Science, Technology and Innovation
ISSN 2544-9125 ▪ 2021; 12 (1): 16–25
doi: 10.5604/01.3001.0015.2458

STI • 2021;12 (1): X-X

Mobilny system przetwarzania obrazu oparty na platformie Raspberry Pi 4

Mobile image processing system based on the Raspberry Pi 4 platform

Łukasz Chlastawa

Państwowa Wyższa Szkoła Zawodowa w Tarnowie, ul. Mickiewicza 8, 33-100 Tarnów, Polska

Artykuł oryginalny

Abstrakt

W artykule zaprezentowano system przetwarzania obrazu oparty na platformie Raspberry Pi (RPi). Na początku artykułu omówiono podstawowe założenia oraz cel wykonania systemu. W dalszej części przedstawiono strukturę i sposób działania systemu. Zaprezentowano aplikację okienkową zarządzającą systemem oraz pozwalającą na wykonanie przekształceń kontekstowych oraz widmowych na obrazach, jak również pomiar parametrów, takich jak: czas przetwarzania obrazu oraz błąd średniokwadratowy (ang. *Mean Square Error* – MSE). Wykonywane przekształcenia oparto zarówno na gotowych formułach zawartych w bibliotece OpenCV, jak również własnych implementacjach, a wśród nich na funkcji realizującej algorytm szybkiej transformacji Fouriera FFT (ang. *Fast Fourier Transform*) radix-2. Zaprezentowano przykłady przekształceń wraz ze wskazaniem ich użyteczności. Na końcu przedstawiono potencjał rozwojowy utworzonego systemu oraz zaproponowano zastosowanie w konkretnych rozwiązaniach.

Słowa kluczowe

- cyfrowe przetwarzanie obrazów
- dwuwymiarowa szybka transformacja Fouriera
- biblioteka OpenCV
- aplikacja okienkowa
- system wizyjny

Abstract

The article presents an image processing system based on the Raspberry Pi (RPi) platform. At the beginning of the article, the basic assumptions and purpose of the system are discussed. The following section presents the structure and operation of the system. The window application managing the system and allowing to perform contextual and spectral transformations on images as well as the measurement of parameters such as image processing time and mean square error (MSE) was discussed. The transformations performed were based both on ready formulas contained in the OpenCV library and the author's implementations, including the function implementing the Fast Fourier Transform algorithm radix-2. Examples of transformations were presented along with their usefulness. In the end, the development potential of the created system is presented and its application in specific solutions is proposed.

Korespondencja

Łukasz Chlastawa
e-mail: l_chlastawa@pwsztar.edu.pl
Państwowa Wyższa Szkoła Zawodowa
w Tarnowie
Wydział Politechniczny
Katedra Elektroniki, Telekomunikacji
i Mechatroniki
ul. Mickiewicza 8
33-100 Tarnów, Poland

Informacje o artykule

Historia artykułu (Article history)

- Otrzymano (Received): 2021-02-07
- Zaakceptowano (Accepted): 2021-08-13
- Opublikowano (Published): 2021-08-13

Wydawca (Publisher)

Państwowa Wyższa Szkoła Zawodowa w Tarnowie
University of Applied Sciences in Tarnow
ul. Mickiewicza 8, 33-100 Tarnow, Poland

Finansowanie (Financing)

Badania nie zostały sfinansowane z grantów pochodzących ze środków publicznych, organizacji komercyjnych lub non-profit.

Konflikt interesów (Conflict of interest)

Nie zadeklarowano konfliktu interesów.

Wstęp

Systemy wizyjne odgrywają obecnie istotną rolę w przemysłowych zakładach produkcyjnych, mając za zadanie nie tylko monitorowanie procesu, ale również sterowanie nim [1]. Rozmaite algorytmy przetwarzania obrazu umożliwiają detekcję obiektów na obrazie, określenie ich pozycji, kształtu, wymiarów, jak również ich śledzenie. Typowy system przetwarzania obrazu stosowany w rozwiązaniach przemysłowych zawiera wyspecjalizowane kamery lub sensory wizyjne, posiadające niejednokrotnie moduł wstępnego przetwarzania i analizy obrazów [2]. Systemy wizyjne mogą znaleźć zastosowania nie tylko w przemyśle, ale również w wielu dziedzinach życia. Na rynku istnieje wiele firm, które oferują kompletne systemy wizyjne specjalizowane do wybranych zastosowań [3]. Takie systemy cechuje duża wydajność dzięki zastosowaniu szybkich przetworników obrazu, wysokiej jakości oświetlenia i zaawansowanych układów elektronicznych.

Pomimo praktycznych korzyści wynikających z zastosowania systemów wizyjnych, ich niejednokrotnie wysoki koszt uniemożliwia wprowadzenie wspomnianych rozwiązań przez niewielkie przedsiębiorstwa lub instytucje, które mogłyby w ten sposób usprawnić swoją pracę. Do wielu zastosowań nie jest konieczne zapewnienie najwyższej niezawodności produktu, a ewentualna awaria nie wiąże się – poza stratami sprzętowymi – z dodatkowymi kosztami. Przedstawiony w artykule system może stanowić alternatywę dla bardziej złożonych, ale zarazem droższych rozwiązań. Może posłużyć do zastosowań niewymagających wysokiej odporności na zakłócające czynniki zewnętrzne, w niewielkich zakładach, również do celów ekperymentalnych, amatorskich, edukacyjnych, za to jego wprowadzenie może być bardzo łatwe i nie wymaga dużej ingerencji w istniejącą infrastrukturę. System jest zbudowany z łatwo dostępnych i niedrogich komponentów, może być ciągle rozwijany i dostosowywany do potrzeb użytkownika. Swoją funkcjonalnością może dorównać wielu specjalizowanym systemom wizyjnym.

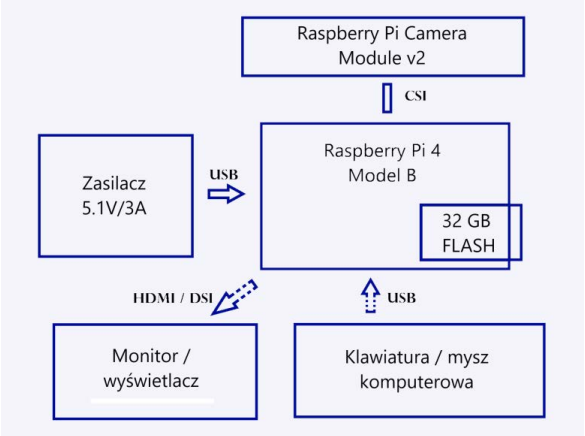
Materiały i metody

Utworzony system przetwarzania jest oparty na module Raspberry Pi 4 Model B, który jest jednopłytkowym komputerem posiadającym szereg urządzeń peryferyjnych i złącz. Podstawowe parametry urządzenia przedstawia tabela 1.

Tabela 1. Podstawowe parametry modułu Raspberry Pi 4 Model B [4]

procesor	Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
pamięć RAM	2GB, 4GB or 8GB LPDDR4-3200 SDRAM (w zależności od modelu)
złącza	2 x micro HDMI, 2 x USB 3.0, 2 x USB 2.0, Gigabit Ethernet, DSI, CSI, 4-polowe złącze audio/video typu minijack, 40-pinowe złącze kołkowe z wyprowadzeniami GPIO
zasilanie	5V/3A USB-C/GPIO
komunikacja bezprzewodowa	2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE

Rysunek 1 przedstawia schemat blokowy systemu. Poza minikomputerem RPi system wyposażono w dedykowany zasilacz, kartę pamięci 32 GB oraz moduł z kamerą Raspberry Pi Camera Module v2. Aparat cechuje się rozdzielczością 3280 x 2464 px. Moduł obsługuje również tryby wideo: 1080p30 (full-HD), 720p60, 640 x 480p90 (VGA), a komunikacja następuje za pomocą interfejsu CSI (ang. *Camera Serial Interface*).



Rysunek 1. Schemat blokowy systemu wizyjnego opartego na RPi

Opcjonalnym wyposażeniem systemu jest dotykowy wyświetlacz TFT (ang. *Thin-Film Transistor*), posiadający interfejs DSI. Istnieje także możliwość podłączenia monitora przez gniazdo HDMI. Do pracy z systemem można wykorzystać klawiaturę i mysz komputerową, jak również połączenie bezprzewodowe. Do działania modułu niezbędne jest odpowiednie oprogramowanie.

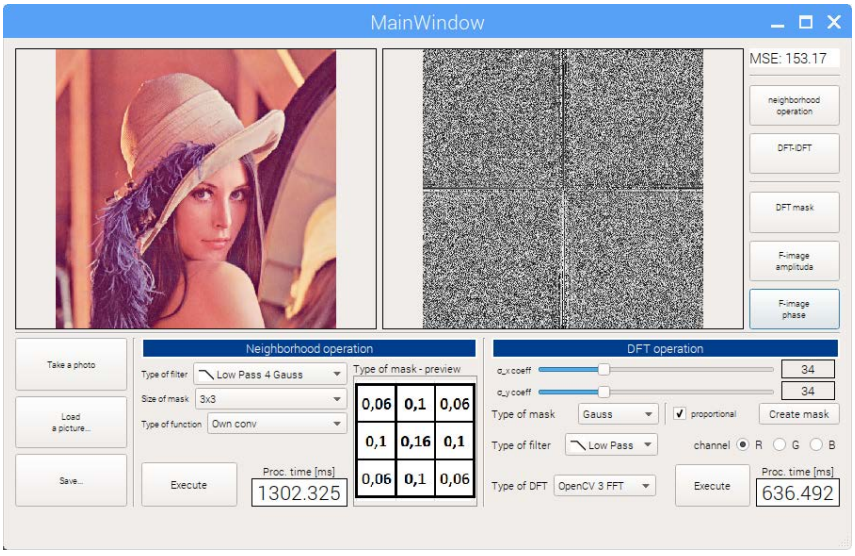
Na karcie pamięci zainstalowano system operacyjny Raspbian, po czym utworzono aplikację okienkową działającą pod jego kontrolą. Rysunek 2 przedstawia fotografię gotowego do pracy systemu. W przedstawionym przykładzie moduł RPi zintegrowany jest w jednej obudowie z wyświetlaczem.

Rysunek 2. Fotografia systemu wizyjnego, w którym wykorzystano dotykowy wyświetlacz TFT

Spełniane przez system funkcje zależą w głównej mierze od oprogramowania, dlatego w łatwy sposób można dostosować system do dowolnego zastosowania, modyfikując lub tworząc nowe oprogramowanie.

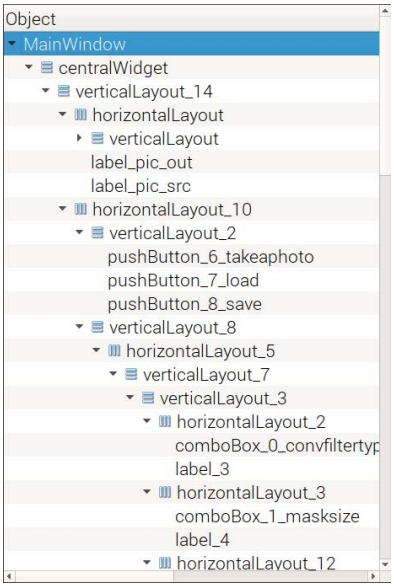
Budowa i użytkowanie aplikacji okienkowej

Opracowana aplikacja umożliwia realizację filtracji obrazu z wykorzystaniem przekształceń kontekstowych oraz widmowych. Kod aplikacji napisano w języku C++, a cały jej projekt zrealizowano w oprogramowaniu Qt Creator, wykorzystując przy tym również moduł Qt Designer. Wspomniany typ operacji na obrazach można wykorzystywać do różnych celów, w zależności od parametrów danego przekształcenia. Filtracja dolnoprzepustowa (ang. *Low Pass* – LP) może posłużyć do usuwania z obrazu zakłóceń, szumu, ale również do celowego rozmycia obrazu [5–6]. Poprzez zastosowanie filtru górnoprzepustowego możliwe jest zwiększenie ostrości obrazu jak również uwypuklenie krawędzi. Aplikacja w tej formie nie znajdzie szerokiego zastosowania, jednak posiada ona wiele istotnych funkcjonalności i stanowi bazę, na której można zbudować system o dowolnym przeznaczeniu. Na rysunku 3 przedstawiono interfejs graficzny aplikacji. Została ona podzielona na kilka sekcji. W górnej części okna znajduje się sekcja wyświetlania obrazu. Po lewej stronie wyświetlany jest obraz wejściowy, a po prawej stronie obraz po wykonanych przekształceniach. Panel znajdujący się po prawej stronie umożliwia dodatkowo przełączanie obrazu obserwowanego w prawym oknie. W wyniku przekształceń są bowiem generowane również pośrednie obrazy. W dolnej części okna po lewej stronie znajdują się przyciski pozwalające na wczytanie obrazu, wykonanie fotografii za pomocą zainstalowanej kamery, a także zapisanie obrazu wynikowego. Kolejno zamieszczone są dwie sekcje. Pierwsza z nich – *Neighborhood operation* – umożliwia przeprowadzenie operacji kontekstowych, a kolejna – *DFT operation* – przetwarzanie w dziedzinie częstotliwości. Dla każdego typu przetwarzania możliwe jest dobranie wielu parametrów.



Rysunek 3. Interfejs graficzny zaprojektowanej aplikacji okienkowej

Do projektowania interfejsu graficznego aplikacji użyto dostępnych w środowisku widżetów, które następnie grupowano w struktury nazywane *Layouts*. Rysunek 4 przedstawia fragment diagramu zawierającego powiązania pomiędzy poszczególnymi widżetami. Dla każdego widżetu zdefiniowano powiązane funkcje obsługujące akcje, takie jak naciśnięcie przycisku bądź wybór opcji z listy. W kodzie programu znalazło się ponadto wiele funkcji niezwiązanych bezpośrednio z widżetami i są to przede wszystkim funkcje realizujące określone algorytmy obliczeniowe. Funkcje te przytoczone są w dalszej części artykułu.



Rysunek 4. Diagram wzajemnych powiązań widgetów przedstawiony w oknie Object edytora Qt Designer

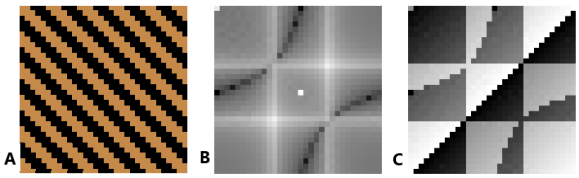
Przekształcenie kontekstowe polega na wyznaczeniu wartości piksela obrazu wyjściowego poprzez wykonanie obliczeń na wielu pikselach obrazu wejściowego, znajdujących się w otoczeniu wyliczanego piksela [5, 7]. Wśród przekształceń tego typu można wyróżnić filtry liniowe oraz nieliniowe. W aplikacji zastosowano filtrację liniową, którą realizuje się z wykorzystaniem splotu funkcji [6, 8]. Do takiej operacji niezbędna jest maska splotu, którą definiuje się w postaci kwadratowej macierzy współczynników [6, 9]. Odpowiedni rozkład współczynników decyduje o typie przeprowadzonej filtracji oraz jej głębokości. Okno aplikacji umożliwia wybór typu filtracji, rozmiaru zastosowanej maski oraz rodzaju funkcji, która ma być użyta do przetwarzania (tabela 2). Wśród typów filtracji dostępne są dolnoprzepustowe o różnych wagach, w tym dobranych zgodnie z funkcją Gaussa, jak również górnoprzepustowe. Wymiar krawędzi dla maski może być równy 3, 7, 11, 15 lub 19. Dostępne są dwie funkcje przetwarzające. Pierwszą z nich jest funkcja biblioteki OpenCV, drugą z kolei jest funkcja napisana we własnym zakresie. Obydwie funkcje dla masek o niewielkim rozmiarze (3 x 3; 7 x 7) dają identyczne rezultaty, co zostało sprawdzone podczas tworzenia oprogramowania. Dla większych masek funkcja biblioteki OpenCV nie wykonuje przekształcenia z definicji, ale wykorzystuje już przetwarzanie widmowe [10].

Tabela 2. Opcje pól wyboru w sekcji przekształceń kontekstowych

Type of filter	Size of mask	Type of function
Low Pass 1	3 x 3	OpenCV Filter2D(), w kodzie programu: filter2D()
Low Pass 2	7 x 7	Own conv, w kodzie programu: conv_own()
Low Pass 3	11 x 11	
Low Pass 4	15 x 15	
High Pass	19 x 19	

Przekształcenia widmowe, inaczej w dziedzinie częstotliwości, również umożliwiają filtrację obrazu [6]. Uzyskanie podobnych rezultatów do filtracji kontekstowej jest najłatwiejsze poprzez przeprowadzenie filtracji dolnoprzepustowej. Filtrację widmową można podzielić na kilka etapów. Pierwszym jest wyznaczenie F-obrazu, a więc wyliczenie reprezentacji częstotliwościowej obrazu źródłowego [6, 9]. Można go wyznaczyć obliczając dwuwymiarową transformatę Fouriera. W rezultacie otrzymuje się dwa obrazy, gdyż wynikowe próbki są liczbami zespolonymi. Podobnie jak dla sygnału jednowymiarowego, również tutaj każda próbka wynikowa powiązana jest z określoną częstotliwością, jednak w tym wypadku jest to para częstotliwości – jedna z nich odnosi się do wierszy, a druga do kolumn. Wygodną, czytelną formą jest przedstawienie F-obrazu w postaci amplitudy oraz fazy (rysunek 5). Wynikowe próbki są celowo przestawione. Po takiej operacji łatwo jest zlokalizować piksele odpowiadające za poszczególne częstotliwości. W środku obrazu znajduje się składowa stała, następnie wokół tej próbki znajdują się piksele odpowiadające niskim częstotliwościom [6]. Wraz z oddalaniem się od środka obrazu częstotliwości są większe. Na tak przygotowanym F-obrazie w łatwy sposób można dokonać wycięcia poszczególnych częstotliwości. Wystarczy zdefiniować okno częstotliwościowe o określonym kształcie oraz wymiarach, wymnożyć próbki F-obrazu przez to okno, a następnie zrealizować

odwrotną dwuwymiarową transformację Fouriera, aby przedstawić zmodyfikowany obraz w dziedzinie pierwotnej.



Rysunek 5. Przykładowy obraz spreparowany cyfrowo o rozdzielczości 32 x 32 px (A) oraz jego reprezentacja częstotliwościowa: moduł F-obrazu (B), faza F-obrazu (C).

W oknie aplikacji w sekcji przekształceń widmowych można dobrać wiele parametrów, badając ich wpływ na przebieg filtracji. Opcje dostępne w oknach wyboru przedstawia tabela 3.

Tabela 3. Opcje pól wyboru w sekcji przekształceń widmowych

Type of mask	Type of filter	Type of function
Gauss	Low Pass	OpenCV DFT, w kodzie programu: dft()
Rect	High Pass	Own DFT, w kodzie programu: dft_own()
		Own FFT radix-2, w kodzie programu: fft_own()

Działania te sprowadzają się do wygenerowania okna częstotliwościowego o żądanym kształcie. Można wybrać jeden z dwóch typów okna – prostokątne lub Gaussa. Okna umieszczane są centralnie. Dla każdego z nich można precyzyjnie dobrać wielkość za pomocą suwaków. Dla okna prostokątnego definiuje się wymiary boków prostokąta. W przypadku okna Gaussa można dostosować odchylenia standardowe dla dwuwymiarowej funkcji Gaussa określonej zależnością:

$$w_G(k,l) = A \cdot e^{-\left(\frac{(i-i_C)^2}{2\sigma_x^2} + \frac{(j-j_C)^2}{2\sigma_y^2}\right)} \quad (1)$$

gdzie A oznacza amplitudę, i oraz j współrzędne piksela, i_c oraz j_c współrzędne piksela centralnego, a σ_x^2 i σ_y^2 wariancje odpowiednio dla wymiaru poziomego oraz pionowego. W aplikacji udostępniono również możliwość wyboru filtracji dolnoprzepustowej lub górnoprzepustowej. Po dokonaniu wszystkich ustawień należy nacisnąć przycisk *Create mask*, który spowoduje wygenerowanie okna częstotliwościowego. Zostaje ono automatycznie wyświetlone w prawym oknie. Również w przypadku filtracji widmowej możliwy jest wybór funkcji przetwarzającej. Do wyliczenia dwuwymiarowej transformaty Fouriera jak również transformaty odwrotnej można wykorzystać funkcję dostępną w bibliotece OpenCV [11]. Dostępne są również dwie funkcje napisane we własnym zakresie. Pierwsza z nich umożliwia wyznaczenie transformaty z definicji. Druga natomiast umożliwia wyznaczenie transformaty z wykorzystaniem algorytmu szybkiej transformacji Fouriera FFT radix-2. Ten drugi sposób zasługuje na szczególną uwagę, gdyż powoduje znaczną redukcję niezbędnych obliczeń i decyduje o przydatności przetwarzania częstotliwościowego. Algorytm FFT radix-2 dla przetwarzania jednowymiarowego można przedstawić w formie rekurencyjnej, gdzie w kolejnych etapach stosowane jest podstawienie $N = N/2$ [12]:

$$\begin{bmatrix} F(k) \\ F(k+N/2) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} F(k) \\ W_N^{-k} F(k+N/2) \end{bmatrix}, \quad k = 0, 1, \ldots, N/2-1 \quad (2)$$

W rekurencyjnej zależności (2) $F(k)$ oznacza wartość próbki o określonym indeksie, N – liczbę próbek, a W_N jest czynnikiem wyodrębnionym z definicyjnego równania transformacji. Jego postać przedstawia zależność (3) [12].

$$W_N = e^{\frac{j2\pi}{N}} \quad (3)$$

W opisywanym zastosowaniu rozszerzono definicję transformacji jednowymiarowej wyrażonej wzorem (2) do operacji dwuwymiarowej (4) oraz (5). Transformacja dwuwymiarowa może zostać zrealizowana poprzez złożenie transformacji jednowymiarowych [6, 7, 9]. Wtedy macierzowa, rekurencyjna postać algorytmu zostaje rozszerzona o drugi wymiar i zostaje podzielona na dwa etapy. W przypadku obrazów oznacza to wyliczenie transformaty każdego wiersza w pierwszym etapie, a następnie wyliczenie z tak uzyskanych wartości transformaty dla każdej kolumny w drugim etapie, przy czym kolejność ta może być odwrotna. Pierwszy etap (dla transformacji poszczególnych wierszy) przedstawia zależność:

$$\begin{bmatrix} F(k,l) \\ F(k,l+N/2) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} F(k,l) \\ W_N^{-l} F(k,l+N/2) \end{bmatrix}, \quad l = 0, 1, \ldots, N/2-1, k = 0, 1, \ldots, M-1 \quad (4)$$

W równaniu (5) przedstawiono zapis algorytmu dla drugiego etapu:

$$\begin{bmatrix} F(k,l) \\ F(k+M/2,l) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} F(k,l) \\ W_M^{-k} F(k+M/2,l) \end{bmatrix}, \quad k = 0, 1, \ldots, M/2-1, \quad l = 0, 1, \ldots, N-1 \quad (5)$$

Taki sposób przeprowadzania filtracji może wydawać się złożony, jednak ma on istotną zaletę – niezależnie od rodzaju filtracji oraz jej głębokości wymaga on tyle samo obliczeń, a w przypadku szybkich algorytmów, takich jak FFT radix-2 cały proces przebiega o wiele szybciej. Inaczej jest w przypadku filtracji kontekstowej, gdzie przy zastosowaniu coraz większych masek czas obliczeń się wydłuża. Złożoność obliczeniową poszczególnych operacji można oszacować na podstawie ich definicji [6, 12]. W przypadku filtracji kontekstowej jest to liczba rzędu $N \cdot M \cdot U \cdot V$, gdzie N i M są wymiarami obrazu, a U oraz V wymiarami zastosowanej maski. Po przyjęciu $M = N$ (obraz kwadratowy) oraz $V = U$ (maska kwadratowa) można określić koszt obliczeniowy jako liczbę rzędu $N^2 \cdot U^2$. Koszt obliczeniowy należy tu rozumieć jako rząd wielkości dla liczby mnożeń podczas realizacji przekształcenia. Złożoność obliczeniowa transformacji Fouriera przeprowadzonej wprost z definicji jest zdecydowanie większa, ponieważ jest to liczba rzędu $N^2 \cdot M^2$, co po założeniu $M = N$ jest równe N^4. W dodatku działania wykonywane są na liczbach zespolonych, a w celu filtracji konieczne jest również wyznaczenie transformaty odwrotnej. Nieco lepsze wyniki daje przeprowadzenie transformacji dwuwymiarowej jako złożenia transformacji jednowymiarowych. Wówczas koszt obliczeniowy takiej operacji jest rzędu $N^2 \cdot M + N \cdot M^2$, a po przyjęciu $M = N$ jest to liczba rzędu N^3. To jednak nadal dużo w porównaniu do filtracji kontekstowej z maską o niewielkich wymiarach. Znaczną poprawę wydajności może dać za to zastosowanie algorytmu FFT radix-2, dla którego koszt obliczeniowy jest rzędu $M \cdot N/2 \cdot (\log_2 N + \log_2 M)$, co przy założeniu $M = N$ daje liczbę rzędu $N^2 \cdot \log_2 N$. Tabela 4 zestawia szacowany koszt obliczeniowy dla poszczególnych przypadków. Przy oszacowaniu kosztu obliczeniowego dla poszczególnych operacji w aplikacji należy wziąć pod uwagę wiele czynników, między innymi sposób implementacji funkcji przetwarzających, ilość oraz rodzaj operacji w pojedynczym kroku, ilość dodatkowych operacji (takich jak wymnażanie przez okno częstotliwościowe, odpowiedni zapis próbek), złożoność podstawowych operacji arytmetycznych (działania na liczbach zmiennoprzecinkowych oraz zespolonych), jak również liczbę przetwarzanych kanałów obrazu. Trudno jest więc określić dokładnie czas wykonywania poszczególnych operacji przez komputer, jednak przytoczone oszacowanie daje rozeznanie o przydatności FFT.

Tabela 4. Szacowany koszt obliczeniowy przeprowadzenia filtracji obrazu w zależności od wybranej metody

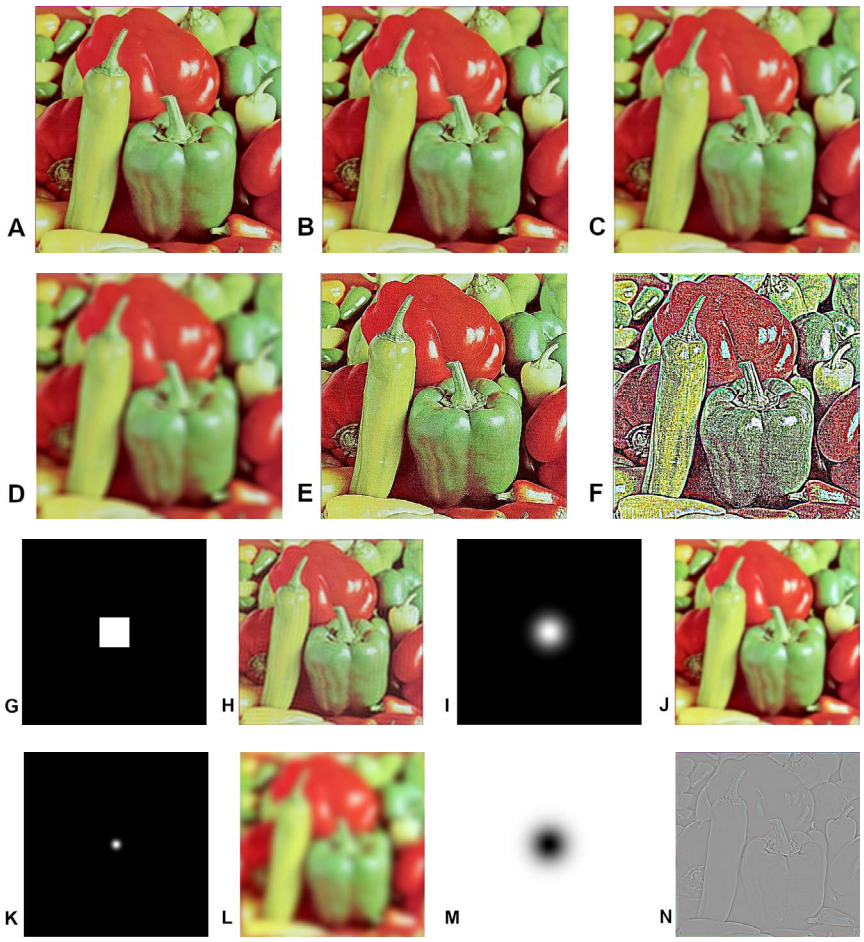
Rodzaj filtracji	Koszt obliczeniowy	Koszt obliczeniowy przy założeniu $N = M$ oraz $U = V$
kontekstowa	$N \cdot M \cdot U \cdot V$	$N^2 \cdot U^2$
DFT dwuwymiarowa z definicji	$N^2 \cdot M^2$	N^4
DFT jako złożenie transformacji jednowymiarowych	$N^2 \cdot M + N \cdot M^2$	N^3
FFT radix-2	$M \cdot N/2 \cdot (\log_2 N + \log_2 M)$	$N_2 \cdot \log_2 N$

Dane wynikające z oszacowania można z łatwością porównywać z rzeczywistym czasem przetwarzania, bowiem w odpowiednich polach w oknie aplikacji po wykonanej operacji wyświetlany jest czas przetwarzania z dokładnością do jednej mikrosekundy. Innym wskaźnikiem pozwalającym ocenić wpływ doboru parametrów na przebieg filtracji jest błąd średniokwadratowy, którego wartość określa, w jakim stopniu obraz wynikowy różni się od źródłowego.

Testy systemu i dyskusja

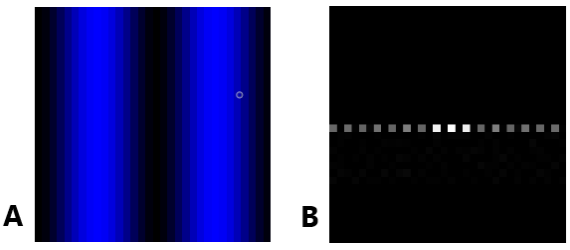
W obecnej formie jednym z największych walorów utworzonej aplikacji jest wartość badawczo-edukacyjna. Aplikacja pozwala śledzić efekty różnego rodzaju filtracji. Z łatwością można ocenić, jaki wpływ ma dobór maski lub okna częstotliwościowego na efekt filtracji. Przykładowe operacje przeprowadzone na obrazie testowym przedstawia rysunek 6.

Istotną zaletą aplikacji jest możliwość śledzenia wyników pośrednich w przetwarzaniu częstotliwościowym. Możliwe jest wyświetlenie amplitudy i fazy F-obrazu jak również okna częstotliwościowego. Analiza próbek F-obrazu, w szczególności dla obrazów o niewielkich wymiarach, pozwala zrozumieć, czym jest reprezentacja częstotliwościowa obrazu. Przykład przedstawia rysunek 7, gdzie przekształceniu został poddany obraz spreparowany według określonych założeń. Wartości każdego wiersza tego obrazu przebiegają w oparciu o funkcję cosinus i w jednym wierszu zawarte są dwa pełne okresy tej funkcji. Każda kolumna zawiera natomiast piksele o stałej wartości. Wszystkie cechy obrazu źródłowego są doskonale widoczne w jego



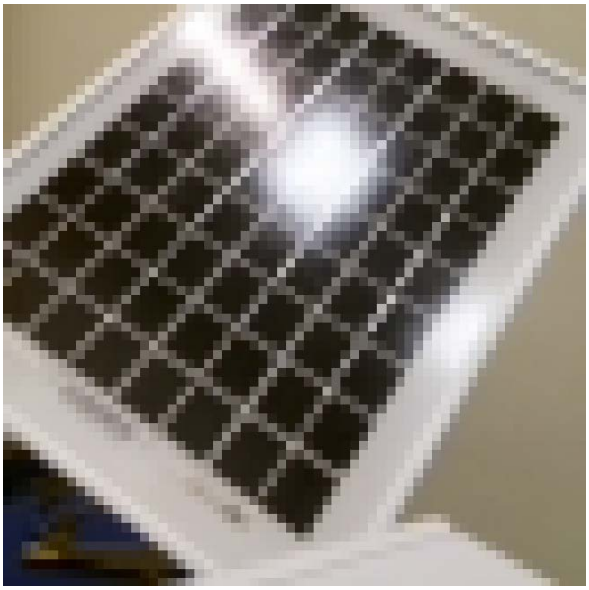
Rysunek 6. Przykładowy obraz o rozdzielczości 512 x 512 px pobrany z bazy USC-SIPI [13] (A), obraz A po filtracji LP maską 3 x 3 ze współczynnikami o wartości 1 (B), maską 7 x 7 (C), maską 15 x 15 (D), po filtracji HP maską 3 x 3 – punkt centralny równy sumie współczynników, pozostałe elementy –1 (E), maską 7 x 7 (F), dolnoprzepustowe okno z kwadratem o boku 83 px (G), wynik filtracji widmowej oknem G (H), dolnoprzepustowe okno Gaussa o parametrach $\sigma_x = \sigma_y = 28$ (I), wynik filtracji oknem I (J), dolnoprzepustowe okno Gaussa o parametrach $\sigma_x = \sigma_y = 8$ (K), wynik filtracji oknem K (L), górnoprzepustowe okno Gaussa o parametrach $\sigma_x = \sigma_y = 35$ (M), wynik filtracji oknem M (N)

widmie amplitudowym, gdzie można dostrzec trzy najjaśniejsze punkty. Piksel środkowy obrazuje składową stałą obrazu, która stanowi średnią wartość wszystkich próbek, natomiast dwa boczne piksele o dużej jasności reprezentują funkcję bazową o częstotliwości 2 wzdłuż wierszy oraz 0 (wartość stała) wzdłuż kolumn. W przedstawionym przykładzie wartość średnia pikseli obrazu źródłowego jest równa 127, co można rozumieć jako dodanie do funkcji cosinus stałej wartości równej amplitudzie sygnału. Jest to konieczne ze względu na fakt, że piksele obrazu przybierają jedynie nieujemne wartości. Wartość częstotliwości, podawana w tym wypadku bez jednostki, oznacza wielokrotność częstotliwości takiej funkcji harmonicznej, której pełen okres zawiera się dokładnie na długości jednego z boków obrazu. Kolejne piksele wzdłuż danego wymiaru w F-obrazie oznaczają kolejne krotności częstotliwości bazowej. Tego typu proste obrazy źródłowe można w łatwy sposób modyfikować, a następnie obserwować zmiany w widmie amplitudowym oraz fazowym.

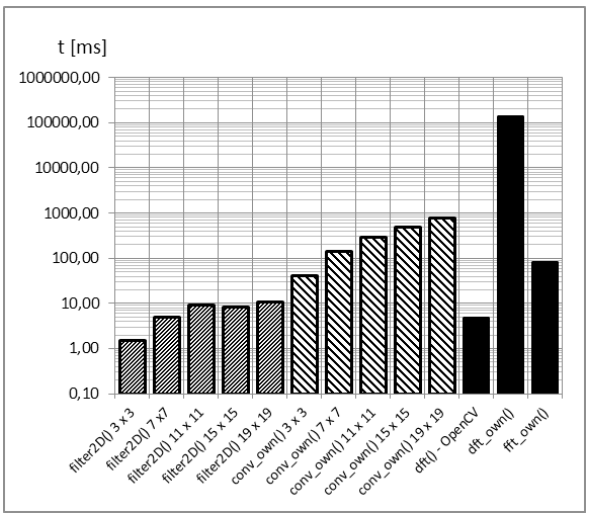


Rysunek 7. Obraz testowy o parametrach: kanały R, G – wartość 0, kanał B – wartości pikseli w wierszach według funkcji cosinus, amplituda – 128 (A) oraz moduł F-obrazu (B)

Kolejne istotne zastosowanie aplikacji w aspekcie badawczym polega na analizie wskaźników charakteryzujących przebieg operacji, takich jak czas przetwarzania oraz błąd średniokwadratowy. Badanie czasu przetwarzania umożliwia nie tylko analizę porównawczą różnych algorytmów przetwarzania wraz z określonym zestawem parametrów, ale również ocenę czasu wykonywania operacji pod kątem zastosowania modułu RPi w konkretnych aplikacjach. W tym wypadku program wykonuje jedynie filtrację określonego typu, jednak bazując na istniejącej strukturze kodu, można go z łatwością dostosować do wykonywania innych zadań.



Rysunek 8. Przykładowa fotografia wykonana za pomocą dołączonej kamery



Rysunek 9. Wykres czasu przetwarzania w zależności od wybranej metody filtracji dla wykonanej fotografii z rysunku 8

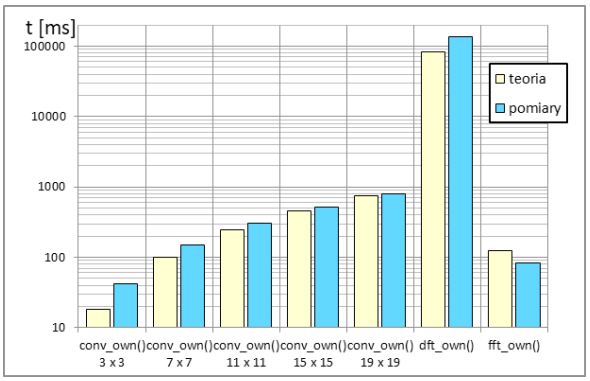
Pomiaru czasu przetwarzania dokonano na przykładowym obrazie wykonanym za pomocą dołączonej kamery (rysunek 8). Rozdzielczość fotografii ustawiono na 64 x 64 piksele, aby z łatwością można było porównać wszystkie metody przetwarzania. Czas przetwarzania w funkcji dobranej metody oraz parametrów maski (dla operacji kontekstowych) przedstawia rysunek 9.

Najmniejszy zarejestrowany czas przetwarzania wyniósł około 1,6 ms. Porównując działanie funkcji napisanych we własnym zakresie, które bazują wprost na definicjach operacji danego typu, można stwierdzić, że czas obliczeń dla operacji kontekstowych o coraz większych maskach wydłuża się. Ponadto można zauważyć, że już przy zastosowaniu maski 7 x 7 ten typ operacji jest bardziej czasochłonny niż przetwarzanie częstotliwościowe z wykorzystaniem funkcji *fft_own()*, utworzonej w oparciu o algorytm szybkiej transformacji FFT radix-2. Jeśli więc operacja kontekstowa wymagałaby użycia dużej maski dla uzyskania żądanego rezultatu, na przykład w celu otrzymania znacznego stopnia rozmycia we filtracji dolnoprzepustowej, to lepiej jest przeprowadzić filtrację widmową z odpowiednio dobranym oknem częstotliwościowym. W przypadku filtracji przeprowadzonej z wykorzystaniem funkcji *filter2D()*, czyli funkcji dostępnej w bibliotece OpenCV, nie zauważa się przyrostu czasu przetwarzania dla masek o wymiarze równym co najmniej 11 x 11. Wspomniano już wcześniej, że ta funkcja operację kontekstową realizuje jedynie dla niewielkich masek, w tym wypadku maski 3 x 3 oraz 7 x 7. Dla pozostałych przypadków funkcja co prawda przyjmuje wielkość oraz współczynniki maski, jednak przetwarzanie następuje w dziedzinie częstotliwości [10]. Ten typ przetwarzania zapewnia taki sam czas wykonywania operacji niezależnie od głębokości filtracji, co widoczne jest na wykresie. Nieznaczne różnice w czasie przetwarzania funkcją *filter2D()* dla większych masek mogą mieć wiele przyczyn, poczynając od algorytmu przeliczającego podaną maskę na odpowiednie okno częstotliwościowe, po bieżące zapotrzebowanie na zasoby procesora powodowane działaniem systemu oraz programów.

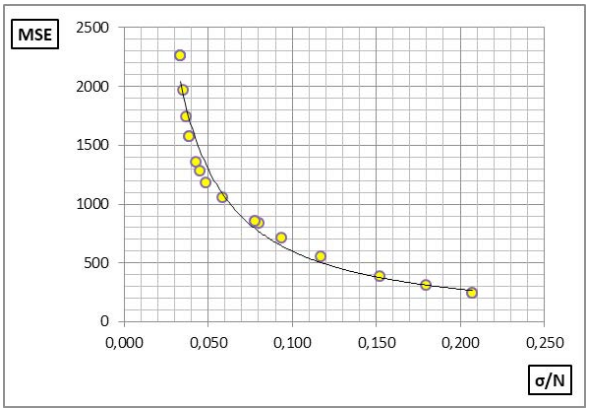
Na rysunku 10 zestawiono uzyskane wyniki pomiarów z teoretycznymi szacunkami kosztu obliczeniowego przedstawionymi we wcześniejszej części artykułu.

W przedstawionej analizie wzięto pod uwagę wymienione wcześniej czynniki mające wpływ na szybkość wykonywania operacji. Szacunkowe wartości wynikające z rozważań teoretycznych zostały wymnożone przez pewną stałą wartość w celu konwersji szacowanej liczby działań na czas ich wykonywania. Teoretyczne przeliczenie liczby operacji na czas wykonywania w przypadku sprzętu działającego pod kontrolą systemu operacyjnego byłoby zadaniem bardzo trudnym, jednak nie ma takiej konieczności i do porównania wystarczy

odpowiednie dobranie stałej w taki sposób, aby uzyskać jak najlepsze dopasowanie szacunków do wyników pomiarów. Istotne jest natomiast zachowanie proporcji pomiędzy szacowanymi wartościami. Z przedstawionego zestawienia jasno wynika, że szacunkowe rozważania znajdują pokrycie w rzeczywistości, ponieważ zmierzone czasy są podobne to tych wynikających z analizy teoretycznej.



Rysunek 10. Wykres zestawiający wyniki pomiaru czasu przetwarzania z wartościami wynikającymi z analizy teoretycznej dla wykonanej fotografii z rysunku 8



Rysunek 11. Wykres MSE = f(σ/N) dla filtracji widmowej fotografii przedstawionej na rysunku 8 z wykorzystaniem okna Gaussa

Badanie błędu średniokwadratowego pozwala określić, w jakim stopniu wynikowy obraz różni się od źródłowego. W aspekcie dydaktycznym pozwala to na obserwację, w jaki sposób parametry filtracji wpływają na zniekształcenie obrazu wejściowego. Użytkownik może obserwować na przykład wzrost wartości błędu średniokwadratowego wraz ze zwiększaniem maski dla dolnoprzepustowej filtracji kontekstowej. Błąd średniokwadratowy wyznaczany dla obrazu wyjściowego można również zestawić z wielkością oraz typem zastosowanego okna częstotliwościowego we filtracji widmowej. Rysunek 11 przedstawia wartość MSE w funkcji odchylenia standardowego, znormalizowanego względem rozmiaru obrazu, dla przekształcenia widmowego fotografii przedstawionej wcześniej na rysunku 8.

Im większa jest wartość odchylenia standardowego, tym większe okno częstotliwościowe, a co za tym idzie więcej częstotliwości jest przepuszczanych przez filtr. Wtedy obraz źródłowy jest w mniejszym stopniu zniekształcany, co objawia się mniejszą wartością MSE.

Wartość błędu średniokwadratowego może posłużyć również jako wskaźnik głębokości filtracji dolnoprzepustowej. Jeśli odpowiednio dobierze się parametry przekształcenia kontekstowego oraz widmowego, to podobna wartość błędu średniokwadratowego przy obydwu typach przekształcenia może świadczyć o uzyskaniu podobnego rezultatu – podobnej głębokości filtracji, redukcji szumów, rozmycia.

Zastosowania oraz możliwości rozwoju aplikacji

Przedstawiona aplikacja jest użyteczna głównie w wymiarze dydaktycznym oraz badawczym. Po pierwsze aplikacja umożliwia zapoznanie się z przekształceniami obrazu takimi jak filtracja kontekstowa oraz widmowa. Po drugie umożliwia generowanie F-obrazów, dzięki czemu można obserwować reprezentację częstotliwościową obrazu oraz badać wpływ zmian wprowadzanych na obrazie w dziedzinie pierwotnej na rozkład widmowy. Dodatkowe narzędzia umożliwiają również ocenę przydatności platformy Raspberry Pi jako systemu wizyjnego do określonych rozwiązań. Częstotliwościowa reprezentacja obrazu może być bardzo użyteczna, gdyż może ukazywać pewne jego specyficzne cechy niedostrzegalne w dziedzinie pierwotnej. W tym aspekcie utworzona aplikacja może być użyteczna na przykład do analizy tekstur (lub ogólniej struktury materiałów), które mają okresowy charakter. Wykorzystując analizę częstotliwościową można wychwycić różne nieprawidłowości. Sama operacja filtracji również może być wykorzystana w konkretnych rozwiązaniach, gdyż pozwala na uzyskanie efektu rozmycia, usunięcie szumów z obrazu, wyostrzenie obrazu, wyodrębnienie krawędzi [2, 6, 9]. Często operacja filtracji może stanowić pewien etap w przygotowaniu obrazu do dalszej obróbki.

Poza wdrożonymi rozwiązaniami do aplikacji można wprowadzić nowe funkcjonalności. Kod programu został zbudowany tak, aby w łatwy sposób można było dodawać nowe funkcje przetwarzające. Utworzona aplikacja stanowi zalążek do stworzenia systemu wizyjnego przeznaczonego do konkretnych specjalizowanych

zastosowań. Przykładowym rozwiązaniem, które aktualnie jest wdrażane, jest wykorzystanie systemu wizyjnego do obserwacji roślin w zautomatyzowanej farmie agrofotowoltaicznej. Zaprojektowany system mógłby wykonywać fotografię liści lub innych części roślin i na podstawie analizy obrazu dostarczać do centralnego systemu informacje dotyczące poprawności rozwoju rośliny, w tym te dotyczące chorób, które można zidentyfikować wizualnie. Dzięki możliwości komunikacji bezprzewodowej z zastosowanym modułem całe urządzenie będzie mobilne i z łatwością będzie można je umieścić w dowolnym miejscu, niezbędne będzie jedynie zasilenie urządzenia. Dostępne porty wejścia-wyjścia umożliwiają ponadto bezpośrednie sterowanie wybranymi urządzeniami wykonawczymi. Główną drogę rozwoju aplikacji i całego systemu będzie stanowiło badanie użyteczności oraz przystosowanie do wybranych rozwiązań.

Wnioski

Niniejsza praca przedstawia kompletny, działający system wizyjny, który może posłużyć do różnych zastosowań. Istotnym efektem pracy jest zaimplementowanie w utworzonym systemie operacji przetwarzania widmowego obrazów z wykorzystaniem własnej funkcji realizującej algorytm dwuwymiarowej szybkiej transformacji Fouriera radix-2. W napisanej aplikacji okienkowej zastosowano wiele autorskich rozwiązań. W aktualnej formie system ma znaczącą wartość edukacyjną dla studentów zdobywających informacje dotyczące przetwarzania obrazów oraz systemów wizyjnych. Stanowi również narzędzie badawcze pozwalające analizować obrazy w dziedzinie częstotliwości. Wykorzystując utworzony system zbadano wiele przykładowych obrazów testowych oraz fotografii, obserwując i interpretując ich charakterystyki widmowe. Przedstawione wyniki pomiarów czasu przetwarzania wykonanych z użyciem aplikacji potwierdzają, że utworzony system z powodzeniem może być wykorzystany do różnych rozwiązań, w tym analizy obrazu z kamery oraz sterowania wybranym procesem.

Bibliografia

[1] Sankowski D, Mosorov V, Strzecha K. Przetwarzanie i analiza obrazów w systemach przemysłowych. Warszawa: PWN; 2011.

[2] Demant Ch, Streicher-Abel B, Waszkewitz P. Industrial image processing. Visual quality control in manufacturing. Berlin: Springer; 1999. doi: https://doi.org/10.1007/978-3-642-58550-0.

[3] Kompleksowy zestaw systemu wizyjnego, seria XG/CV-X. Keyence catalog.

[4] Raspberry Pi Foundation [Internet, cytowane 2 lutego 2021]. Dostępne na: www.raspberrypi.org.

[5] Malina W, Smiatacz M. Cyfrowe przetwarzanie obrazów. Warszawa: Exit; 2008.

[6] Tadeusiewicz R, Korohoda P. Komputerowa analiza i przetwarzanie obrazów. Kraków: WFPT; 1997.

[7] Owen M. Przetwarzanie sygnałów w praktyce. Warszawa: WKŁ; 2009.

[8] Jahne B. Digital image processing. Berlin: Springer; 2005. doi: https://doi.org/10.1007/3-540-27563-0.

[9] Sundararajan D. Digital image processing. A signal processing and algorithmic approach. Singapur: Springer; 2017. doi: https://doi.org/10.1007/978-981-10-6113-4.

[10] Bradski G, Kaehler A. Learning OpenCV. Computer vision with the OpenCV library. Farnham, Cambridge: O'Reilly; 2008.

[11] Brahmbhatt S. Practical OpenCV. Technology in action. Berkeley: Apress; 2013. doi: https://doi.org/10.1007/978-1-4302-6080-6.

[12] Zieliński T. Cyfrowe przetwarzanie sygnałów od teorii do zastosowań. Warszawa: WKŁ; 2005.

[13] The USC-SIPI Image Database. USC University of Southern California, USC Viterbi School of Engineering [Internet, cytowane 10 kwietnia 2021]. Dostępne na: http://sipi.usc.edu/database/.