

# Algorytmy korejestracji przestrzennej dla angiograficznego obrazowania w optycznej tomografii koherencyjnej

---

Aleksander Grzyb

2015



Politechnika Poznańska



---

**POLITECHNIKA POZNAŃSKA**

---

Wydział Informatyki

**Algorytmy korejestracji przestrzennej dla  
angiograficznego obrazowania w optycznej  
tomografii koherencyjnej**

Aleksander Grzyb

Praca zrealizowana w ramach projektu „Rozwój interferometrycznych metod optycznych do badania dynamiki układów biologicznych” (RIMO-BIOL), NCBiR  
PBS1/A9/20/2013.

*Promotor* dr hab. inż. Krzysztof Krawiec

2015

**Aleksander Grzyb**

*Algorytmy korejestracji przestrzennej dla angiograficznego obrazowania w optycznej tomografii koherencyjnej*

v 2015

Promotor: dr hab. inż. Krzysztof Krawiec

**Politechnika Poznańska**

Wydział Informatyki

Poznań

## Podziękowania

Składam serdeczne podziękowania dla mojego promotora, profesora Krzysztofa Krawca za wsparcie, inspirację oraz poświęcony czas w trakcie realizacji niniejszej pracy. Również chciałbym podziękować doktorowi Danielowi Rumińskiemu za owocną współpracę oraz wsparcie merytoryczne.



# Spis treści

<b>1 Wstęp</b>	<b>1</b>
1.1 Opis problemu . . . . .	1
1.1.1 Projekt RIMO-BIOL . . . . .	2
1.1.2 Cele szczegółowe . . . . .	2
1.2 Organizacja pracy . . . . .	3
<b>2 Obrazowanie OCT</b>	<b>5</b>
2.1 Zasada działania OCT . . . . .	6
2.1.1 Metoda uzyskania trójwymiarowego obrazu tkanki . . . . .	7
2.1.2 OCT w domenie częstotliwości . . . . .	8
2.2 Angiografia OCT . . . . .	9
2.2.1 Sposób powstania obrazu angiograficznego . . . . .	9
2.3 Zastosowania OCT . . . . .	11
2.3.1 OCT w okulistyczce . . . . .	11
2.3.2 OCT w gastroenterologii i dermatologii . . . . .	11
2.3.3 OCT w przemyśle . . . . .	11
<b>3 Algorytmy korejestracji przestrzennej obrazów OCT</b>	<b>13</b>
3.1 Model deformacji kafelków . . . . .	14
3.1.1 Matematyczny zapis modelu transformacji ciała sztywnego . .	15
3.2 Korejestracja kafelków . . . . .	15
3.2.1 Korejestracja na podstawie wartości pikseli . . . . .	17
3.2.2 Korejestracja na podstawie cech . . . . .	18
3.3 Łączenie kafelków . . . . .	21
<b>4 Proponowane algorytmy</b>	<b>23</b>
4.1 Wiedza dziedzinowa . . . . .	23
4.2 Rejestracja kafelków z wykorzystaniem cech . . . . .	23
4.2.1 Algorytm SIFT . . . . .	24
4.2.2 Dopasowanie wykrytych cech . . . . .	30
4.2.3 Filtrowanie dopasowań . . . . .	30
4.3 Rejestracja kafelków poprzez wykrycie położen naczyń krwionośnych w kafelkach . . . . .	37
4.3.1 Zasada działania . . . . .	38

4.4	Estymacja macierzy transformacji pomiędzy kafelkami . . . . .	42
4.4.1	Proces decyzyjny . . . . .	42
4.4.2	Estymacja macierzy transformacji na podstawie wykrytych naczyń krwionośnych . . . . .	43
4.4.3	Estymacja macierzy transformacji z wykorzystaniem metody OpenCV na podstawie cech SIFT . . . . .	45
4.4.4	Estymacja macierzy transformacji na podstawie średnich różnic pomiędzy cechami SIFT . . . . .	45
4.4.5	Estymacja macierzy transformacji na podstawie wiedzy dziedzinowej . . . . .	45
4.5	Globalna rejestracja kafelków . . . . .	46
4.6	Łączenie kafelków . . . . .	47
<b>5</b>	<b>Oprogramowanie</b>	<b>49</b>
5.1	Plik konfiguracyjny . . . . .	49
5.2	Dokumentacja . . . . .	50
5.3	OpenCV . . . . .	50
<b>6</b>	<b>Wyniki</b>	<b>51</b>
6.1	Zbiór 1 . . . . .	51
6.2	Zbiór 2 . . . . .	51
6.3	Zbiór 3 . . . . .	51
6.4	Ocena wyników . . . . .	52
6.4.1	Ocena wizualna . . . . .	52
6.4.2	Ocena na podstawie średniego odchylenia standardowego . .	53
6.4.3	Ocena na podstawie czasu wykonywania . . . . .	55
6.4.4	Podsumowanie . . . . .	56
<b>7</b>	<b>Podsumowanie i wnioski końcowe</b>	<b>59</b>
7.1	Napotkane trudności . . . . .	59
7.2	Möżliwości rozwoju . . . . .	60
<b>Bibliografia</b>		<b>61</b>

# Wstęp

Gwałtowny rozwój technologii w ostatnim wieku ma ogromny wpływ na metody diagnostyczne w medycynie. Dzięki nowym technologiom takim jak optyczna tomografia koherencyjna (ang. *optical coherence tomography, OCT*, sekcja 2) lekarze są w stanie zastąpić inwazyjne metody diagnostyczne, których przeprowadzenie stanowi ryzyko dla zdrowia pacjenta. Postęp w dziedzinie informatyki, a w szczególności algorytmów przetwarzania obrazów przyczynia się do wzrostu jakości obrazów diagnostycznych przez co lekarze są w stanie rozpoznać choroby skuteczniej oraz szybciej.

## 1.1 Opis problemu

Jednym z najbardziej popularnych zastosowań optycznej tomografii koherencyjnej jest badanie siatkówki oka. Wykorzystanie OCT pozwala zastąpić inwazyjną metodę diagnozy siatkówki polegającej na wstrzyknięciu środka kontrastowego do krewobiegu pacjenta, a następnie wykorzystaniu jednej z metod obrazowania opierającej się na promieniach rentgenowskich. Użycie środka kontrastowego wiąże się z ryzykiem szkodliwych powikłań na zdrowiu pacjenta tj. reakcje alergiczne, czy bóle głowy. Optyczna tomografia koherencyjna dzięki swojej nieinwazyjnej charakterystyce w badaniu siatkówki jest mocno rozwijana przez wiele ośrodków laboratoryjnych na całym świecie (jednym z przedsięwzięć zajmujących się doskonaleniem OCT jest projekt RIMO-BIOL opisany dokładnie w sekcji 1.1.1, w którego skład wchodzi niniejsza praca). Wynikiem przeprowadzenia optycznej tomografii koherencyjnej są trójwymiarowe obrazy oddające strukturę siatkówki oka (prawy obraz na rysunku 2.4). Na podstawie trójwymiarowych obrazów są tworzone angiograficzne obrazy *en face* przedstawiające przebieg naczyń krwionośnych (rysunek 2.2). Każdy obraz angiograficzny przedstawia fragment siatkówki oka.

**Celem niniejszej pracy** jest zaimplementowanie algorytmu, który poprzez pobranie angiograficznych obrazów siatkówki oka wraz z informacją o ich wzajemnych relacjach przestrzennych ma za zadanie automatycznie stworzyć jednolitą mozaikę przedstawiającą całość siatkówki oka. Potrzeba na stworzenie autorskiego algorytmu powstała ze względu na to, że dostępne metody tworzenia mozaiki (ang. *mosaic stitching*) nie dają zadowalających rezultatów ze względu na charakterystykę angiograficznych obrazów OCT.

### 1.1.1 Projekt RIMO-BIOL

**Projekt RIMO-BIOL** po tytulem „Rozwój interferometrycznych metod optycznych do badania dynamiki układów biologicznych”, w którego skład wchodzi niniejsza praca skupia się na rozwoju mikroskopowych metod optycznych w celu badania dynamiki układów biologicznych. Projekt jest przeprowadzany przez konsorcjum składające się z m. in. Uniwersytetu Mikołaja Kopernika w Toruniu i Politechniki Poznańskiej w Poznaniu. Praca jest finansowana przez Narodowe Centrum Badań i Rozwoju w ramach pierwszej edycji konkursu „Program Badań Stosowanych”.

RIMO-BIOL pracuje obecnie nad rozwojem optycznej tomografii koherencyjnej oraz sprzętu jej wykonującej. Wszystkie obrazy wykorzystywane w niniejszej pracy pochodzą z aparatury OCT zbudowanej w ramach projektu. Stworzona technologia w ramach RIMO-BIOL testowana jest w badaniach:

1. Siatkówki oka ludzkiego w celu diagnozy cukrzycy na podstawie struktur mikronacyjnych w siatkówce oka.
2. Mózgu myszy w celu analizy przebiegu udaru mózgu na podstawie oceny morfologii i przepływu krwi w sieci mikronacyjnej mózgu.
3. Komórek jajowych na podstawie pomiaru ruchu cytoplazmy.

### 1.1.2 Cele szczegółowe

Wynikiem niniejszej pracy ma być oprogramowanie realizujące cel niniejszej pracy opisany w 1.1, natomiast do celów szczegółowych można zaliczyć:

1. Rozpoznanie możliwości bibliotek przetwarzania i analizy obrazów w zakresie korejestracji przestrzennej.
2. Wybór bibliotek i środowiska deweloperskiego.
3. Projekt, implementacja i przetestowanie algorytmów przetwarzania obrazów realizujących cel niniejszej pracy.
4. Stworzenie w pełni funkcjonalnego oraz prostego w użyciu programu komputerowego o nazwie `mostitch`.
5. Przygotowanie dokumentacji technicznej oprogramowania.

## 1.2 Organizacja pracy

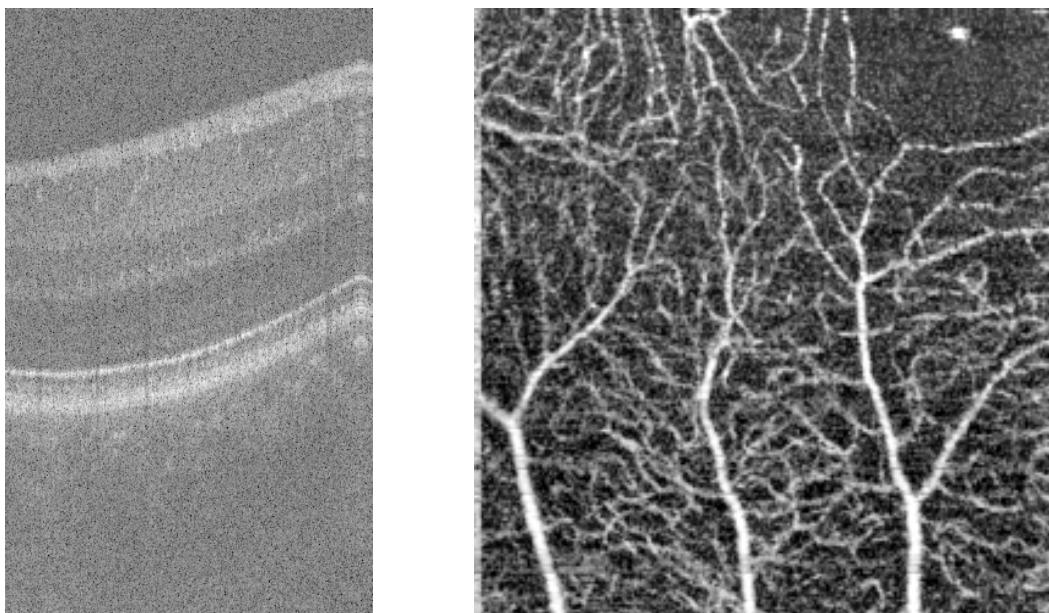
Praca zaczyna się od przedstawienia w rozdziale 2 technologii optycznej tomografii komputerowej oraz wyjaśnienia zasady jej działania. Następnie w rozdziale 3 opisano dostępne metody oraz poszczególne kroki niezbędne do stworzenia mozaiki (ang. *mosaic stitching*) w kontekście angiograficznych obrazów OCT. Kolejno w rozdziale 4 szczegółowo opisano zasadę działania wybranych oraz autorskich algorytmów realizujących cel niniejszej pracy. W rozdziale 5 zaprezentowano stworzone oprogramowanie, zasadę jego użycia oraz wykorzystane biblioteki. W rozdziale 6 przedstawiono wynikowe mozaiki powstałe z przykładowych angiograficznych obrazów OCT. Na koniec w rozdziale 7 opisano trudności napotkane podczas pracy oraz możliwy rozwój oprogramowania w przyszłości.



## Obrazowanie OCT

**Optyczna tomografia koherencyjna** (ang. *optical coherence tomography, OCT*) jest metodą umożliwiającą nieinwazyjne oraz *in vivo* przechwycenie obrazu wnętrza tkanki biologicznej. Zasada działania OCT opiera się na wykorzystaniu fal świetlnych, dzięki czemu rozdzielcość obrazów jest o wiele wyższa niż w ultrasonografii (wykorzystanie fal dźwiękowych), czy rezonansie magnetycznym (wykorzystanie pola magnetycznego). Następnym powodem dużej popularności OCT w medycynie jest bezkontaktowe badanie pacjenta oraz brak wymogu przygotowania pacjenta przed badaniem.

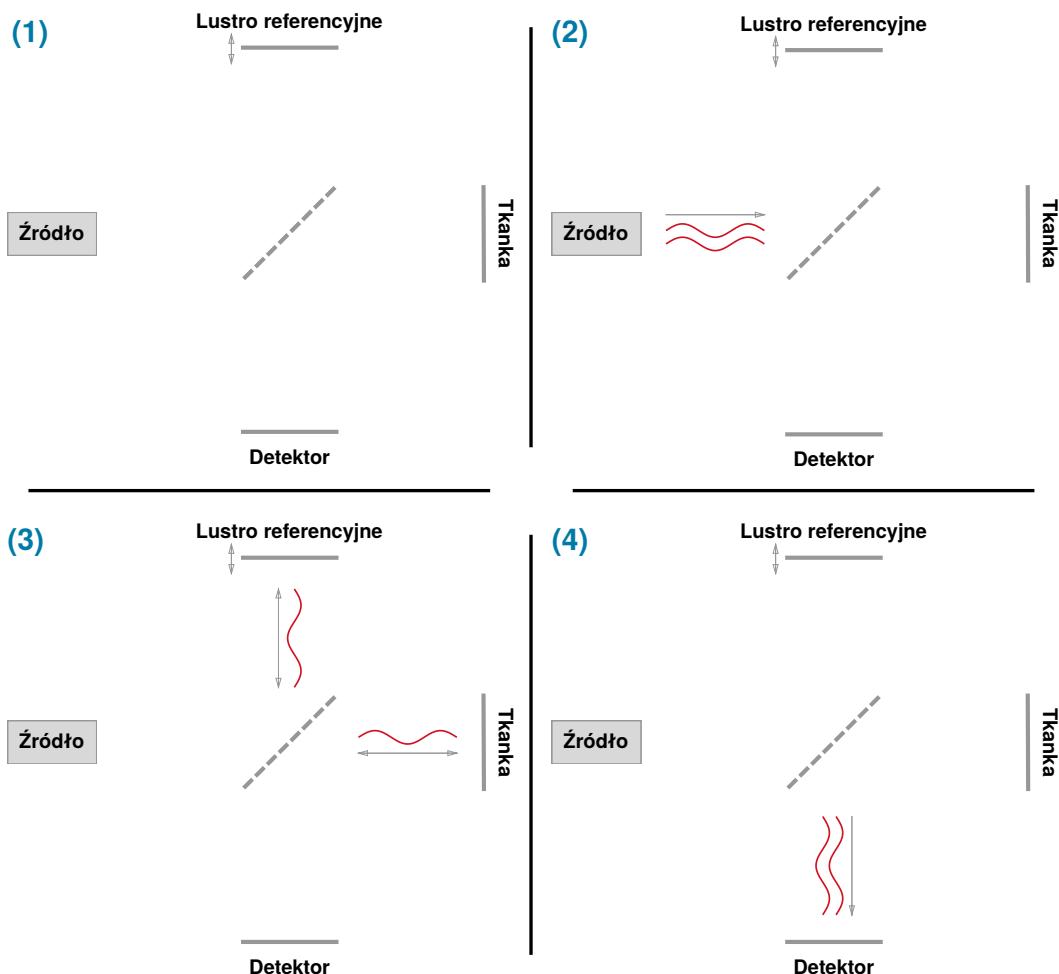
W projekcie RIMO-BIOL (sekcja 1.1.1) OCT wykorzystano do uzyskania szczegółowych obrazów naczyń krwionośnych siatkówki oka. Rysunek 2.1 przedstawia przykładowe obrazy siatkówki oka uzyskane poprzez wykorzystanie OCT (prawy obraz to angiografia siatkówki oka, która jest obrazem wejściowym do algorytmów omawianych w niniejszej pracy). Sposób powstania angiografii z danych OCT przedstawiono w sekcji 2.2, natomiast ogólną zasadę działania OCT zaprezentowano w sekcji 2.1. Na koniec rozdziału w sekcji 2.3 opisano zastosowania OCT.



**Rys. 2.1:** Lewy obraz: Dwuwymiarowy przekrój siatkówki oka (B-skan). Obraz został uzyskany poprzez połączenie jednowymiarowych A-skanów, które zawierają informację o strukturze tkanki w głąb siatkówki oka. Prawy obraz: Angiografia siatkówki oka uzyskana poprzez przetworzenie danych z OCT.

## 2.1 Zasada działania OCT

Optyczna tomografia koherencyjna przechwytuje obrazy wnętrza tkanki poprzez wykorzystanie fal świetlnych. OCT za pomocą generatora wytwarza falę świetlną, która jest skierowana na tkankę pacjenta. Następnie po odbiciu fali poprzez tkankę wiązka jest przechwycona przez detektor. Jedną z dostępnych metod, która umożliwiłaby zlokalizowanie miejsca odbicia fali byłoby zmierzenie czasu pomiędzy wygenerowaniem fali, a zarejestrowaniem jej przez detektor (mechanizm stosowany np. w ultrasonografii z wykorzystaniem fal dźwiękowych), natomiast prędkość światła ( $3 \times 10^8 \frac{m}{s}$ ) wyklucza zastosowanie tego mechanizmu. Zjawisko, które umożliwia dokładne zlokalizowanie miejsca odbicia to **interferencja światła o niskiej spójności**.

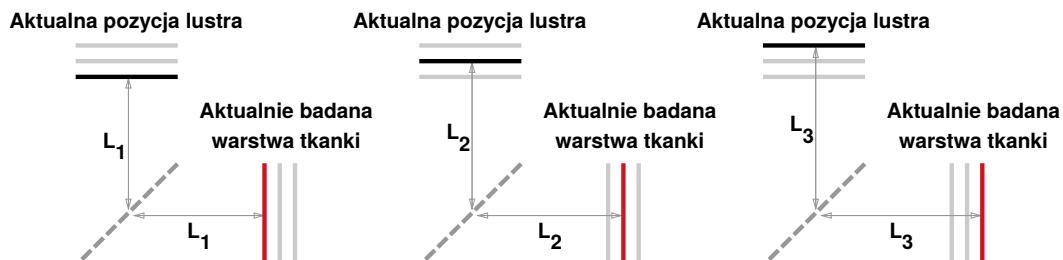


**Rys. 2.2:** Kolejne etapy działania metody OCT. (1) - Etap początkowy. (2) - Źródło wyemitowało wiązkę światła. (3) - Fala rozdzieliła się za pomocą interferometru na wiązkę referencyjną (skierowaną na lustro referencyjne) oraz na wiązkę próbki (skierowaną na tkankę). (4) - Wiązki po odbiciu od lustra referencyjnego i tkanki ponownie łączą się za pomocą interferometru. W tej części występuje zjawisko interferencji, które jest zarejestrowane przez detektor.

Rysunek 2.2 składa się z bardzo uproszczonych schematów obrazujących kolejne etapy działania metody. Schematy na rysunku 2.2 składają się z pięciu elementów:

- **Źródła** - Źródło (np. dioda superluminescencyjna) światła podczerwonego będąca falą o niskiej spójności.
- **Rozdzielacza wiązek** (na rysunku 2.2 przedstawiony za pomocą przerywanych linii na środku każdego schematu) - Interferometr (np. Michelsona) umożliwiający rozdzielenie fali na dwie wiązki oraz następne ich połączenie.
- **Lustro referencyjne** - Lustro, które odbija wiązkę referencyjną. Posiada możliwość oddalania oraz przybliżania się względem interferometru.
- **Tkanka** - Badana tkanka odbijająca wiązkę próbki.
- **Detektor** - Rejestruje zjawisko interferencji związek.

Najbardziej istotnym etapem wymaganym do zrozumienia mechanizmu OCT jest etap (4) pokazany na rysunku 2.2. W tym kroku wiązka referencyjna i wiązka próbki łączą się i zachodzi zjawisko interferencji. Dzięki temu, że wiązki są falami o niskiej spójności interferencja zachodzi tylko na małej długości zwanej *coherence length*. Odkrywając za pomocą detektora charakterystyczny wzorzec interferencji występujący na *coherence length* OCT jest w stanie precyzyjnie odczytać informację o strukturze wnętrza tkanki. Badanie warstw tkanki o różnej głębokości przedstawia rysunek 2.3.

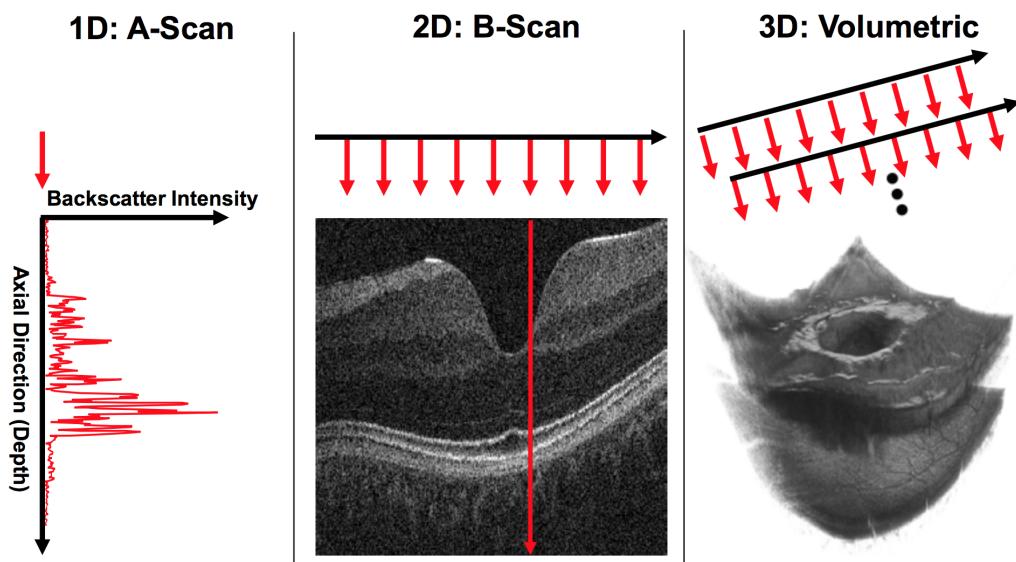


**Rys. 2.3:** Kolejne etapy badania głębszych warstw tkanki dzięki przesuwaniu lustra referencyjnego. Aktualna pozycja lustra referencyjnego jest zaznaczona kolorem czarnym, natomiast aktualnie badana warstwa tkanki jest zaznaczona kolorem czerwonym.

### 2.1.1 Metoda uzyskania trójwymiarowego obrazu tkanki

Poprzez poruszenie lustra referencyjnego pomiary przeprowadzane są w głąb tkanki (wzdłuż osi Z). Zbiór pomiarów w głąb tkanki nazywa się A-skanem (obraz jednowymiarowy). Powtarzając ten proces w osi X lub Y i następnie poprzez połączenie sąsiadujących A-skanów otrzymuje się przekrój tkanki zwany B-skanem (obraz dwuwymiarowy). Proces uzyskania B-skanów można powtórzyć dla sąsiadujących

przekrojów. Poprzez połączenie otrzymanych B-skanów otrzymuje się trójwymiarowy obraz tkanki. Na rysunku 2.4 przedstawione są poszczególne skany.



**Rys. 2.4:** Lewy obraz: Pojedynczy A-skan wykonany w głąb tkanki. Środkowy obraz: Otrzymany B-skan poprzez połączenie A-skanów. Prawy obraz: Trójwymiarowy obraz tkanki stworzony poprzez połączenie B-skanów (obraz pochodzi z pracy [10]).

### 2.1.2 OCT w domenie częstotliwości

Metoda OCT wykonująca ruchy lustrem referencyjnym jest zwana metodą OCT w domenie czasu (ang. *time-domain OCT, TdOCT*). Alternatywną oraz nowszą metodą od TdOCT jest OCT w domenie częstotliwości (ang. *Fourier-domain OCT, FdOCT*). FdOCT umożliwia 100 razy szybsze [@17] skanowanie w porównaniu do TdOCT. TdOCT jest w stanie wykonać ok. 400 A-skanów w przeciągu sekundy, natomiast FdOCT jest ich w stanie wykonać dziesiątki tysięcy. Szybsze skanowanie poprawia również jakość skanów ze względu na to, że pacjent ma mniejszą szansę poruszenia okiem podczas skanowania (ruch oka w trakcie skanowania przyczynia się do powstania artefaktów ruchu na obrazach OCT). Oprócz poprawy szybkości skanowania FdOCT ma wyższą rozdzielcość w przedziale od 3 do 7 mikrometrów [@17]. Jest to poprawa względem TdOCT o 8-10 mikrometrów.

Większa szybkość oraz rozdzielcość FdOCT w porównaniu do TdOCT jest możliwa dzięki dwóm modyfikacjom technicznym:

1. FdOCT, jako źródło światła, wykorzystuje laser o wysokiej szerokości pasma, co znacząco zwiększa rozdzielcość.
2. FdOCT, jako detektor, wykorzystuje spektrometr, który przeprowadza analizę widma fali (połączona wiązka referencyjna i próbki), która dotarła do spektro-

metru z interferometru. Transformata Fouriera na widmie fali tworzy A-skan tkanki. Dzięki tej technice FdOCT wyeliminowało potrzebę ruszania lustrem referencyjnym, co znacząco zwiększa szybkość skanowania.

## 2.2 Angiografia OCT

Angiografia to technika służąca do wizualizacji naczyń krwionośnych i organów ciała. W większości przypadków w medycynie angiografia jest wykonywana poprzez wstrzyknięcie pacjentowi środka kontrastowego, który nie przepuszcza promieni rentgenowskich. Następnie, by stworzyć obrazy angiograficzne, wykorzystuje się jedną z technik obrazowania (np. fluoroskopię) opartą o promienie rentgenowskie. Metoda oparta na stosowaniu środka kontrastowego ma jedną znaczącą wadę, jako technika inwazyjna może prowadzić do reakcji alergicznych na środek kontrastowy oraz jest przeciwwskazana kobietom w ciąży i dzieciom. Z tego powodu nieustannie poszukiwane są metody, które jednocześnie są nieinwazyjne i tworzą obrazy o jakości porównywalnej do metod inwazyjnych.

Angiografia OCT poprzez czasową analizę informacji tomograficznej OCT jest w stanie odtworzyć trójwymiarową informację o strukturze naczyń krwionośnych siatkówki oka. Przykład takiego obrazu znajduje się na rysunku 2.2.

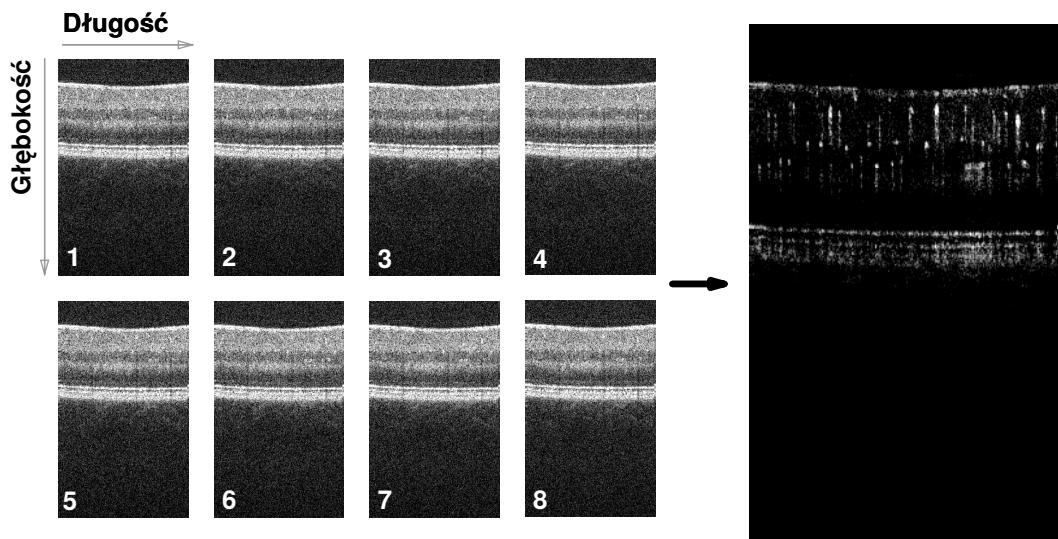
### 2.2.1 Sposób powstania obrazu angiograficznego

Obrazy angiograficzne użyte w niniejszej pracy powstały za pomocą metody zwanej *Speckle Variance* polegającej na liczeniu wariancji dla każdego piksela z grupy B-skanów zmierzonych w tym samym miejscu w  $N$  kolejnych chwilach czasu. Wartość wariancji w obszarach gdzie występuje przepływ krwi (naczynia krewionośne) jest lokalnie wyższa od obszarów, w których znajduje się struktura statyczna. Rezultatem tej metody jest obraz angiograficzny. Na rysunku 2.5 został przedstawiony przykładowy zbiór B-skanów zmierzonych w tym samym miejscu w kolejnych chwilach czasu.

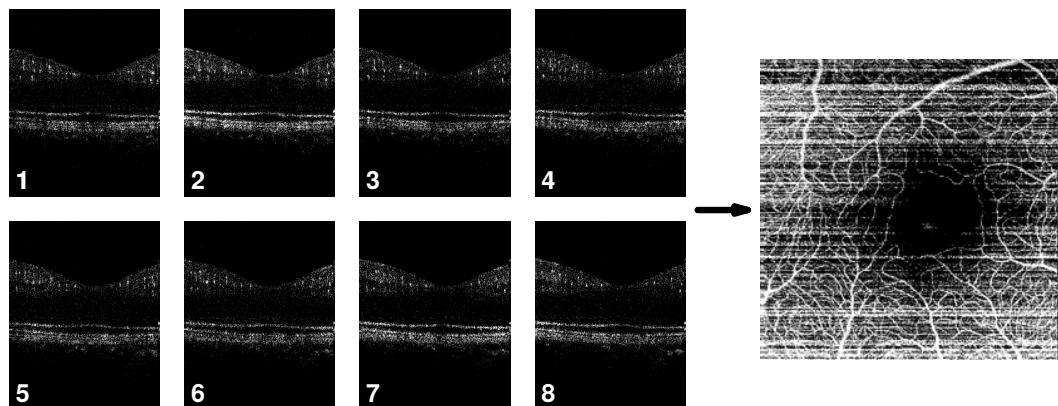
Wartości pikseli  $V_{jk}$  obrazu przepływowego liczone są na podstawie wartości pikseli  $I_{ijk}$  z grupy  $N$  B-skanów za pomocą wzoru (na rysunku 2.5  $N = 8$ ):

$$V_{jk} = \frac{1}{N} \sum_{i=1}^N (I_{ijk} - I_{mean})^2 \quad (2.1)$$

gdzie  $j$  i  $k$  to współrzędne przestrzenne B-skanu (numery pikseli), a  $I_{mean}$  to średnia natężenie grupy pikseli, z których liczona jest wariancja.



**Rys. 2.5:** Z lewej strony przedstawiono osiem B-skanów zmierzonych w kolejnych chwilach czasu na postawie których powstaje jeden obraz przepływowego pokazany z prawej strony.



**Rys. 2.6:** Z lewej strony znajduje się fragment zbioru sąsiadujących obrazów przepłybowych na podstawie których powstaje jeden obraz *en face* pokazany z prawej strony (obraz pochodzi z pracy [15]).

Rezultatem połączenia dwuwymiarowych sąsiadujących obrazów przepłybowych jest trójwymiarowy obraz przepływowego, który poddaje się projekcji maksymalnego natężenia (ang. *maximum intensity projection, MIP*) polegającej na projekcji woksela o najwyższej intensywności z obrazu 3D na obraz 2D. Po zastosowaniu projekcji maksymalnego natężenia z trójwymiarowego zestawu danych przepłybowych otrzymujemy angiograficzny obraz *en face* naczyń krwionośnych. Rysunek 2.6 przedstawia część z obrazów przepłybowych, na podstawie których stworzono jeden obraz *en face*.

## 2.3 Zastosowania OCT

Optyczna tomografia koherencyjna ze względu na swoje właściwości (badanie nieważne oraz *in vivo*) jest metodą, która ma szerokie zastosowanie w medycynie oraz w innych specjalizacjach.

### 2.3.1 OCT w okulistyczce

Najbardziej popularnym zastosowaniem OCT w medycynie jest badanie oka [7]. Technika OCT umożliwia przechwycenie trójwymiarowych obrazów części oka takich jak dno, czy warstwy przednie. Obrazy wykorzystywane są do diagnozowania takich chorób jak stwardnienie rozsiane, zwyrodnienie plamki żółtej, czy jaskra.

### 2.3.2 OCT w gastroenterologii i dermatologii

OCT w porównaniu do innych metod diagnostycznych w medycynie jest techniką nową. Lekarze i naukowcy nieustannie starają się znaleźć nowe zastosowania dla OCT, która jest metodą obiecującą i szybko rozwijającą się. OCT jest potencjalnym kandydatem by w niektórych diagnozach zastąpić konwencjonalną biopsję wymagającą usunięcia kawałka tkanki z organizmu. Przykładowe zastosowania OCT:

- **Badanie struktury błon śluzowych i podśluzowych w układzie pokarmowym.** OCT dostarczyło czyste obrazy dostarczające lekarzom dużo diagnostycznej informacji [14].
- **Diagnoza raka skóry** (obecnie diagnozowany jest poprzez biopsję). Niestety OCT w obecnym stopniu zaawansowania nie jest w stanie dostarczyć na tyle dokładnych danych by stać się jedyną metodą diagnozy.
- **Diagnoza zapalnych chorób skóry** [9].

### 2.3.3 OCT w przemyśle

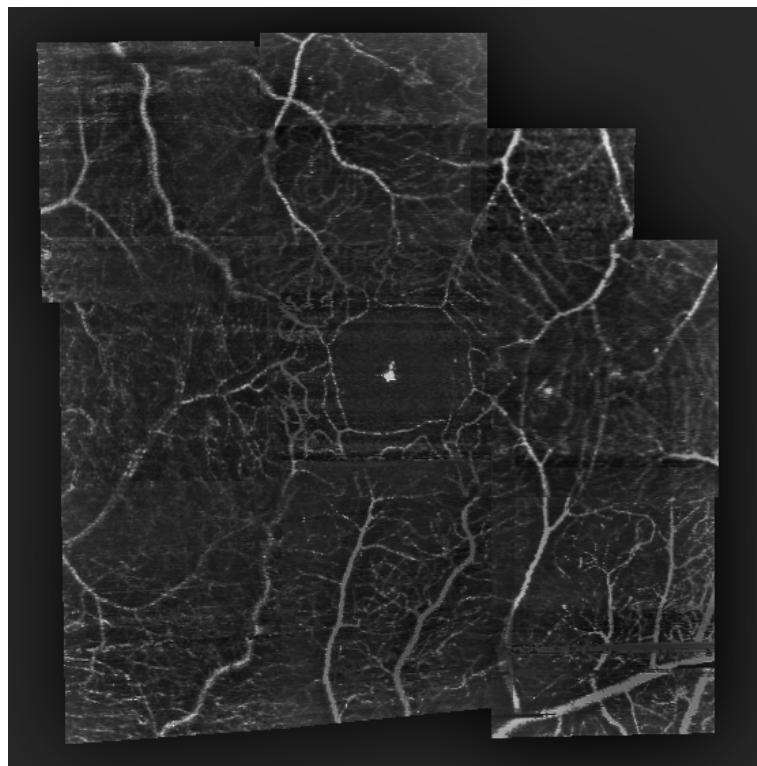
OCT wykorzystywane jest również w przemyśle. Umożliwia badanie np. grubości materiałów [20], czy badanie grubości warstwy pancerza tabletek podczas ich produkcji w przemyśle farmaceutycznym [13].



## Algorytmy korejestracji przestrzennej obrazów OCT

**Mozaiką** nazywa się obraz powstały poprzez połączenie grupy obrazów zwanych w żargonie „kafelkami” na podstawie ich wzajemnych relacji. Znanym oraz popularnym przykładem łączenia obrazów w jeden większy jest funkcja panoramy w telefonach komórkowych czy aparatach fotograficznych. Od strony użytkownika proces tworzenia panoramy polega na powolnym przesuwaniu telefonem po linii poziomej do momentu aż żądany krajobraz zostanie uchwycony. Od strony urządzenia proces polega na wykonywaniu serii zdjęć oraz następnie łączenie nachodzących klatek w jeden obraz. Rezultatem jest jednolita panorama, która składa się z grupy mniejszych, węższych zdjęć.

Celem niniejszej pracy jest stworzenie mozaiki OCT (przykład na rysunku 3.1) z połączenia mniejszych nachodzących na siebie nawzajem angiograficznych obrazów OCT (przykład obrazu angiograficznego OCT znajduje się z prawej strony na rysunku 2.1).

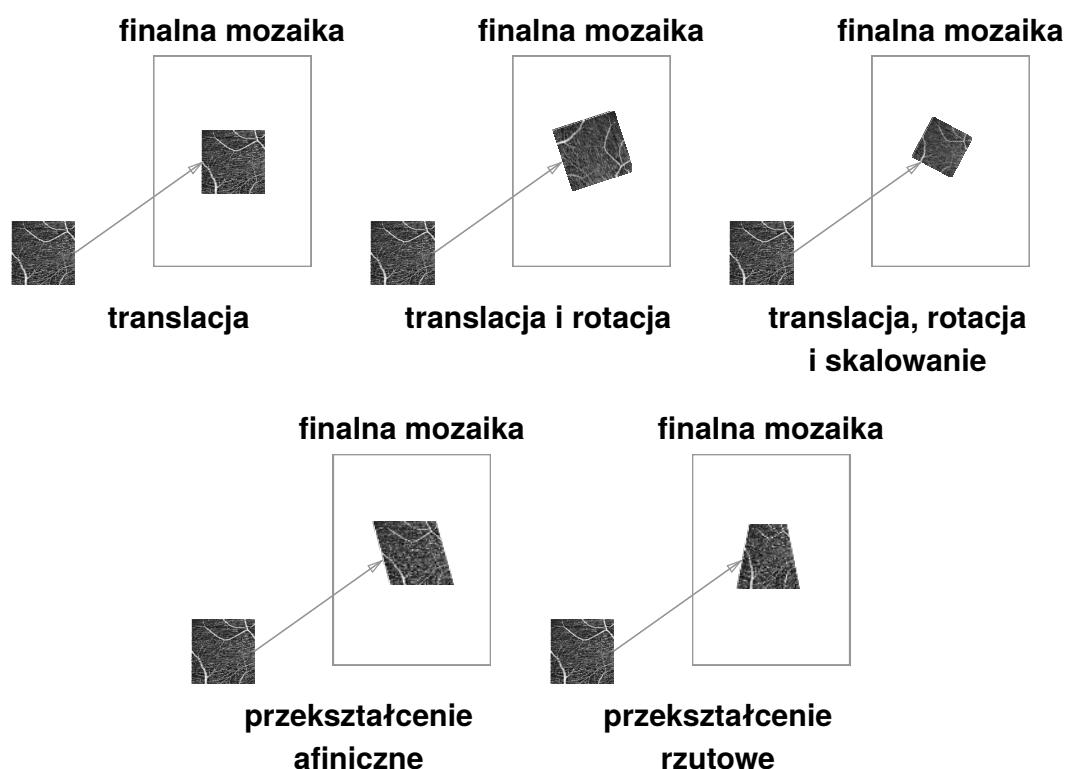


Rys. 3.1: Mozaika OCT stworzona z połączenia angiograficznych obrazów OCT.

Proces automatycznego stworzenia mozaiki (ang. *stitching*) takiej jak na rysunku 3.1 jest zadaniem nietrywialnym i wymaga dokładnej analizy wiedzy dziedzinowej oraz precyzyjnego wyboru metod przetwarzania obrazów. Pierwszym krokiem jest wybór modelu deformacji kafelków (sekcja 3.1), następnym etapem jest wybór metody wzajemnej korejestracji kafelków (sekcja 3.2). Posiadając zdefiniowane wzajemne relacje kafelków oraz ich docelowe położenie w finalnej mozaice należy wykonać proces łączenia kafelków (sekcja 3.3). W każdej z tych sekcji jest wyjaśniona idea metody w kontekście stworzenia mozaiki OCT, natomiast szczegółowy opis zaimplementowanych metod znajduje się w rozdziale 4.

### 3.1 Model deformacji kafelków

Model deformacji kafelków określa dozwolone przekształcenia geometryczne odwzorowujące piksele kafelka do pikseli kafelka w finalnej mozaice. Ze względu na to, że angiograficzne obrazy OCT znajdują się na jednej płaszczyźnie<sup>1</sup> możliwy zbiór modeli deformacji ogranicza się do transformacji dwuwymiarowych przedstawionych na rysunku 3.2.



**Rys. 3.2:** Zbiór transformacji dwuwymiarowych dla przykładowego angiograficznego obrazu OCT.

<sup>1</sup>W rzeczywistości powierzchnia siatkówki jest wklęsła. Uproszenie polegające na przedstawieniu mozaiki na płaszczyźnie znacznie ułatwia implementację, a nie powoduje widocznych zniekształceń.

Idealnie OCT powinno tworzyć obrazy angiograficzne, które są względem siebie tylko przesunięte. W rzeczywistości pojawiają się zniekształcenia wynikające z niedokładności urządzenia oraz ruchu oka pacjenta, przez co niektóre kafelki są nieznacznie obrócone względem siebie. Z tego względu wybranym modelem deformacji kafelków został **model transformacji ciała sztywnego**, czyli połączenie translacji i rotacji.

### 3.1.1 Matematyczny zapis modelu transformacji ciała sztywnego

Współrzędne piksela w kafelku to trójelementowy wektor  $\tilde{x} = (x, y, 1)$ , gdzie  $x$  i  $y$  to współrzędne piksela w układzie współrzędnych kafelka. Tak zdefiniowany piksel poddaje się transformacji ciała sztywnego. Po transformacji piksel posiada współrzędne  $\hat{x} = (x', y')$  w układzie współrzędnych finalnej mozaiki. Transformacja ciała sztywnego jest zdefiniowana następująco:

$$\hat{x} = \begin{bmatrix} R & t \end{bmatrix} \tilde{x} \quad (3.1)$$

gdzie

$$R = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \quad \text{i} \quad t = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad (3.2)$$

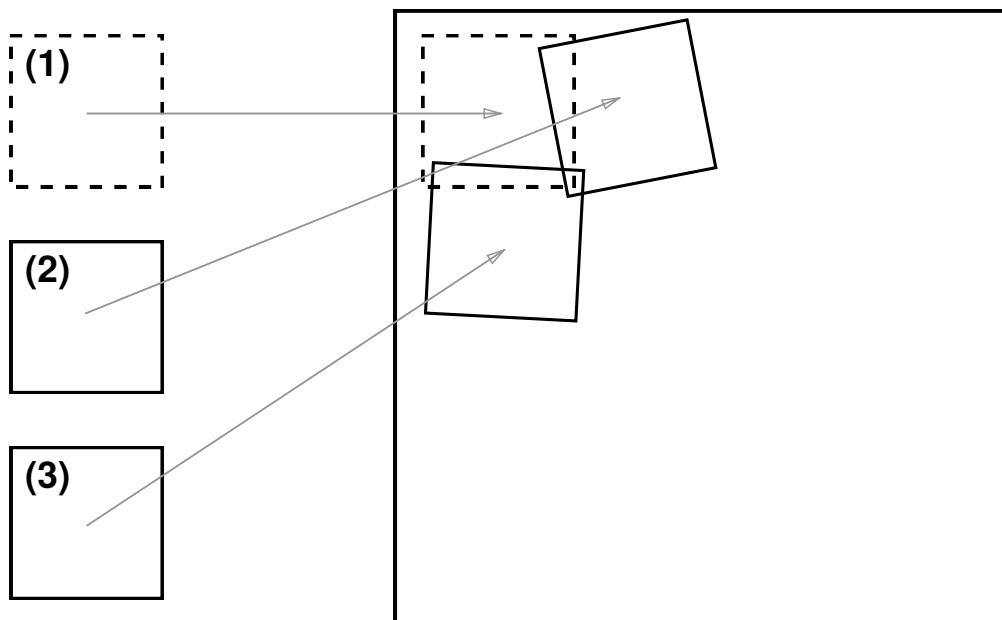
W równaniu 3.2  $\theta$  to kąt obrotu (rotacja) względem początku układu współrzędnych, a  $t_x$  i  $t_y$  to odpowiednio przesunięcia względem osi x i osi y. Parametry  $\theta$ ,  $t_x$  i  $t_y$  są niewiadomymi równania 3.1, których sposób obliczenia przedstawiono w sekcji 3.2.

## 3.2 Korejestracja kafelków

Po wyborze modelu deformacji kafelków można przejść do wyboru metody określającej jego parametry (w przypadku niniejszej pracy są to parametry przesunięcia i obrotu). Metoda powinna zwrócić takie wartości by kafelek znalazł się w odpowiednim miejscu w finalnej mozaice z możliwie najmniejszym błędem. By rozwiązać ten problem najpierw trzeba poznać położenie kafelków względem siebie oraz ustalić kafelek referencyjny. Rysunek 3.3 przedstawia przykładowe rozmieszczenie kafelków. Kafelek referencyjny oznaczony przerywaną linią jest jedynie poddany translacji, natomiast żeby odpowiednio umieścić kafelki (2) i (3) trzeba najpierw znać dopasowanie kafelków (2) i (3) do kafelka referencyjnego (1). W przykładzie na rysunku 3.3 zostało założone, że kafelki (2) i (3) mogą być dopasowane do kafelka (1). Informacja na temat relacji geometrycznych pomiędzy kafelkami wynika z wiedzy dziedzinowej (sekcja 4.1).

## Zbiór kafelków

## Finalna mozaika



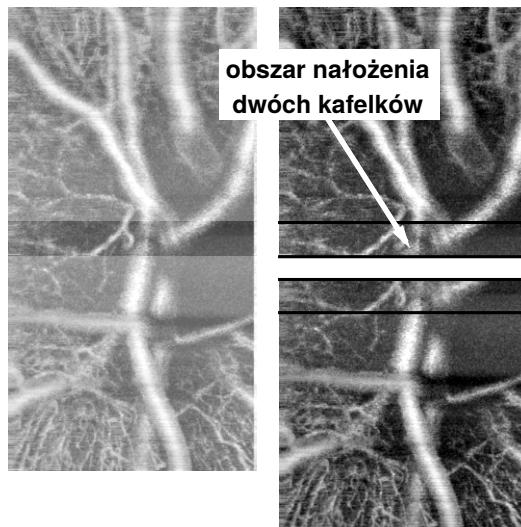
Rys. 3.3: Przykładowe rozmieszczenie kafelków na finalnej mozaice. Kafelek referencyjny jest wyróżniony przerywaną linią.

Dopasowanie dwóch kafelków do siebie nazywa się również ich korejestracją, czyli przeniesieniem dwóch kafelków do wspólnego układu współrzędnych w taki sposób by były względem siebie dopasowane. Prosty przykład korejestracji przedstawiony jest na rysunku 3.4, gdzie kafelki mają wspólny obszar nałożenia.

Dopasowanie kafelków z rysunku 3.4 jest przykładem bardzo prostym, ponieważ wymaga tylko translacji jednego kafelka względem drugiego. W rzeczywistości pomiary OCT nie są tak dokładne. Prosta translacja wzdłuż jednej z osi nie wystarcza, przez co wymagane jest użycie metody automatycznie wyznaczającej parametry translacji i rotacji. Dwie najpopularniejsze metody to:

1. **Korejestracja na podstawie wartości pikseli** (sekcja 3.2.1) - Metoda wielokrotnie przesuwająca jeden obraz względem drugiego i następnie oceniącą każdą z tych unikalnych pozycji pod względem dopasowania obrazów do siebie.
2. **Korejestracja na podstawie cech** (sekcja 3.2.2) - Metoda wykrywająca charakterystyczne cechy z każdego obrazu i następnie dopasowującą te cechy by osiągnąć globalną spójność. Dopasowania są później użyte przez algorytm do wyznaczenia macierzy transformacji pomiędzy obrazami.

**kafelki po procesie korejestracji      kafelki z nakładającym się obszarem**



**Rys. 3.4:** Przykładowa korejestracja dwóch kafelków ze wspólnym obszarem nałożenia.

### 3.2.1 Korejestracja na podstawie wartości pikseli

Korejestracja na podstawie wartości pikseli jest często nazywana metodą bezpośrednią, ponieważ wymaga bezpośredniego porównania wartości pikseli nakładających się obrazów. Pierwszym krokiem w tej metodzie jest wybranie odpowiedniej miary błędu. Taką miarą błędu może być funkcja suma błędów kwadratowych  $E_{RSS}$  (ang. *residual sum of squares, RSS*). Mając daną funkcję  $I_S(p_i)$  zwracającą wartość piksela w obrazie źródłowym (przesuwany) w lokalizacji  $p_i = (x_i, y_i)$  oraz funkcję  $I_T(p_i)$  zwracającą wartość piksela w obrazie docelowym (referencyjnym, pozostającym na miejscu)  $E_{RSS}$  można zdefiniować jako:

$$E_{RSS} = \sum_i [I_S(p_i + u) - I_T(p_i)]^2 = \sum_i e_i^2 \quad (3.3)$$

gdzie  $u = (t_x, t_y)$  to wartość przesunięcia obrazu źródłowego, a  $e_i = I_S(p_i + u) - I_T(p_i)$  to błąd resztowy pomiędzy wartościami pikseli przesuniętego obrazu źródłowego i stacjonarnego obrazu docelowego.

Mając tak zdefiniowaną miarę błędu najprostszą metodą wyznaczającą najlepsze dopasowanie obrazów jest obliczenie błędu dla każdego możliwego przesunięcia obrazu źródłowego. Posiadając wartości wszystkich błędów z odpowiednimi dla nich wartościami przesunięcia można wybrać wartość najmniejszego błędu. Takie podejście natomiast wiąże się z ogromnym kosztem obliczeniowym w przypadku, gdy obrazy są dużej rozdzielczości i jest ich wiele. Proces można przyspieszyć używając algorytm optymalizacji Levenberga-Marquardta [19] albo poprzez wykorzystanie piramid obrazu w metodzie *Hierarchical Motion Estimation* [18].

Ze względu na specyficzny charakter problemu niniejszej pracy zaimplementowano autorską metodę bazującą na bezpośrednim odczytaniu wartości pikseli (sekcja 4.3).

### 3.2.2 Korejestracja na podstawie cech

Korejestracja na podstawie cech jest techniką rozpoczęającą się od detekcji cech z każdego obrazu. Następnie na podstawie odkrytych cech przeprowadza się ich dopasowanie. Na podstawie dopasowań można wyestymować macierz transformacji jednego obrazu względem drugiego. W niniejszej sekcji opisano metody realizujące wymienione kroki.

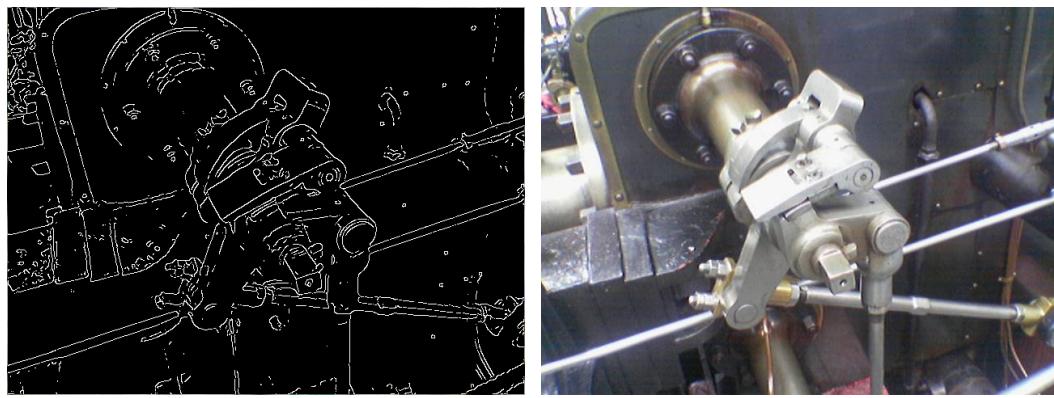
#### Detekcja cech

Detekcja cech (ang. *keypoints detection*) jest techniką, w której algorytm poprzez skanowanie obrazu piksel po pikselu podejmuje decyzję, czy dany punkt jest cechą, czy nie. Rezultatem tej techniki może być zbiór punktów, krawędzi, rogów, czy obszarów. Każda cecha ma swój deskryptor (ang. *keypoint descriptor*) ją określający. W dziedzinie widzenia komputerowego jest zaimplementowanych wiele algorytmów wykrywających cechy obrazów. Poniżej zostały wymienione niektóre algorytmy będące zaimplementowane przez bibliotekę przetwarzania obrazów OpenCV:

- **SIFT** [12] - Algorytm potrafi znaleźć charakterystyczne cechy niezależnie od lokalizacji, rotacji, czy skali obrazu. Jest jednym z najpopularniejszych algorytmów i jest wykorzystywany w niniejszej pracy. Jest opisany dokładniej w sekcji 4.2.1.
- **SURF** [2] - Algorytm oparty na tych samych zasadach co algorytm SIFT, natomiast detale poszczególnych kroków są różne. Autorzy algorytmu w swojej pracy pokazują, że SURF potrafi być szybszy of SIFT nie tracąc przy tym jakości. Tak jak SIFT, SURF jest niezależny od lokalizacji, rotacji, czy skali obrazu.
- **Canny** [6] - Algorytm potrafi wykryć krawędzie obiektów w obrazie, przez co dostarcza dużo informacji o ich strukturze. Na rysunku 3.5 jest zaprezentowane działanie algorytmu Canny.

#### Dopasowanie wykrytych cech

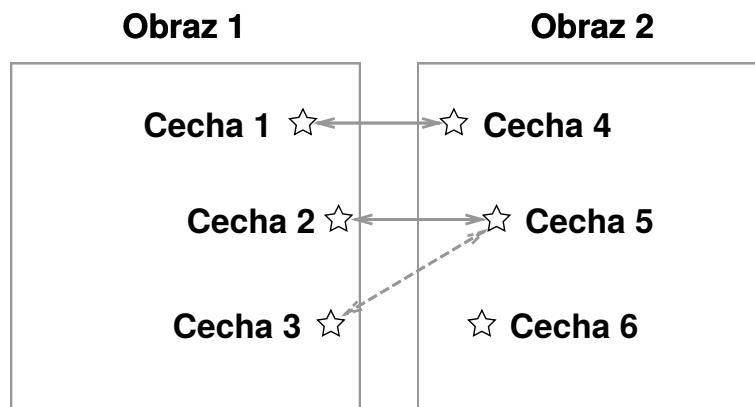
Następnym krokiem po detekcji cech w dwóch obrazach jest ich dopasowanie na podstawie deskryptorów. Najprostszym rozwiązaniem może być porównanie każdej



**Rys. 3.5:** Lewy obraz: Wynik zaaplikowanego algorytmu Canny na kolorowym obrazie z prawej strony. Prawy obraz: Oryginalny obraz.

cechy z pierwszego obrazu z każdą cechą z drugiego obrazu, natomiast dla niektórych zastosowań jest to rozwiązywanie zbyt wymagające obliczeniowo. Z tego względu zostały opracowane bardziej wydajne algorytmy. Przykładem jest wykorzystanie drzewa k-d przez Beis i Lowe [3].

Kolejnym wyzwaniem w dopasowaniu cech jest wykrycie obserwacji odstających (ang. *outliers*). Problem obrazuje rysunek 3.6.



**Rys. 3.6:** Dwa obrazy, w których wykryto po trzy cechy (na rysunku oznaczone jako gwiazdy). Algorytm szukający dopasowań wykrył dopasowania (na rysunku oznaczone jako strzałki) pomiędzy cechami: 1 i 4, 2 i 5, 3 i 5. Z wiedzy dziedzinowej wiemy, że obrazy nachodzą na siebie (tak jak angiograficzne obrazy OCT). Patrząc na dopasowania pomiędzy obrazami można zauważyć, że jedno z nich nie pasuje do reszty (strzałka narysowana przerywaną linią). Zakładając, że można tylko obracać i przesuwać obrazy niemożliwa jest estymacja macierzy transformacji biorąc pod uwagę wszystkie dopasowania. Niepoprawne dopasowanie trzeba odrzucić by otrzymać poprawną macierz transformacji.

Problem z rysunku 3.6 można rozwiązać poprzez wykorzystanie odpowiedniej metody filtrowania dopasowań. Jedną z takich metod jest RANSAC (ang. *random sample consensus*, *RANSAC*), który jest algorytmem iteracyjnym i jego zadaniem jest eliminacja obserwacji odstających. RANSAC jest wykorzystywany w niniejszej pracy i jest dokładniej opisy w sekcji 4.2.3. Do rozwiązania tego problemu można również

wykorzystać relacje geometryczne pomiędzy obrazami. Relacje geometryczne definiuje się na podstawie wiedzy dziedzinowej (np. obraz nie może się przesunąć dalej niż 10 pikseli – wszystkie dopasowania cech znajdujące się dalej niż 10 pikseli są niepoprawne). Filtracja na podstawie wiedzy dziedzinowej jest wykorzystywana w niniejszej pracy i jest opisana dokładniej w sekcji [4.2.3](#).

## Estymacja macierzy transformacji na podstawie dopasowań

Następnym krokiem po wyznaczeniu dopasowań pomiędzy dwoma obrazami jest estymacja macierzy transformacji jednego obrazu względem drugiego. Estymacja macierzy transformacji wykonuje się na podstawie ustalonego modelu deformacji (sekcja [3.1](#)). Biorąc pod uwagę jako model transformacji ciało sztywne (dozwolona rotacja i translacja) równanie pozwalające obliczyć parametry rotacji i translacji można zapisać jako:

$$T = SR + t \quad (3.4)$$

gdzie  $T$  to zbiór cech w obrazie docelowym,  $S$  to zbiór cech w obrazie źródłowym,  $R$  to macierz rotacji, a  $t$  to wektor translacji. Wyznaczenie  $R$  i  $t$  z tego równania byłoby proste, jeżeli wszystkie cechy z obrazu źródłowego dałoby się idealnie przekształcić na cechy obrazu docelowego. Istnieje bardzo mała szansa, że macierz transformacji obliczona na podstawie jednego dopasowania zaaplikowana do cechy z innego dopasowania zwróciłaby miejsce odpowiadającej cechy z drugiego obrazu (będzie blisko niej, ale nie idealnie w tym samym miejscu). Różnica wynika z tego, że obrazy są zaszumione (nałożenie szumu pomiarowego na rzeczywiste jasności punktów) oraz algorytmy zwracające cechy nie są tak dokładne. Jednym z prostszych rozwiązań tego problemu jest wykorzystanie metody minimalizacji błędu średniokwadratowego polegającej na znalezieniu takiej macierzy transformacji dla której błąd średniokwadratowy pomiędzy cechą w obrazie docelowym, a cechą z obrazu źródłowego po transformacji jest najmniejszy. Mając dane współrzędne cechy w obrazie docelowym  $T_i$  oraz współrzędne odpowiadającej cechy w obrazie źródłowym  $S_i$  oraz macierz transformacji  $M$  błąd średniokwadratowy można zapisać jako:

$$E_{LS} = \sum_i d(T_i, MS_i)^2 \quad (3.5)$$

gdzie  $E_{LS}$  to minimalny błąd średniokwadratowy, a funkcja  $d(\dots)$  to odległość euklidesowa pomiędzy dwoma punktami.

Równanie [3.5](#) można rozwiązać algorytmami optymalizacji. W niniejszej pracy do wyznaczenia parametrów rotacji i translacji na podstawie dopasowań została wykorzystana funkcja OpenCV, której użycie jest opisane w sekcji [4.4](#).

### 3.3 Łączenie kafelków

Po wyznaczeniu macierzy translacji pomiędzy dwoma kafelkami następnym krokiem jest ich złączenie w obszarze, w którym na siebie nachodzą. Problem by nie istniał jeżeli dopasowanie byłoby idealne oraz ekspozycja dwóch kafelków byłaby identyczna. Niestety w rzeczywistym świecie po złączeniu widoczne są krawędzie kafelków oraz inne artefakty wynikające z błędu rejestracji, czy niedokładności aparatu. Stworzenie ładnej oraz jednorodnej mozaiki wymaga określenia, które piksele z dwóch kafelków należy wykorzystać w procesie łączenia oraz jakie przypisać im wagę. Najprostszą metodą jest wzięcie średniej ważonej każdego piksela z dwóch łączonych kafelków:

$$f(x) = \frac{\sum_i \omega_i(x) f_i(x)}{\sum_i \omega_i(x)} \quad (3.6)$$

gdzie  $i$  to liczba nakładających się kafelków,  $f(x)$  to wartość piksela  $x$  w finalnej mozaice,  $f_i(x)$  to wartość piksela  $x$  w kolejnych nakładających się kafelkach, a  $w(x)$  to waga piksela  $x$ . Następnym krokiem w tej metodzie jest ustalenie wartości wag. Jednym z pomysłów jest wykorzystanie zwykłej średniej ( $w(x) = 1$  dla każdego piksela  $x$ ). Nie jest to jednak najlepszy sposób, ponieważ krawędzie kafelków po zastosowaniu średniej są dalej widoczne. Lepszym podejściem jest uzależnienie wartości wag od odległości piksela od środka kafelka do którego należy (piksel znajdujący się przy krawędzi swojego kafelka będzie miał mniejszą wagę niż piksel znajdujący się blisko środka swojego kafelka). Metoda wykorzystana jest w niniejszej pracy i dokładniej jest opisana w sekcji 4.6. Istnieją również bardziej zaawansowane metody takie jak algorytm *multi-band blending* zaproponowany przez Burt i Adelson [5] dający jeszcze lepsze rezultaty [4].



# Proponowane algorytmy

Niniejszy rozdział szczegółowo opisuje kolejne kroki rozwiązania problemu dokładnie przedstawionego w sekcji 1.1. Sekcja 4.1 opisuje wiedzę dziedzinową na temat danych wejściowych (kafelków). Następnie sekcje od 4.2.1 do 4.6 opisują zasadę działania poszczególnych metod w kolejności zgodnej z ich wykonywaniem w programie.

## 4.1 Wiedza dziedzinowa

Wiedza dziedzinowa określa zbiór informacji związanych z problemem i danymi wejściowymi. Dokładne zdefiniowanie oraz analiza wiedzy dziedzinowej prowadzi do ułatwienia problemu oraz lepszych rezultatów. W przypadku problemu niniejszej pracy danymi wejściowymi do stworzenia mozaiki są angiograficzne obrazy OCT (kafelki). Są to monochromatyczne obrazy o rozmiarze 240 na 240 pikseli. Wraz z obrazami do programu dostarczane są jeszcze dwie informacje:

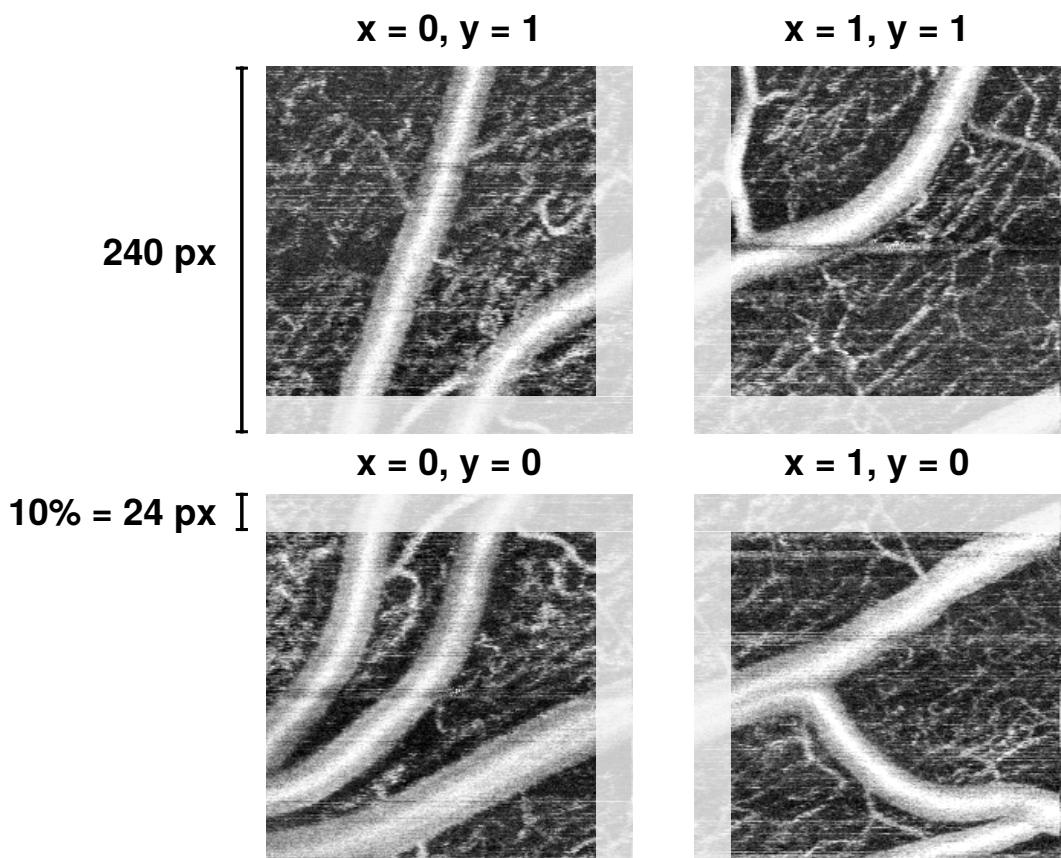
- Względna pozycja kafelków względem siebie zapisana w nazwie obrazu. Lokalizacja jest podana za pomocą współrzędnych x i y układu kartezjańskiego.
- Szacowany obszar nałożenia kafelków na siebie wyrażony w procentach szerokości obrazu.

Na rysunku 4.1 jest przedstawiony przykładowy zbiór kafelków przeznaczony do stworzenia jednej mozaiki wraz z dostarczoną wiedzą dziedzinową.

Informacje dostarczone wraz z kafelkami mają istotny wpływ na działanie algorytmu. Położenie kafelków informuje algorytm, który kafelek należy dopasowywać z którym (sekcja 4.2.3), a szacowany obszar nałożenia określa obszar detekcji cech (sekcja 4.2.1). Bez tych informacji stworzenie mozaiki byłoby o wiele bardziej skomplikowane.

## 4.2 Rejestracja kafelków z wykorzystaniem cech

Detekcja cech jest pierwszym krokiem tworzenia mozaiki. W niniejszej sekcji najpierw przedstawiono wybrany algorytm służący do detekcji cech oraz sposób jego użycia w



**Rys. 4.1:** Cztery kafelki rozmieszczone zgodnie z ich pozycją w docelowej mozaice. Na każdym obrazie białym przeźroczystym prostokątem zaznaczony jest oszacowany obszar nałożenia sąsiadujących kafelków.

programie (sekcja 4.2.1). Następnie opisano proces dopasowania wykrytych cech do siebie (sekcja 4.2.2), a na koniec tej sekcji przedstawiono filtrowanie znalezionych dopasowań (sekcja 4.2.3).

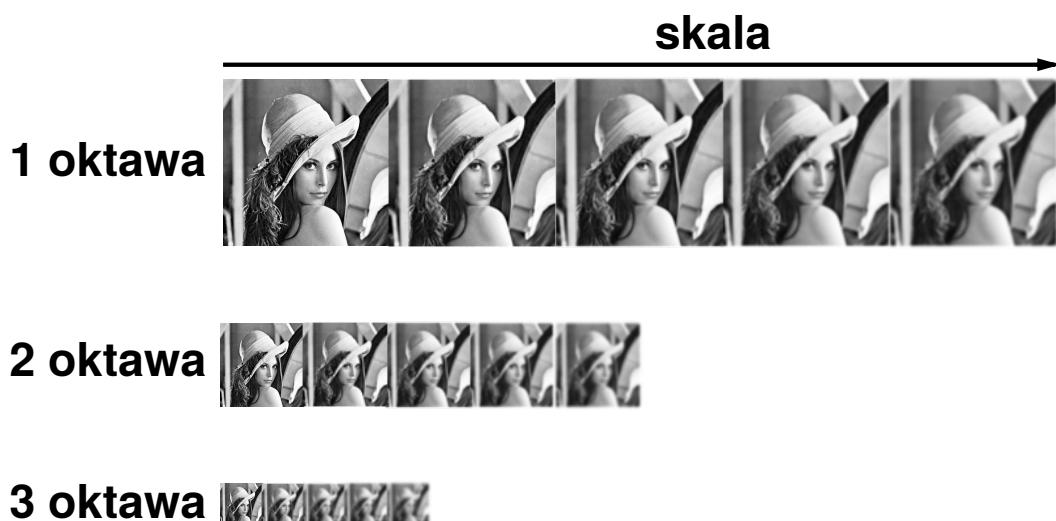
### 4.2.1 Algorytm SIFT

SIFT (ang. *scale-invariant feature transform*) to algorytm służący do wykrycia i opisania cech w obrazie. Metoda została opublikowana w 1999 roku przez David Lowe [12]. Dzięki umiejętności detekcji cech, które są niewrażliwe na rotacje, skalowanie, lokalizacje i przekształcenie afiniczne obrazu SIFT jest stosowany w wielu aplikacjach (np. lokalizacja robota [16], czy rozpoznawanie zachowań człowieka [11]).

Działanie algorytmu można podzielić na 6 kroków:

## 1. Reprezentacja obszaru za pomocą przestrzeni *scale-space*

Algorytm rozpoczyna swoją pracę tworząc przestrzeń *scale-space* obszaru obrazu służący do detekcji cech. *Scale-space* generuje się poprzez kilkakrotne użycie filtra Gaussa (z kolejnymi wartościami siły zdefiniowanych przez wariancję rozkładu normalnego) na docelowym obszarze. Następnie oryginalny obraz próbujemy zmniejszając jego wielkość dwukrotnie i powtarzamy proces użycia filtra Gaussa. Obrazy o tej samej wielkości tworzą oktawę. Na rysunku 4.2 pokazana jest przestrzeń *scale-space* dla przykładowego obrazu.

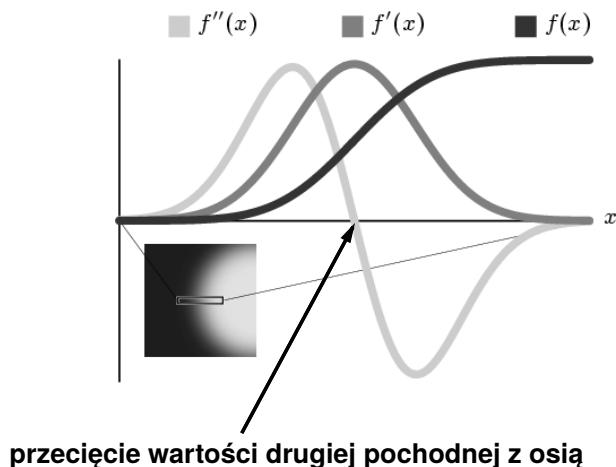


**Rys. 4.2:** Przestrzeń *scale-space* dla przykładowego obrazu. Skala rośnie wraz z siłą filtru Gaussa. Obrazy tej samej wielkości tworzą oktawę.

Autor algorytmu SIFT zalecana jest użycie czterech oktaw i pięciu poziomów siły filtru Gaussa.

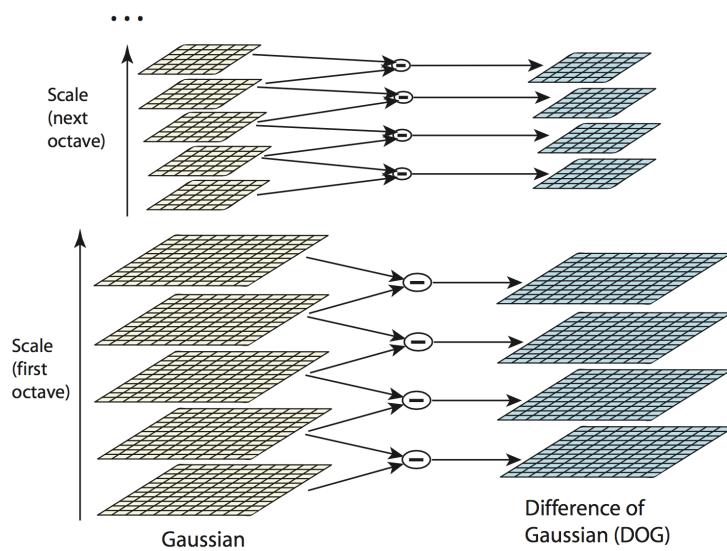
## 2. Obliczenie *difference of Gaussians*

Charakterystycznymi punktami obrazu są krawędzie i narożniki. Są to miejsca najlepiej nadające się na znalezienie cech. Operacja *laplacian of Gaussian*, *LoG* świetnie sobie radzi w wykrywaniu krawędzi i rogów w obrazie. Polega ona na wyeliminowaniu szumu poprzez zaaplikowanie filtra Gaussa, a następnie obliczenie drugiej pochodnej. Na rysunku 4.3 przedstawione jest działanie *LoG* na jednowymiarowym fragmencie obrazu.



**Rys. 4.3:** Przykład wykrycia krawędzi przez LoG. Czarny wykres przedstawia jednowymiarowy sygnał z obrazu. Miejsce przecięcia się z osią drugiej pochodnej tzw. *zero-crossing* wskazuje miejsce krawędzi w oryginalnym obrazie.

Obliczenie drugiej pochodnej całego obrazu jest natomiast kosztownym procesem. SIFT implementuje operację *difference of Gaussians*, DoG dającą niemal identyczny wynik do LoG, ale jest o wiele szybsza. DoG polega na odjęciu dwóch kolejnych obrazów z tej samej oktawy. Rysunek 4.4 przedstawia operację DoG. DoG jest obliczane dla każdej pary w każdej oktawie.

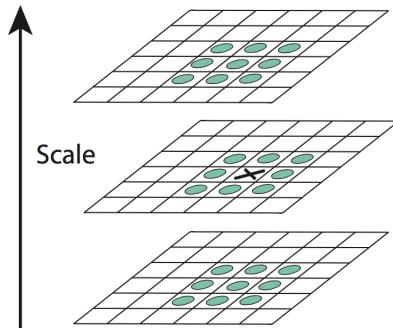


**Rys. 4.4:** Poprzez proces odjęcia kolejnych zdjęć z jednej oktawy otrzymujemy obrazy DoG doskonale odzwierciedlający LoG (obraz pochodzi z pracy [12]).

### 3. Detekcja cech

Następnym krokiem po obliczeniu obrazów DoG jest detekcja cech. Na rysunku 4.4 druga pochodna poprzez przecięcie osi (*zero-crossing*) wskazuje miejsce występo-

wania krawędzi w oryginalnym obrazie, natomiast można też zauważyc, że lokalne maksimum z lewej strony i lokalne minimum z prawej strony wykresu wskazują dwie strony krawędzi. SIFT wykrywa cechy poprzez wyznaczenie lokalnych maksimów i minimów w obrazach DoG. Proces wyjaśniony jest na rysunku 4.5.



Rys. 4.5: Maksima i minima wykrywane są za pomocą porównania wartości aktualnie badanego piksela (zaznaczony czarnym krzyżykiem) z jego 26 sąsiadami powstałymi poprzezłączenie kolejnych obrazów DoG z jednej oktawy. Cecha jest wykrywana w miejscu aktualnie badanego piksela jeżeli wartość tego piksela jest największa lub najmniejsza spośród 26 sąsiadów (obraz pochodzi z pracy [12]).

#### 4. Filtrowanie cech

W większości przypadków detekcja cech może wygenerować ich ogromną ilość. SIFT poprzez zastosowanie następujących dwóch filtrów pozbywa się cech najgorszej jakości:

1. Pierwszy filtr sprawdza intensywności piksela obrazu DoG w miejscu gdzie wykryto cechę. Jeżeli wartość piksela jest mniejsza niż ustalony wcześniej próg cecha jest odrzucana.
2. Drugi filtr polega na obliczeniu dwóch prostopadłych do siebie gradientów w miejscu wystąpienia cechy. W zależności od otoczenia cechy możliwe są 3 opcje:
  - Wartości dwóch gradientów będą małe z czego wynika, że w otoczeniu cechy jest region płaski. Cecha jest odrzucana.
  - Wartość jednego gradientu będzie duża, a drugiego mała. Oznacza to, że wzdłuż gradientu z małą wartością występuje krawędź w oryginalnym obrazie (gradient z dużą wartością jest prostopadły do krawędzi). Cecha jest odrzucana.

- Wartość dwóch gradientów będzie duża. Oznacza to, że w cecha znajduje się w miejscu występowania rogu w oryginalnym obrazie. Cecha jest zachowana ze względu na to, że narożniki są najlepszymi cechami.

## 5. Przydzielenie orientacji cechom

Kolejny krok algorytmu SIFT polega na obliczeniu orientacji każdej cechy na podstawie jej otoczenia, przez co deskryptory cech mogą być przedstawione względnie do swojej orientacji. Dzięki temu algorytm SIFT jest niezależny od rotacji obrazów. Proces obliczenia orientacji zaczyna się od wyznaczenia wartości gradientu  $m(x, y)$  i orientacji  $\theta(x, y)$  dla każdego piksela z otoczenia wokół cechy za pomocą wzorów:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \quad (4.1)$$

$$\theta(x, y) = \tan^{-1}((L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y))) \quad (4.2)$$

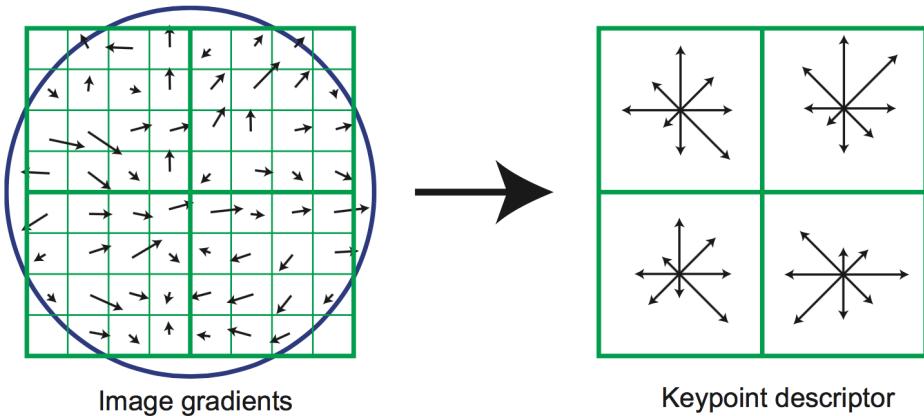
Następnie na podstawie wartości orientacji tworzony jest histogram posiadający 36 przedziałów przez co pokrywa zakres orientacji od 0 do 360 stopni. Każdy piksel dodany do histogramu jest ważony poprzez użycie jego wartości gradientu i wartości okrągłego okna filtra Gaussa (piksele znajdujące się dalej od cechy mają mniejszą ważność niż te znajdujące się bliżej). Orientacja odpowiadająca najwyższej wartości w histogramie jest wybrana jako orientacja cechy. Dodatkowo każda orientacja posiadająca wartość większą niż 80% maksymalnej wartości w histogramie jest zapisywana jako nowa cecha obrazu.

## 6. Generacja deskryptorów cech

Deskryptor cechy musi ją charakterystycznie określać i być łatwy do obliczenia. SIFT przedstawia cechę jako 128 elementowy wektor powstały poprzez wykonanie następujących kroków:

1. Tworzone jest otoczenie 16 na 16 pikseli wokół cechy.
2. Dla każdego piksela w otoczeniu cechy jest liczona wartość gradientu oraz orientacja zgodnie ze wzorami 4.1 i 4.2. Każdą orientację piksela w otoczeniu odejmujemy od orientacji cechy obliczonej w poprzednim punkcie, przez co algorytm SIFT jest niezależny od rotacji obrazu.
3. Wartości gradientu oraz orientacji są ważone przez okrągłe okno filtru Gaussa i są następnie gromadzone poprzez utworzenie histogramu orientacji, który określa informacje w regionach 4 na 4 piksele. Rysunek 4.6 przedstawia

proces kumulacji wartości gradientu oraz orientacji poprzez wykorzystanie histogramu.



**Rys. 4.6:** Lewy obraz: Wartości orientacji oraz gradientu z uwzględnieniem okrągłego okna filtru Gaussa. Prawy obraz: Wartości z obszarów 4 na 4 piksele są gromadzone w histogramie, który posiada 8 przedziałów. Wartości orientacji z lewego obrazu mieszczące się w jednym przedziale histogramu są sumowane, np. piksel z orientacją 12 stopni wchodzi do przedziału 0 - 44 stopni histogramu i jego wartość jest dodawana z każdą orientacją 0 - 44 stopni (obraz pochodzi z pracy [12]).

4. Prawy obraz z rysunku 4.6 przedstawia deskryptor z obszaru 8 na 8 pikseli posiadający 32 elementy (4 obszary, każdy posiada 8 orientacji). Autor algorytmu SIFT podaje, że najlepsze rezultaty można uzyskać poprzez policzenie deskryptora z obszaru 16 na 16 pikseli wokół cechy, wtedy deskryptor posiada 128 elementów (16 obszarów, każdy posiada 8 orientacji).
5. Na koniec 128 elementowy wektor jest normalizowany.

### Sposób wykorzystania algorytmu SIFT w programie

Biblioteka OpenCV posiada zaimplementowany algorytm SIFT i jego użycie sprowadza się do paru linii kodu:

```
SIFT featuresFinder = SIFT::SIFT(...);
featuresFinder(...);
```

Konstruktor `SIFT::SIFT(...)` tworzy nam obiekt SIFT oferujący funkcjonalność algorytmu SIFT. Do konstruktora `SIFT::SIFT(...)` podajemy parametry algorytmu tj. ilość najlepszych cech do zwrócenia, ilość oktaf do wykorzystania, progi do filtrowania wykrytych cech (sekcja 4.2.1) oraz siłę filtru Gaussa. Operator `featuresFinder(...)`

przyjmuje obraz w którym mają być znalezione cechy i zwraca miejsca wystąpienia tych cech oraz ich deskryptory. Więcej informacji na temat obiektu SIFT znajduje się w oficjalnej dokumentacji OpenCV<sup>1</sup>.

#### 4.2.2 Dopasowanie wykrytych cech

Cecha jest uważana za dopasowaną do drugiej cechy, jeżeli deskryptory (w przypadku SIFT 128 elementowy wektor) tych cech są do siebie podobne. W świecie rzeczywistym bardzo rzadko zdarza się, że deskryptory są identyczne, dlatego algorytmy zwracają zazwyczaj deskryptor, który jest najbardziej podobny. Trudność problemu dopasowania wykrytych cech jest uzależniona od ilości wykrytych cech w każdym z obrazów. Im więcej cech do dopasowania tym problem staje się bardziej kosztowny obliczeniowo. W przypadku problemu niniejszej pracy mamy do czynienia z obrazami o rozmiarze 240 na 240 pikseli z czego w większości zbiorów danych obrazy nakładają się w 10% szerokości obrazu, czyli algorytm SIFT szuka cech w obszarze 24 na 240 pikseli. Jest to obszar na tyle mały, że ilość cech wykrytych przez SIFT jest niewielka. Dzięki temu w niniejszej pracy w procesie rejestracji kafelków każda cecha jednego obrazu jest porównywana z każdą cechą drugiego obrazu. Jeżeli w jednym kafelku wykryto  $n$  cech, a w drugim  $m$  cech to algorytm wykonuje  $nm$  dopasowań. Taki mechanizm dopasowania nazywa się *Brute Force Matching* i tak jak algorytm SIFT również jest zaimplementowany w OpenCV. Użycie *Brute Force Matching* w OpenCV tak jak w przypadku SIFT sprowadza się do kilku linii kodu:

```
BFMatcher bruteForceMatcher;  
bruteForceMatcher.match(...);
```

Obiekt BFMatcher posiada funkcjonalność pozwalającą na dopasowanie każdego deskryptora z jednego zbioru do każdego deskryptora z drugiego zbioru. Funkcja `match(...)` przyjmuje deskryptory z dwóch zbiorów i zwraca dopasowania. Więcej informacji na temat obiektu BFMatcher znajduje się w oficjalnej dokumentacji OpenCV<sup>2</sup>.

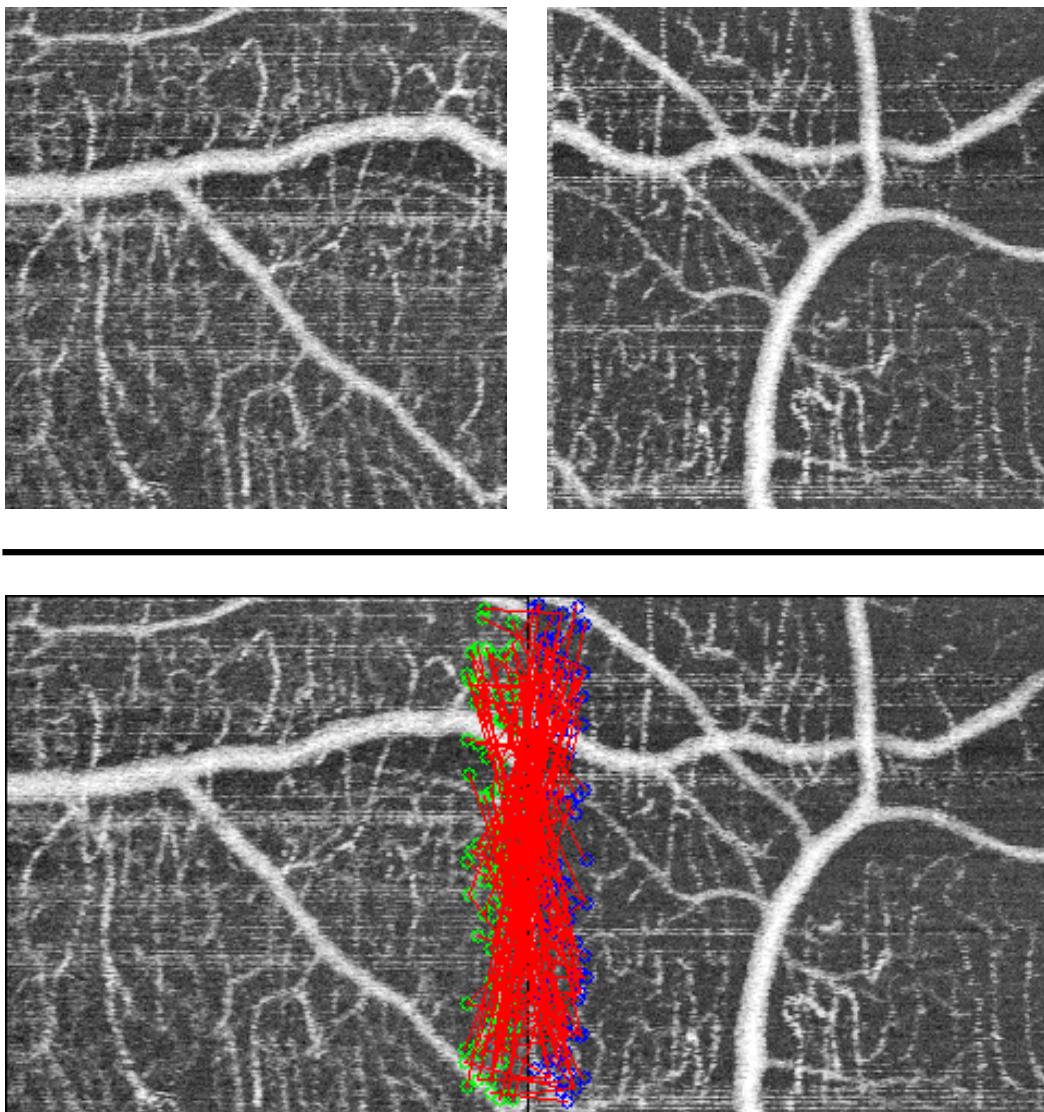
#### 4.2.3 Filtrowanie dopasowań

Na tym etapie algorytm już wykrył cechy z obrazów i wyznaczył dopasowania pomiędzy nimi. Ze względu na to, że naczynia krwionośne w angiograficznych obrazach OCT są bardzo do siebie podobne wiele z tych dopasowań będzie niepoprawnych. Rysunek 3.6 z sekcji 3.2.2 przedstawia istotę problemu niepoprawnych dopasowań,

<sup>1</sup>[http://docs.opencv.org/modules/nonfree/doc/feature\\_detection.html](http://docs.opencv.org/modules/nonfree/doc/feature_detection.html)

<sup>2</sup>[http://docs.opencv.org/modules/features2d/doc/common\\_interfaces\\_of\\_descriptors\\_matchers.html](http://docs.opencv.org/modules/features2d/doc/common_interfaces_of_descriptors_matchers.html)

natomast rysunek 4.7 przedstawia wykryte dopasowania przez mechanizm *Brute Force Matching* na dwóch angiograficznych obrazach OCT.



Rys. 4.7: Na górze znajdują się dwa angiograficzne obrazy OCT, które mają być złączone na ok. 10% swojej szerokości. Na dole znajdują się wykryte cechy (kółka zielone i niebieskie) w dwóch obrazach oraz znalezione dopasowania (czerwone linie) pomiędzy cechami za pomocą techniki *Brute Force Matching*.

Na rysunku 4.7 przedstawiono dopasowania zwrócone przez algorytm. Łatwo zauważać, że część wykrytych dopasowań jest sprzeczna z resztą. Z tego względu wymagane jest dalsze filtrowanie tych dopasowań na podstawie wiedzy dziedzino- wej oraz dostępnych algorytmów. Poniżej opisano poszczególne procesy filtrowania dopasowań w takiej kolejności w jakiej wykonuje je program.

## Filtrowanie na podstawie położenia

Pierwsza technika filtrowania polega na usunięciu dopasowań doprowadzających do nadmiernego przesunięcia kafelka poza określoną normą. Ta norma jest określona poprzez dwa parametry *shiftParameter* i *percentOverlap*, które ustawia się w pliku konfiguracyjnym. W zależności od sposobu ułożenia kafelków, dopasowanie przejdzie ten filtr jeżeli spełnia dwa warunki:

1. Dopasowane cechy nie będą się znajdować zbyt daleko od siebie w osi pionowej (w przypadku ułożenia kafelków w pionie) lub osi pionowej (w przypadku ułożenia kafelków w poziomie). Ten warunek określa wzory – 4.3 dla przypadku ułożenia kafelków w pionie, 4.4 dla przypadku ułożenia kafelków w poziomie:

$$|f_{1x} - f_{2x}| < imageSize * shiftParameter \quad (4.3)$$

$$|f_{1y} - f_{2y}| < imageSize * shiftParameter \quad (4.4)$$

gdzie  $f_{1x}$  i  $f_{2x}$  to współrzędne  $x$  dopasowanych cech z dwóch obrazów (dla  $f_{1y}$  i  $f_{2y}$  są to współrzędne  $y$ ), a *imageSize* to rozmiar angiograficznego obrazu OCT (w przypadku niniejszej pracy ma 240 pikseli).

2. Dopasowane cechy nie będą znajdować się zbyt daleko od siebie. Ten warunek określa wzór:

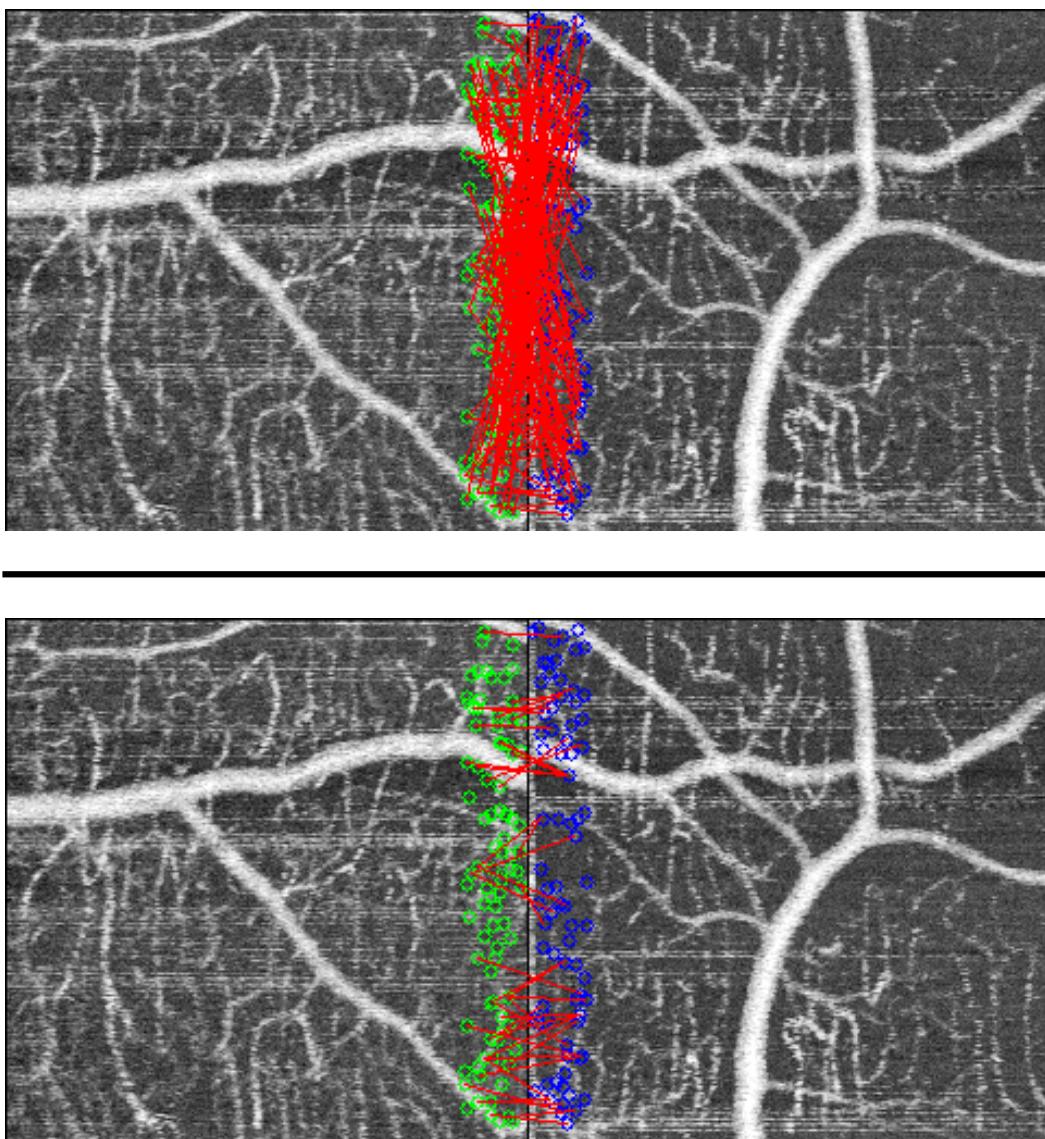
$$length < imageSize * percentOverlap * 2.5 \quad (4.5)$$

gdzie *length* to długość wektora pomiędzy cechami z uwzględnieniem położenia kafelków. W zależności od położenia kafelków *length* jest zdefiniowane jako – 4.6 dla położenia, w którym kafelki ustawione są w poziomie i kafelek  $f_1$  znajduje się z lewej strony kafelka  $f_2$ , 4.7 dla położenia, w którym kafelki ustawione są w pionie i kafelek  $f_1$  znajduje się nad kafelkiem  $f_2$ :

$$length = \sqrt{(f_{1x} - (f_{2x} + imageSize))^2 + (f_{1y} - f_{2y})^2} \quad (4.6)$$

$$length = \sqrt{(f_{1x} - f_{2x})^2 + (f_{1y} - (f_{2y} + imageSize))^2} \quad (4.7)$$

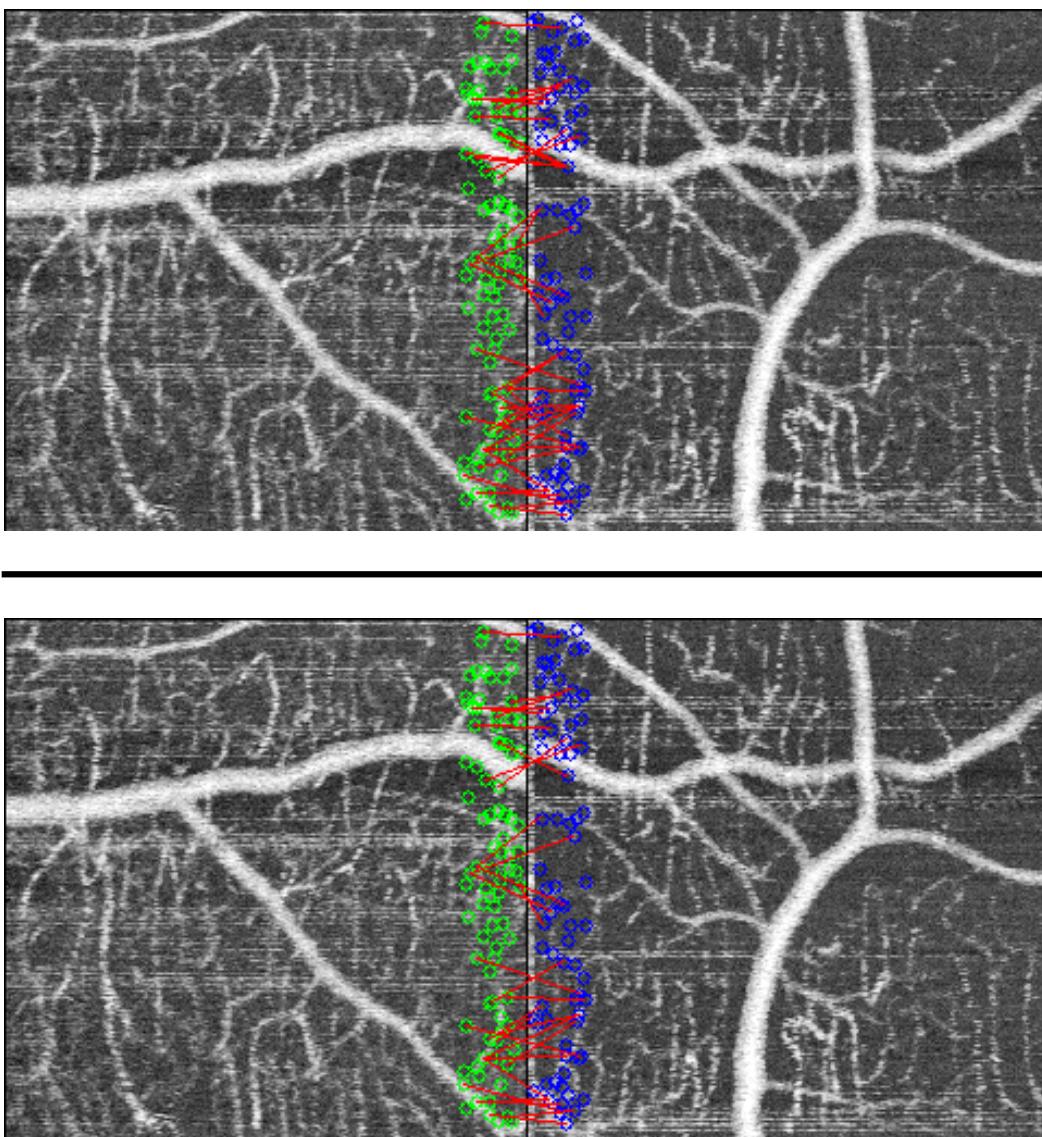
Rysunek 4.8 przedstawia wynik filtracji na tych samych dwóch obrazach angiograficznych użytych jako przykład w rysunku 4.7.



Rys. 4.8: Górnny obraz: Dopasowanie wyznaczone po zastosowaniu *Brute Force Matching*.  
Dolny obraz: Dopasowania pozostałe po filtrowaniu na podstawie położenia.

### Usuwanie niejednoznacznych dopasowań przypisanych do jednej cechy

Czasem występuje sytuacja, w której jedną cechę z jednego kafelka dopasowano do kilku cech z drugiego kafelka. Na podstawie wiedzy dziedzinowej wiemy, że jest to sytuacja niepożądana i cecha może być dopasowana tylko i wyłącznie do jednej cechy. Niniejszy etap filtracji polega na iteracji po wszystkich cechach posiadających dopasowania do więcej niż jednej cechy i na pozostawieniu tylko najbardziej podobnej cechy (na podstawie wartości deskryptora cechy). Ten etap filtrowania jest wykonywany dwukrotnie – pierwszy raz dla cech z pierwszego kafelka i drugi raz dla cech z drugiego kafelka. Rysunek 4.9 przedstawia wynik filtracji poprzez eliminację niejednoznacznych dopasowań na dopasowaniach będących wynikiem filtracji poprzedniej (rysunek 4.8).



**Rys. 4.9:** Górnny obraz: Dopasowania wyznaczone po zastosowaniu filtrowania na podstawie położenia. Dolny obraz: Dopasowania pozostałe po eliminacji niejednoznacznych dopasowań.

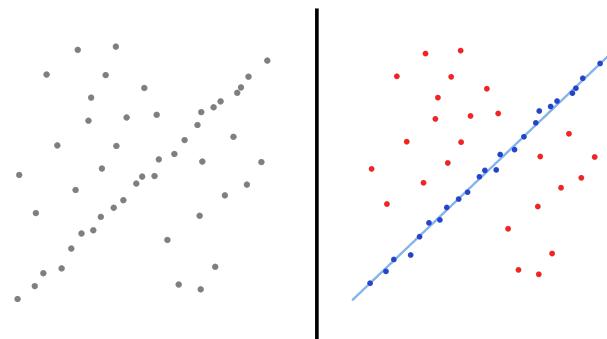
## RANSAC

Poprzednie dwa kroki filtrowania dopasowań wyeliminowały znaczącą część dopasowań wykrytych na początku. Niestety nadal można zauważać (prawy obraz na rysunku 4.9), że niektóre dopasowania są nieprawidłowe. Algorytm RANSAC (ang. *random sample consensus, RANSAC*) pierwszy raz opublikowany w roku 1981 przez Fischler i Bolles [8] zaimplementowano by wyeliminować próbki stanowiące szum w celu znalezienia poprawnego modelu. W niniejszej pracy próbками są dopasowania cech, a modelem jest poprawna macierz transformacji pomiędzy kafelkami.

Działanie algorytmu RANSAC składa się z dwóch kroków powtarzanych w sposób iteracyjny:

1. W pierwszym kroku algorytm losowo wybiera podzbior próbek. Następnie używając tylko próbek z tego podzbioru oblicza pasujący model.
2. W drugim kroku algorytm sprawdza, które próbki z całego zbioru są zgodne z modelem obliczonym w pierwszym kroku. Próbka będzie uznana za szum jeżeli nie będzie pasować do modelu w obrębie progu błędu.

Algorytm RANSAC będzie powtarzał powyższe dwa kroki dopóki nie znajdzie w kroku drugim wystarczającej ilości próbek pasujących do modelu wyznaczonego w kroku pierwszym lub nie przekroczy liczby iteracji, która jest parametrem algorytmu. Rysunek 4.10 przedstawia działanie algorytmu RANSAC na problemie, w którym próbками są punkty, a szukanym modelem jest funkcja prostej.



Rys. 4.10: **Lewy obraz:** Zbiór danych, w którym wiele punktów jest szumem. **Prawy obraz:** Funkcja prostej (niebieska linia) obliczona przez algorytm RANSAC. Punkty, które są szumem (czerwone kropki) nie mają wpływu na model, czyli funkcję prostej (obrazy pochodzą ze strony internetowej <http://en.wikipedia.org/wiki/RANSAC>).

Biblioteka OpenCV implementuje algorytm RANSAC w sposób niejawny wewnątrz funkcji `findHomography(...)`, której celem jest znalezienie afiniycznej transformacji pomiędzy dwoma zbiorami punktów. Dodatkowo funkcja `findHomography(...)` implementuje krok pozbywający się szumu ze swoich danych wejściowych. Ten krok może być wykonany za pomocą metody RANSAC lub LMedS (ang. *least-median of squares, LMedS*). Żeby otrzymać wynik algorytmu RANSAC na dopasowaniach pomiędzy punktami `featuresSource` z pierwszego kafelka i `featuresTarget` z drugiego kafelka należy do funkcji `findHomography(...)` przekazać opcjonalny wektor `mask`, w którym będzie zapisany wynik algorytmu RANSAC. Wywołanie funkcji wygląda następująco:

```
findHomography(featuresSource, featuresTarget, CV_RANSAC, threshold, mask)
```

Wartość CV\_RANSAC powoduje użycie algorytmu RANSAC zamiast LMedS. Jeżeli element o indeksie  $i$  wektora mask posiada wartość, która jest równa 0 to cechy o indeksie  $i$  wektorów featuresSource i featuresTarget są szumem. Argument threshold określa maksymalny błąd metody RANSAC. Para cech  $i$  uznawana jest jako szum, gdy:

$$\text{featuresSource}_i - H * \text{featuresTarget}_i > \text{threshold} \quad (4.8)$$

gdzie  $H$  to macierz transformacji pomiędzy punktami featuresSource, a featuresTarget. Domyślnie funkcja `findHomography(...)` ustawia wartość 3 dla argumentu `threshold` i ta wartość używana jest w niniejszej pracy.

Rysunek 4.11 przedstawia wynik algorytmu RANSAC na dopasowaniach będących wynikiem poprzedniego procesu filtracji (rysunek 4.9).

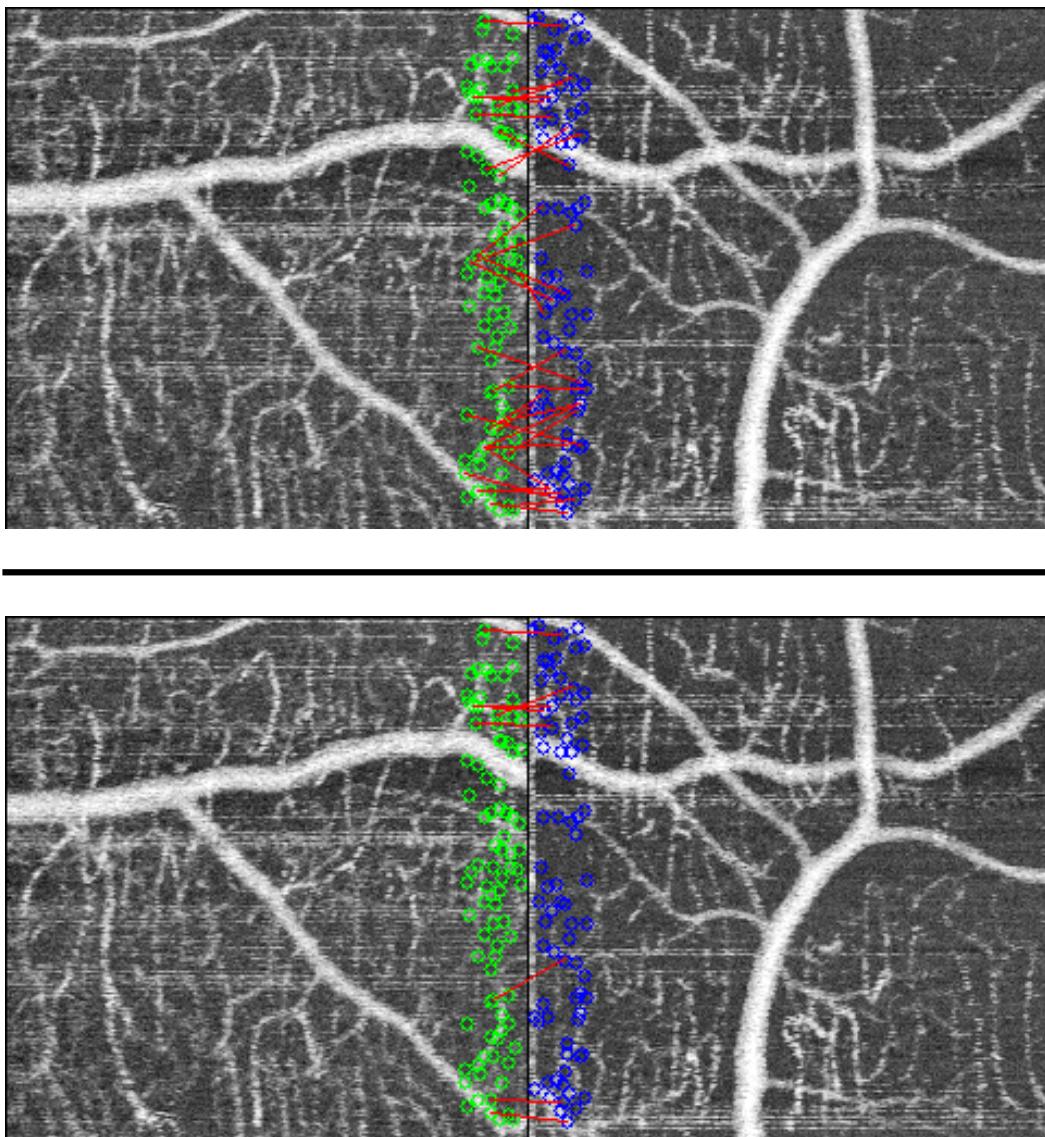
### Filtrowanie na podstawie nachylenia linii dopasowania

Ostatni etap filtrowania składa się z trzech kroków:

1. Dla każdego dopasowania obliczany jest kąt nachylenia  $angle$  funkcji prostej przechodzącej przez cechy z tego dopasowania. W zależności od ułożenia kafelków cechy mają współrzędne odpowiednio zmodyfikowane poprzez rozmiar kafelka.
2. Z obliczonych w poprzednim kroku wartości kątów nachylenia wyznaczana jest wartość mediany  $medianAngle$ .
3. Dopasowanie przechodzi ten filtr jeżeli kąt nachylenia  $angle$  obliczony w pierwszym kroku zawiera się w przedziale ograniczonym przez wartość mediany  $medianAngle$  wyznaczonej w kroku drugim i parametr  $angleParameter$ , który ustawia się w pliku konfiguracyjnym:

$$medianAngle + angleParameter > angle > medianAngle - angleParameter \quad (4.9)$$

Filtr ma na celu pozostawić dopasowania z podobnym nachyleniem. Rysunek 4.12 przedstawia wynik filtrowania na podstawie nachylenia dopasowań będących wynikiem poprzedniego procesu filtracji (rysunek 4.11).

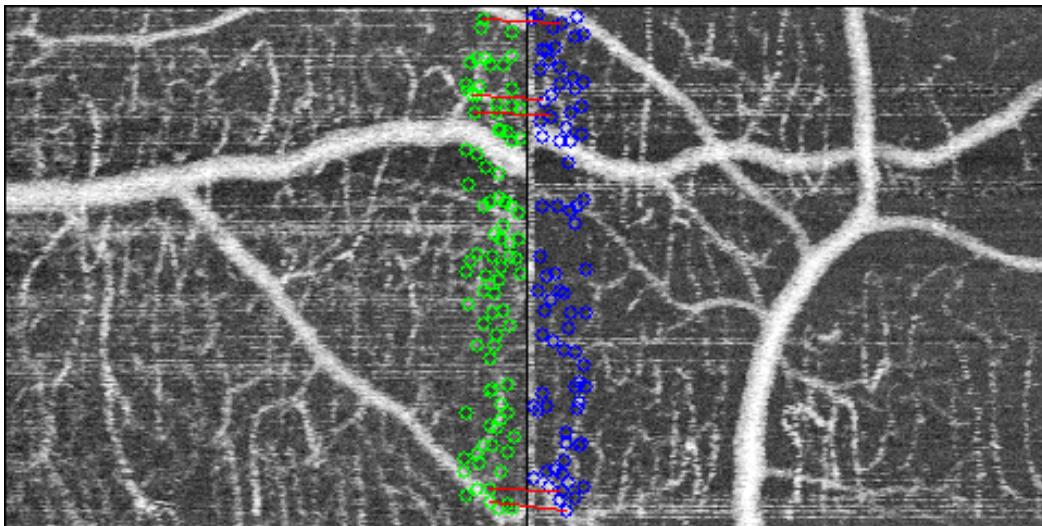
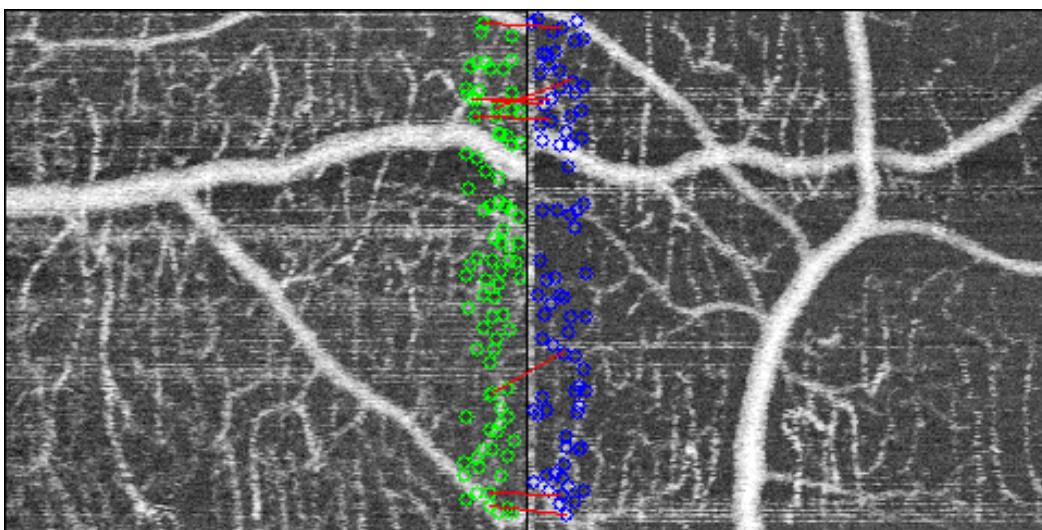


**Rys. 4.11:** **Górny obraz:** Dopasowania pozostałe po eliminacji niejednoznacznych dopasowań. **Dolny obraz:** Dopasowania pozostałe po użyciu algorytmu RANSAC.

Wynik ostatniego filtra przedstawiony na prawym obrazie na rysunku 4.12 pozostawia poprawne dopasowania, które mogą być użyte do estymacji macierzy transformacji (sekcja 4.4).

### 4.3 Rejestracja kafelków poprzez wykrycie położen naczyń krwionośnych w kafelkach

Technika rejestracji kafelków za pomocą cech opisana w sekcji 4.2 jest jedną z dwóch metod rejestracji zaimplementowanych w programie. Niniejszy rozdział przedstawia drugą metodę, która została zaimplementowana ze względu na pojawienie się zbioru kafelków o niestandardowym rozmiarze tj. 40 na 240 pikseli. Dla takiego



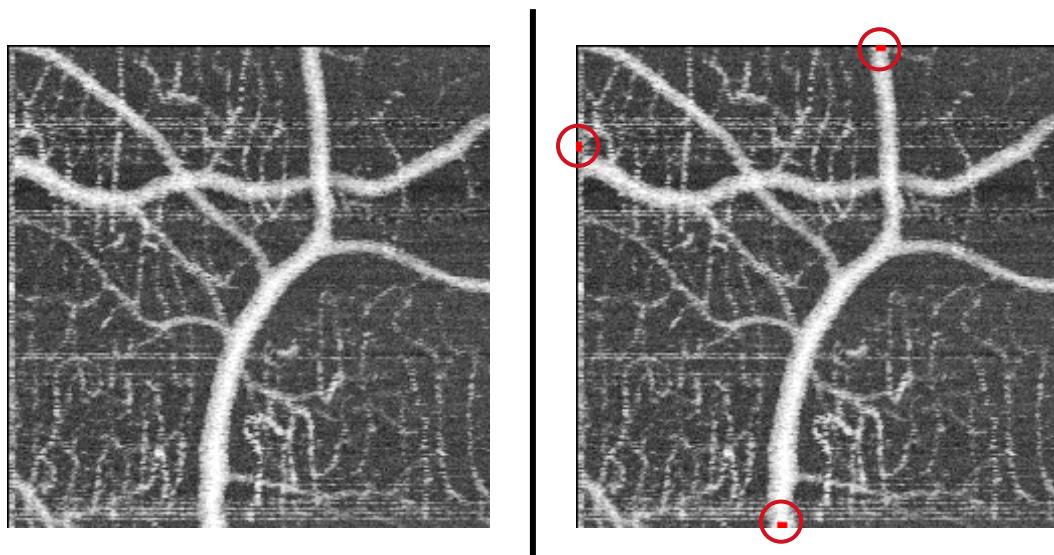
**Rys. 4.12:** Górný obraz: Dopasowania pozostałe po użyciu algorytmu RANSAC. Dolny obraz: Dopasowania posiadające podobny kąt nachylenia.

zbioru szerokość obszaru nałożenia wynosi 4 piksele co jest zdecydowanie zbyt małą wartością by wykryć cechy. By zarejestrować kafelki w takim rozmiarze należało wziąć pod uwagę obszar całego kafelka, a nie tylko obszar nałożenia. Metoda zaproponowana w niniejszej sekcji ma na celu wykrycie miejsc występowania naczyń krwionośnych na krawędziach obrazu kafelka. Wynik tej metody jest przedstawiony na rysunku 4.13.

W sekcji 4.3.1 opisano zasadę działania algorytmu.

### 4.3.1 Zasada działania

Działanie algorytmu można podzielić na dwa etapy:



**Rys. 4.13:** Lewy obraz: Przykładowy angiograficzny obraz OCT. Prawy obraz: Wykryte naczynia krwionośne na krawędziach obrazu (zaznaczone czerwonymi prostokątami).

### 1. Tworzenie szkieletu obrazu angiograficznego

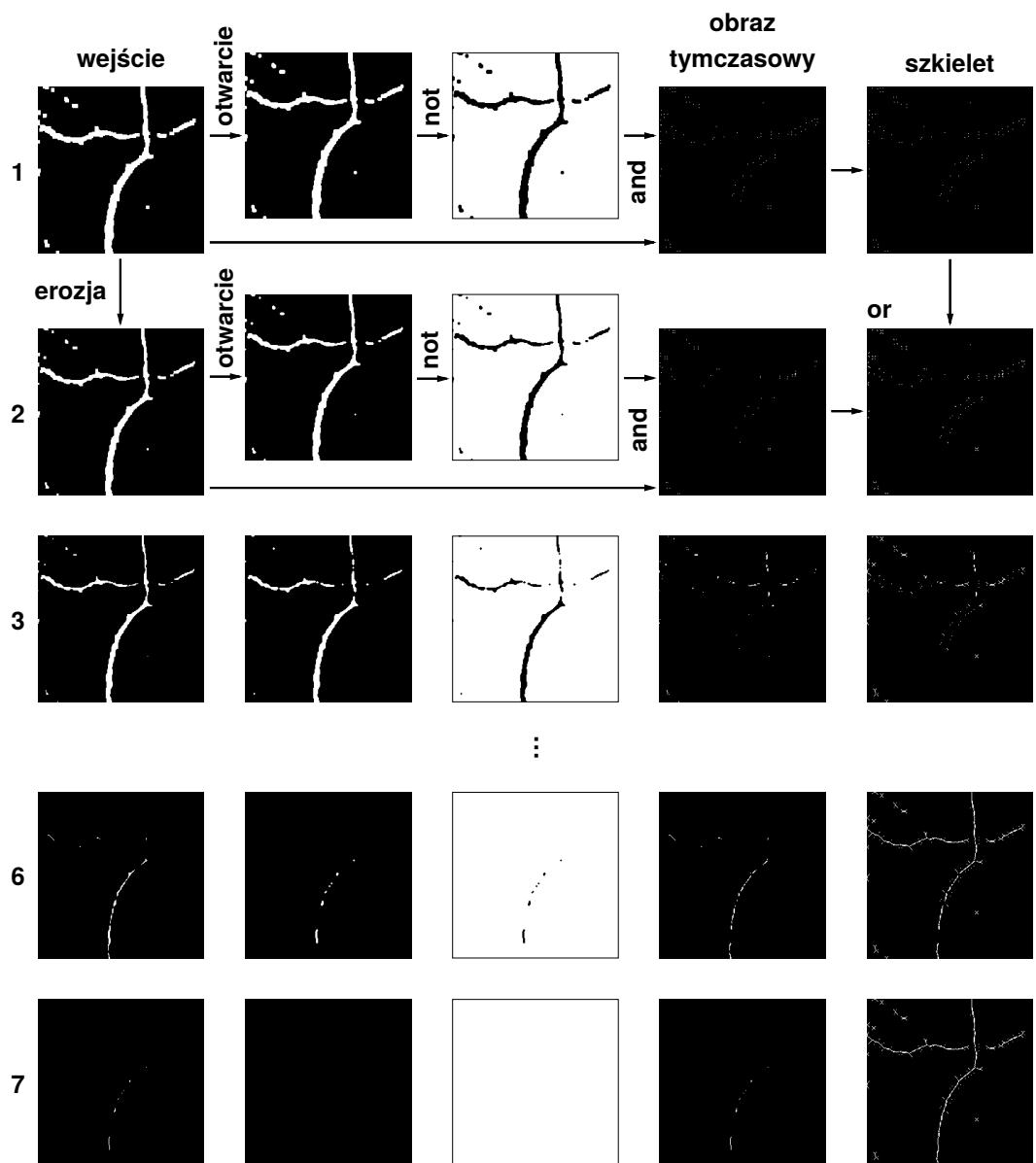
Szkieletem obiektu nazywamy kształt znajdujący się wewnątrz tego obiektu, którego krawędzie są równo oddalone od krawędzi obiektu. Przykład szkieletu znajduje się na rysunku 4.14.



**Rys. 4.14:** Szkielet (zaznaczony białą linią) litery B (obraz pochodzi ze strony [https://en.wikipedia.org/wiki/Topological\\_skeleton](https://en.wikipedia.org/wiki/Topological_skeleton)).

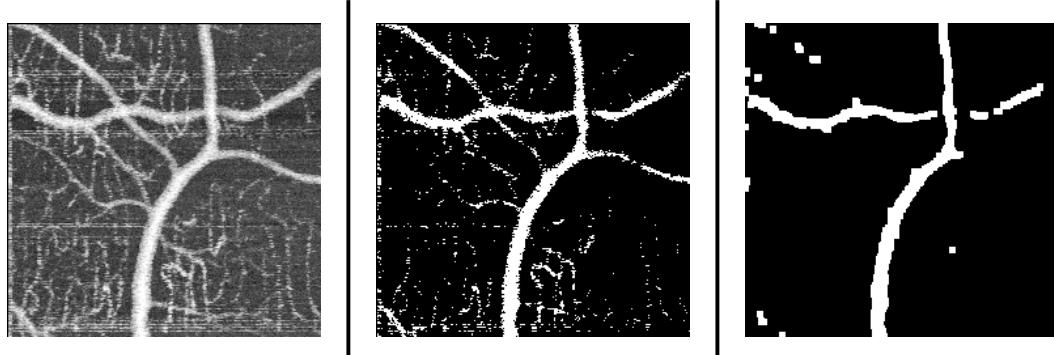
Proces tworzenia szkieletu obrazu [4.1] zaczyna się od wstępniego przetwarzania (rysunek 4.16) składającego się z dwóch etapów:

1. Binaryzacji obrazu.
2. Zastosowania operacji morfologicznych na obrazie (zamknięcie, a następnie otwarcie).



**Rys. 4.15:** Rysunek przedstawia przykładowy proces tworzenia szkieletu. Obraz w lewym górnym rogu jest obrazem wejściowym i jest wynikiem wstępniego przetwarzania. Algorytm potrzebował w tym przypadku 7 iteracji by stworzyć szkielet (iteracje są zaznaczone numerami z lewej strony). W każdej iteracji proces zaczyna się od otwarcia, a następnie negacji obrazu wejściowego (obraz w 1 kolumnie). Wynik poprzedniej operacji wraz z obrazem wejściowym jest poddany operatorowi logicznemu *and*, którego wynik jest zapisany w obrazie tymczasowym (obraz w 4 kolumnie). Szkielet powstaje poprzez zastosowanie operatora logicznego *or* na szkielecie z poprzedniej iteracji i obrazie tymczasowym z aktualnej iteracji. Po każdej iteracji obraz wejściowy jest poddawany erozji. Algorytm kończy działanie, jeżeli obraz wejściowy po operacji erozji będzie posiadał wszystkie piksele o wartości 0. Obraz w prawym dolnym rogu jest wynikiem algorytmu oraz finalnym szkieletem.

Kolejnym krokiem jest proces stworzenia szkieletu na obrazie wynikowym powstały po wstępny przetwarzaniu. Proces składa się z 6 etapów, które wykonywane są



**Rys. 4.16:** Lewy obraz: Przykładowy angiograficzny obraz OCT. Środkowy obraz: Wynik binaryzacji obrazu z progiem o wartości 180. Prawy obraz: Wynik operacji morfologicznych z elementem strukturalnym 3 na 3 piksele dla zamknięcia i 5 na 5 pikseli dla otwarcia.

pętli do momentu stworzenia szkieletu. Technika jest zobrazowana oraz wyjaśniona na rysunku 4.15.

## 2. Eksploracja szkieletu za pomocą *depth first search*

Szkielet obrazu angiograficznego OCT jest następnie wykorzystany do wykrycia położen naczyń krwionośnych. Proces rozpoczyna się od ustalenia położenia białych pikseli na krawędziach szkieletu (są to potencjalne miejsca, w których może zacząć się naczynie krwionośne). Następnie algorytm rozpoczyna eksploracje podążając białymi pikselami szkieletu z wykorzystaniem techniki *depth first search*<sup>3</sup>, dzięki czemu eksploracja wzdłuż naczynia krwionośnego przebiega szybciej. Algorytm wykrywa naczynie krwionośne jeżeli dojdzie do piksela  $p$  spełniającego warunki w zależności od początku drogi eksploracji – 4.10 dla początku w górnej lub dolnej krawędzi obrazu, 4.11 dla początku w lewej lub prawej krawędzi obrazu.

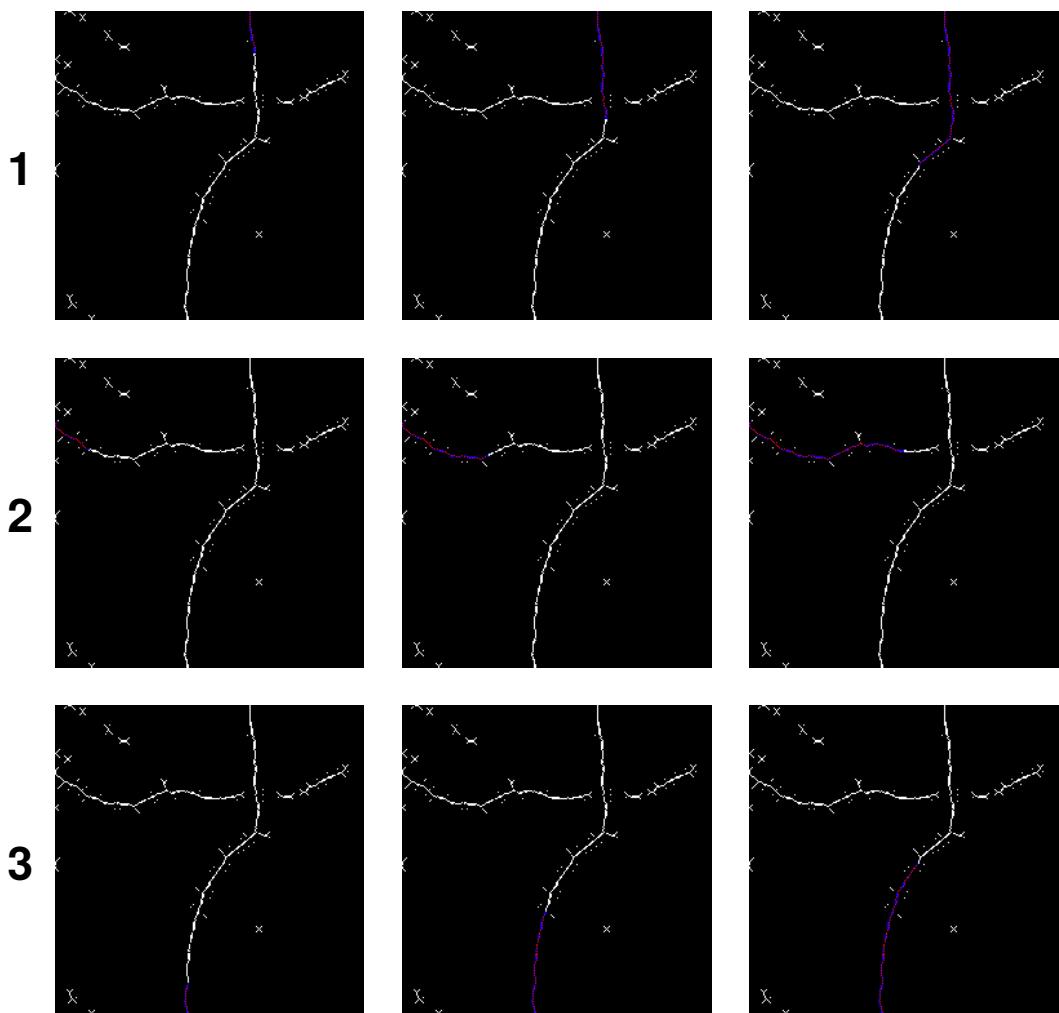
$$distance > imageSize * 0.5 \wedge |p_y - startp_y| > 0.1 * imageSize \quad (4.10)$$

$$distance > imageSize * 0.5 \wedge |p_x - startp_x| > 0.1 * imageSize \quad (4.11)$$

gdzie  $imageSize$  to rozmiar obrazu (w przypadku niniejszej pracy jest to 240 pikseli),  $startp$  to piksel od którego rozpoczęła się eksploracja, a  $distance$  to odległość euklidesowa pomiędzy  $p$ , a  $startp$ . Współrzędne  $p.x$ ,  $p.y$ ,  $startp_x$  i  $startp_y$  są wyrażone w układzie współrzędnych z początkiem w lewym górnym rogu obrazu.

Rysunek 4.17 przedstawia przykładową eksplorację szkieletu.

<sup>3</sup>[https://en.wikipedia.org/wiki/Depth-first\\_search](https://en.wikipedia.org/wiki/Depth-first_search)



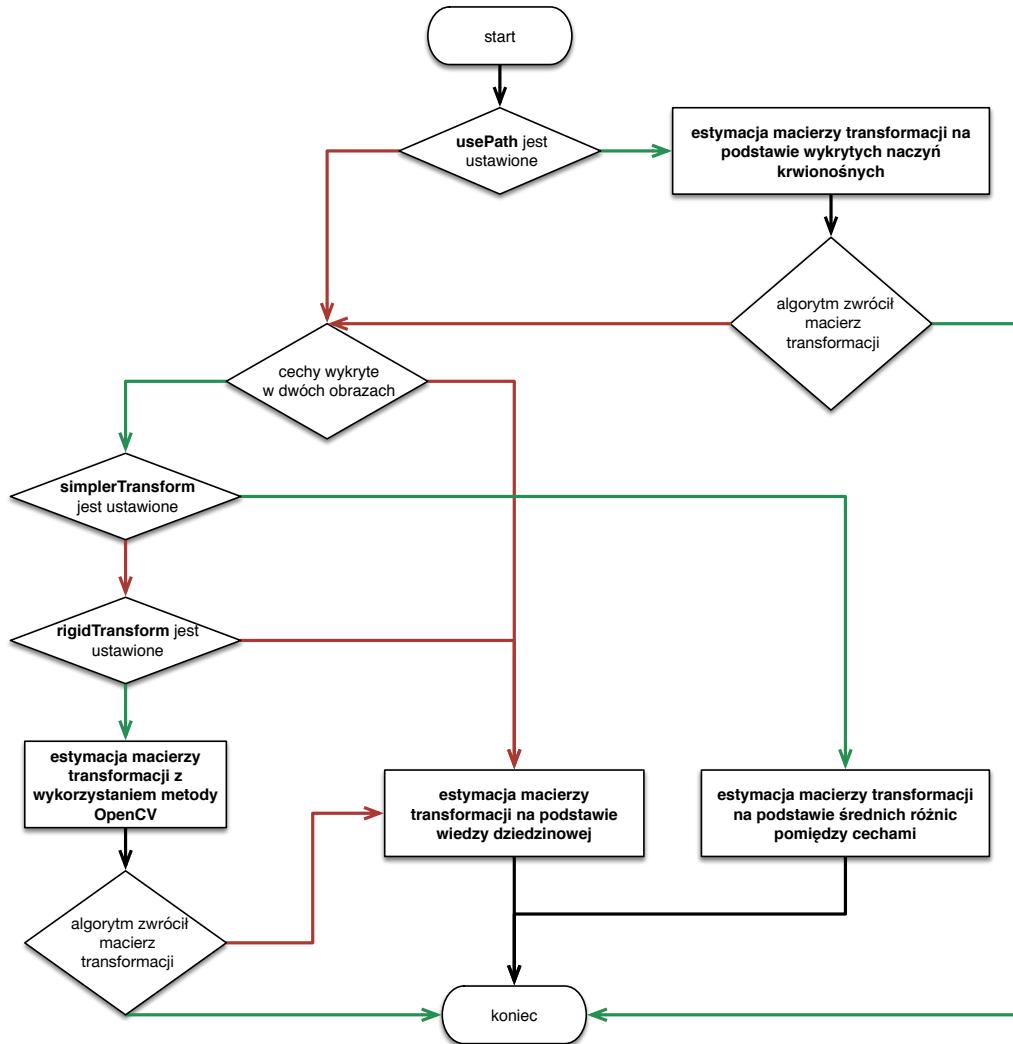
**Rys. 4.17:** Odkrywanie naczyń krwionośnych w różnych momentach działania algorytmu w zależności od początku eksploracji. Każdy rząd reprezentuje inne naczynie krwionośne. Wynik eksploracji jest przedstawiony w prawym obrazie na rysunku 4.13.

## 4.4 Estymacja macierzy transformacji pomiędzy kafelkami

W programie zaimplementowano cztery różne metody wyznaczające macierz transformacji pomiędzy dwoma kafelkami. Proces decyzyjny określający, która metoda zostanie użyta przez program opisano na początku niniejszej sekcji w sekcji 4.4.1. Następnie opisano każdą z czterech metod w sekcjach 4.4.2, 4.4.3, 4.4.4 i 4.4.5.

### 4.4.1 Proces decyzyjny

Wybór użytej metody do estymacji macierzy transformacji jest głównie zdeterminowany przez trzy zmienne w programie `simplerTransform` (odpowiada metodzie



**Rys. 4.18:** Schemat blokowy przedstawiający wybór metody estymacji macierzy transformacji w zależności od wewnętrznej zmiennej w programie (czerwone strzałki odpowiadają odpowiedzi ‘nie’, a zielone strzałki odpowiadają odpowiedzi ‘tak’).

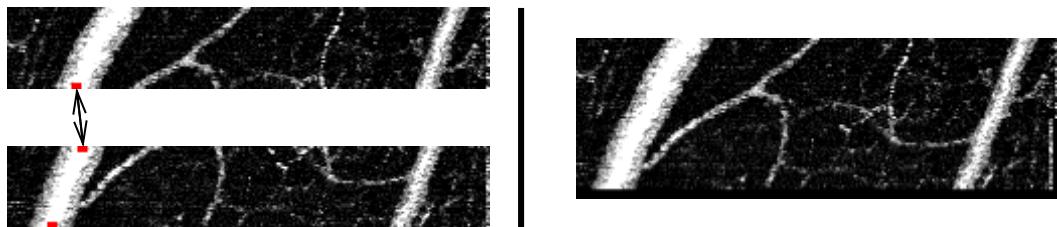
z sekcji 4.4.4), `rigidTransform` odpowiada metodzie z sekcji 4.4.3) i `usePaths` (opowiada metodzie z sekcji 4.4.2). Algorytm samodzielnie tworzy różne kombinacje wartości zmiennych by na wyjściu uzyskać kilka mozaik. Dzięki temu mechanizmowi użytkownik ma możliwość wybrania najlepszego rezultatu. Rysunek 4.18 obrazuje proces wyboru metody w zależności od ustawionych parametrów.

#### 4.4.2 Estymacja macierzy transformacji na podstawie wykrytych naczyń krwionośnych

Metoda wykrywająca naczynia krwionośne opisana w sekcji 4.3 została zaimplementowana ze względu na pojawienie się niestandardowego zbioru danych z kafelkami o rozmiarze 40 na 240 pikseli. Wyjściem metody jest zbiór punktów wskazujący

miejsce występowania naczynia krwionośnego na krawędzi obrazu, które przez resztę tej sekcji są nazywane punktami charakterystycznymi. Przykładowo punkt o współrzędnych (24, 0) informuje algorytm, że naczynie krwionośne zaczyna się lub kończy w tym punkcie.

Proces estymacji macierzy transformacji jest zależny od relacji pomiędzy dwoma kafelkami. Rysunek 4.19 przedstawia przykładową relację kafelek z zaznaczonymi wykrytymi naczyniami krwionośnymi oraz wynik transformacji wyznaczony macierzą.



**Rys. 4.19:** **Lewy obraz:** Przykładowe dwa obrazy z zaznaczonymi wykrytymi naczyniami krwionośnymi. Czarna strzałka wskazuje punkty charakterystyczne użyte do wyznaczenia macierzy transformacji. **Prawy obraz:** Wynik zaaplikowania wyznaczonej macierzy transformacji (z dodatkowym łączniem opisanym w sekcji 4.6).

By wyznaczyć macierz transformacji algorytm wybiera punkty charakterystyczne leżące najbliżej lewej krawędzi (dla relacji kafelków z rysunku 4.19) lub leżące najbliżej górnej krawędzi (dla kafelków ustawionych poziomo). Wybrane punkty charakterystyczne  $p_1$  i  $p_2$  należą do tego samego naczynia krwionośnego jeżeli różnica ich odpowiednich współrzędnych mieści się w ustalonym przedziale  $range$  – 4.12 dla ułożenia pionowego, 4.13 dla ułożenia poziomego.

$$|p_{1x} - p_{2x}| < range \quad (4.12)$$

$$|p_{1y} - p_{2y}| < range \quad (4.13)$$

Na podstawie zweryfikowanych punktów charakterystycznych macierz transformacji  $T$  wyznacza się następująco – 4.14 dla ułożenia pionowego, 4.15 dla ułożenia poziomego:

$$T = \begin{bmatrix} 1 & 0 & -(p_{1x} - p_{2x}) \\ 0 & 1 & height * (1 - overlap) \end{bmatrix} \quad (4.14)$$

$$T = \begin{bmatrix} 1 & 0 & width * (1 - overlap) \\ 0 & 1 & -(p_{1y} - p_{2y}) \end{bmatrix} \quad (4.15)$$

gdzie  $width$  to szerokość kafelka,  $height$  to jego wysokość, a  $overlap$  to miara szerokości obszaru nakładania się kafelków wyrażona w procentach.

#### 4.4.3 Estymacja macierzy transformacji z wykorzystaniem metody OpenCV na podstawie cech SIFT

OpenCV implementuje funkcję `estimateRigidTransform(...)`, która przyjmuje jako argumenty dwa zbiory punktów i na ich podstawie zwraca optymalną macierz transformacji (wewnątrz OpenCV używa metody *least squares*<sup>4</sup>). W programie funkcja jest wykorzystana w następujący sposób:

```
trans = estimateRigidTransform(imageOneKey, imageTwoKey, false)
```

gdzie `imageOneKey` to cechy SIFT z obrazu pierwszego, `imageTwoKey` to cechy SIFT z obrazu drugiego, a ostatni argument o wartości `false` powoduje, że użyty jest model ciała sztywnego (rotacja i translacja).

#### 4.4.4 Estymacja macierzy transformacji na podstawie średnich różnic pomiędzy cechami SIFT

W niniejszej metodzie do wyznaczenia macierzy jest liczona średnia różnica dopasowanych punktów w osi x 4.16 i osi y 4.17:

$$t_x = \frac{\sum_{i=1}^n k_{ix}^2 - k_{ix}^1}{n} \quad (4.16)$$

$$t_y = \frac{\sum_{i=1}^n k_{iy}^2 - k_{iy}^1}{n} \quad (4.17)$$

gdzie  $k_i^1$  to cecha SIFT  $i$  z obrazu pierwszego,  $k_i^2$  to cecha SIFT  $i$  z obrazu drugiego, a  $n$  to ilość dopasowań pomiędzy cechami. Macierz transformacji wyznacza się następująco:

$$T = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix} \quad (4.18)$$

#### 4.4.5 Estymacja macierzy transformacji na podstawie wiedzy dziedzinowej

Wyznaczenie macierzy transformacji na podstawie wiedzy dziedzinowej jest metodą najprostszą i jedynie bierze pod uwagę obszar nałożenia kafelków, który ustalany jest za pomocą parametru `percentOverlap` w pliku konfiguracyjnym. Macierz trans-

---

<sup>4</sup>[https://en.wikipedia.org/wiki/Least\\_squares](https://en.wikipedia.org/wiki/Least_squares)

formacji wyznacza się następująco – 4.20 dla ułożenia kafelków w poziomie, 4.19 dla ułożenia kafelków w pionie:

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & \text{imageSize} * (1 - \text{overlap}) \end{bmatrix} \quad (4.19)$$

$$T = \begin{bmatrix} 1 & 0 & \text{imageSize} * (1 - \text{overlap}) \\ 0 & 1 & 0 \end{bmatrix} \quad (4.20)$$

gdzie  $\text{imageSize}$  to rozmiar kafelka, a  $\text{overlap}$  to wartość parametru  $\text{percentOverlap}$ .

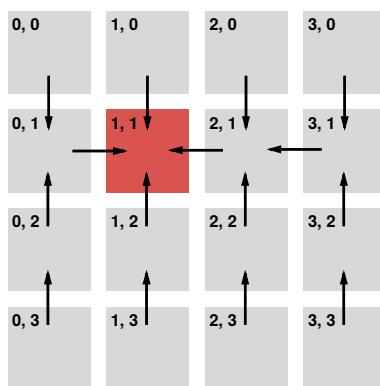
## 4.5 Globalna rejestracja kafelków

Następnym krokiem tworzenia mozaiki jest decyzja, który kafelek rejestrować z którym i w jakiej kolejności. Proces rozpoczyna się od wybrania kafelka referencyjnego wobec którego wszystkie inne mozaiki będą transformowane. Zakładając, że mozaika składa się z  $r$  wierszy i  $c$  kolumn współrzędne kafelka referencyjnego  $\text{ref}_x$  i  $\text{ref}_y$  są wybierane na podstawie:

$$\text{ref}_x = \left\lfloor \frac{c - 1}{2} \right\rfloor \quad (4.21)$$

$$\text{ref}_y = \left\lfloor \frac{r - 1}{2} \right\rfloor \quad (4.22)$$

Mając wyznaczony kafelek referencyjny następnym krokiem jest ustalenie kolejności rejestracji pomiędzy kafelkami. Proces wyjaśniony jest na przykładzie na rysunku 4.20.



**Rys. 4.20:** Przykład mozaiki składającej się z 4 wierszy i 4 kolumn kafelków. Kafelek referencyjny wybrany zgodnie z wzorami 4.21 i 4.22 oznaczono kolorem czerwonym. Rejestracja kafelków rozpoczyna się od porównywania kafelków parami na lewo i prawo od kafelka referencyjnego. Następnie od każdego kafelka z wiersza kafelka referencyjnego kafelki są porównywane parami idąc w górę i dół. W lewym górnym rogu każdego kafelka jest zapisana jego współrzędna w mozaice.

Posługując się przykładem z rysunku 4.20 oraz oznaczając  $T_{xy \rightarrow zw}$  jako macierz transformacji pomiędzy kafelkiem  $I_{xy}$  o współrzędnych  $x$  i  $y$  i kafelkiem  $I_{zw}$  o współrzędnych  $z$  i  $w$  macierz transformacji kafelka  $I_{33}$  wygląda następująco:

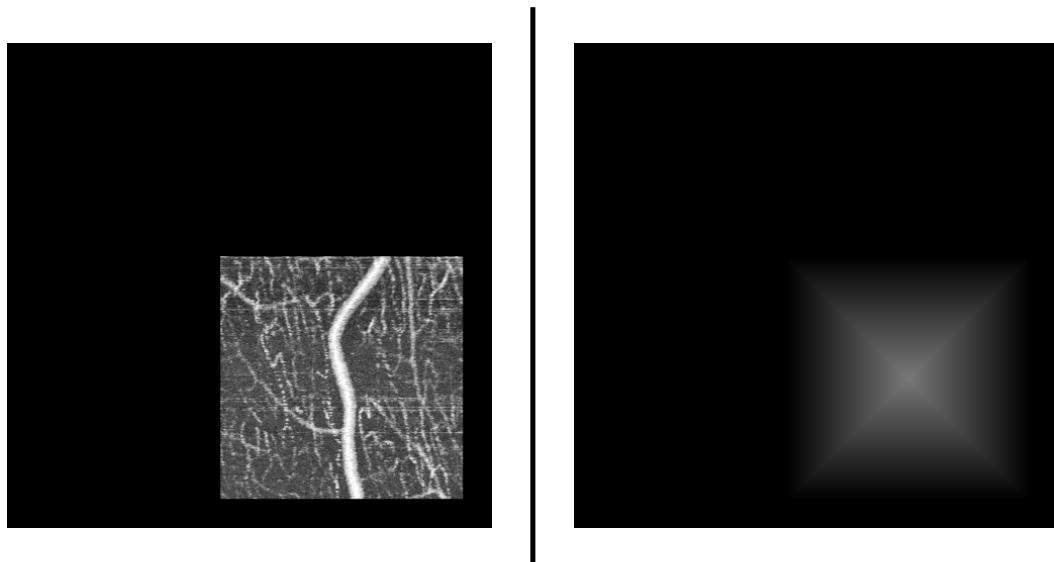
$$T_{33 \rightarrow 11} = T_{21 \rightarrow 11} T_{31 \rightarrow 21} T_{32 \rightarrow 31} T_{33 \rightarrow 32} \quad (4.23)$$

## 4.6 Łączenie kafelków

Ostatnim etapem w tworzeniu mozaiki jest wyświetlenie wszystkich kafelków po transformacji na wspólnej płaszczyźnie. Żeby mozaika spełniała wrażenie spójnej i jednolitej musi być zaimplementowany mechanizm, który będzie odpowiednio wybierał i ważył piksele w miejscach gdzie kafelki się nakładają. Wartość piksela finalnej mozaiki  $f$  jest obliczana za pomocą wzoru:

$$f(p) = \frac{\sum_{n=0}^{N-1} w_n(p) f_n(p)}{\sum_{n=0}^{N-1} w_n(p)} \quad (4.24)$$

gdzie  $N$  to liczba kafelków znajdujących się w miejscu piksela  $f(p)$ ,  $f_n(p)$  to wartość piksela  $p$  w kafelku  $n$ , a  $w_n(p)$  to waga piksela  $p$  w kafelku  $n$ . Wagi pikseli są obliczane za pomocą *distance transform*<sup>5</sup>, której wartość to odległość do najbliższego piksela o wartości 0. *Distance transform* jest zaimplementowane w OpenCV za pomocą funkcji `distanceTransform(...)` zwracającą obraz z obliczonymi wartościami dla całego obrazu. Rysunek 4.21 przedstawia wyjście funkcji na przykładowym obrazie.



**Rys. 4.21:** Lewy obraz: Przykładowy kafelek po transformacji. Prawy obraz: Wyjście funkcji `distanceTransform(...)`.

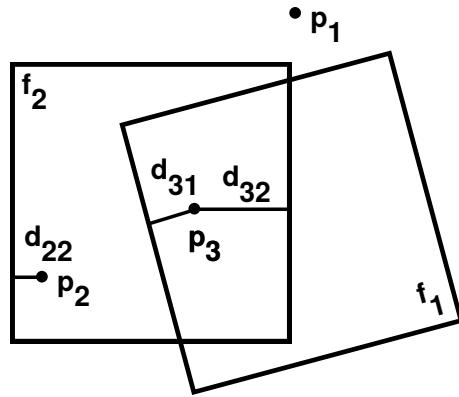
<sup>5</sup>[https://en.wikipedia.org/wiki/Distance\\_transform](https://en.wikipedia.org/wiki/Distance_transform)

Dzięki wykorzystaniu *distance transform* piksele znajdujące się bliżej środka kafelka będą bardziej istotne w przeciwnieństwie do pikseli znajdujących przy krawędziach obrazu. Rysunek 4.22 przedstawia przykładowe dwa kafelki z zaznaczonymi trzema punktami  $p_1$ ,  $p_2$  i  $p_3$  w finalnej mozaice. Punkt  $p_1$  nie należy do żadnego kafelka przez co jego wartość jest równa 0, punkt  $p_2$  należy do kafelka  $f_2$  i jego wartość jest równa:

$$f(p_2) = \frac{d_{22}f_2(p_2)}{d_{22}} \quad (4.25)$$

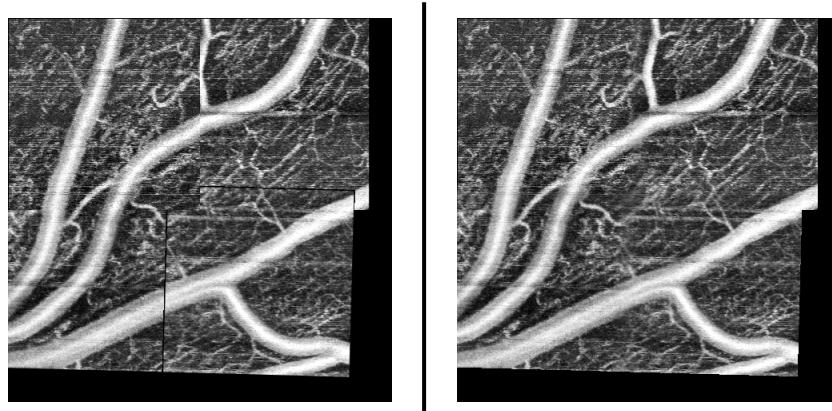
gdzie  $d_{nk}$  to wartość *distance transform* piksela  $p_n$  w obrazie  $f_k$ . Punkt  $p_3$  należy do kafelka  $f_2$  i  $f_3$  i jego wartość jest równa:

$$f(p_3) = \frac{d_{32}f_2(p_3) + d_{31}f_1(p_3)}{d_{32} + d_{31}} \quad (4.26)$$



**Rys. 4.22:** Przykładowy schemat łączenia dwóch kafelków  $f_1$  i  $f_2$  z zaznaczonymi punktami  $p_n$  oraz wartościami *distance transform*  $d_{nk}$ .

Rysunek 4.23 przedstawia wynik łączenia kafelków niniejszą metodą na przykładowej mozaice.



**Rys. 4.23:** Lewy obraz: Mozaika stworzona bez wykorzystania łączenia. Prawy obraz: Mozaika stworzona z wykorzystaniem łączenia.

# Oprogramowanie

Program o nazwie `mostitch`, który jest celem niniejszej pracy jest całkowicie napisany w języku C++. Język wybrano ze względu na to, że wykorzystywana biblioteka przetwarzania obrazów OpenCV (opisana w skrócie w sekcji 5.3) posiada interfejs w języku C++. Program można ściągnąć z repozytorium<sup>1</sup>, a następnie zainstalować postępując zgodnie z instrukcją napisaną w pliku `README.md`. Po udanej instalacji program można uruchomić za pomocą komendy:

```
mostitch path_to_config_file
```

gdzie `path_to_config_file` to obowiązkowy argument do programu wskazujący ścieżkę do pliku konfiguracyjnego (opisanego dokładniej w sekcji 5.1), który określa wszystkie parametry niezbędne do prawidłowego działania programu.

Program umożliwia konstrukcję mozaik dla dowolnej ilości zbiorów kafelków. Dodatkowo dla każdego zbioru kafelków powstają różne wersje mozaik, ze względu na wykorzystanie różnych metod ich tworzenia. Dzięki tej funkcjonalności użytkownik może wybrać najbardziej odpowiednią mozaikę. Wyjściem programu więc jest zbiór mozaik (obrazy `.png`), w którym każda mozaika ma przypisaną metodę jej tworzenia oraz odpowiadający zbiór kafelków.

## 5.1 Plik konfiguracyjny

Do zarządzania plikiem konfiguracyjnym wykorzystano bibliotekę `libconfig`<sup>2</sup> umożliwiającą wygodny odczyt pliku konfiguracyjnego z rozszerzeniem `.cfg`. Format pliku konfiguracyjnego jest bardziej czytelny w porównaniu do powszechnie wykorzystywanych plików XML. Biblioteka również jest świadoma typu zmiennej przez co unika się konwersji typu `string` na typy takie jak `int`, czy `float`.

W pliku konfiguracyjnym zawarte są informacje:

- Ścieżka do miejsca z folderami zawierającymi zbiory kafelków do złączenia.

<sup>1</sup><http://git.tesla.cs.put.poznan.pl/agrzyb/mostitch/tree/master>

<sup>2</sup><http://www.hyperrealm.com/libconfig/>

- Ścieżka do miejsca, w którym będą zapisane wynikowe mozaiki.
- Parametry pozwalające na automatyczne wczytanie obrazów kafelków.
- Parametry modyfikujące działanie metod przetwarzania kafelków.

Wszystkie parametry zawarte w pliku konfiguracyjnym są szczegółowo opisane w przykładzie pliku konfiguracyjnego dołączonego do niniejszej pracy o nazwie config.cfg. Poniżej znajduje się fragment pliku konfiguracyjnego config.cfg:

```
mosaicsDirectoryAbsolutePath = "/Users/thesis/mosaics";
numberOfMosaics = 4;
tilesBaseName = "tile_XX_YX.tif";
mosaicsSaveAbsolutePath = "/Users/thesis/output_images";
angleParameter = 1.00;
percentOverlap = 0.12;
shiftParameter = 0.10;
```

## 5.2 Dokumentacja

Dokumentacje programu przygotowano za pomocą narzędzia do generacji dokumentacji **Doxxygen**<sup>3</sup>. Żeby wyświetlić dokumentację należy otworzyć plik index.html w przeglądarce znajdujący się w folderze docs w repozytorium<sup>4</sup> programu.

## 5.3 OpenCV

Wszystkie rozwiązania zaimplementowane w niniejszej pracy bazują na bibliotece przetwarzania obrazów o nazwie **OpenCV**. Biblioteka jest dozwolona do bezpłatnego wykorzystania w projektach prywatnych i komercyjnych. OpenCV jest używane w ogromnej ilości projektów z różnych dziedzin i bibliotekę ściągnięto już ponad 9 milionów razy<sup>5</sup>. OpenCV cieszy się taką popularnością ze względu na szybkość działania, implementację większości metod przetwarzania obrazu oraz umiejętnością pracy na wielu rdzeniach. Kod źródłowy biblioteki jest dostępny publicznie w serwisie Github<sup>6</sup> przez co każdy może rozwijać OpenCV i ma wgląd do zaimplementowanych metod.

---

<sup>3</sup><http://www.stack.nl/~dimitri/doxygen/>

<sup>4</sup><http://git.tesla.cs.put.poznan.pl/agrzyb/mostitch/tree/master>

<sup>5</sup><http://opencv.org>

<sup>6</sup><https://github.com/Itseez/opencv>

# Wyniki

W niniejszym rozdziale przedstawiono wynik działania programu `mostitch` na trzech przykładowych zbiorach angiograficznych obrazów OCT dostarczonych w ramach projektu RIMO-BIOL (sekcja 1.1.1). Wynik działania programu na zbiorze to cztery mozaiki powstałe za pomocą różnych kombinacji wartości parametrów (proces opisano w 4.4.1):

1. **Wersja 1** – `simplerTransform = false, rigidTransform = true, usePaths = false.`
2. **Wersja 2** – `simplerTransform = true, rigidTransform = true, usePaths = true.`
3. **Wersja 3** – `simplerTransform = true, rigidTransform = true, usePaths = false.`
4. **Wersja 4** – `simplerTransform = true, rigidTransform = false, usePaths = false.`

## 6.1 Zbiór 1

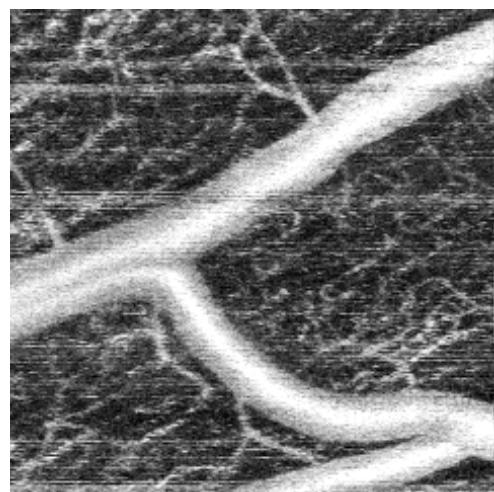
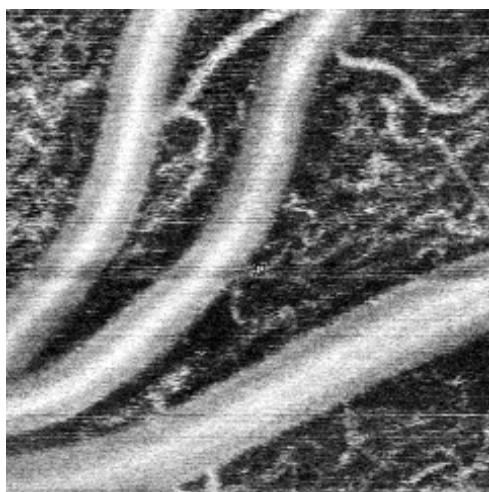
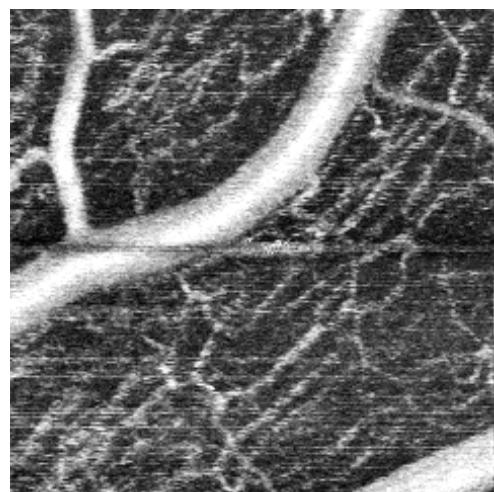
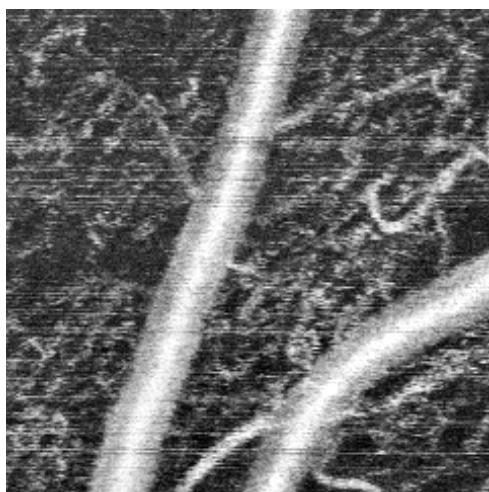
Rysunek 6.1 przedstawia zbiór angiograficznych obrazów OCT, natomiast rysunek 6.2 przedstawia wynik działania programu `mostitch`.

## 6.2 Zbiór 2

Rysunek 6.3 przedstawia zbiór angiograficznych obrazów OCT, natomiast rysunek 6.4 przedstawia wynik działania programu `mostitch`.

## 6.3 Zbiór 3

Rysunek 6.5 przedstawia zbiór angiograficznych obrazów OCT, natomiast rysunek 6.6 przedstawia wynik działania programu `mostitch`.



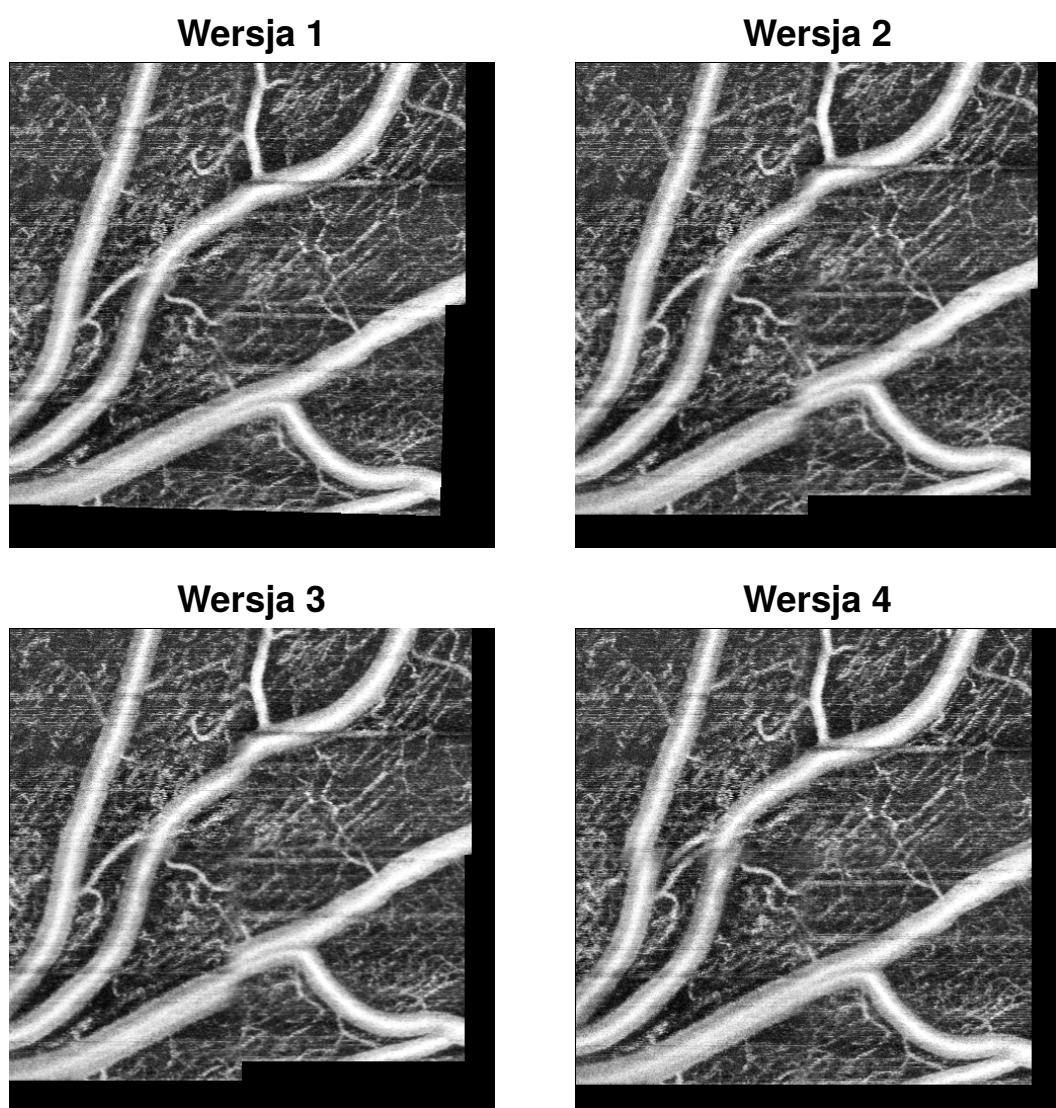
**Rys. 6.1:** Przykładowy zbiór angiograficznych obrazów OCT umieszczonych zgodnie z ich współrzędnymi. Obraz referencyjny znajduje się w prawym górnym rogu.

## 6.4 Ocena wyników

W niniejszej sekcji przeprowadzono analizę wyników rozpoczęając od oceny wizualnej ‘na oko’ w sekcji 6.4.1, następnie mozaiki oceniano poprzez policzenie średniego odchylenia standardowego w miejscach nałożenia kafelków w sekcji 6.4.2. Czas wykonywania programu `mostitch` przedstawiono w sekcji 6.4.3. Na koniec podsumowano wyniki w sekcji 6.4.4.

### 6.4.1 Ocena wizualna

Na podstawie zaprezentowanych wyników dla trzech przykładowych zbiorów trudno wybrać wersję sprawdzającą się najlepiej dla każdego zbioru. Dla zbioru pierwszego z sekcji 6.1 najlepszą wersją jest wersja pierwsza, natomiast dla zbioru drugiego z

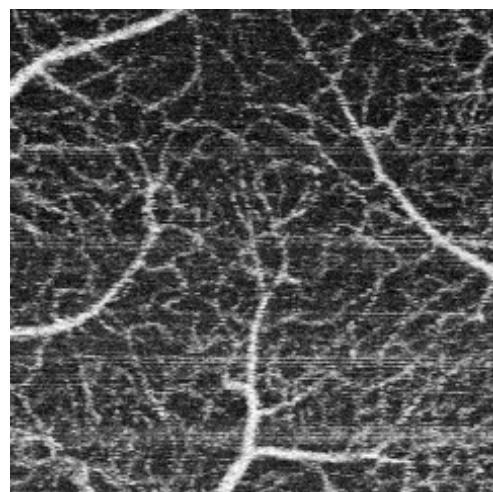
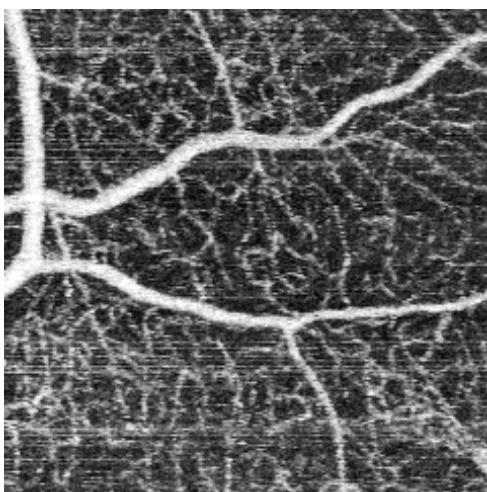
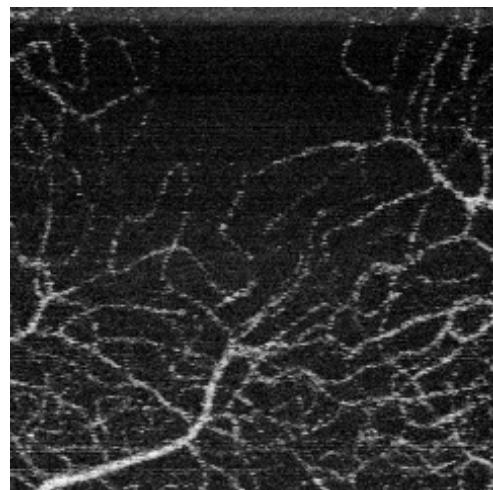
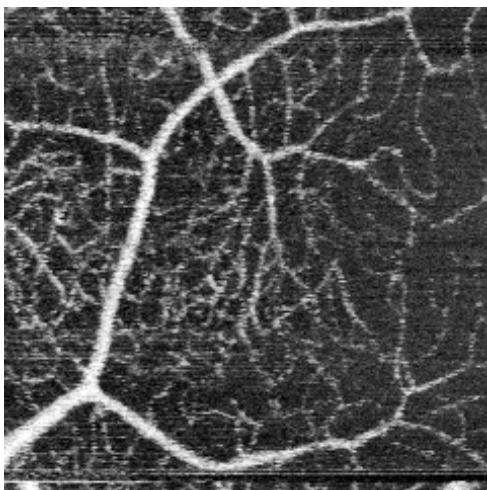


**Rys. 6.2:** Cztery mozaiki będą wynikiem działania programu `mostitch` na zbiorze obrazów OCT z rysunku 6.1.

sekcji 6.2 najlepiej prezentującym się wynikiem jest wersja druga lub trzecia. Dla zbioru trzeciego z sekcji 6.3 najlepiej wygląda mozaika stworzona wersją drugą lub trzecią.

#### 6.4.2 Ocena na podstawie średniego odchylenia standardowego

Chcąc stworzyć wiarygodną miarę jakości końcowej mozaiki trzeba najpierw zdefiniować wzorzec, czyli mozaikę złożoną idealnie. W przypadku tematu niniejszej pracy definicja takiego wzorca jest problemem nietrywialnym ze względu na unikalne artefakty występujące w obrazach. Nie jest możliwe by nałożyć kafelki na siebie w taki sposób by piksele idealnie się pokrywały pod względem wartości. Z tego względu zdecydowano się oceniać jakość mozaiki  $Q$  na podstawie średniego



**Rys. 6.3:** Przykładowy zbiór angiograficznych obrazów OCT umieszczonych zgodnie z ich współrzędnymi. Obraz referencyjny znajduje się w prawym górnym rogu.

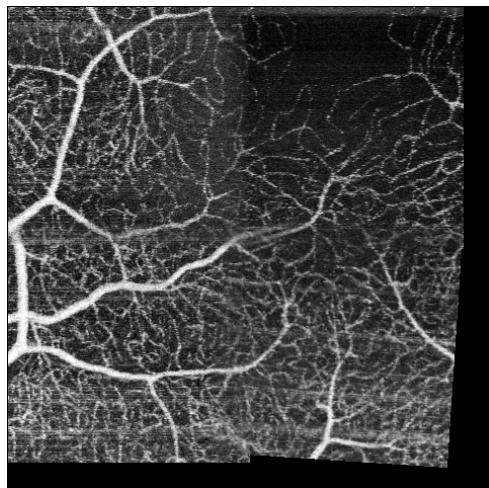
odchylenia standardowego wartości pikseli w  $n$  miejscach gdzie kafelki nakładają się na siebie w mozaice:

$$Q = \frac{\sum_{i=1}^n \sigma_i}{n} \quad (6.1)$$

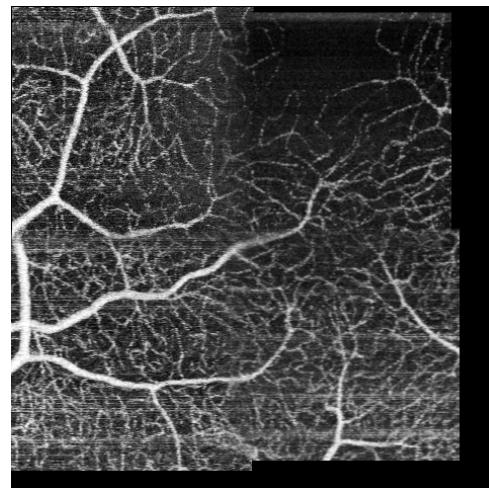
gdzie  $\sigma_i$  to wartość odchylenia standardowego wartości pikseli kafelków nakładających się w miejscu  $i$ . Wykres na rysunku 6.7 przedstawia wartość jakości finalnych mozaik przedstawionych w niniejszym rozdziale dla czterech wersji tworzenia mozaiki.

Wartości jakości nie pokrywają się z oceną wizualną (sekcja 6.4.1). Jedynie dla zbioru trzeciego wartość jakości oddaje jakość rzeczywistą mozaiki. Stworzenie wiarygodnej miary jakości mozaiki jest problemem nietrywialnym i wymaga wzięcia pod uwagę wielu czynników. Warto również zauważyć, że dla każdego zbioru jakość wersji drugiej i trzeciej jest taka sama. Wersja druga i trzecia różni się występowaniem w procesie algorytmu detekcji naczyń krwionośnych (sekcja 4.3). W wersji drugiej,

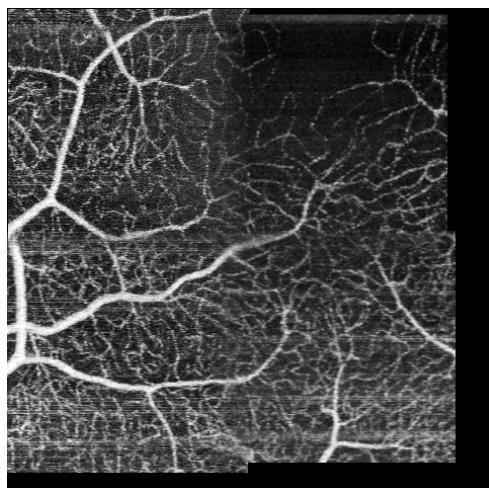
**Wersja 1**



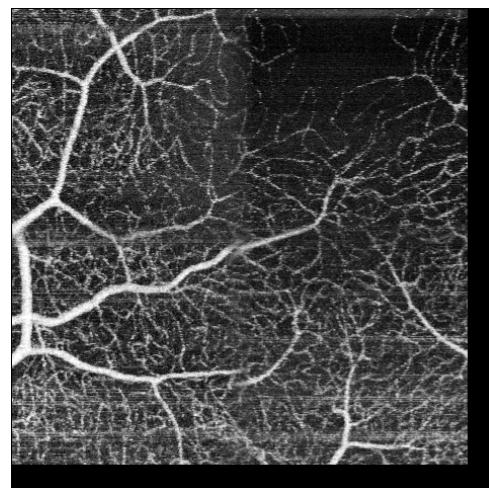
**Wersja 2**



**Wersja 3**



**Wersja 4**



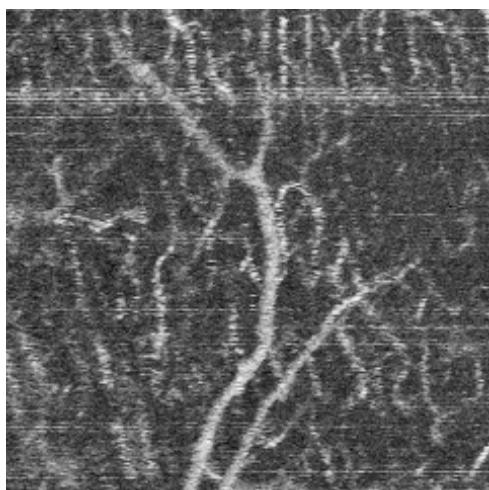
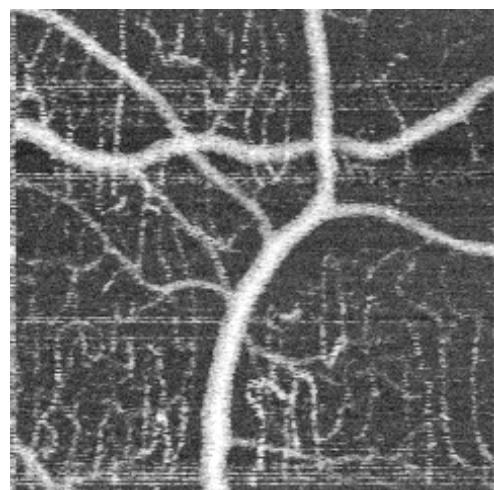
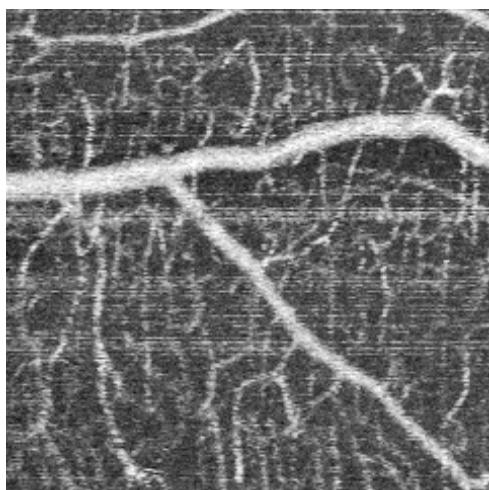
**Rys. 6.4:** Cztery mozaiki będą wynikiem działania programu `mostitch` na zbiorze obrazów OCT z rysunku 6.3.

w której występuje, algorytm najwyraźniej nie zwrócił macierzy transformacji na podstawie wykrytych naczyń krwionośnych, przez co wersja druga jest identyczna do wersji trzeciej (proces dokładniej opisano w sekcji 4.4.1).

### 6.4.3 Ocena na podstawie czasu wykonywania

Wykres na rysunku 6.8 przedstawia czas wykonywaniu programu `mostitch` na zbiorach przedstawionych w niniejszym rozdziale dla czterech wersji tworzenia mozaiki.

Program `mostitch` najszybciej ( $0.333s$ ) stworzył mozaikę ze zbioru trzeciego za pomocą wersji czwartej. Na podstawie wyników zaprezentowanych na rysunku 6.8 trudno wybrać wersję, która jest najszybsza, albo najwolniejsza. Dla każdej wersji



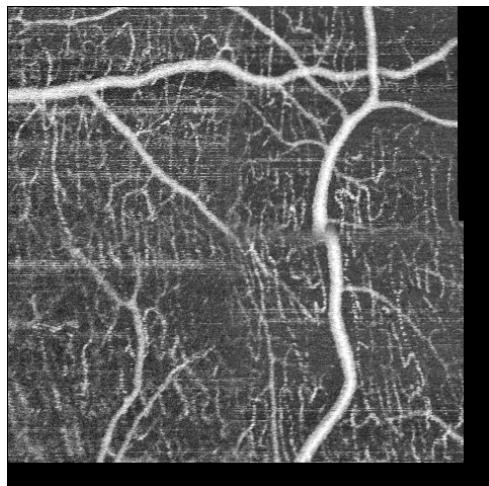
**Rys. 6.5:** Przykładowy zbiór angiograficznych obrazów OCT umieszczonych zgodnie z ich współrzędnymi. Obraz referencyjny znajduje się w prawym górnym rogu.

tworzenia mozaiki większość kosztownych obliczeń (np. odkrywanie cech SIFT, filtrowanie dopasowań) jest wykonywana (pomimo różnie ustawionych parametrów), z tego względu czasy wykonywania wersji są do siebie podobne. Czasy wykonywania programu pomiędzy różnymi zbiorami również są bardzo podobne. Ta kwestia nie powinna dziwić, ponieważ każdy kafelek jest tego samego rozmiaru.

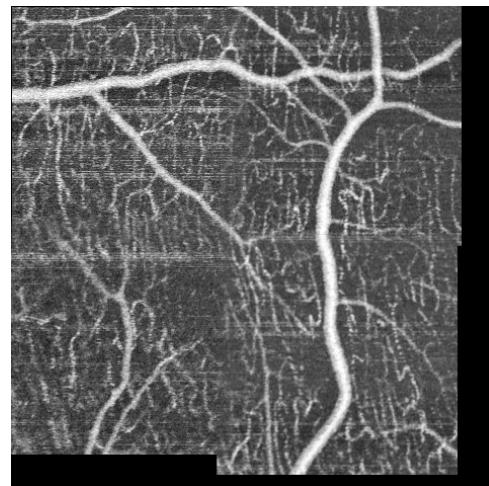
#### 6.4.4 Podsumowanie

Podsumowując analiza wizualna (sekcja 6.4.1), oraz analiza na podstawie czasu wykonania (sekcja 6.4.3) nie wyznaczyły jednoznacznie wersji, która osiąga najlepsze rezultaty. Analiza z wykorzystaniem średniego odchylenia standardowego (sekcja 6.4.2) wyłoniła na faworytów wersję drugą oraz trzecią (najmniejsza wartość średniego odchylenia standardowego dla zbioru pierwszego i trzeciego), natomiast nie pokrywa się z rzeczywistą jakością mozaiki. Dla zbioru pierwszego wartość

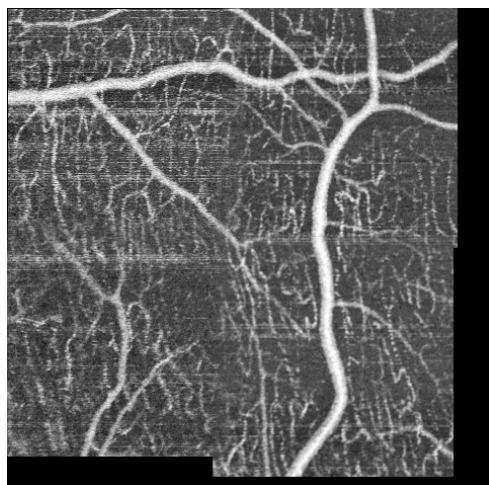
**Wersja 1**



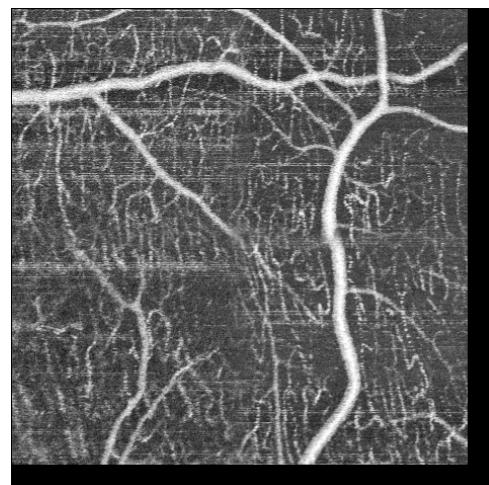
**Wersja 2**



**Wersja 3**

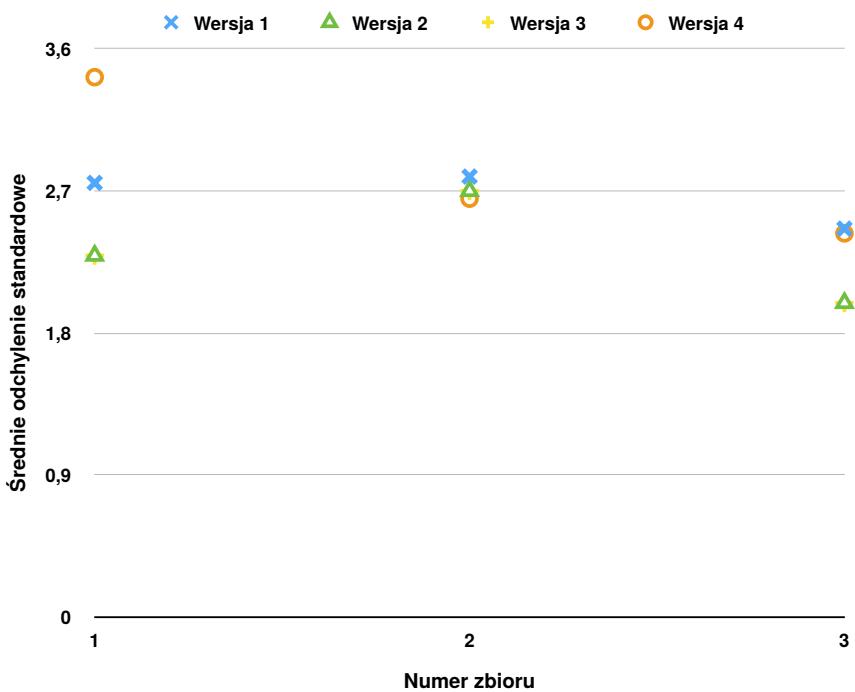


**Wersja 4**

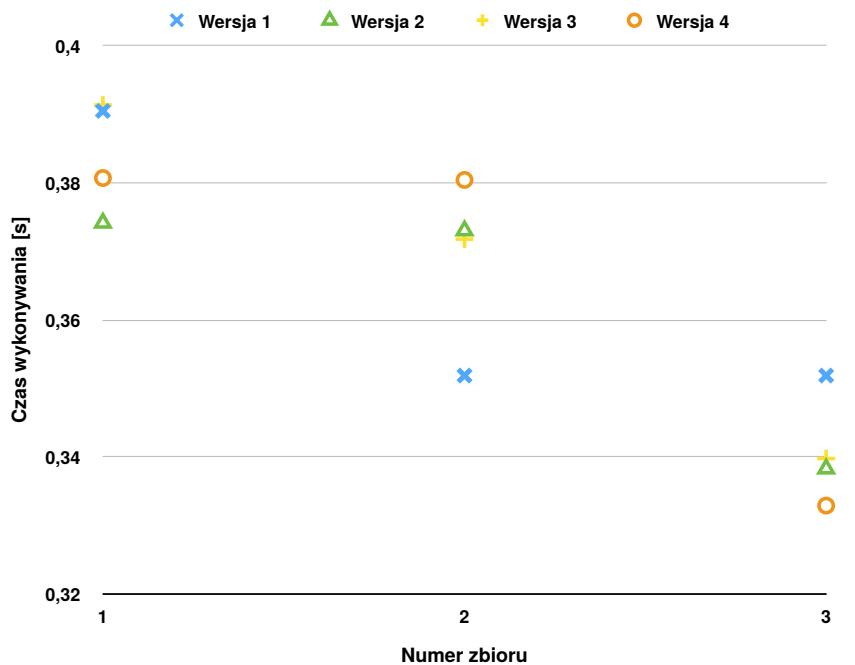


**Rys. 6.6:** Cztery mozaiki będą wynikiem działania programu `mostitch` na zbiorze obrazów OCT z rysunku 6.5.

średniego odchylenia standardowego była najmniejsza dla wersji drugiej i trzeciej, natomiast bezkonkurencyjnie wersja pierwsza stworzyła najlepszą mozaikę. Na podstawie trzech przeprowadzonych analiz nie jest możliwym wyłonić najlepszą wersję. Rozbieżność rezultatów dla różnych wartości parametrów jest głównym powodem tworzenia czterech różnych wersji mozaik przez program `mostitch`. Dzięki takiemu rozwiązaniu osoba z wiedzą ekspercką jest w stanie stwierdzić, który obraz nadaje się najlepiej do wybranego zastosowania. Co więcej, otrzymanie czterech różnych wynikowych mozaik oszczędza czas uruchamiania programu z różnymi wartościami parametrów.



**Rys. 6.7:** Jakość finalnych mozaik przedstawionych w sekcjach 6.1, 6.2 i 6.3 (im mniej tym lepiej).



**Rys. 6.8:** Czas wykonywania programu mostitch na zbiorach przedstawionych w sekcjach 6.1, 6.2 i 6.3.

# Podsumowanie i wnioski końcowe

Cel niniejszej pracy w pełni zrealizowano. Mozaiki obrazów angiograficznych stworzone za pomocą programu `mostitch` wyglądają prawidłowo oraz spełniają kryteria jakościowe. Cele szczegółowe z sekcji 1.1.2 również osiągnięto:

1. Zapoznano się z istniejącymi metodami i algorytmami realizującymi *stitching*. Finalnie wybrano korejestrację na podstawie cech SIFT z autorskimi modyfikacjami.
2. Wybrano jedną z najbardziej popularnych bibliotek przetwarzania obrazów OpenCV. Środowiskiem deweloperskim był Xcode (zintegrowane środowisko programistyczne firmy Apple Inc.).
3. Oprogramowanie napisano w języku C++. Program przetestowano na testowych zbiorach obrazów angiograficznych OCT dostarczonych w ramach projektu RIMO-BIOL.
4. Program `mostitch` zaimplementowano i jest dostępny do ściągnięcia z repozytorium<sup>1</sup>. Instrukcja instalacji znajduje się w tym samym repozytorium w pliku README.md.
5. Dokumentację oprogramowania przygotowano za pomocą narzędzia Doxygen (sekcja 5.2).

## 7.1 Napotkane trudności

Podczas realizacji celu niniejszej pracy zdecydowanie najwięcej czasu spędzono na implementacji oraz testowaniu programu `mostitch`. W większości praca była usystematyzowana i poszczególne zadania realizowanie zgodnie z założonym harmonogramem prac. W trakcie tworzenia programu `mostitch` można wyznaczyć dwa przypadki w których praca trwała dłużej niż to pierwotnie zakładano:

1. Bardzo ważnym czynnikiem wpływającym na rezultat oraz przyszłość pracy był wybór algorytmu detekcji cech w obrazach angiograficznych. Po zapoznaniu

<sup>1</sup><http://git.tesla.cs.put.poznan.pl/agrzyb/mostitch/tree/master>

się z większością dostępnych algorytmów ostatecznie zdecydowana się na algorytm SIFT, który pozwolił uzyskać bardzo dobre wyniki.

2. Na początku zakładano, że obrazy angiograficzne będą w rozmiarze 240 na 240 pikseli. Na tym założeniu wybrano algorytm SIFT oraz inne rozwiązania. W trakcie projektu natomiast pojawił się zbiór z obrazami 24 na 240 pikseli. Wykorzystanie wówczas zaimplementowanego algorytmu dało bardzo złe rezultaty, przez co trzeba było stworzyć nowe metody i rozwiązania. Powstał autorski algorytm opisany w sekcji 4.3.

## 7.2 Możliwości rozwoju

Program `mostitch` ma duży potencjał i poprzez modularną budowę jego elementy mogą być wykorzystane w innych projektach w przyszłości. Na obecną chwilę istnieje kilka pomysłów na jego ulepszenia i rozwój:

- Dołączenie innych algorytmów wykrywających cechy w angiograficznych obrazach OCT (np. SURF, FAST), a następnie porównanie wynikowych mozaik.
- Implementacja programu okienkowego, w którym użytkownik poprzez przyjazny interfejs ładowałby angiograficzne obrazy z dysku, a następnie przeciągając obrazy ustalałby ich relacje przestrzenne. Takie rozwiązanie wyeliminowałoby plik konfiguracyjny, który jest mniej przyjazny dla użytkownika.
- W trakcie tworzenia pracy zdecydowano się na model transformacji bryły sztywnej (translacja i rotacja) ze względu na to, że operacja skalowania generowała zbyt dużo błędów. Czasem jednak lekkie skalowanie jest niezbędne by uzyskać pożądany wynik. Dodanie operacji skalowania, której parametry byłyby mocno ograniczone i kontrolowane przyczyniłoby się do poprawy jakości mozaik.

# Bibliografia

- [2]Herbert Bay, Andreas Ess, Tinne Tuytelaars i Luc Van Gool. „Speeded-Up Robust Features (SURF)“. W: *Comput. Vis. Image Underst.* 110.3 (czer. 2008), s. 346–359.
- [3]Jeffrey S. Beis i David G. Lowe. „Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces“. W: *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*. CVPR '97. Washington, DC, USA: IEEE Computer Society, 1997, s. 1000–.
- [4]Matthew Brown i David G. Lowe. „Automatic Panoramic Image Stitching Using Invariant Features“. W: *Int. J. Comput. Vision* 74.1 (sierp. 2007), s. 59–73.
- [5]Peter J. Burt i Edward H. Adelson. „A Multiresolution Spline with Application to Image Mosaics“. W: *ACM Trans. Graph.* 2.4 (paź. 1983), s. 217–236.
- [6]J Canny. „A Computational Approach to Edge Detection“. W: *IEEE Trans. Pattern Anal. Mach. Intell.* 8.6 (czer. 1986), s. 679–698.
- [7]A F Fercher, W Drexler, C K Hitzenberger i T Lasser. „Optical coherence tomography - principles and applications“. W: *Reports on Progress in Physics* 66.2 (2003), s. 239.
- [8]Martin A. Fischler i Robert C. Bolles. „Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography“. W: *Commun. ACM* 24.6 (czer. 1981), s. 381–395.
- [9]Welzel J. „Optical coherence tomography in dermatology: a review.“ W: *Skin. Res. Technol.* 7.1 (2001).
- [10]Martin F. Kraus, Benjamin Potsaid, Markus A. Mayer, Ruediger Bock, Bernhard Baumann, Jonathan J. Liu, Joachim Hornegger i James G. Fujimoto. „Motion correction in optical coherence tomography volumes on a per A-scan basis using orthogonal scan patterns“. W: *Biomed. Opt. Express* 3.6 (2012), s. 1182–1199.
- [11]Ivan Laptev, Barbara Caputo, Christian Schüldt i Tony Lindeberg. „Local Velocity-adapted Motion Events for Spatio-temporal Recognition“. W: *Comput. Vis. Image Underst.* 108.3 (grud. 2007), s. 207–229.
- [12]David G. Lowe. „Distinctive Image Features from Scale-Invariant Keypoints“. W: *Int. J. Comput. Vision* 60.2 (list. 2004), s. 91–110.
- [13]D. Markl, G. Hanneschläger, M. Leitner, S. Sacher, D. Koller i J. Khinast. *A device and a method for monitoring a property of a coating of a solid dosage form during a coating process forming the coating of the solid dosage form*. EP Patent App. EP20,140,166,452. 2014.

- [14] Andrew M. Rollins, Rujchai Ung-arunyawee, Amitabh Chak, Richard C. K. Wong, Kenji Kobayashi, Michael V. Sivak i Joseph A. Izatt. „Real-time in vivo imaging of human gastrointestinal ultrastructure by use of endoscopic optical coherence tomography with a novel efficient interferometer design“. W: *Opt. Lett.* 24.19 (1999), s. 1358–1360.
- [15] Daniel Ruminski, Bartosz L. Sikorski, Danuta Bukowska, Maciej Szkulmowski, Krzysztof Krawiec, Grazyna Malukiewicz, Lech Bieganowski i Maciej Wojtkowski. „OCT angiography by absolute intensity difference applied to normal and diseased human retinas“. W: *Biomed. Opt. Express* 6.8 (2015), s. 2738–2754.
- [16] Stephen Se, David G. Lowe i James J. Little. „Vision-based Mobile Robot Localization And Mapping using Scale-Invariant Features.“ W: *ICRA*. IEEE, 2001, s. 2051–2058.
- [18] Richard Szeliski. *Image Alignment and Stitching: A Tutorial*. Spraw. tech. MSR-TR-2004-92. Microsoft Research, 2004, s. 89.
- [19] P. Thévenaz i M. Unser. „User-Friendly Semiautomated Assembly of Accurate Image Mosaics in Microscopy“. W: *Microscopy Research and Technique* 70.2 (2007), s. 135–146.
- [20] W.J. Walecki i P. Van. *Determining thickness of slabs of materials by inventors*. US Patent 7,116,429. 2006.

## Strony internetowe

- [@1] Félix Abecassis. *OpenCV - Morphological Skeleton*. 2011. URL: <http://felix.abecassis.me/2011/09/opencv-morphological-skeleton/>.
- [@17] James Strong. *Retinal OCT Imaging*. 2011. URL: <http://www.opsweb.org/?page=RetinalOCT>.

