 59-63, Apr. 1976.

[17] Z. Ouyang, J. Feng, F. Su, A. Cai, "Fingerprint Matching with Rotation-Descriptor Texture Features", Proc. 18th Int'l Conf. on Pattern Recognition (ICPR'06), vol. 4, pp. 417-420, 2006.

[18] Ryszard Tadeusiewicz „Komputerowa analiza. Przetwarzanie obrazów”, 1997.

[19] Przemysław Skurowski, „Analiza możliwości wykorzystania liniowych przekształceń geometrycznych w kompresji sekwencji wizyjnych”, Politechnika Śląska, Instytut Informatyki, „Studia Informatica”, Vol. 31, No. 1, 2010.

Spis rysunków

Spis tabeli