AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

KATEDRA TELEKOMUNIKACJI

PRACA DYPLOMOWA INŻYNIERSKA

Urządzenie do wyznaczania widma światła białego

The application for spectrum estimation of white light sources

Autor: *Agnieszka Job*
Kierunek studiów: Elektronika i Telekomunikacja
Opiekun pracy: *dr inż. Jarosław Bułat*

Kraków, 2016

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „ Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystyczne wykonanie albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.”, a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.) „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej „sądem koleżeńskim”, oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście, samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

_____

Spis treści

Wstęp

Człowiek nauczył się funkcjonować w warunkach nie zależnych od pory dnia lub nocy. Umożliwiły mu to, wszechobecne w dzisiejszych czasach sztuczne źródła światła. Na rynku mamy dostępne różne rodzaje „zamienników" Słońca. Najczęściej używanymi przez nas są żarowe źródła światła, w tym halogenowe, oparte o białe diody LED lub fluorescencyjne (świetlówki kompaktowe). Jednak żaden substytut nie odwzoruje w stu procentach oryginału w zakresie takich parametrów jak odcień barwy, intensywność oraz jakość widzianych kolorów w danym świetle.

Aby opisać właściwości światła oraz określić jak bardzo zbliżone ono jest do promieniowania słonecznego, wykorzystuje się takie parametry jak współczynnik oddawania barw oraz temperatura barwowa. Wskaźnik oddawania barw oznaczany jako CRI, niesie informację o tym, w jakim stopniu dane źródło światła umożliwia obserwację kolorów [2]. Temperatura barwowa jest obiektywną miarą wyrażania barwy danego źródła światła białego [2]. Wartości tych parametrów mają wpływ na jakość odbieranych bodźców wzrokowych przez człowieka. Do określenia tych parametrów potrzebny jest rozkład widmowy informujący o zakresie fal, które są emitowane przez źródło światła. Widmo światła można uzyskać dzięki spektrofotometrowi.

Dodatkowo analizę spektralną stosuje się w astronomii oraz chemii. Wykorzystuje się wtedy widmo atomowe powstające na skutek przejścia elektronów pomiędzy poziomami energetycznymi w atomie. Technikę tą nazywamy spektralną analizą widmową. Polega ona na tym, że analizuje się widma emisyjne umożliwiające zidentyfikowanie zawartości substancji w podgrzanej próbce.

Istnieją profesjonalne oraz drogie urządzenia badające widmo. Jednak zadaniem tej pracy jest zaprojektowanie rozwiązania składającego się z dostępnych i tanich elementów, które będzie mogło być zbudowane oraz zastosowane przez każdego w domowym zaciszu.

Pierwszym etapem tego projektu było zbudowanie urządzenia pomiarowego składającego się z kamery internetowej oraz siatki dyfrakcyjnej. Następnie zaprojektowanie i wykonanie programu dla systemu operacyjnego Linux do akwizycji, wizualizacji i obróbki widma światła białego. Aplikacja powinna działać poprawnie z spektrometrami zbudowanymi w oparciu o różne kamery internetowe oraz zawierać możliwość kalibracji. Taki projekt jest gotowy do prawidłowego wykonania pomiarów, co więcej składa się z tanich i dostępnych elementów, z których można łatwo zrobić spektrofotometr. Obraz wychwytywany z kamery internetowej zostaje przekształcony w taki sposób, aby znajdujące się w nim widmo zostało odseparowane od zbędnego tła. Następnie na podstawie jasności pikseli określane jest spektrum. Istnieje również możliwość dalszego rozwoju oprogramowania o nowe funkcjonalności.

Praca składa się z trzech rozdziałów zawierających opis poszczególnych etapów tworzenia układu pomiarowego oraz oprogramowania. W pierwszym rozdziale przedstawiono warianty spektrometrów oraz wytłumaczono zasadę ich działania. Kolejna część pracy została poświęcona na opisanie projektu oraz implementacji. Zawiera ona informacje na temat tego z czego i w jaki sposób zbudowano spektrometr, jakie wykorzystano narzędzia programistyczne oraz zawiera objaśnienie zastosowanych algorytmów. W trzecim rozdziale zaprezentowano rezultaty pomiaru widma światła białego oraz przeprowadzono dyskusję nad czynnikami mającymi na nie wpływ.

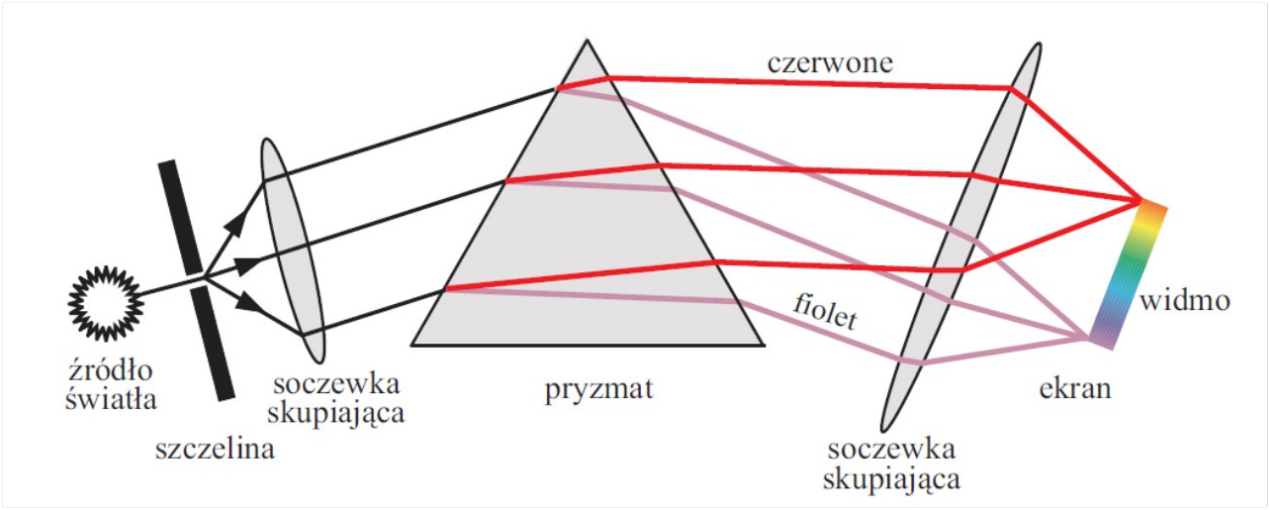
1. Zasada działania spektrometru

Po odkryciach Isaaca Newtona, Jamesa Clerka Maxwella oraz wielu innych fizyków wiadomo, że światło białe składa się z fal elektromagnetycznych o różnych długościach. Spektrometry wykorzystują falowe zachowanie światła, aby rozszczepić je na składowe monochromatyczne. Wykonane może być to na dwa sposoby.

Pierwszym z nich jest zastosowanie pryzmatu, w którym zachodzi zjawisko dyspersji. Współczynnik załamania światła n, w ośrodkach nie będących próżnią, zależy od długości fali światła. Oznacza to że, składowe strumienia światła białego, będą załamywane pod różnymi kątami, a co za tym idzie zostaną „rozszczepione" i widoczne w postaci tęczy na ekranie (patrz rysunek 1). Gdzie współczynnik załamania światła n obliczmy według zależności:

$$n^2 = A + B/\lambda^2 \qquad (1)$$

gdzie A i B to stałe dla ośrodka, a λ to długość fali [1]. Można zauważyć że, współczynnik załamania światła maleje wraz ze wzrostem długości fali. Czyli fale o krótszej długości np. kolor niebieski, jest załamywany mocniej, a niżeli fale o większej długości np. kolor czerwony. Zachowanie to jest niezależne od tego czy światło przechodzi z powietrza do szkła, czy na odwrót [3]. Na rysunku 1 przedstawiono zasadę działania protego spektrometru opartego na pryzmacie.

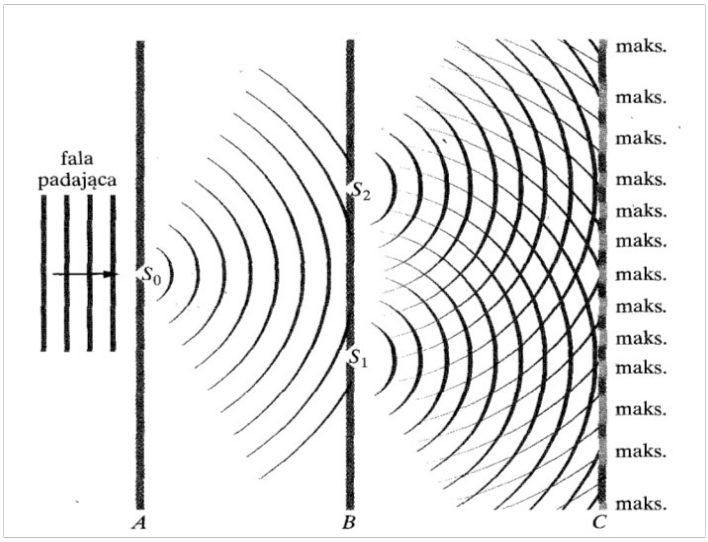


Rys. 1 Bieg wiązki światła w pryzmacie [1]

Metoda ta w tym projekcie nie zostanie wykorzystana. Spowodowane jest to tym, że pryzmat jest stosunkowo nieporęczny. Co więcej, takie urządzenie będzie wymagało cechowania przy użyciu widma wzorcowego. Taki układ pomiarowy będzie bardzo trudny do skonstruowania w warunkach domowych.

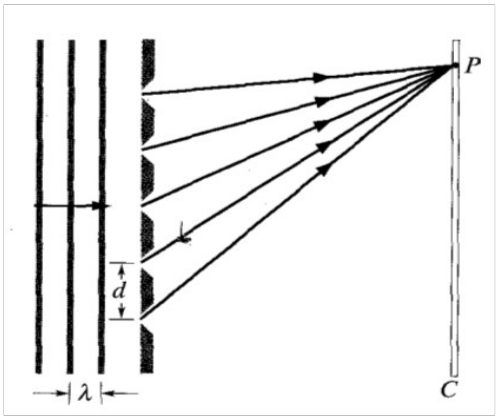
Kolejnym sposobem na stworzenie układu rozszczepiającego wiązkę światła białego jest zbudowanie spektrometru składającego się z siatki dyfrakcyjnej. Metoda ta wykorzystuje zjawisko dyfrakcji oraz interferencji. Interferencja polega na nałożeniu się na siebie dwóch lub więcej fal, w danym punkcie przestrzeni. Zjawisko dyfrakcji polega uginaniu się fali, która na swojej drodze napotka przeszkodę, np. zawierającą szczelinę o rozmiarach zbliżonych do jej długości. Jeżeli fala napotka na swojej drodze ekran zawierającą więcej niż jedną szczelinę, zgodnie z zasadą Huygensa, każda szczelina staje się źródłem nowej fali kulistej. Obraz interferencyjny możemy zaobserwować, wówczas gdy źródła światła emitują fale o jednej długości lub źródła interferujących fal są spójne

co oznacza, że emitowane fale zachowują stałą w czasie różnice faz. Światło słoneczne jest światłem częściowo spójnym, dla tego aby uległo interferencji po przejściu przez dwie szczeliny musi wcześniej przejść przez jedną. W takim przypadku fale kuliste posiadają tą samą długość oraz jednakową różnice faz dzięki czemu, możemy zaobserwować interferencję. Rysunek 2 przedstawia wcześniej wyjaśnione zachowanie.



Rys. 2 Zjawisko dyfrakcji oraz interferencji [3]

Ekran B jest najbardziej podstawowym układem szczelin. Układy które zawierają o wiele większą liczbę szczelin nazywane są siatką dyfrakcyjną, co przedstawiono na Rysunku 3.

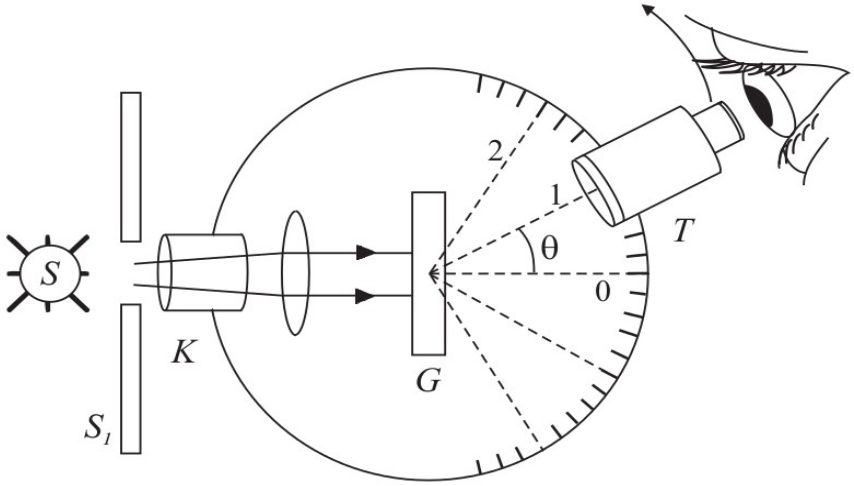


Rys. 3 Uproszczona siatka dyfrakcyjna [3]

Kiedy światło monochromatyczne przechodzi przez szczeliny, powstają wąskie prążki interferencyjne. Położenie kątowe każdego z nich zależy od długości fali jaką oświetliliśmy siatkę. Opisuje to równanie:

$$\theta = \arcsin\left(m\lambda / d\right) \tag{2}$$

Gdzie m to numer rzędu linii, λ to długość fali, a d to stała siatki dyfrakcyjnej określająca odległość między szczelinami. Zaletą tego rozwiązania jest to, że nawet jeśli światło zawiera kilka składowych o nie znanej długości fali, możemy je rozróżnić na podstawie rzędu linii i położeniu kątowemu. Urządzenie służące do mierzenia długości fali, zbudowane na podstawie siatki dyfrakcyjnej nazywane jest spektroskopem siatkowym. W jego skład wchodzą siatka dyfrakcyjna, szczelina, kolimator oraz lunetka. Rysunek 4 przedstawia schemat takowego urządzenia.



Rys. 4 Schemat spektrometru siatkowego[5]

Światło ze źródła jest ogniskowane przez kolimator na pionową szczelinę. Takie światło jest falą płaską, które następnie pada na siatkę dyfrakcyjną i ulega ugięciu. Dzięki temu stworzony zostanie obraz dyfrakcyjny na którym kąt ugięcia θ dla zerowego rzędu, względem osi siatki jest równy 0. Dla światła białego będzie to biały prążek. Po obu stronach owego maksimum będzie można zauważyć widma dyfrakcyjne pierwszego, drugiego i kolejnych rzędów. Ilość widocznych rzędów zależy od wartości stałej siatki dyfrakcyjnej.

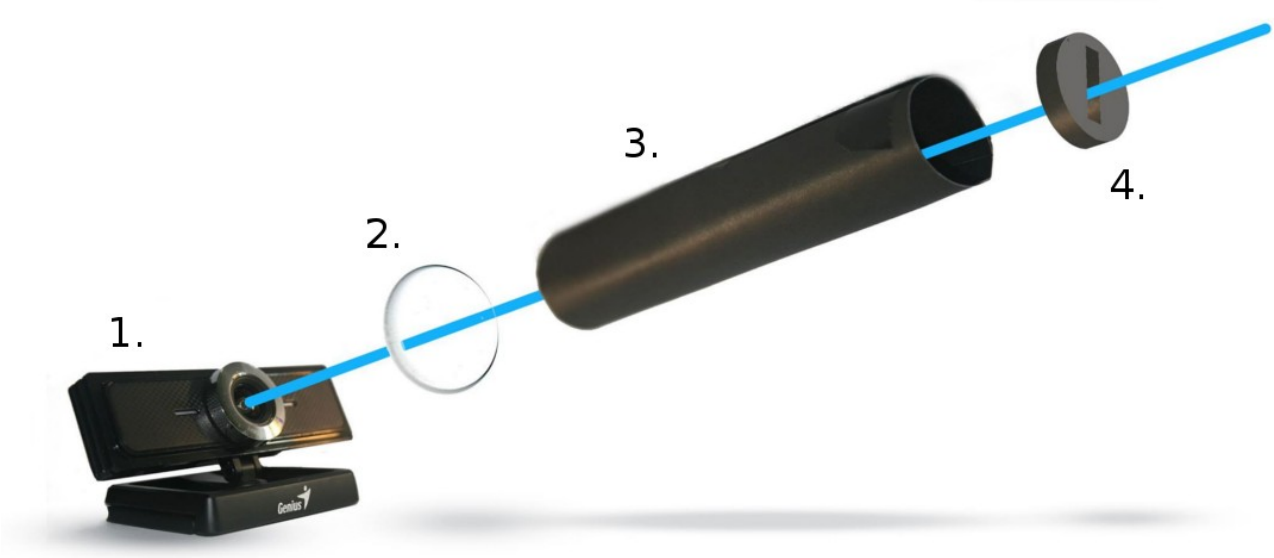
2. Implementacja

W tym rozdziale zostanie pokazane w jaki sposób zbudować spektrometr przy użyciu prostej kamery internetowej oraz siatki dyfrakcyjnej. Dodatkowo zostaną opisane biblioteki oraz środowisko programistyczne jakie zastosowano do stworzenia oprogramowania. Na koniec rozdziału wyjaśnione zostaną algorytmy do akwizycji obrazu oraz kalibracji.

2.1 Budowa spektrometru

W skład spektroskopu w tym projekcie będą wchodzić elementy, które są tanie i łatwo dostępne. Do rozszczepiania światła zastosowana zostanie siatka dyfrakcyjna. Jako sensor wychwytujący rozkład widmowy i przesyłający dane do programu, posłuży kamera internetowa z interfejsem UVC. Dodatkowo jako kolimator posłuży tekturowa tuba ze szczeliną wykonaną z żyletki.

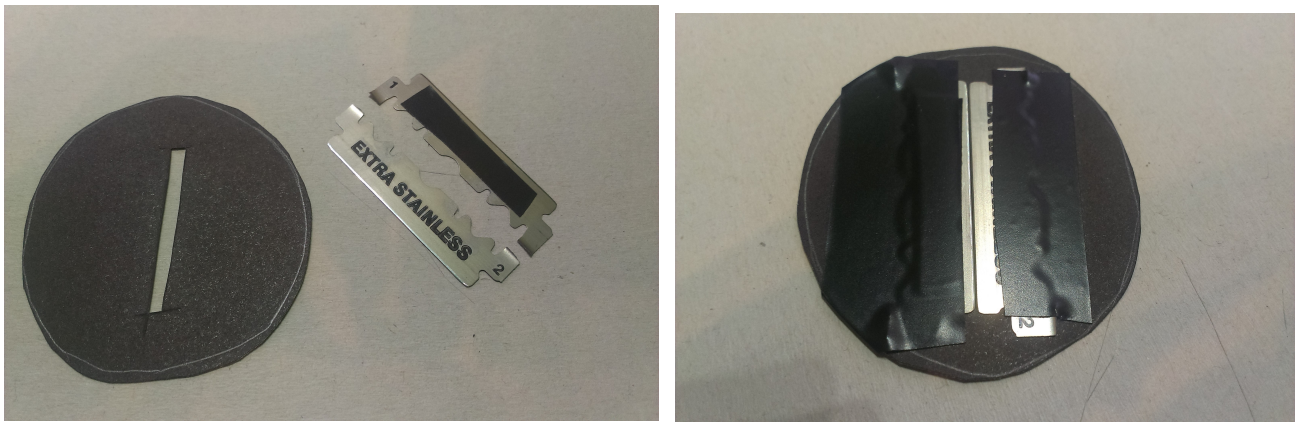
Pierwszym etapem budowy spektrometru jest skonstruowanie układu według rysunku 5.



Rys. 5 Schemat poglądowy spektrometru gdzie: 1. kamera internetowa 2. siatka dyfrakcyjna 3. tuba 4. szczelina

Na wymiary kamerki internetowej użytkownik nie ma wpływu. Każda kamerka internetowa posiada własny system soczewek dostosowany do jej matrycy. Dla tego każdy układ mierzący rozkład widmowy musi być dostosowany do właściwości danej kamery. Jedyne na co powinniśmy zwrócić uwagę przy wyborze tego instrumentu, to aby kąt widzenia był jak najmniejszy oraz aby nie posiadała żadnego oświetlenia przy obiektywie, mogącego zafałszować wynik doświadczenia. Resztę elementów powinniśmy dostosować do kamery. Jednym z elementów jest tuba, która powinna być zbudowana z czarnego, matowego papieru. Taką kartkę zawija się w taki sposób, aby rura była dopasowana do obiektywu kamery internetowej, najlepiej aby na niego nachodziła. Dzięki temu światło wpadające do środka nie odbija się od ścian tuby. Początkowa długość tuby powinna wynosić około 25 cm. Szczelina powinna być zamontowana w postaci wieczka tuby. Można ją

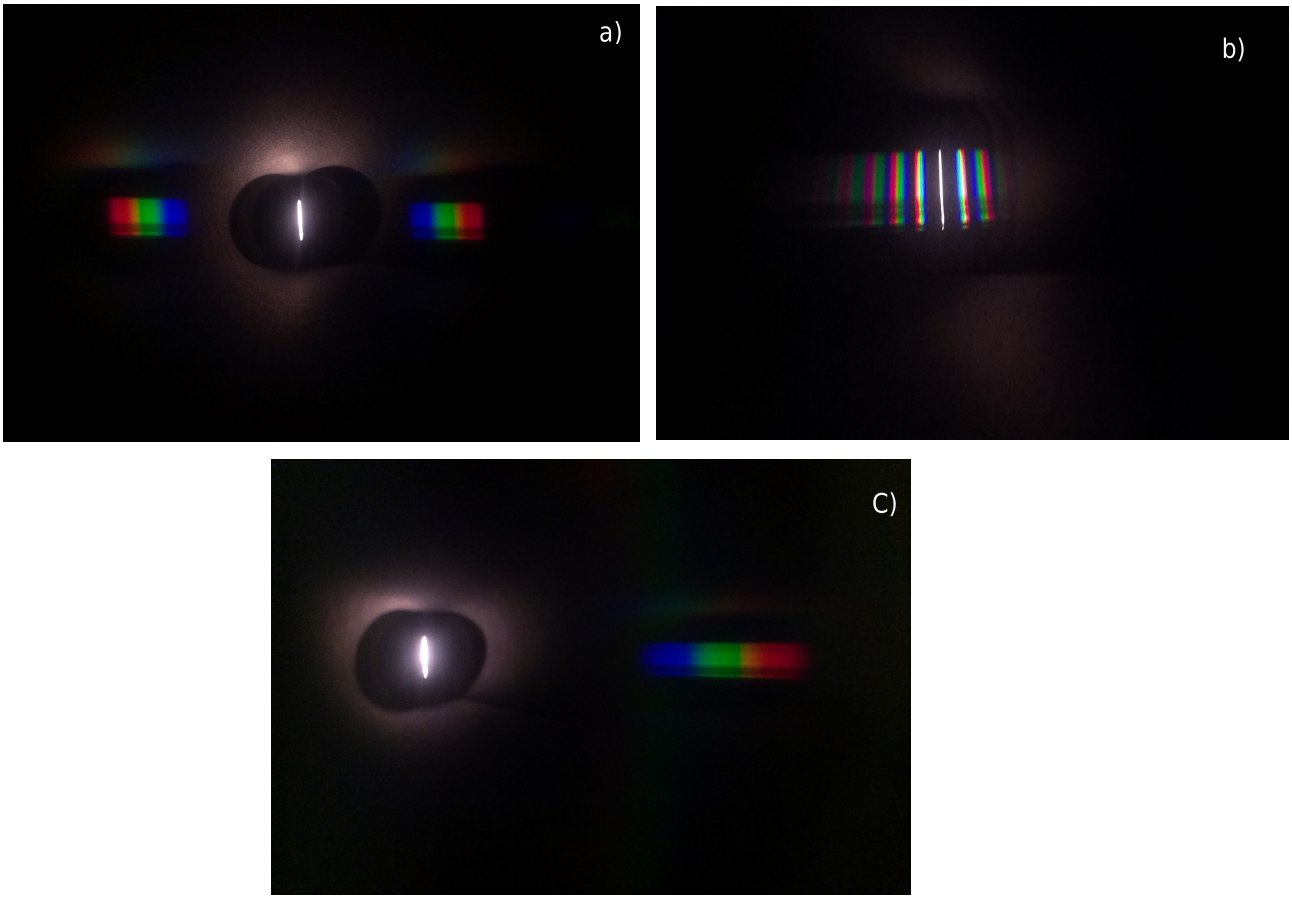
zrobić na kilka sposobów. Jednym z nich jest wycięcie w papierowym dekielku cienkiej szpary ostrym nożykiem. Jednak wtedy brzegi szczeliny nie są proste i często ciężko jest wyciąć otwór o odpowiednich wymiarach, co ma wpływ na jakość uzyskanego widma. Tam gdzie szczelina jest zbyt wąska mamy zaniki widma, a w miejscach gdzie szczelina jest zbyt duża obraz jest prześwietlony. Najlepszym rozwiązaniem jest zastosowanie dekielka z przyklejonymi dwoma ostrzami żyletki, które w miarę możliwości usytuowane są bardzo blisko oraz równolegle do siebie. Rysunek 6 przedstawia w jaki sposób powinna być zamontowana żyletka do wieczka.



Rys. 6 Po lewej elementy szczeliny, po prawej gotowa szczelina sporządzona z żyletki

Tak skonstruowana szczelina ma równe brzegi i może zostać zamontowana do jednego z końców tuby. Należy zadbać, aby od strony wieczka światło dostawało się wyłącznie przez szczelinę, reszta układu musi być światłoszczelna. Miejsca które przepuszczają światło najlepiej zakleić czarną taśmą izolacyjną.

Kolejnym etapem budowy spektrometru jest zamocowanie siatki dyfrakcyjnej. Stałe siatek dyfrakcyjnych jakie udało się uzyskać wynoszą 500 rys/mm, 50 rys/mm, 1000 rys/mm. W tym przypadku najlepsza jest siatka o stałej wynoszącej 500 rys/mm, ponieważ uzyskane widmo światła białego jest widoczne dla rzędu pierwszego, a kąt ugięcia jest na tyle mały, że jest możliwość zaobserwowania widma po obu stronach szczeliny bez dodatkowych modyfikacji położenia tuby. W przypadku siatki o stałej wynoszącej 50 rys/mm widzimy kilka rzędów o małym kącie załamania, dodatkowo widma są stosunkowo wąskie. Dla siatki dyfrakcyjnej o stałej równej 1000 rys/mm kąt załamania jest na tyle duży, że jednocześnie można zaobserwować tylko widmo po jednej stronie rzędu zerowego. Taka siatka nie nadaje się dla kamer wąskokątnych, ponieważ będzie trudno zadbać o poprawne ustawienie siatki względem szczeliny, punktu ogniskowej oraz kąta pod jakim kamera ma widzieć widmo. Siatkę dyfrakcyjną o stałej 1000 rys/mm, można zastosować do kamer szerokokątnych. Na rysunku 7 pokazano obrazy widm dla siatek dyfrakcyjnych o różnych stałych.



Rys. 7 Zastosowanie siatek różnych stałych siatki dyfrakcyjnej,

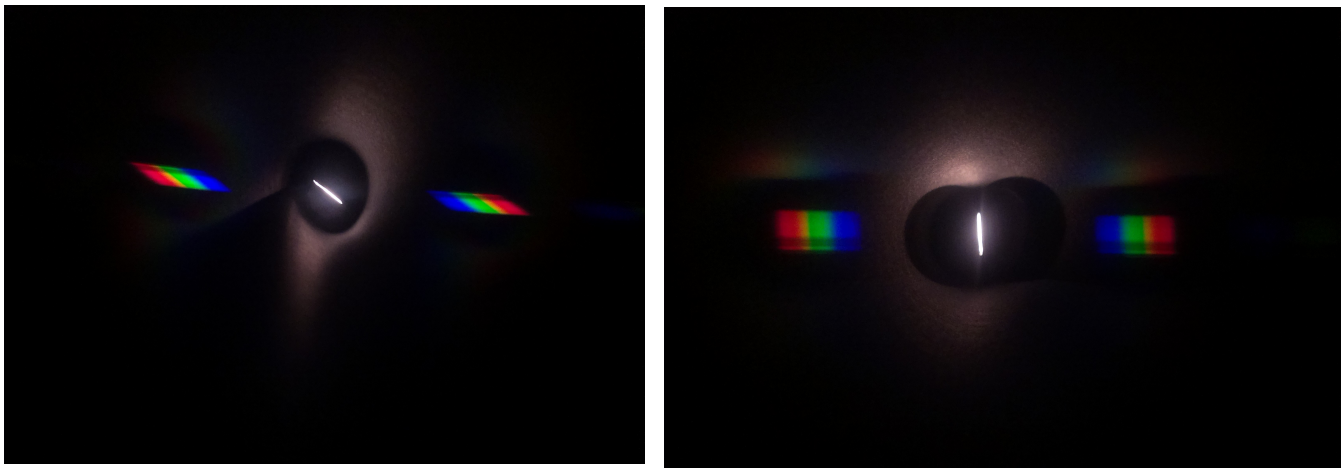
a- stała siatki 500 rys/mm, b- 50 rys/m, c- 1000 rys/mm

Aby zamontować siatkę dyfrakcyjną należy wyciąć koło o rozmiarach obiektywu i nakleić ją na obiektyw (patrz rysunek 8). Pamiętać trzeba, żeby taśma, którą mocujemy siatkę, nie zasłaniała pola widzenia.

Rys. 8 Kamerka internetowa z zamontowaną siatką dyfrakcyjną

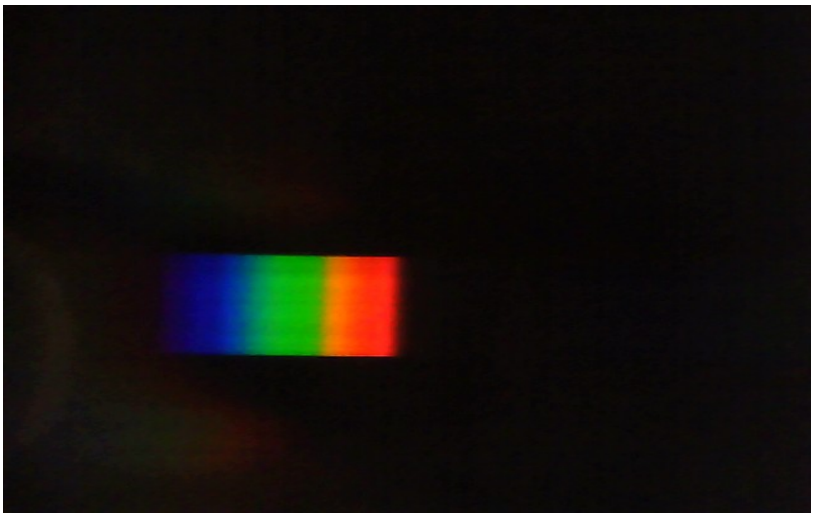
Następnie mocujemy wcześniej zrobiona tubę. Konieczne jest, by szczelina względem rys na siatce dyfrakcyjnej była położona w takiej samej płaszczyźnie i była równoległa do ich położenia. Aby

ułożenie szczeliny względem siatki było poprawne należy sprawdzić czy widmo nie jest pochylone w żadnym kierunku. Rysunek 9 przedstawia poprawne oraz złe ustawienie szczeliny względem rys na siatce.

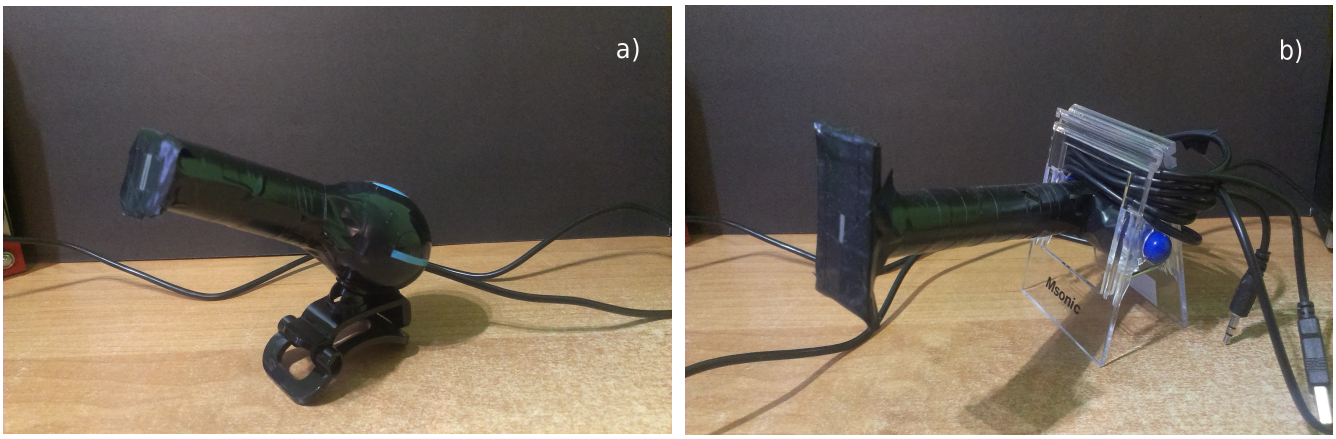


Rys. 9 Złe (rysunek lewy) oraz poprawne (rysunek prawy) ułożenie szpary względem rys na siatce dyfrakcyjnej

Ostatnią czynnością w konstrukcji urządzenia pomiarowego jest ustawienie i zamocowanie tuby na obiektywie kamerki internetowej. Najpierw skracamy papierowa rurę do takiego stopnia, aby widmo zajmowało jak największą powierzchnię kadru. Pamiętając o tym, by odpowiednio do długości tuby zmieniać punkt ogniskowej obiektywu tak, żeby obraz był jak najostrzejszy. Dodatkowo należy zwracać uwagę na położenie szczeliny względem rys siatki dyfrakcyjnej. Następnie tubę trzeba ułożyć pod takim kątem, aby w kadrze znajdowało się wyłącznie prawe widmo pierwszego rzędu. W takim ułożeniu długości fal w widmie rozkładają się rosnąco od strony prawej do lewej. Dzięki temu wykres widma w aplikacji zostanie poprawnie odwzorowany w układzie kartezjańskim. Kolejnym powodem takiego ustawienia jest fakt, że gdy kamera widzi bardzo jasny punkt, w tym przypadku szczelinę, tło widma jest zbyt jasne aby dobrze je zinterpretować. Rysunek 10 przedstawia rozkład widmowy w dobrze zbudowanym układzie pomiarowym.



Rys. 10 Rozkład widmowy widoczny w kadrze dobrze zbudowanego układu



Rys. 11 Skompletowane układy pomiarowe na kamerach internetowych firmy:

a) Media-tech model MT4047 , b) Msonic model MR1803B

Rysunek 11 przedstawia układy pomiarowe zbudowane na dwóch różnych kamerkach internetowych firmy Media-tech model MT4047 oraz firmy Msonic model MR1803B. Widać, że dla każdej z osobna dopasowano długość tuby jak i nachylenie względem obiektywu. Spowodowane jest to tym, że każda kamea posiada inny tor optyczny.

2.2 Wykorzystane narzędzia programistyczne

Oprogramowanie w tym projekcie zaprojektowano dla systemu operacyjnego Linux. Wykorzystano do tego język programowania C++. Jego głównymi atutami są szybkość oraz możliwość programowania obiektowego. W tym celu wykorzystano środowisko Eclipse oraz Qt Creator.

Do akwizycji obrazu oraz dalszej jego modernizacji posłużyła biblioteka OpenCV (Open Source Computer Viision library) wersji 3.0. „Jest ona bezpłatną, opensource'ową biblioteka napisaną w języku C oraz C++. Została zaprojektowana na potrzeby aplikacji czasu rzeczywistego, gdzie wydajność oprogramowania jest główną cechą. Posiada interfejsy do takich języków programowania jak C, C++ Python, Java i MatLab oraz wspiera systemy operacyjne takie jak Windows, Linux, Android, Mac OS. Biblioteka ta zawiera ponad 25000 zoptymalizowanych algorytmów, w których skład wchodzą kompleksowe zestawy algorytmów zarówno klasycznej jak i „state-of-the-art” do przetwarzania obrazów.”[8] W tej pracy skorzystano z funkcji, które maiły na celu zarejestrować obraz widziany przez kamerkę internetową oraz pomóc w kalibracji układu. Podczas pisania kodu wspomagano się również przykładami opisanymi w [8].

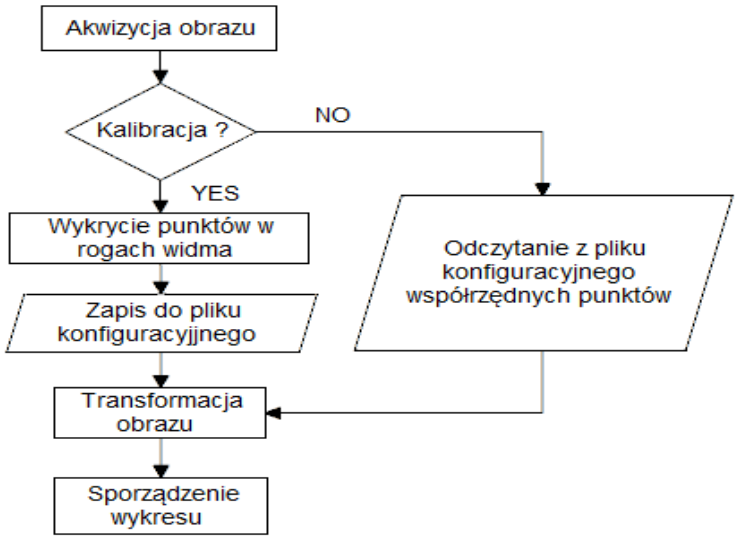
Biblioteki libconfg użyto do zachowywania parametrów kalibracji układu. „Jest to prosta biblioteka do czytania, tworzenia oraz modyfikacji plików z rozszerzeniem cfg. Ten format plików jest bardziej zwarty i bardziej czytelny niż XML. Dodatkowo ten format jest „type-aware” co oznacza, że posiada świadomość jakiego typu zmienne zostały do niego zapisane, więc nie ma potrzeby robienia ciągu parsowania w kodzie. Biblioteka ta dysponuje parserem i obejmuje powiązania dla języków programowania C i C++ .”[9]

Do zaprojektowania i wykonania graficznego interfejsu użytkownika skorzystano z platformy programistycznej Qt, w której skład wchodzi zintegrowanie środowisko programistyczne Qt Creator oraz biblioteka. „Biblioteka został napisany w języku C++ i rozszerza ten język o takie

funkcjonalności jak sygnały i sloty. Proces MOC (Meta-Object-Compiler), przed kompilacją analizuje pliki źródłowe napisane w języku C++ poszerzonym o QT i generuje standardowe zgodne z C++ źródła. Tak więc aplikacje używające Qt mogą zostać skompilowane przez dowolny standardowy kompilator C++.”[10] Dodatkowym atutem platformy Qt, są dodatki takie jak QCustomPlot wspomagające tworzenie różnego typu wykresów oraz wizualizacje danych. Co więcej Qt działa na systemach Linux, OS X, Android oraz Windows. „Środowisko programistyczne Qt Creator oferuje inteligentne uzupełnianie kodu, podświetlanie składnie, zintegrowany system pomocy oraz debugger. Ponadto Qt Creator posiada pełen zestaw narzędzi. Ta platforma programistyczna posiada kilka wersji licencji. Jedną z nich jest licencja komercyjna, ale również dostępna jest licencja darmowa przeznaczona dla osób uczących się.”[10] Konkurencyjną platformą programistyczną jest GTK. Jest to równie dobre środowisko do tworzenia interfejsu graficznego użytkownika. Jednak nie posiada wsparcia dla systemów operacyjnych przeznaczonych dla smartfonów, co wykluczyło by rozwój aplikacji na telefony komórkowe.

2.3 Algorytm do akwizycji widma światła białego

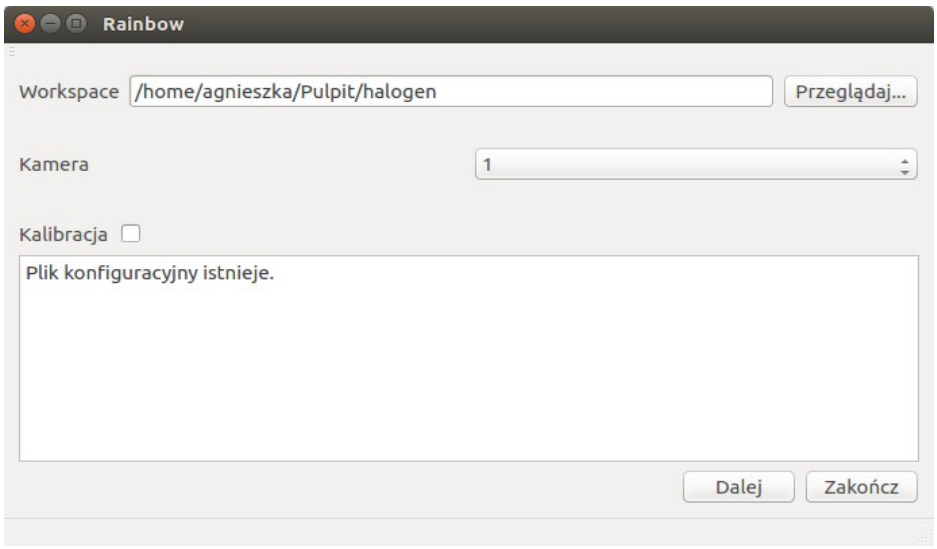
W tym rozdziale omówione zostaną elementy z jakich składa się algorytm do akwizycji widma światła białego. Dodatkowo wyjaśniona będzie metoda sprawdzania intensywności światła oraz algorytmy umożliwiające poprawne zbadanie widma. Na rysunku 12 przedstawiony jest diagram blokowy obrazujący podstawowe działanie programu.



Rys. 12 Diagram blokowy programu do wyznaczania widma światła białego

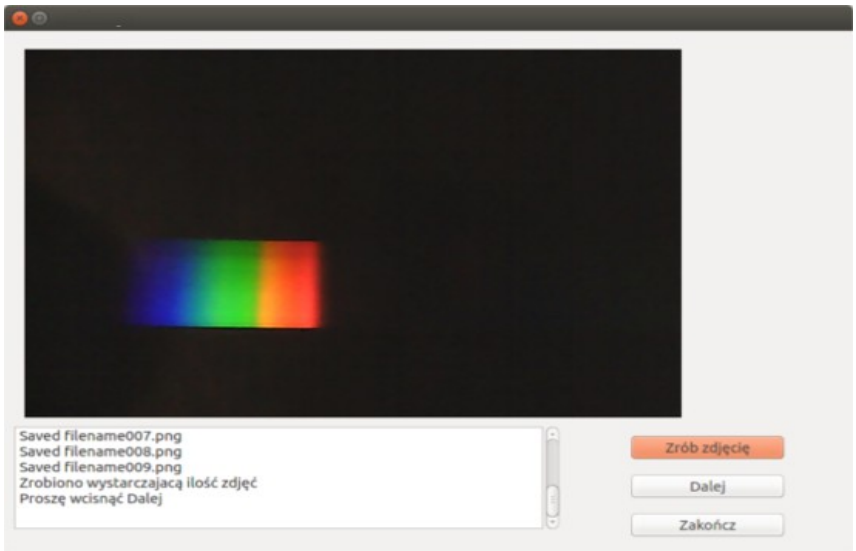
Oprogramowanie podzielone jest na pięć lub cztery moduły w zależności od tego czy układ będzie kalibrowany. Operacja kalibracji zostanie opisana w podrozdziale 2.4 . W tym rozdziale założono, że układ został poprawnie skalibrowany.

Pierwszy etap algorytmu zajmuje się akwizycją obrazu z kamery internetowej. Aby program mógł zacząć poprawnie wykonywać swoje działanie, użytkownik wprowadza podstawowe ustawienia: ścieżkę do katalogu w którym będą zapisywane pliki, numer kamery, kalibracja. Dodatkowo w oknie znajduje się pole informujące użytkownika czy plik konfiguracyjny istnieje. W przeciwnym wypadku wyświetlana jest informacja o potrzebie skalibrowania aplikacji. Przedstawia to rysunek 13.



Rys. 13 Okno do wprowadzenia podstawowych ustawień

Po wprowadzeniu danych i zaakceptowaniu przyciskiem „Dalej” program zaczyna rejestrować obraz z kamery internetowej. Wykonane jest to za pomocą klasy `VideoCapture`, która otwiera wybraną przez użytkownika kamerę. Proces odtwarzania obrazu polega na czytaniu kolejnych klatek w nieskończonej pętli z zadanym opóźnieniem. Dla aplikacji nie korzystających z Qt, za opóźnienie odpowiedzialna jest funkcja z biblioteki OpenCV, `waitKey()`. Argumentem jest wartość opóźnienia w milisekundach, a wartością zwracaną liczba odpowiadająca wydarzeniu wykonanemu w czasie czekania. Jednak funkcja ta działa wyłącznie w przypadku, gdy aktywne jest okno HighGUI. W przypadku gdy obraz przedstawiany jest w oknie widget'u Qt za opóźnienie odpowiedzialny jest Qtimer z biblioteki Qt. Na koniec odliczanego czasu tego licznika wywoływany jest sygnał uruchamiający metodę przechwytującą kolejną klatkę z kamery. Zadaniem użytkownika jest zrobienie dziesięciu zdjęć, które później zostaną zintegrowane w celu zmniejszenia szumu. Ma to szczególne znaczenie w przypadku słabej jakości kamer. Rysunek 14 przedstawia okno, które jest odpowiedzialne za wyświetlanie obrazu z kamerki. Dodatkowo posiada przyciski do zrobienia zdjęć oraz przejścia do kolejnego etapu.



Rys. 14 Okno programu do wyświetlania obrazu kamerki

Po zsumowaniu zdjęć zrobionych przez użytkownika, program usiłuje otworzyć za pomocą klasy `Config` plik zawierający współrzędne widma na zdjęciu. Położenie przedstawiane jest na podstawie czterech punktów o współrzędnych „x” i „y”, określających rogi widma wykryte podczas kalibracji. Następnie zapisuje dane do klas reprezentujących pojedynczy piksel.

W kolejnym etapie programu korygowana jest geometria układu pomiarowego. Człowiek nie jest w stanie idealnie ułożyć szczeliny względem rys na siatce dyfrakcyjnej. Zawsze widmo jest w pewnym stopniu pochylone. Dodatkowo należy wyodrębnić z zdjęcia pole gdzie znajduje się samo widmo bez tła. Do tego posłużyły funkcje `getPerspectiveTransform(InputArray src, InputArray dst)` i `warpPerspective(InputArray src, OutputArray dst, InputArray M, Size dsize)` z biblioteki OpenCV. Ich zadaniem jest rozciągnąć pole znajdujące się w środku czterech punktów do prostokąta o założonych rozmiarach 600 pikseli szerokości i 300 pikseli wysokości. Geometryczna transformacja obrazu polega na deformacji siatki pikseli oraz na mapowaniu tej zdeformowanej siatki do obrazu docelowego. Taka transformacja przekształci widmo do odpowiedniej formy, niezależnie od tego jakie miało wymiary i gdzie znajdowało się na zdjęciu. Najważniejszym elementem są dobrze określone rogi czworokąta, w którym znajduje się analizowane widmo. Tabela 1 przedstawia funkcję która po otrzymaniu odpowiednich punktów zastosuje transformacje geometryczna.

Tabela 1. Funkcja wykonująca transformację geometryczną obrazu

```
Mat dst( 300, 600, img.type() );
Mat transform_matrix;
Point2f source_points[4];
Point2f dest_points[4];

source_points[0] = Pt1;
source_points[1] = Pt2;
source_points[2] = Pt3;
source_points[3] =Pt4;

Point2f lu_result( 0, 0);
Point2f ru_result( 600, 0);
Point2f rd_result( 600, 300);
Point2f ld_result( 0, 300);
dest_points[0] = lu_result;
dest_points[1] = ru_result;
dest_points[2] = rd_result;
dest_points[3] = ld_result;

transform_matrix = getPerspectiveTransform(source_points,
dest_points);
warpPerspective(img, dst, transform_matrix,dst.size());
```

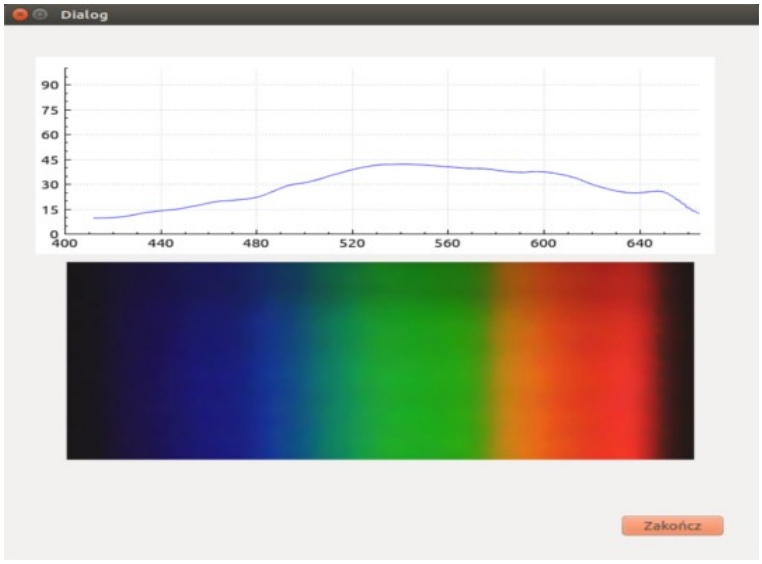
W przypadku gdy dostępny jest już obraz o odpowiednich rozmiarach i formie, w której skład wchodzi wyłącznie widmo bez zbędnego tła, program może wyznaczyć luminancje pikseli, która jest związana z energią światła w zakresie widmowym padającym na poszczególne piksele. Gdy obraz byłby reprezentowany w odcieniach szarości, jasność piksela określała by wartość jego

luminancji. Jednak większość tanich kamer internetowych posiada matrycę w formacie RGB więc potrzebna jest konwersja do formatu YUV. W tym formacie Y- odpowiada luminancji, a U i V reprezentują barwy różnicowe barw podstawowych RGB [13]. Po to by obliczyć średnią luminancję każdej z kolumn zawartej w macierzy `Mat imgYUV` przedstawiającej widmo, wartości luminancji każdego piksela są sumowane w kolumnie po czym dzielone przez wysokość kolumn, w tym przypadku 300. Następnie wartości te umieszczane są kolejno w tablicy `Power[600]`. Tabela 2 przedstawia algorytm wyznaczający średnią wartości luminancji dla każdej z kolumn macierzy.

Tabela 2. Algorytm do wyliczania średniej luminancji kolumn

```
cvtColor(img1, imgYUV, CV_BGR2YUV);
rows = imgYUV.rows;
cols = imgYUV.cols;
Mat dstY(rows, cols,  CV_8UC1);
int Power[cols];
for(int y = 0; y < rows; y++){
   for(int x = 0; x < cols; x++){
       Vec3b color = imgYUV.at<Vec3b>(Point(x,y));
       dstY.at<uchar>(Point(x,y)) = color[0];
       colorint = int(color[0]);
       Power[x] = Power[x] + colorint;
   }
}
for( int i = 0; i < cols; i++ ){
   Power[i]= Power[i] / rows;
}
```

Ostatnią częścią programu jest sporządzenie wykresu. Wykonano to dzięki dodatkowi do biblioteki Qt o nazwie QcustomPlot. Wykres na rysunku 15 przedstawia zależność długości fali od luminancji. Długości fali są w zakresie od 385nm do 670 nm, a luminancja określona w procentach. Wygenerowany wykres zostaje zapisany w folderze wybranym przez użytkownika pod nazwą graph.png . Rysunek 15 przedstawia ostatnie okno programu z finalnym wynikiem.



Rys. 15 Okno programu zawierające ostateczny wynik

2.4 Kalibracja układu

Aby program działał poprawnie oraz wspierał każdą kamerkę internetową z interfejsem UVC, niezależnie od tego jaki zestaw soczewek posada, potrzebna jest kalibracja. Składa się ona z czterech etapów, które mają na celu najdokładniejsze znalezienie czterech punktów określających położenie widma w obrazie. Założono, że z powodu niedokładnego wykonania urządzenia, obraz widma rejestrowany przez kamerę będzie miał kształt czworoboku. W celu kalibracji urządzenia wymagane jest źródło światła o ciągłym i pełnym widmie. Zatem najlepszym źródłem wzorcowym będzie żarowe źródło światła np. żarówka halogenowa.

W pierwszym etapie kalibracji wydzielone zostają kolorowe piksele widma od ciemnego tła. Wykonywane jest to za pomocą funkcji `varianceRGB(uchar b, uchar g, uchar r)`oraz `matVariance(string pathtofile)`. Pierwsza z nich oblicza wartość wariancji z trzech kolorów, zapisanych w macierzy obrazu, mianowicie koloru niebieskiego, zielonego, oraz czerwonego. Algorytm ten wykorzystuje fakt, że jeżeli barwy RGB są w równych proporcjach, dają odcienie szarości odpowiadające tłu widma. Tabela 3 przedstawia implementację funkcji varianceRGB().

Tabela 3. Funkcja obliczająca wariancję kolorów RGB

```
uchar varianceRGB(uchar b, uchar g, uchar r){
    uchar mean;
    uchar variance;
    uchar componentB;
    uchar componentG;
    uchar componentR;
    mean = (b + g + r )/3;
    componentB = ( b - mean ) * ( b - mean );
    componentG = ( g - mean ) * ( g - mean );
    componentR = ( r - mean ) * ( r - mean );
    variance =(componentB + componentG + componentR ) / 3;
    return variance;
}
```

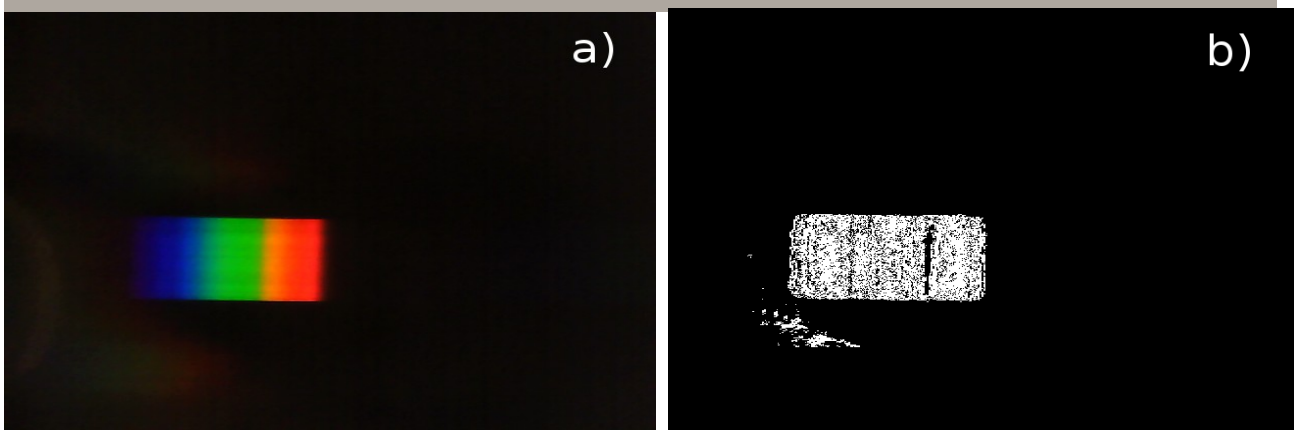
Funkcja `matVariance` przetwarza macierz kolorowego trójkanałowego obrazu wejściowego na macierz parametryczną. Wartość każdego piksela pochodzi z wyliczonej wariancji obliczanej według wzoru 3.

$$\sigma^2 = \frac{\sum_{i=1}^{n} (X-\mu)^2}{N} \tag{3}$$

Gdzie X to wartość danego koloru, μ to średnia wszystkich kolorów, a N to ilość kolorów. Jeżeli wariancja jest większa lub równa 80 wtedy piksel jest ustawiany na „255” i reprezentuje część obrazu w której znajduje się czworokąt z widmem. W pozostałych przypadkach piksel ustawiany jest na „0” i reprezentuje tło. Tabela czwarta przedstawia fragment funkcji zajmujący się przekształceniem obrazu wejściowego.

Tabela 4. Fragment funkcji przetwarzającej obraz kolorowy na obraz czarno-biały odpowiadający wariancji kolorów pikseli

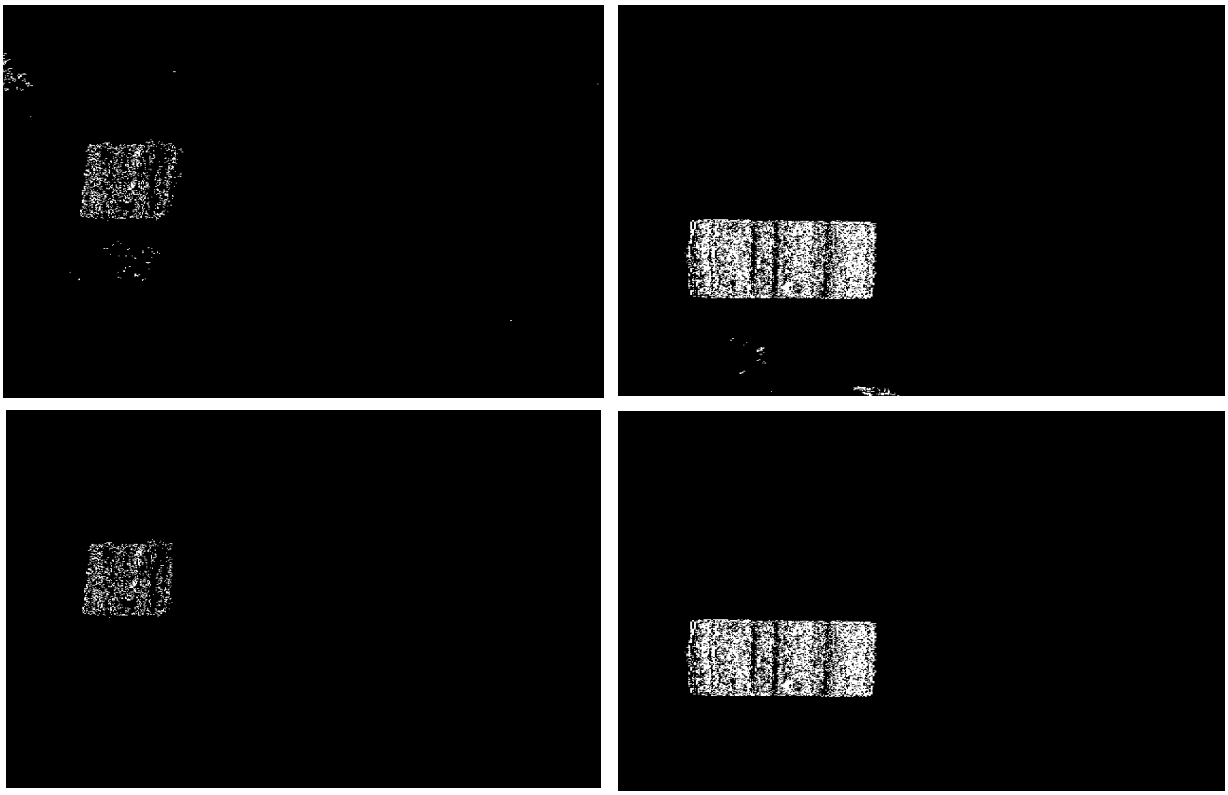
```
int cols = img.cols;
int rows = img.rows;

for(int y = 0; y < rows; y++){
   for(int x = 0; x < cols; x++){
       Vec3b color = img.at<Vec3b>(Point(x,y));
       var=varianceRGB(color[0] ,color[1], color[2]);
       dst.at<uchar>(Point(x,y)) = var;
       if( var >= 80 ){
           dstn.at<uchar>(Point(x,y)) = 255;
           xList.push_back(x);
           yList.push_back(y);
       }
       else{
           dstn.at<uchar>(Point(x,y)) = 0;
       }
   }
}
```



Rys. 16 Efekt przekształcenia obrazu względem wariancji a) obraz wejściowy, b) obraz wyjściowy

Jak widać na rysunku 16, kolor biały uzyskały nie tylko piksele należące do obszaru należącego do widma. Te punkty spowodowane są szumami cyfrowymi i nieidealnym układem pomiarowym powodującym odbicia widma na soczewce lub tubie. Niestety w tym przypadku nie można użyć funkcji z biblioteki OpenCV, które pozwoliłyby zlikwidować szum na zdjęciu. Przyczyną tego jest szumowy charakter obszaru określającego widmo. Po zastosowaniu funkcji likwidujących szum, zostają usunięte także niektóre piksele należące do powierzchni widma. Aby pozbyć się pikseli nie należących do interesującej nas powierzchni współrzędne wszystkich białych punktów zapisywane są do dwóch list, `xList` oraz `yList`. Następnie dla każdej punktu w danej przestrzeni, liczony jest błąd względny w zależności od średniego położenia wszystkich punktów w tej przestrzeni. Jeśli błąd względny dla wartości punktów osi odciętych jest większy od 65 lub dla wartości punktu osi rzędnych jest większy od 25, to punkt o tych współrzędnych zostanie wyeliminowany z grupy punktów odpowiadających położeniu widma. Wartości ograniczające powierzchnie zostały określone na podstawie wartości największego obszaru jaki udało się uzyskać dla kamer internetowych. Rysunek 17 porównuje różne zdjęcia przed obliczaniem błędu względnego i po, by

przedstawić korzyść z zastosowania tej metody.

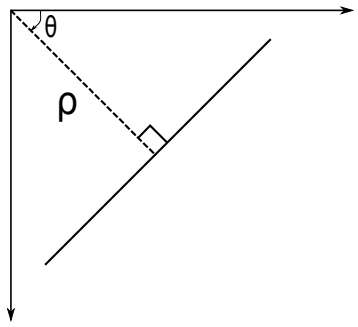


Rys. 17 Efekty zastosowania algorytmu korzystającego z błędu względnego

Kolejna funkcja `fillingWhite(string pathtofile)` odpowiedzialna jest za wypełnienie kolorem białym jak najwięcej miejsca między punktami reprezentującymi widmo. Takie przetwarzanie potrzebne jest po to by przygotować układ do wykrywania krawędzi. Wykorzystano do tego funkcję `matchTemplate()` z biblioteki OpenCV. Funkcja ta bierze szablon i przesuwa go po całym obrazie wejściowym obliczając dopasowanie obszaru, następnie generując macierz wyjściową zawierającą wartość dopasowania. Zwykle funkcja ta używana jest do wykrywania elementu znajdującego się na obrazie najlepiej odpowiadającemu szablonowi. W tym przypadku użyta została by w miejscach, w których znajduje się duża ilość białych punktów, przestrzeń między nimi wypełnić kolorem białym. Dla tego jako szablon funkcja pobiera biały prostokąt o wymiarach 4 x 4 piksele. Dzięki takiemu zabiegowi łatwiej jest wykryć brzegi czworokąta wypukłego.

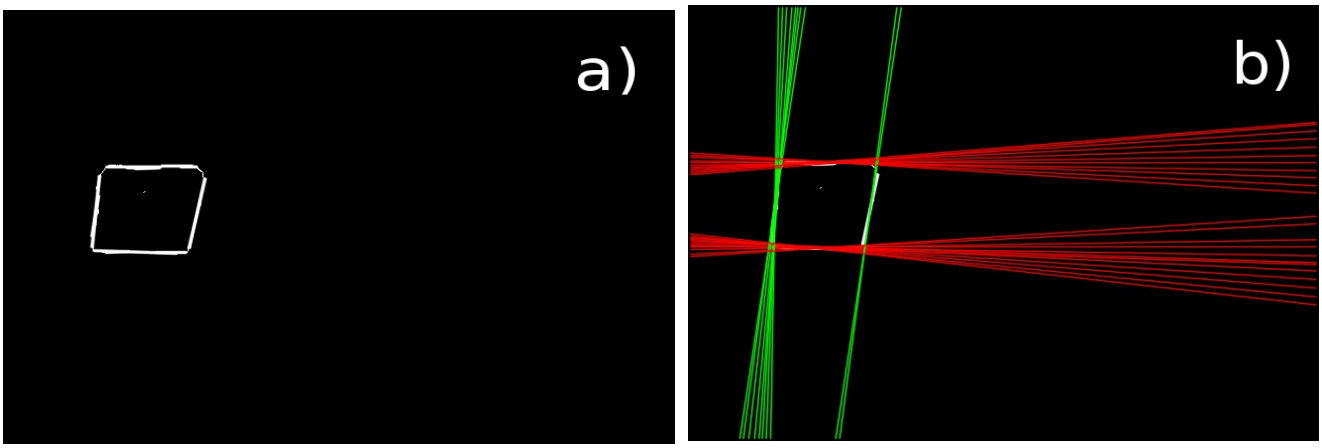
Punty określające rogi widma wyznaczane będą jako punkty przecięcia prostych przechodzących przez brzegi figury (założono, że będzie to czworokąt wypukły) określonej białymi pikselami. Aby taki zabieg był jak najdokładniejszy krawędzie widma wykrywane są dwupoziomowo. Najpierw wykrywana jest obwoluta grupy białych pikseli odpowiadających położeniu widma. Wynik tego etapu przedstawia rysunek 19 a. Aby to wykonać użyto funkcji `Canny()` oraz `HoughLinesP()`. Funkcja `HoughLinesP()` wykrywa linie na podstawie wyodrębnionych przez funkcje `Canny()` krawędzi. Jednak wyodrębnione brzegi są nieciągłe i nierówne. Na tym poziomie wykryte linie nie określają prawidłowo brzegów pola, a punty przecięcia nie określają rogów czworokąta wypukłego. Tak wykryte linie posłużyły wyłącznie do wygładzenia nierównych boków. Gdy brzegi figury określającej położenie, są już wystarczająco gładkie obraz ponownie podlega wykrywaniu linii dzięki algorytmowi opartemu o transformacie Hough'a. Tak wykryte linie reprezentowane są za pomocą dwóch zmiennych (θ, ρ). ρ to odległość

między odnaleziona prostą, a początkiem układu współrzędnych, który znajduje się w lewym górnym rogu zdjęcia. θ to kat między liną prostopadłą do ρ określającą odległość, a osią współrzędnych x. Na podstawie tych wartości podejmowana jest decyzja czy dana linia jest pozioma lub pionowa. Rysunek 18 przedstawia zmienne (θ, ρ) w układzie kartezjańskim.



Rys. 18 Zmienne θ oraz ρ w układzie kartezjańskim [10]

Rysunek 19 przedstawia efekt działania wykrywania prostych.



Rys. 19 Rezultat dwupoziomowego działania funkcji do wykrywania brzegów a) wstępne wyszukanie krawędzi, b) wykrycie linii przechodzących przez krawędzie

Dla poprawienia dokładności pomiaru ograniczono przeszukiwany zakres nachylenia prostych. Dla linii czerwonych (patrz rysunek 19) jest to $|0.999| < \sin(\theta)$, co oznacza że górny i dolny bok mogą być pochylone względem osi rzędnych o kąt od 87,44° do 92,56°. Dla linii zielonych $|0.8888| < \cos(\theta)$, co daje maksymalny kąt nachylenia względem osi odciętych znajdujący się w przedziale od 62,82° do 117.18°. Wartości kątów nachylenia prostych wyznaczono doświadczalnie, zmieniając położenie szczeliny względem rys na siatce dyfrakcyjnej. Nakłada to na użytkownika ograniczenia dotyczące budowy układu pomiarowego, takie aby brzegi widma nie były zbyt pochylone.

Po poprawnym wykryciu linii ich współrzędne zapisywane są do odpowiednich wektorów. Do wektorów Vlines1 oraz Vlines2 zapisywane są kolejno parametry linii wertykalnych. Natomiast do wektorów Hlines1 oraz Hlines2 zapisywane są parametry prostych horyzontalnych. Dzięki takiemu podziałowi wykryte zostaną punkty przecięć prostych znajdujące się tylko w rogach obszaru zawierającego widmo. Wyliczaniem tych punktów zajmuje się algorytm opisany w tabeli 4.

Tabela 4. Algorytm wyznaczający punkty przecięcia linii

```
vector<Point2f> Cpoints;
Point2f x, d1, d2, r;
for(size_t n = 0; n < Vlines1.size(); n++){
   for(size_t m = 0; m < Hlines1.size(); m++){
       x = Hlines1[m]-Vlines1[n];
       d1 = Vlines2[n] – Vlines1[n];
       d2 = Hlines2[m] - Hlines1[m];
       float cross = d1.x*d2.y – d1.y*d2.x;
       double t1 = (x.x * d2.y - x.y * d2.x)/cross;
       r = Vlines1[n] + d1 * t1;
       Cpoints.push_back(r);
   }
}
```

Funkcja transformacji obrazu wymaga, aby podawane do niej punkty były w odpowiedniej kolejności. Pierwszym punktem jest lewy górny, następnym prawy górny, kolejnym prawy dolny, a ostatnim lewy dolny. Do określenia orientacji grup wykrytych punktów oraz wyznaczenia po jednym z każdej grupy, punkty muszą być posortowane względem położenia. Z powodu braku ścisłego położenia punktów wykonywane jest to w taki sposób, że najpierw obliczana jest średnia z współrzędnych x-owych oraz y-owych najbardziej skrajnych punktów. Wartości te posłużą do wyznaczenia granic między czterema grupami. Aby wyróżnić po jednym punkcie z każdego zbioru wybierane są kolejno punkty:

1. o najmniejszej wartości x oraz największej wartości y w danej grupie,
2. o największej wartości x oraz największej wartości y w danej grupie,
3. o największej wartości x oraz najmniejszej wartości y w danej grupie,
4. o najmniejszej wartości x oraz najmniejszej wartości y w danej grupie.

Dodatkowo na samym początku punkty określane są z domyślną wartością. Tabela 5 przedstawia opisany wyżej algorytm do sortowania oraz wyboru punktów.

Tabela 5. Algorytm do sortowania punktów

```
float avrx = (xp1+xp2)/2;
float avry = (yp1+yp2)/2;
for(size_t j = 0; j < Cpoints.size(); j++){
       Pt = Cpoints[j];
       if(Pt.x < avrx && Pt.y < avry){ //lewy gorny
              Pt1 = Pt;
       }
       else if(Pt.x > avrx && Pt.y < avry){ // prawy gorny
              Pt2= Pt;
       }
       else if(Pt.x < avrx && Pt.y > avry){ //lewy dolny
              Pt4 = Pt;
       }
```

```
      else if(Pt.x > avrx && Pt.y > avry){ //prawy dolny
              Pt3 = Pt;
      }
}
for(size_t j = 0; j < Cpoints.size(); j++){
    Pt = Cpoints[j];
    if(Pt.x < avrx && Pt.y < avry){ //lewy gorny
        if(Pt.x < Pt1.x ){
            Pt1.x = Pt.x;
        }
        if(Pt.y > Pt1.y){
            Pt1.y=Pt.y;
        }
    }
    else if(Pt.x > avrx && Pt.y < avry){ // prawy gorny
        if(Pt.x > Pt2.x){
            Pt2.x= Pt.x;
        }
        if(Pt.y > Pt2.y){
            Pt2.y= Pt.y;
        }
    }
    else if(Pt.x < avrx && Pt.y > avry){ //lewy dolny
        if(Pt.x < Pt4.x){
            Pt4.x = Pt.x;
        }
        if(Pt.y < Pt4.y){
            Pt4.y = Pt.y;
        }
    }
    else if(Pt.x > avrx && Pt.y > avry){ //prawy dolny
        if(Pt.x > Pt3.x ){
            Pt3.x = Pt.x;
        }
        if(Pt.y < Pt3.y){
            Pt3.y = Pt.y;
        }
    }
}
```

Tak określone punkty zapisywane są w pliku konfiguracyjnym newconfig.cfg. Pozwala to na pobranie współrzędnych punktów przy kolejnym otworzeniu programu oraz niweluje potrzebę kalibracji przed każdym użyciem programu. Kalibracja powinna być powtórzona co około 4-5 pomiarów oraz za każdym razem gdy tuba z szczeliną ulegnie przesunięciu. Spowodowane jest to zmianą toru optycznego przy minimalnym przesunięciu tuby względem obiektywu kamery.

Gdy program zostanie poprawnie skalibrowany dodatkową czynnością za jaką jest odpowiedzialny użytkownik, jest ustawienie parametrów kamery internetowej. Do tego potrzebny

będzie program, który pozwoli na zarządzanie parametrami takimi jak jasność, nasycenie oraz ostrość. W tym przypadku zastosowano program GUVCViewer. Na wstępie należy ustawić kontrast na wartość równą zero, aby nie zostały wprowadzone do obrazu żadne filtry, które mogłyby zafałszować wynik. Kolejnym krokiem jest odpowiednie ustawianie jasności oraz nasycenia, aby otrzymane zdjęcia nie były prześwietlone. Metody do radzenia sobie z zbyt intensywnymi źródłami światła przedstawione zostaną w kolejnym rozdziale.

Kalibracja wykresu rozkładu widmowego została zrobiona podczas tworzenia kodu. Pozwala na to fakt, że transformacja obrazu przekształca pole zawierające widmo, do widoku o stałych wymiarach. Sprawdzano jak po dobrym wykryciu punktów brzegowych układają się prążki widma świetlówki kompaktowej. Jej widmo posiada charakterystyczne wąskie paski, dla tego najłatwiej można było określić długości fali nawet jeśli transformacja nie rozciągła wykrytego obrazu w sposób liniowy. Wartości długości fali w wykresie sporządzanym w ostatnim punkcie działania aplikacji wzorowane były na pomiarach sporządzonych fotometrem ColorMunki. Jeżeli widmo zostało dobrze wykryte, średnie wartości luminacji w tabeli określają długości fali od 400nm do 665nm. Dodatkowo wartości każdego elementu przedstawiającego wartości luminancji dla danej długości fali zmieniają się co 0.447 nm/piksel dla długości mniejszych od 490nm, dla wartości w zakresie od 490nm do wartości mniejszych od 545nm o 0.445 nm/piksel, a dla długości większych od 545 wartości zmieniają się o 0.473 nm/piksel. Taka nieliniowa zależność wynika z faktu, że obraz jest poddawany transformacji nie zachowując proporcji wymiarów obrazu.

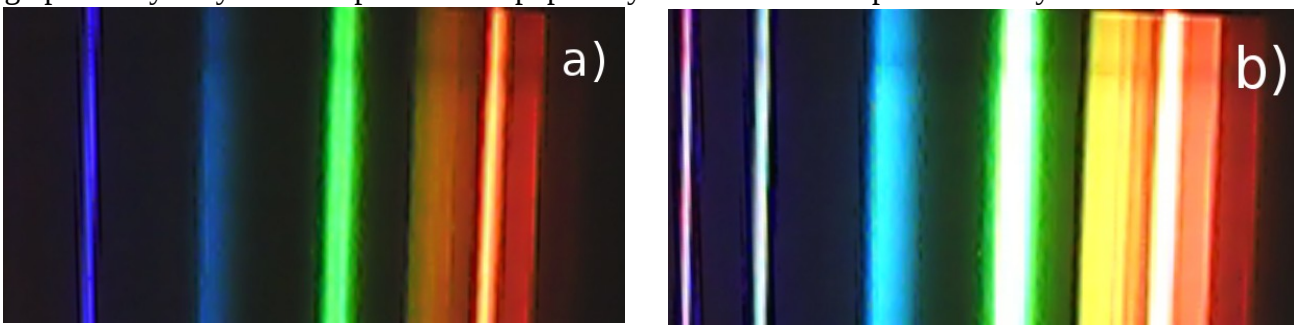
3. Analiza wyników

Ten rozdział ma na celu przedstawić rezultaty pomiarów. Został podzielony na dwa podrozdziały opisujące różne czynniki mające wpływ na wynik pomiaru. W pierwszym znajdować się będzie opis działania aplikacji z różnymi rodzajami źródeł światła białego oraz porównanie jakościowe do spektrofotometru ColorMunki. W drugim podrozdziale porównana zostanie praca z użyciem różnych kamer internetowych, posiadających różne układy soczewek oraz matryce o różnej czułości.

3.1 Działanie aplikacji z użyciem różnych źródeł światła białego

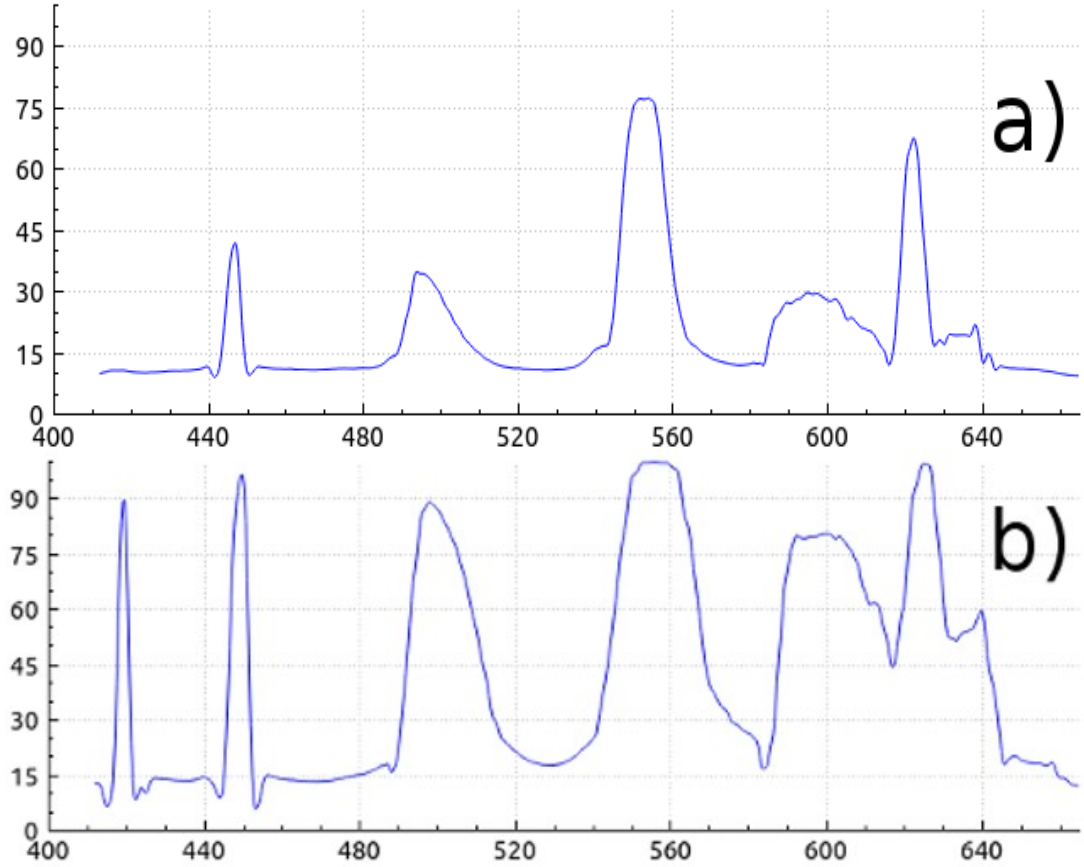
Emitowanie światła białego może zostać wywołane w różny sposób. Dostępne są źródła światła, które wykorzystują wyładowanie w gazach lub parach rtęci (lampy rtęciowe), sodu (lampy sodowe), w mieszaniach par rtęci i halogenków metali (lampy metalohalogenkowe), a także rekombinacje elektronów i dziur w złączu półprzewodnikowym (diody LED) oraz żarowe źródła światła [4]. W zależności od tego z jakiego źródła pochodzi światło, widmo może wyglądać inaczej.

W tym projekcie do pomiarów użyto głównie żarówek halogenowych, świetlówek kompaktowych potocznie zwanych żarówkami energooszczędnymi oraz źródeł światłą LED. W zależności od tego jakie źródło zostało zbadane oraz jakiej mocy ono było pomiary mogły wyjść nie doświetlone lub prześwietlone. Jeżeli na zdjęciach uzyskanych z kamery, w miejscach prążków kolorowych znajduje się obraz o białym kolorze, oznacza to że pomiar jest prześwietlony i należy go powtórzyć. Rysunek 20 przedstawia poprawny obraz oraz obraz prześwietlony.



Rys. 20 Pomiar a) poprawny, b) prześwietlony

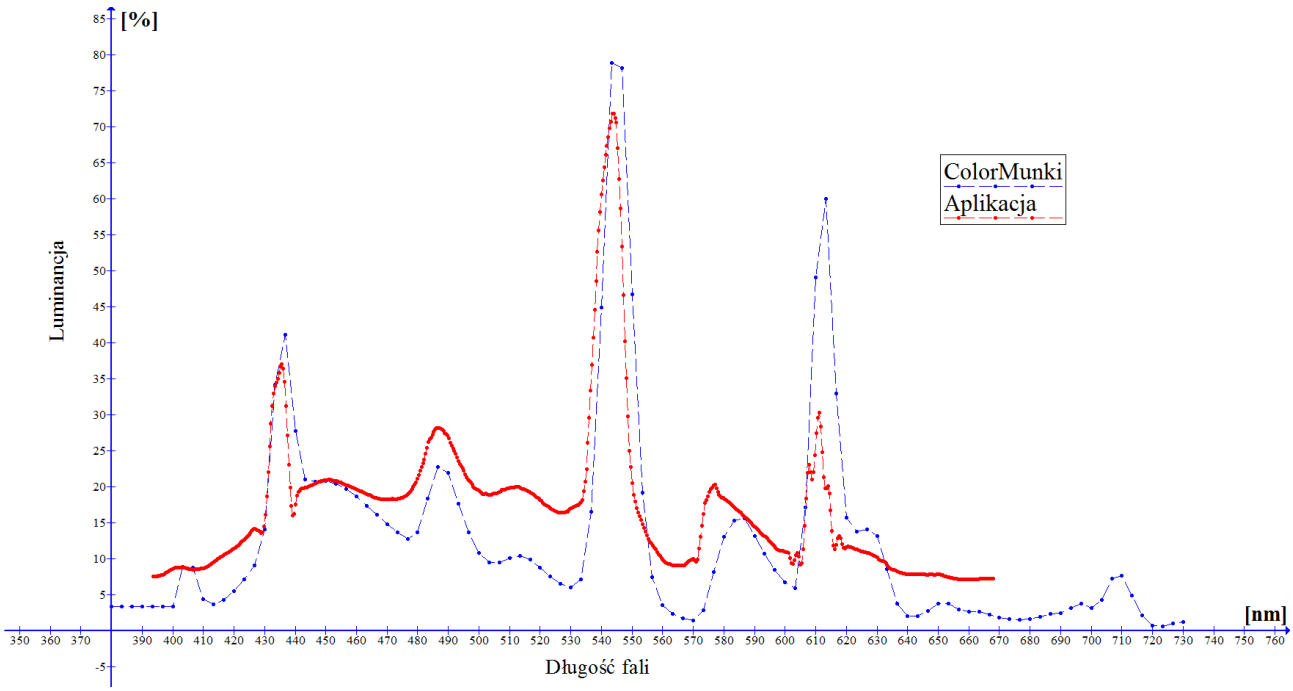
Skutkiem prześwietlonych zdjęć jest zła interpretacja luminancji. Wykres przedstawiający rozkład widma jest zawyżony, aczkolwiek w przypadku z byt małej energii prążka może on nie zostać zarejestrowany przez kamerę. Spowodowane jest to tym, że tanie kamery internetowe mogą zarejestrować dany kolor od pewnego progu wartości intensywności. Rysunek 21 przedstawia wykresy widma dla tego samego źródła odpowiadające poprawnemu pomiarowi oraz pomiarowi prześwietlonemu.



Rys. 21 Wykres rozkładu widmowego dla pomiaru a-poprawnego, b- prześwietlonego

Najbardziej wrażliwymi pomiarami na prześwietlenie są pomiary świetlówek. Powodem tego jest duża energia skupiona w wąskich prążkach widma. Mniej wrażliwe są źródła halogenowe oraz LED, dla których widmo jest ciągłe. Dodatkową wadą zbyt intensywnych źródeł światła są niechciane obrazy widma wchodzące w kadr spowodowane załamywaniem się światła na soczewce obiektywu kamery. Aby poradzić sobie z prześwietlonymi zdjęciami oraz niechcianymi widmami źródło światła można odseparować od urządzenia pomiarowego cieką białą kartką na przykład papieru pergaminowego.

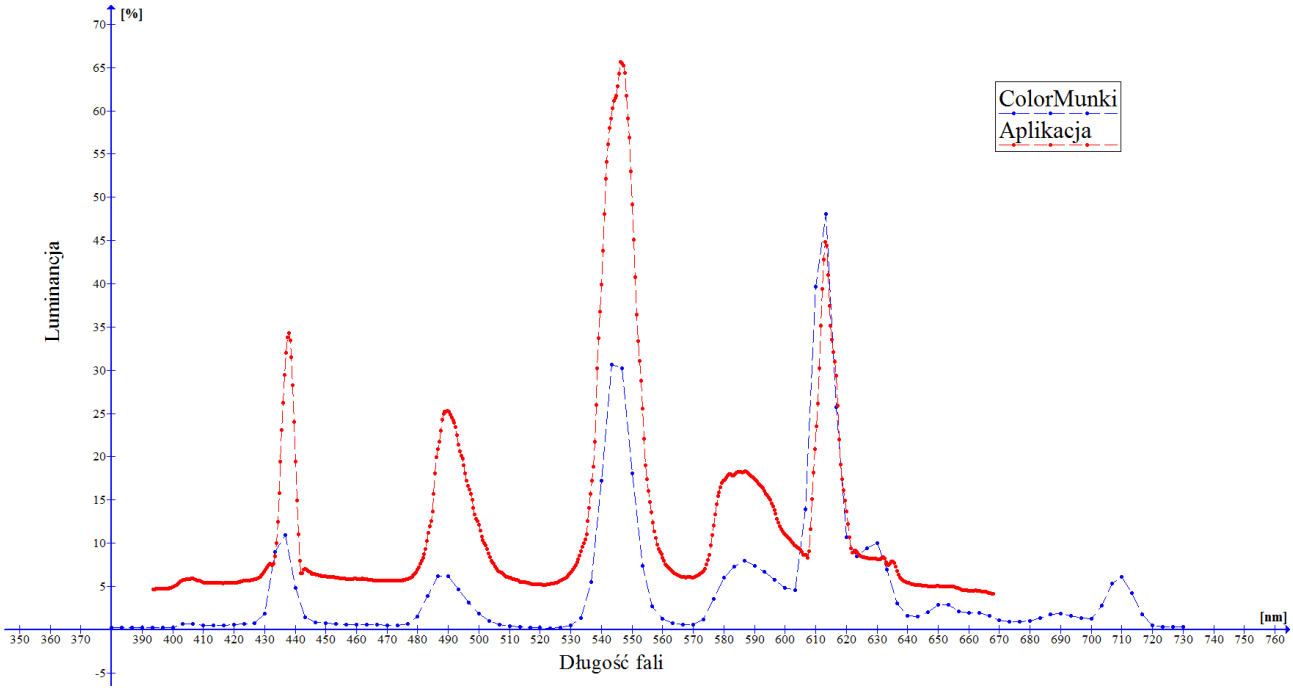
Gdy układ jest już prawidłowo oświetlony można porównać wyniki układu z wzorcem pomiarowym spektrofotometrem ColorMunki dla różnych źródeł światła białego. Jako pierwsze porównanie będzie przedstawiony wynik (patrz rysunek 22) pomiaru świetlówki kompaktowej firmy OSRAM model DST MITW 15W/865 E27.



Rys. 22 Porównanie wyników żarówki DST MITW 15W/865 E27

kolor czerwony-spektrometr, kolor niebieski- ColorMunki

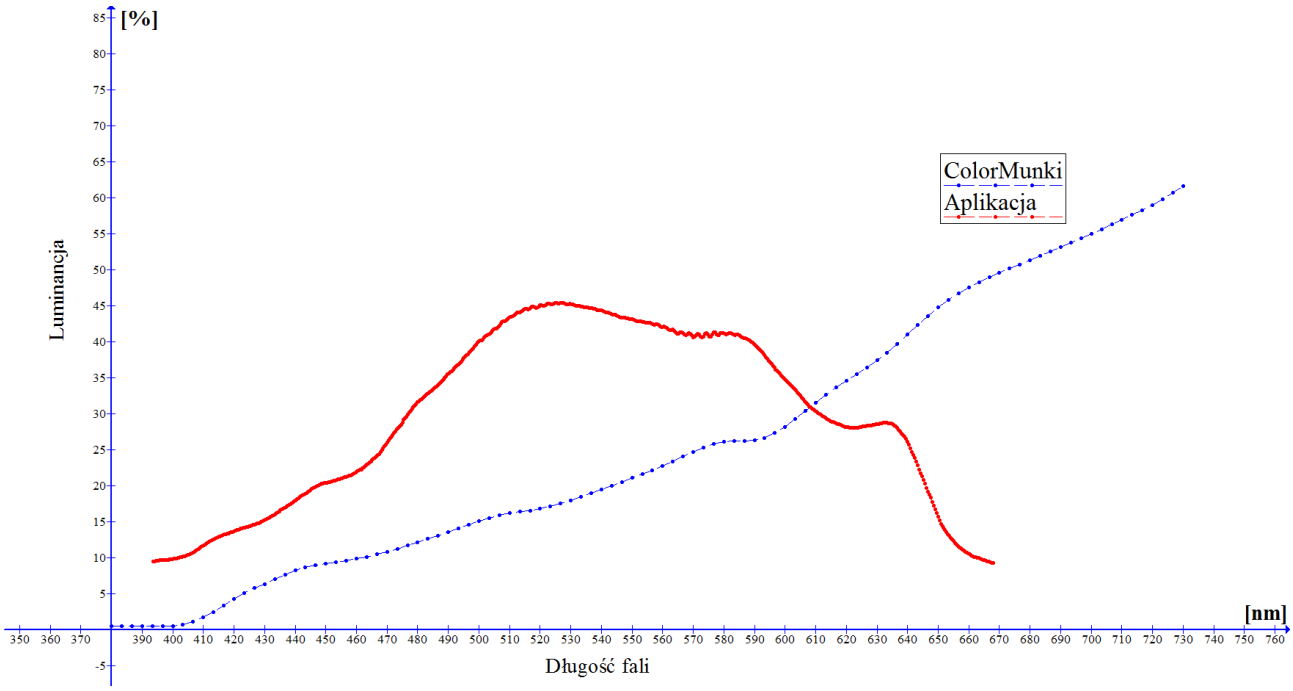
Kolejnym źródłem światła do porównania jest świetlówka firmy OSRAM model DST STICK 11W/827 E27. Porównanie wyników przedstawia rysunek 23.



Rys. 23 Porównanie wyników żarówki DST STICK 11W/827 E27

kolor czerwony-spektrometr, kolor niebieski- ColorMunki

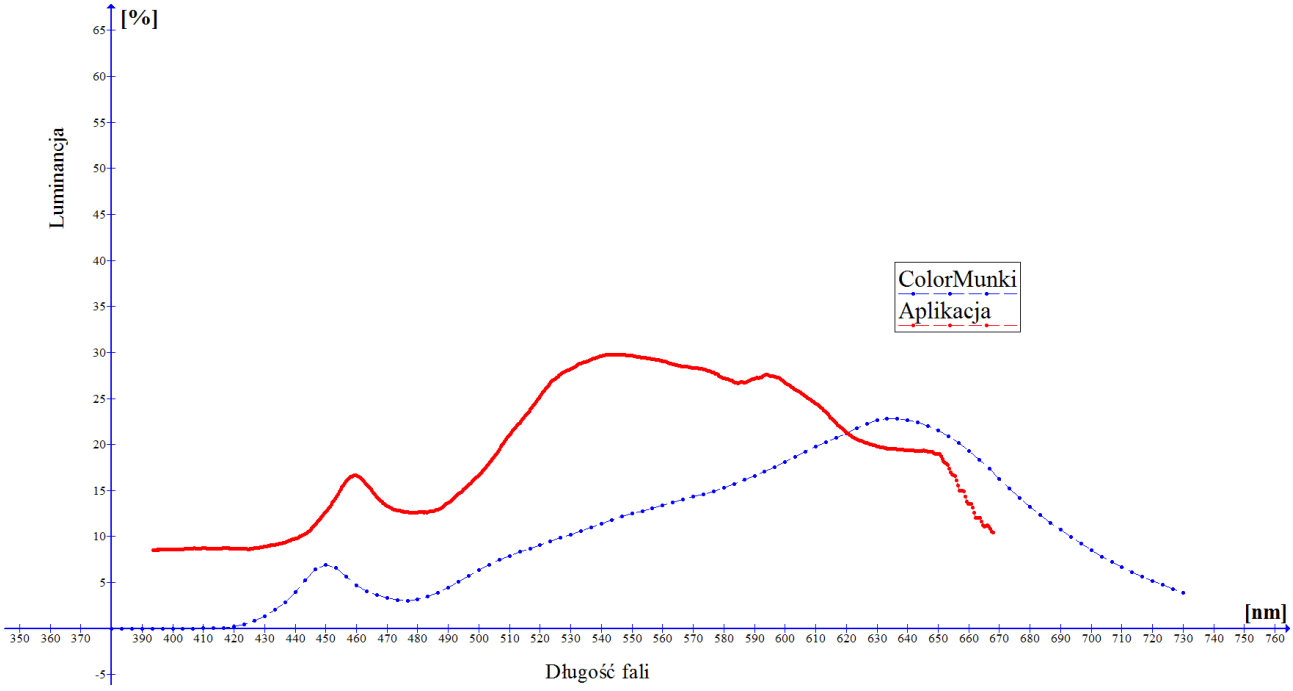
Następne źródło światła białego porównane w tym przykładzie to żarówka halogenowa firmy OSRAM model HAL CL A 105 W 230 V E27. Porównanie wyników przedstawia rysunek 24.



Rys. 24 Porównanie wyników żarówki HAL CL A 105 W 230 V E27

kolor czerwony-spektrometr, kolor niebieski- ColorMunki

Ostatnim źródłem porównywanym z pomiarami ColorMunki jest żarówka LED o mocy 6W. Porównanie wyników przedstawia rysunek 25.



Rys. 25 Porównanie wyników żarówki LED 6W

kolor czerwony-spektrometr, kolor niebieski- ColorMunki

Wyniki przedstawione na rysunkach 22 oraz 23 pokazują, że długości fali określane są z dużą dokładnością. Dla źródeł światła o nieciągłym widmie jesteśmy w stanie określić jakie długości fali biorą udział w promieniowaniu. Z rys 22 można wywnioskować, że czułość kamery w zakresie długich fal jest mniejsza. Spowodowane jest to źle odbierana przez kamerę luminancją. Im długość

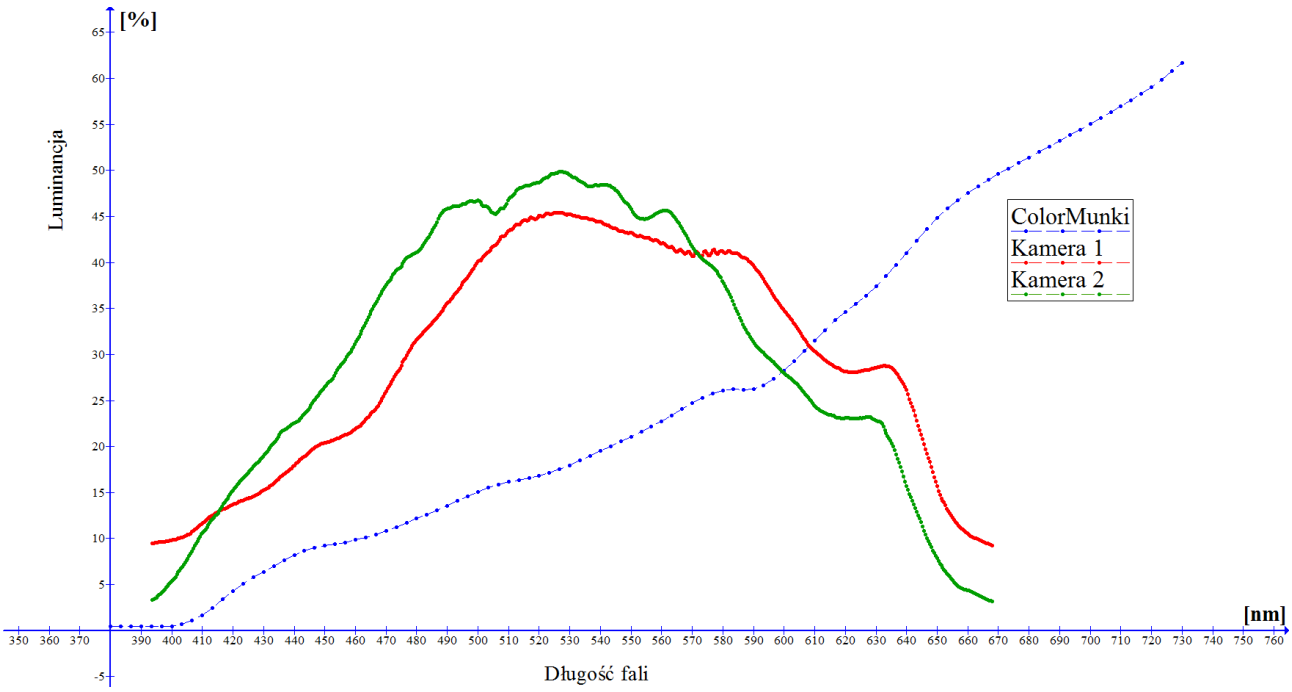
fali jest większa tym z większym błędem przedstawiana jest wartość luminancji. W skutek tego wyniki na rysunkach 24 i 25 określają wyłącznie czy dana długość fali w zakresie od 385 nm do 670 nm wchodzi w skład promieniowania źródła światła białego. Niestety wartości luminancji dla aplikacji nie są satysfakcjonujące. Dodatkowym problemem stają się artefakty spowodowane cyfrową poprawą ostrości. Widoczne one są na wykresach jako wcięcia w miejscach przechodzenia wykresu z dużej wartości luminancji do mniejszej. Widoczne jest to na rysunku 22 na wykresie czerwonym w okolicach długości fali o 440 nm.

3.2 Praca aplikacji z użyciem różnych kamer internetowych

Każdy model kamery internetowej posiada innej jakości matryce oraz indywidualnie dobrany zestaw soczewek. Ma to wpływ na budowę układu pomiarowego jak i na jakość odbieranego obrazu. W tym projekcie zastosowano dwie budżetowe kamery firmy Media-tech model MT4047 oraz firmy Msonic model MR1803B.

Pierwszymi widocznymi problemami przy takich kamerkach jest szum. Jego wartość wzrasta wraz z rozgrzewaniem się kamery. Dla kamery internetowej firmy Media-Tech, po czasie pracy przekraczającym 30 min, obraz jest zaszumiony w takim stopniu, że nie jest możliwa ponowna poprawna kalibracja. Spowodowane jest to wzrostem temperatury układu elektronicznego w środku kamerki internetowej oraz przekazywaniem energii cieplnej przez dłoń trzymającą układ. Z tego powodu nie zaleca się, aby pomiary robione były dłużej niż 20 min.

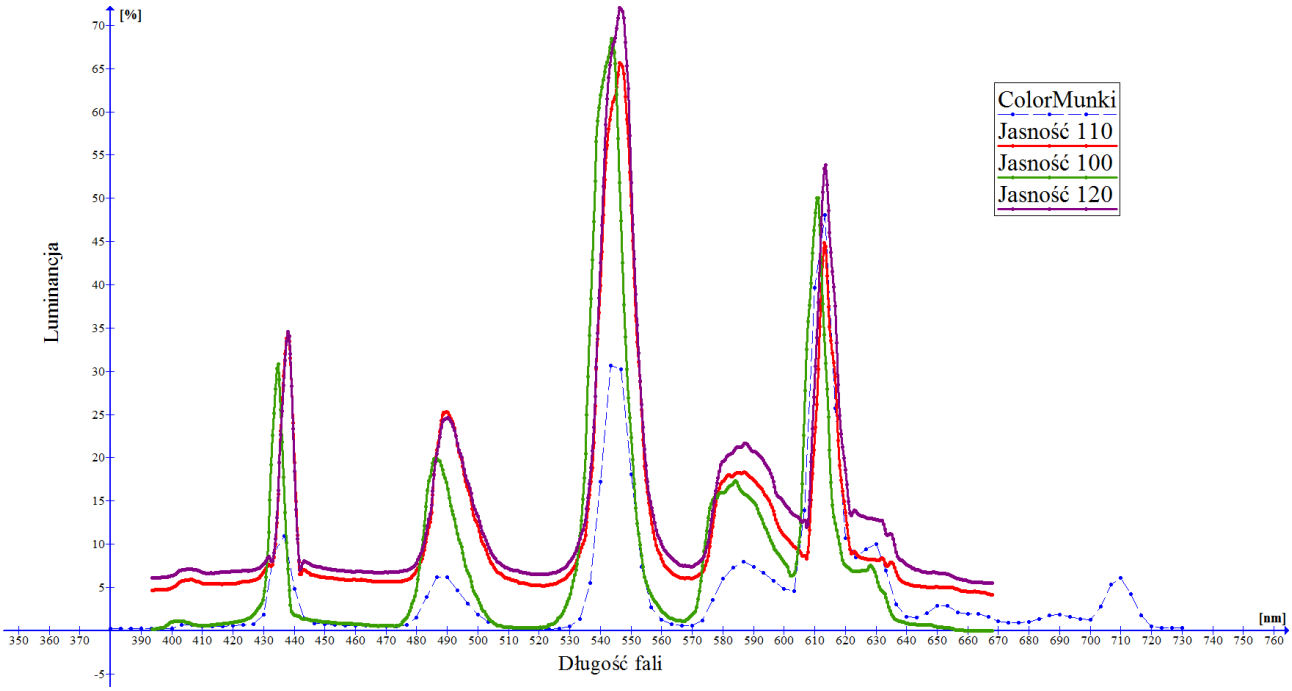
Kolejnym aspektem korzystania z różnych kamer jest różna czułość matryc na kolor oraz luminancje. Porównując pomiary ręcznie zrobionymi spektrometrami do urządzenia wzorcowego, widać, że luminancja w zależności od długości fali jest inaczej rejestrowana. Dodatkowo dla różnych kamer internetowych w różnym stopniu wykresy są zdeformowane. Najlepiej widać to zjawisko dla pomiaru rozkładu widmowego dla żarówki halogenowej gdzie widmo jest ciągłe.



Rys. 26 Pomiary lampy halogenowej kolor niebieski- ColorMunki, kolor zielony- kamera Media-Tech kolor czerwony- kamera Msonic

Jak widać na rysunku 26, luminancja powinna stopniowo wzrastać wraz z wzrostem długości fali. Jednak w przypadku spektrometrów opartych na kamerze internetowej wartość luminancji rośnie do około długości fali równej 520nm, dla większych długości wartość luminancji spada. Kamery internetowe odwzorowują kolory z pewnym błędem, co ma wpływ na wartość wyliczonej luminancji. Dodatkowym problemem jest fakt, że każda kamera internetowa posiada inną czułość na kolory. Widoczne jest to na rysunku 26 dla długości od 480 nm do 590 nm, poza poziomem wartości luminancji wykresy dla kamer Media-Tech oraz Msonic w znaczącym stopniu różnią się kształtem. Z tego powodu nie była możliwa poprawna kalibracja luminancji względem wzorcowych pomiarów.

Użytkownik może w swoim zakresie poprawić odczyt wyniku ustawiając odpowiednio do kamery wartość jasności w programie GUVCViewer. Rysunek 27 przedstawia pomiar świetlówki kompaktowej zrobiony za pomocą ColorMunki oraz za pomocą ręcznie zbudowanego układu pomiarowego z ustawionymi różnymi wartościami jasności.



Rys. 27 Pomiar świetlówki kompaktowej kolor niebieski: ColorMunki, kolor zielony: kamera Msonic jasność równa 100, kolor czerwony: kamera Msonic jasność równa 110 kolor fioletowy: kamera Msonic jasność równa 120

Jak widać na rysunku 27 poza obniżeniem poziomu czerni, zmniejszając jasność niwelowane są artefakty spowodowane cyfrową poprawą ostrości.

Dodatkowym aspektem jest czułość kamery. Każda kamera od pewnego progu intensywności koloru jest w stanie go zaobserwować. Ma to wpływ na wyszukanie 4 punktów wyznaczających położenie widma w kadrze kamery. Podczas kalibracji programu do danego układu pomiarowego użytkownik musi zadbać aby widmo zajmowało jak największy obszar kadru oraz aby wszystkie kolory były bardzo dobrze widoczne. Żeby to zapewnić należy manewrować parametrami jasności oraz nasycenia, pamiętając o tym aby w kadrze znajdowało się tylko jedno widmo. Czułość kamery internetowej ma także wpływ na późniejsze pomiary, a mianowicie może nie wykryć fali która bierze udział w promieniowaniu, ponieważ ma zbyt małą intensywność.

Podsumowanie

W projekcie udało się zbudować układ pomiarowy skonstruowany z tanich i dostępnych urządzeń. Dodatkowo wykonany został program z graficznym interfejsem użytkownika służący do akwizycji widma światła białego. Dzięki funkcji kalibracji urządzenia, użytkownik jest w stanie zbudować układ pomiarowy w oparciu o większość kamer internetowych. Po skalibrowaniu układu pomiarowego, najważniejsze informacje zapisywane są w pliku konfiguracyjnym, co pozwala na zrobienie kilku pomiarów bez konieczności ciągłej kalibracji. Aplikacja posiada kilka algorytmów pozwalających radzić sobie z kamerkami słabej jakości oraz niwelować niedoskonałości, zbudowanych metodami domowymi, spektrometrów. Między innymi algorytm do zintegrowania dziesięciu zdjęć oraz algorytm dwupoziomowego wykrywania brzegów widma. Bez względu na to gdzie znajduje się widmo w kadrze kamery, program po poprawnej kalibracji, korzystając z geometrycznej transformacji, rozciągnie je do obrazu o rozmiarach 300 pikseli wysokości na 600 pikseli szerokości w celu łatwiejszej analizy. Wyniki pomiarów zapisywane są w postaci wykresu luminancji w funkcji długości fali.

Z powodu różnej jakości kamer internetowych, luminancja jest rejestrowana z pewnym błędem. Nakłada to na użytkownika konieczność dodatkowej kalibracji układu pomiarowego zewnętrznymi programami, pozwalającymi na sterowanie takimi parametrami jak ostrość, jasność oraz nasycenie. Dodatkowo, aby program mógł być poprawnie skalibrowany, osoba korzystająca z aplikacji musi zadbać, by obraz nie był prześwietlony stosując np. papierowe zasłony.

Oprogramowanie zostało napisane w taki sposób, aby można było je łatwo rozwinąć o nowe funkcjonalności. Dla przykładu można dodać funkcje generujące wartości współczynnika CRI lub obliczające temperaturę barwową źródła światła białego. Bardzo dobrym pomysłem na rozbudowanie aplikacji jest dodanie możliwości zmiany parametrów kontrastu, jasności oraz ostrości. Dzięki temu użytkownik nie będzie musiał używać dodatkowych programów. Program można także rozwinąć o kalibrację luminancji względem pomiarów wzorcowych. Dodatkowo istnieje możliwość zaprojektowania aplikacji mobilnej dla systemu operacyjnego Android. Aplikacja oparta na algorytmach zawartych w tym projekcie umożliwiałaby badanie widma światła białego w każdej chwili, w każdym miejscu. Aparaty w smartfonach zazwyczaj są lepszej jakości niż zwykła kamera internetowa. Ich atutem w przeciwieństwie do budżetowych kamer jest także posiadanie funkcji auto focus, zwalniającej użytkownika z ręcznego ustawiania punktu ostrości w układzie pomiarowym.

Podobne układy były już projektowane. Na przykład spektrofotometr zaprojektowany przez społeczność Public Lab. Jest budowany w oparciu o kosztowną kamerę, którą można kupić na stronie [14] lub w oparciu o kamerę telefonu komórkowego. Dodatkowo posiada darmową aplikację do akwizycji danych. Jednak w tym przypadku należy wykupić kosztowne elementy do budowy układu pomiarowego .

Literatura:

[1] Barbara Bukowska, Maria Goryl *„Badanie widm emisyjnych za pomocą spektroskopu pryzmatycznego”* http://www.kariera.fais.uj.edu.pl/media/pliki/PF-10.pdf(z dnia 6.12.2015)

[2] Przemysław Oziemblewski *„Technika świetlna od podstaw”*, www.swiatlo.tak.pl Bezpłatny e-book dla czytelników„Biuletynu informacyjnego serwisu Światło i oświetlenie” www.swiatlo.tak.pl WydanieI-wersja 1.01

[3] David Halliday, Robert Resnick, Jearl Walker *„Podstawy Fizyki”* tom 4

[4] Wojciech Żagan *„Podstawy techniki świetlnej”*

[5] *„Pomiar długości fali świetlnej za pomocą siatki dyfrakcyjnej(O16)”* http://www.uj.edu.pl/c/document_library/get_file?uuid=8c3af751-df0a-4b65-b247-0eed0e08ccad&groupId=5046939 (z dnia 6.12.2015)

[6] Angela Turricchia, Ariel Majcher *„Domowy spektroskop”* http://www.pl.eu-hou.net/index.php/wiczenia-mainmenu-13/mierzymy-otaczajcy-nas-wiat-mainmenu-139/132-domowy-spektroskop (z dnia 6.12.2015)

[7] Sebastian Wojas *„Metody przetwarzania obrazów z wykorzystaniem biblioteki OpenCV”*

[8] Biblioteka OpenCV http://opencv.org (z dnia 6.12.2015)

[9] Biblioteka libconfig http://www.hyperrealm.com/libconfig (z dnia 7.12.2015)

[10] Biblioteka Qt http://www.qt.io/developers/ (z dnia 7.12.2015)

[11] GTK vs Qt https://www.wikivs.com/wiki/GTK_vs_Qt (z dnia 7.12.2015)

[12] Gary Bradski, Adrian Kaehler *„Learning OpenCV: Computer Vision with the OpenCV Library*”, „O'Reilly Media, Inc.”, 24.09.2008 - 580

[13] A.Materka, P.Strumiłło *„Wstęp do komputerowej analizy obrazów”* https://www.researchgate.net/profile/Andrzej_Materka/publication/256079247_Wstp_do_komputerowej_analizy_obrazw/links/00b7d52174f92803c2000000.pdf#page=53

[14] Public Lab https://publiclab.org/ (z dnia 30.12.2015)