



AKADEMIA GÓRNICZO-HUTNICZA

im. St. Staszica w Krakowie

WEAliB, Katedra Automatyki

Laboratorium Biocybernetyki

Przedmiot:		Wieloprocessorowe Systemy Wizyjne			
				pr16wsw503	
Temat projektu:					
		Detekcja kształtu oraz orientacji obiektów za pomocą momentów geometrycznych w czasie rzeczywistym			
Wykonali:	Piotr Pałucki Filip Kubicz				
				Automatyka i Robotyka	
Rok akademicki		Semestr zimowy			
Prowadzący		dr inż. Mirosław Jabłoński			
wersja 2.7 Kraków, styczeń 2017					

Spis treści:

1.	ABSTRAKT.....	3
2.	WSTĘP.....	3
3.	ANALIZA ZŁOŻONOŚCI I ESTYMACJA ZAPOTRZEBOWANIA NA ZASOBY.....	5
4.	KONCEPCJA PROPONOWANEGO ROZWIĄZANIA.....	5
5.	SYMULACJA I TESTOWANIE.....	6
	Modelowanie i symulacja.....	6
	Testowanie o weryfikacja.....	6
6.	REZULTATY I WNIOSKI.....	6
7.	PODSUMOWANIE.....	7
8.	LITERATURA.....	7
9.	DODATEK A: SZCZEGÓŁOWY OPIS ZADANIA.....	8
	Specyfikacja projektu.....	8
	Szczegółowy opis realizowanych algorytmów przetwarzania danych.....	8
10.	DODATEK B: DOKUMENTACJA TECHNICZNA.....	8
	Dokumentacja oprogramowania.....	8
	Dokumentacja sprzętowa.....	9
	Procedura symulacji i testowania:.....	9
	Procedura uruchomieniowa i weryfikacji sprzętowej:.....	9
11.	DODATEK D: SPIS ZAWARTOŚCI DOŁĄCZONEGO NOŚNIKA (PŁYTA CD ROM).....	9
12.	DODATEK E: HISTORIA ZMIAN.....	10

1. Abstrakt

Celem projektu realizowanego w ramach laboratorium Wieloprosesorowych Systemów Wizyjnych była automatyczna detekcja orientacji i kształtu nasion. Do analizy zostały wykorzystane momenty geometryczne i momenty niezmiennicze Hu.

Podczas projektu przystosowano minikomputer ODROID-XU4 do akwizycji obrazu z kamery szybkostrzelnej, wykonywania obliczeń wizyjnych i ich wizualizacji. Następnie w celu przyspieszenia systemu zaproponowano sposób równoległego prowadzenia obliczeń momentów z użyciem języka OpenCL. Do osiągnięć należy zaliczyć uruchomienie na minikomputerze ODROID systemu Debian wraz z biblioteką OpenCV i sterownikami umożliwiającymi współpracę z kamerą szybkostrzelną Basler.

Słowa kluczowe: Moment Hu, Akceleracja GPU, OpenCL, ODROID, system wizyjny czasu rzeczywistego

2. Wstęp

a) Cele i założenia projektu

Celem pracy jest zbudowanie fragmentu toru wizyjnego, wykorzystywanego w systemie segregacji nasion, który umożliwiłby detekcję ich kształtu i orientacji. Odpowiednie ułożenie obiektu jest niezbędne, aby na następnym etapie dokonać jego nacięcia z właściwej strony. Wizyjna analiza tak przetworzonego nasiona umożliwi jego późniejszą klasyfikację jako nadające się – lub nie – do późniejszego sadzenia.

b) Ogólny zarys proponowanego rozwiązania

Do obserwacji ruchu obiektów została użyta kamera szybkostrzelna Basler. Obraz z kamery zostaje przesłany na miniatury komputer ODROID-XU4, gdzie obraz jest binaryzowany w oparciu o wynik wykrywania krawędzi metodą Canny'ego. Dla znalezionych metodą śledzenia konturów obiektów są obliczane momenty geometryczne oraz momenty Hu.

Moment geometryczny rzędu $(p+q)$ obrazu $I(x,y)$ w odcieniach szarości można obliczyć jako

$$M_{pq} = \sum_x \sum_y x^p y^q I(x,y) \quad (1)$$

Na ich podstawie można konstruować niezmienniki, czyli parametry które pozostają stałe względem wybranej transformacji obrazu [6]. Niezmiennikiem względem przesunięcia jest moment centralny

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x, y) \quad (2)$$

gdzie współrzędne środka ciężkości:

$$\bar{x} = \frac{M_{10}}{M_{00}}, \quad \bar{y} = \frac{M_{01}}{M_{00}} \quad (3)$$

Niezmienniki względem skalowania obrazu to tzw. znormalizowane momenty centralne

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{1 + \frac{p+q}{2}}} \quad (4)$$

Momenty niezmiennicze Hu [2] pozwalają na opisanie kształtu nasiona w sposób odporny na przesunięcie, obrót i zmianę skali.

$$\begin{aligned} I_1 &= \eta_{20} + \eta_{02} \\ I_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\ I_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\ I_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\ I_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ &\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\ I_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\ I_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ &\quad - (\eta_{30} - 3\eta_{12})(\eta_{12} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \quad (5)$$

Dzięki obliczonym w ten sposób parametrom można określić kształt obiektu i odróżnić go od innego kształtu. Do określenia orientacji można wykorzystać momenty centralne, budując z nich macierz kowariancji

$$\text{cov}[I(x, y)] = \frac{1}{\mu_{00}} \begin{bmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{bmatrix} \quad (6)$$

A następnie wyznaczając jej wartości własne i wektory własne.

c) Alternatywne rozwiązania

Alternatywna metoda obliczania **momentów niezmienniczych** wyższego rzędu została zaproponowana przez J. Flussera [7]. Czeski badacz opracował ogólny algorytm obliczania niezmienników oparty na momentach zespolonych. Stanowi on uogólnienie wzorów podanych przez M. K. Hu.

Alternatywną metodą określania **orientacji** jest PCA, czyli analiza głównych składowych. W tej metodzie należy wyznaczyć transformatę Karhunen–Loèvego znalezionej obiektu. W praktyce, dla obrazu dwuwymiarowego sprowadza się to do wyznaczenia momentów centralnych i badania wariancji kształtu obiektu w prostopadłych do siebie osiach, zatem jest to metoda podobna do zaproponowanej w [6].

3. Analiza złożoności i estymacja zapotrzebowania na zasoby

Przetwarzane dane to filmy w formacie .avi złożone z obrazów w odcieniach szarości o rozmiarze ramki 640x480px, z prędkością 100 klatek na sekundę i głębią kolorów 8-bit. Używana kamera Basler acA2000-165um pozwala na akwizycję obrazu w odcieniach szarości z maksymalną prędkością 165 fps, z głębią kolorów 8-bit, 10-bit lub 12-bit.

Dostępne zasoby obliczeniowe to obecne w komputerze ODROID-XU4 procesor Samsung Exynos 5422 z czterema rdzeniami Cortex-A15 2Ghz oraz czterema rdzeniami Cortex-A7, a także kartą graficzną Mali-T628 MP6, która obsługuje OpenCL 1.1.

Komputer dysponuje pamięcią RAM DDR3 o wielkości 2GB. Aby zapewnić odpowiednią ilość miejsca pod system operacyjny Linux, potrzebne biblioteki i przechowywanie danych, w projekcie użyto karty pamięci MicroSD-HC o rozmiarze pamięci 8GB.

4. Koncepcja proponowanego rozwiązania

a) Ogólny zarys rozwiązania z użyciem biblioteki OpenCV

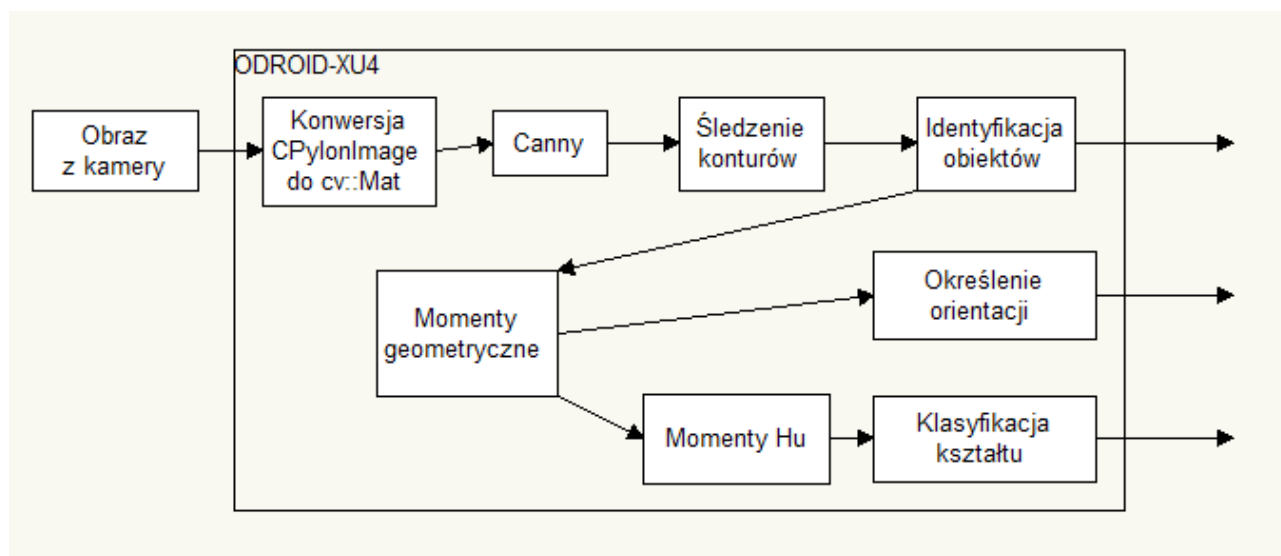
Aby uzyskać założone cele, użyto biblioteki algorytmów wizyjnych *OpenCV* z interfejsem w języku C++. Do akwizycji obrazu zostały wykorzystane funkcje udostępnione przez moduł *Pylon 5*, współpracujący z użytą kamerą Basler. Użycie tych komponentów było możliwe, ponieważ na minikomputerze ODROID zainstalowano system *Linux - Debian Wheezy* z systemem graficznym *LXDE*, który zapewnia prawidłowe działanie sterowników kamery, możliwość zainstalowania

biblioteki *OpenCV* i wyświetlania okien podglądu, co znacznie ułatwia testowanie algorytmów. Instrukcja instalacji systemu *Debian* zgodnego z komputerem ODROID na karcie microSD zostanie przedstawiona w **dodatku B**.

Wybrano **rozdzielczość obrazu** 640x480px (czyli VGA lub 0,3Mpx), ponieważ taki rozmiar wystarcza do dokładnej analizy kształtu i orientacji, a większy (dostępny nawet 2Mpx) może wyraźnie spowolnić przesył danych z kamery do minikomputera. **Głębina koloru**, lub rozdzielczość koloru została ustalona na 10-bit, ponieważ na etapie akwizycji danych z kamery wartości pikseli zostają skonwertowane na głębię 8-bitową przez moduł Pylon. **Szybkość ramek** podlega z kolei optymalizacji i oczekujemy osiągnięcia rezultatu powyżej 100 ramek na sekundę.

Po zapisaniu obrazu z kamery w formie macierzy modułu Pylon (*CPylonImage*) zostaje ona przepisana do macierzy OpenCV (*cv::Mat* - gdzie *cv::* oznacza przestrzeń nazw klas i funkcji C++ biblioteki OpenCV). Na obrazie w tej formie mogą działać wszystkie algorytmy biblioteki OpenCV. W pierwszej kolejności algorytmem Canny'ego (*cv::Canny*) zostają wykryte krawędzie obecne na obrazie. Następnie należy wyodrębnić obiekty, podążając za ich konturami (*cv::findContours*) i odrzucając te obiekty, dla których długość konturu jest zbyt mała.

Kiedy zostały już znalezione rozważane obiekty, algorytm oblicza dla każdego zestaw momentów geometrycznych, centralnych i znormalizowanych (*cv::moments*). Na ich podstawie obliczone zostają momenty niezmiennicze Hu (*cv::HuMoments*), które są podstawą do klasyfikacji kształtu.



Rys. 1: Algorytm rozpoznawania kształtu i orientacji z wykorzystaniem momentów geometrycznych

Obliczanie momentów geometrycznych zostało wybrane jako część algorytmu, która może zostać **zrównoleglona**, a obliczenia przeprowadzone na karcie GPU z użyciem języka OpenCL. Przygotowanie własnego sposobu obliczania momentów geometrycznych i niezmienników Hu jest

również korzystne, ponieważ funkcja *cv::moments* dla każdego obiektu oblicza wszystkie momenty aż do trzeciego rzędu, z których nie wszystkie są potrzebne do późniejszej analizy.

b) Akceleracja obliczania momentów geometrycznych z użyciem środowiska OpenCL

Algorytm do obliczania momentów geometrycznych obrazu w środowisku OpenCL opracowany został na podstawie pracy [3]. Autorzy zaproponowali tam niezależne (równoległe) obliczanie najpierw pionowych a później poziomych składowych momentów obrazu. Rozbicie na etapy i obliczanie równoległe jest możliwe, gdyż składowe pionowe momentów nie zależą od wyników innych obliczeń a jedynie od pozycji i wartości piksela.

Dopiero do obliczenia ostatecznej wartości momentu geometrycznego niezbędne są wyniki sum częściowych momentów uzyskane w poprzednim etapie.

Algorytm ostatecznie wykorzystany w projekcie jest nieco zmodyfikowaną wersją tego przedstawionego w [3]. Modyfikacje dotyczą głównie dopasowania go do użycia w OpenCL. Etapy algorytmu:

1) Wyliczenie składowych pionowych momentów geometrycznych w wierszach.

Składowe obliczane powinny być zgodnie z definicją momentów, tj. z wykorzystaniem informacji o pozycji piksela podniesionej do potęgi wyznaczonej przez rząd momentu.

$$MY_{pq}(x) = \sum_y y^q I(x,y) \quad (7)$$

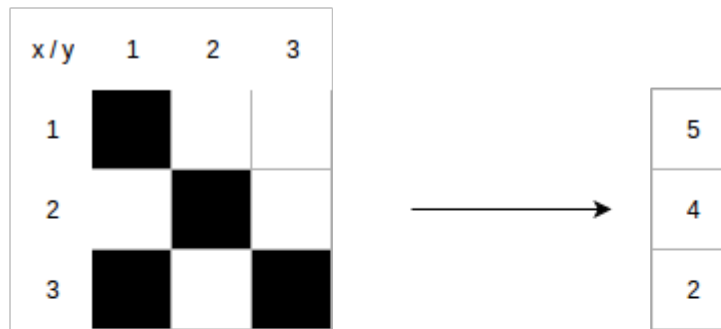
2) Wyliczenie momentów geometrycznych na podstawie 1)

Dla każdego obliczonego wiersza obrazu należy pomnożyć sumę otrzymaną w 1) przez numer wiersza zgodnie z definicją momentu.

$$M_{pq} = \sum_x x^p MY_{pq}(x) \quad (8)$$

Podobne podejście wykorzystano na następnym etapie przy obliczaniu momentów centralnych..

Przykład opisanej operacji dla momentu rzędu 1+1 przedstawiono poniżej:



$$\begin{aligned} MY_{11}(1) &= 1 * 0 + 2 * 1 + 3 * 1 = 5 \\ MY_{11}(2) &= 1 * 1 + 2 * 0 + 3 * 1 = 4 \\ MY_{11}(3) &= 1 * 0 + 2 * 1 + 3 * 0 = 2 \end{aligned}$$

$$M_{11} = 1 * 5 + 2 * 4 + 3 * 2 = 19$$

Rys. 2: Przykład algorytmu zrównoleglającego obliczenia

Składowe pionowe są obliczane przy wykorzystaniu kerneli, z których każdy zajmuje się przetwarzaniem 8 kolejnych pikseli z wiersza. Kernele te pogrupowane są w *work-grupy*, z których każda zajmuje się przetwarzaniem jednego wiersza obrazu. Uzyskujemy dzięki temu przyspieszenie w dwóch wymiarach:

- zrównoleglenie wykonywania kerneli
- 1 dostęp do pamięci globalnej zamiast 8 dla każdego z 8 pikseli (zapewnia nam to użycie wektorowego typu **uchar8** (unsigned char na 8 bitach) w OpenCL)

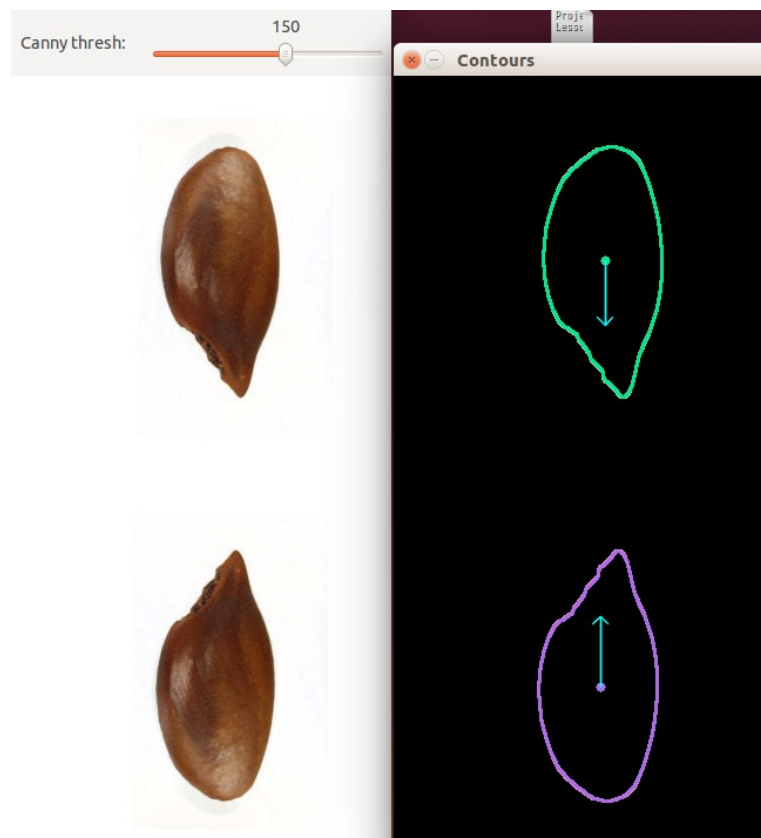
Takie podejście automatycznie nakłada ograniczenie na wielkość obrazu – jego szerokość w pikselach musi być liczbą podzielną przez 8.

5. Symulacja i Testowanie

Modelowanie i symulacja





Opisany w rozdziale 4 algorytm detekcji kształtu i orientacji został wykonany w postaci modelu programowego z użyciem algorytmów zaimplementowanych w bibliotece OpenCV. Jako wektory testowe przyjęto obrazy w formacie **jpg** o rozmiarze 640x480, przedstawiające nasiona różnych gatunków i w różnej orientacji. Wyniki otrzymane z użyciem modelu programowego będą służyć za punkt odniesienia dla implementacji na karcie graficznej Mali z akceleracją OpenCL.

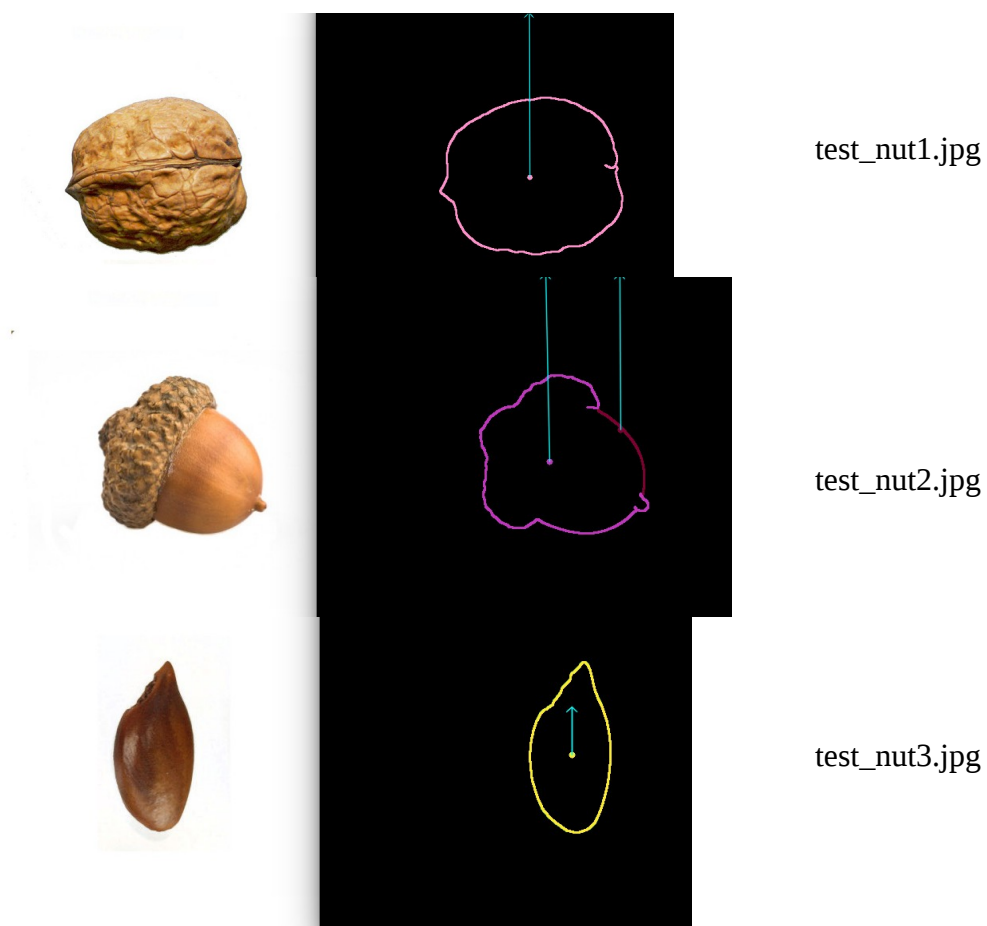
W tabeli 1 zgromadzono obliczone dla trzech wektorów testowych momenty Hu. Z kolei na rysunku 3 przedstawiono, jak siódmy moment Hu może posłużyć do odróżnienia obiektów stanowiących swoje odbicie lustrzane. Strzałka symbolizuje na rysunku znak i wartość momentu I_7 . Na rysunku 4 dla jednego z obrazów testowych zostały wykryte 2 osobne kontury, które nie zostały połączone, stąd 2 strzałki reprezentujące siódmy moment Hu każdej części konturu.



Rys. 3: Przykład z zaznaczonym siódmym momentem Hu

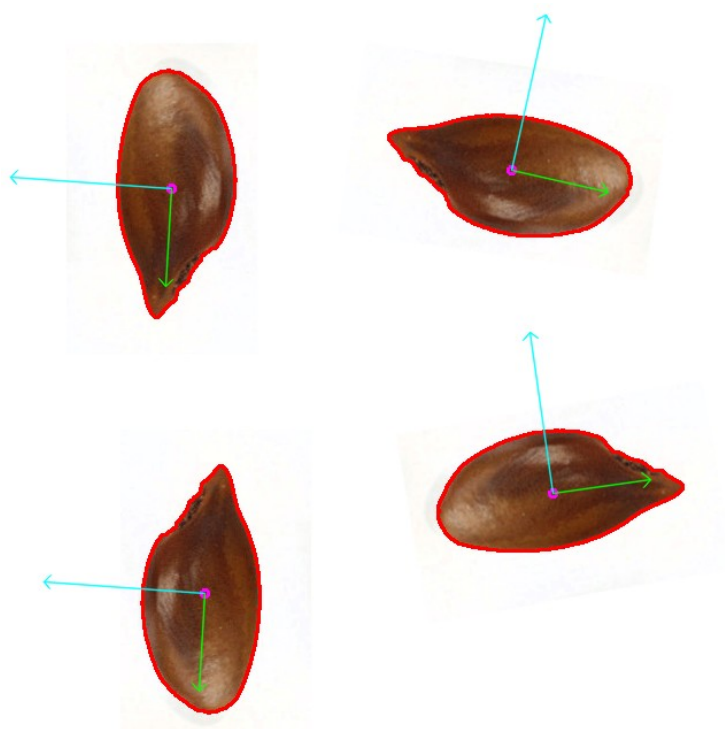
Tabela 1. Momenty Hu oraz długości konturu głównego rozpoznanego obiektu
dla wektorów testowych

Momenty Hu	 test_nut1.jpg	 test_nut2.jpg	 test_nut3.jpg	 test_nut3 odbicie
I_1	8.3017e+01	7.4276e+01	2.0372e-01	2.0372e-01
I_2	4.6719e+02	4.1967e+02	1.5154e-02	1.5154e-02
I_3	7.8750e+03	8.6549e+04	3.2166e-04	3.2166e-04
I_4	1.5661e+03	1.6162e+04	5.5621e-05	5.5621e-05
I_5	-1.6756e+06	5.4097e+08	6.8346e-09	6.8346e-09
I_6	3.0876e+04	1.5655e+05	4.6264e-06	4.6264e-06
I_7	5.2386e+06	2.6964e+08	2.9390e-09	-2.9390e-09
Długość konturu	1759.31	1218.60	530.76	530.76



Rys. 4: Obrazy testowe wraz z rozpoznanymi konturami

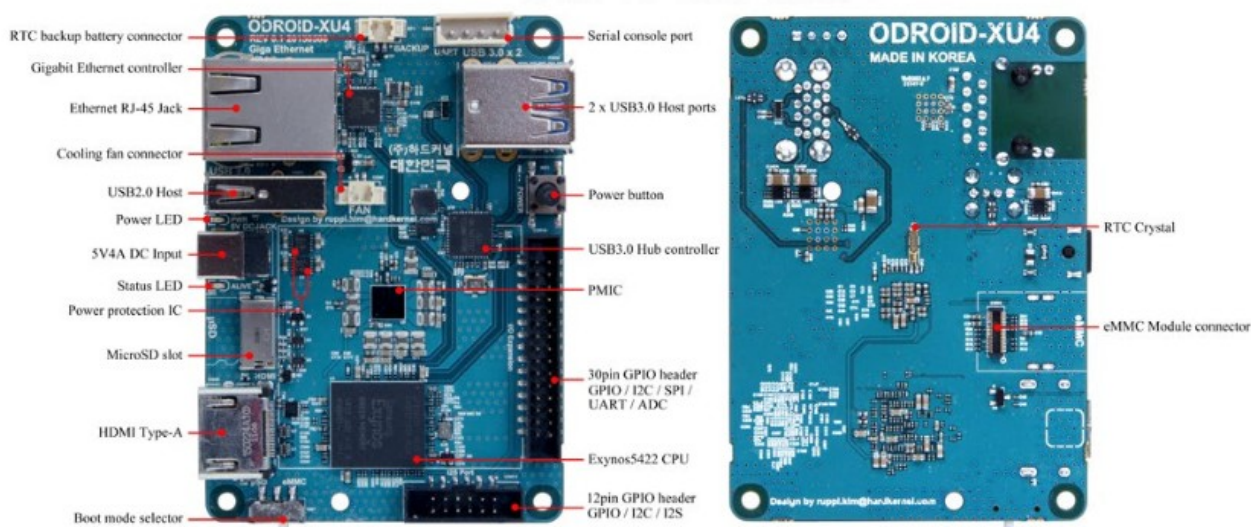
Na rysunku 5 przedstawiono udaną próbę rozpoznania orientacji obiektów na płaszczyźnie z użyciem algorytmu PCA. Planuje się zintegrowanie tego algorytmu z algorytmem detekcji kształtu i wykorzystanie obliczanych na potrzeby kształtu momentów w celu minimalizacji ilości obliczeń.



Rys. 5: Orientacja nasion rozpoznana z użyciem PCA – analizy głównych składowych

Testowanie i weryfikacja

Poniżej przedstawiono budowę docelowej platformy wraz z opisem peryferiów.



Rys. 6: Budowa minikomputera Odroid-XU4

Uruchomienie platformy ODROID i jej przystosowanie do akwizycji i przetwarzania obrazu wymagało wykonania pewnych czynności, które opisano poniżej:

1. Flashowanie obrazu

Odroid umożliwia bootowanie z kart pamięci microSD lub eMMC. Umieszczenie systemu operacyjnego na nośniku bootowalnym nazywany jest **flashowaniem**[1]. Dla potrzeb aplikacji wybrano minimalną instalację systemu **Debian Wheezy**.

2. Uruchomienie i wstępna konfiguracja systemu

3. Konfiguracja interfejsów sieciowych

Aby umożliwić sobie wygodną pracę i dostęp do repozytoriów z aplikacjami należy skonfigurować interfejs sieciowy.

4. Instalacja środowiska graficznego

Środowisko graficzne jest niezbędne do przetestowania działania całego systemu akwizycji i przetwarzania obrazu. Dla potrzeb aplikacji wybrano środowisko graficzne **LXDE**.

Po szczegóły i komendy związane ze wspomnianymi tutaj krokami odsyłamy do Dodatku B.

6. Rezultaty i wnioski

W ramach projektu uruchomiono tor wizyjny na platformie ODROID współpracujący z kamerą Basler a także osobną aplikację wyliczającą momenty obrazu z użyciem akceleracji w OpenCL na podstawie obrazu statycznego. Aplikacja pobierająca obraz z kamery do wyliczania momentów wykorzystuje kontury obiektów znalezione funkcją *findContours*. Dzięki operacji tylko na zbiorze punktów stanowiących krawędź obiektu funkcja ta cechuje się dużą szybkością działania. Aplikacja obliczająca momenty na podstawie obrazu statycznego wykorzystuje pełną informację o obrazie, zostaje jednak przyspieszona dzięki wyodrębnieniu i procesowaniu dalej tylko ROI (Region of Interest), czyli fragmentu obrazu niosącego interesujące dla celów analizy informacje (tutaj – zawierającą obiekt) . Poniżej przedstawiono wyniki eksperymentów związanych z obliczaniem momentów oraz określaniem orientacji z użyciem PCA.

6.1 Akceleracja obliczeń momentów

Niestety wybrana minimalna instalacja Debiana nie zawierała odpowiednich sterowników karty graficznej, które dostarczałyby odpowiednich bibliotek współdzielonych (**libOpenCL.so**) i w trakcie zajęć nie udało się ich zainstalować toteż wszystkie testy w OpenCL zostały przeprowadzone na komputerze osobistym z akceleracją wyłącznie z użyciem CPU.

6.1.1 Struktura testów

W celu umożliwienia powtórzenia eksperymentów wszystkie istotne parametry podano w poniższej tabeli. Parametrami umieszczonymi w zestawieniu są: nazwa obrazu, rozdzielczość obrazu, próg binaryzacji ręcznie wybrany do obliczenia momentów, czas obliczeń momentów w wersji OpenCV (sekwencyjnej) oraz OpenCL (docelowo równoległej), wartości momentów scentralizowanych(w celu porównania poprawności implementacji obiema metodami) oraz siódmy momentu H_u rozróżniający obraz i jego lustrzane odbicie.

Testy przeprowadzono w 3 różnych rozdzielczościach:

- 640 x 480 – rozdzielczość obrazu otrzymywanego z kamery Basler
- 320 x 240 – rozdzielczość obrazu low defintion
- 1440 x 1080 – rozdzielczość obrazu zbliżona do rozdzielczości FullHD (ale w proporcjach 4:3)

Testowane obrazy pochodzące z zasobów Internetu przedstawiono poniżej:



test1.png



test2.png



test3.png

Wszystkie obrazy są w formacie png i znajdują się w lokalizacji **PC/img/test**. Program umożliwiający wyliczenie momentów dla statycznego obrazu znajduje się w lokalizacji: **PC/moments/moments**. Proces jego budowania opisano w Dodatku B.

6.1.2. Wyniki

Wydajność i poprawność liczenia momentów geometrycznych została przedstawiona w tabeli. Do tabeli dołączono również informacje o czasie, jaki trwało określenie orientacji obiektów, razem z binaryzacją i znalezieniem konturów.

Tabela 2. Wydajność i poprawność obliczeń momentów geometrycznych dla obrazu 1 w różnych rozdzielczościach

Parametr	test1.png	test1HD.png	test1LD.png
Rozdzielczość	640x480	1440x1080	320x240
Próg binaryzacji	150	150	150
Czas OpenCV[ms]	3.56	15.68	1.07
Czas OpenCL[ms]	8.94	76.43	2.56
η_{11} OpenCV	-1.490658e-03	-3.216488e-03	-7.746070e-04
η_{11} OpenCL	-1.490658e-03	-3.216488e-03	-7.746070e-04
η_{20} OpenCV	1.186956e-02	2.665232e-02	5.981495e-03
η_{20} OpenCL	1.186956e-02	2.665232e-02	5.981495e-03
η_{30} OpenCV	-2.686508e-04	-8.792476e-04	-9.940832e-05
η_{30} OpenCL	-2.686504e-04	-8.792467e-04	-9.940886e-05
Czas detekcji orientacji PCA [ms]	3.31	8.04	1.55

Tabela 3. Wydajność i poprawność obliczeń momentów geometrycznych dla obrazu 2 w różnych rozdzielczościach

Parametr	test2.png	test2HD.png	test2LD.png
Rozdzielczość	640x480	1440x1080	320x240
Próg binaryzacji	150	N/A	150
Czas OpenCV[ms]	4.06	N/A	1.607
Czas OpenCL[ms]	7.57	N/A	2.97
η_{11} OpenCV	1.312930e-02	N/A	-5.319681e-04
η_{11} OpenCL	1.312930e-02	N/A	-5.319681e-04
η_{20} OpenCV	4.809689e-02	N/A	1.588166e-02
η_{20} OpenCL	4.809689e-02	N/A	1.588166e-02
η_{30} OpenCV	5.173657e-03	N/A	-5.215099e-04
η_{30} OpenCL	5.173655e-03	N/A	-5.215107e-04
Czas detekcji orientacji PCA	3.80596	N/A	1.59926

Dla rozdzielczości HD nie udało się dobrać odpowiedniego progu binaryzacji, by akceptowalnie wykryć kontury obiektu.

Tabela 4. Wydajność i poprawność obliczeń momentów geometrycznych dla obrazu 3 w różnych rozdzielczościach

Parametr	test3.png	test3HD.png	test3LD.png
Rozdzielczość	640x480	1440x1080	320x240
Próg binaryzacji	95	150	116
Czas OpenCV[ms]	3.59	17.63	0.94
Czas OpenCL[ms]	7.33	61.71	3.31
η_{11} OpenCV	-1.887887e-04	1.441995e-02	-1.076393e-04
η_{11} OpenCL	-1.887886e-04	1.441995e-02	-1.076393e-04
η_{20} OpenCV	3.328412e-02	2.863050e-02	1.306965e-02
η_{20} OpenCL	3.328412e-02	2.863050e-02	1.306965e-02
η_{30} OpenCV	-1.873339e-04	-3.535635e-03	-3.789751e-05
η_{30} OpenCL	-1.873357e-04	-3.535637e-03	-3.789773e-05
Czas detekcji orientacji PCA	4.01	8.40	2.17

Poprawność doboru i implementacji metody ($I_7 = I_7(\text{mirror})$):

Parametr	test1	test1_mirror	test2	test2_mirror	test3	test3_mirror
I_7 OpenCV	-2.947137e-09	2.947137e-09	1.290962e+10	2.083523e+07	7.002284e-12	-7.014382e-12
I_7 OpenCL	2.493379e-15	-2.493379e-15	9.082784e-10	-1.024589e-10	-8.756926e-14	8.770097e-14

Dla obrazu drugiego nie uzyskano poprawnych wyników. W połączeniu z poprzednimi uwagami odnośnie tego obrazu można wysnuć wniosek, że obraz nie był wystarczającej jakości.

Otrzymane wyniki mają wartość wyłącznie weryfikacyjną – widać, że momenty są obliczane poprawnie zaś obrana metoda rozpoznawania orientacji obiektów działa. Wyliczanie momentów w sposób zrównoleglony bez dyspozycji GPU jest zgodnie z przypuszczeniami znacznie wolniejsze niż przetwarzanie sekwencyjne. Warto wspomnieć tutaj o różnicy w wynikach dla siódmego momentu H_u – wg. Dokumentacji OpenCV używa innych wzorów (niepoprawnych?) na wyliczanie tej wartości.

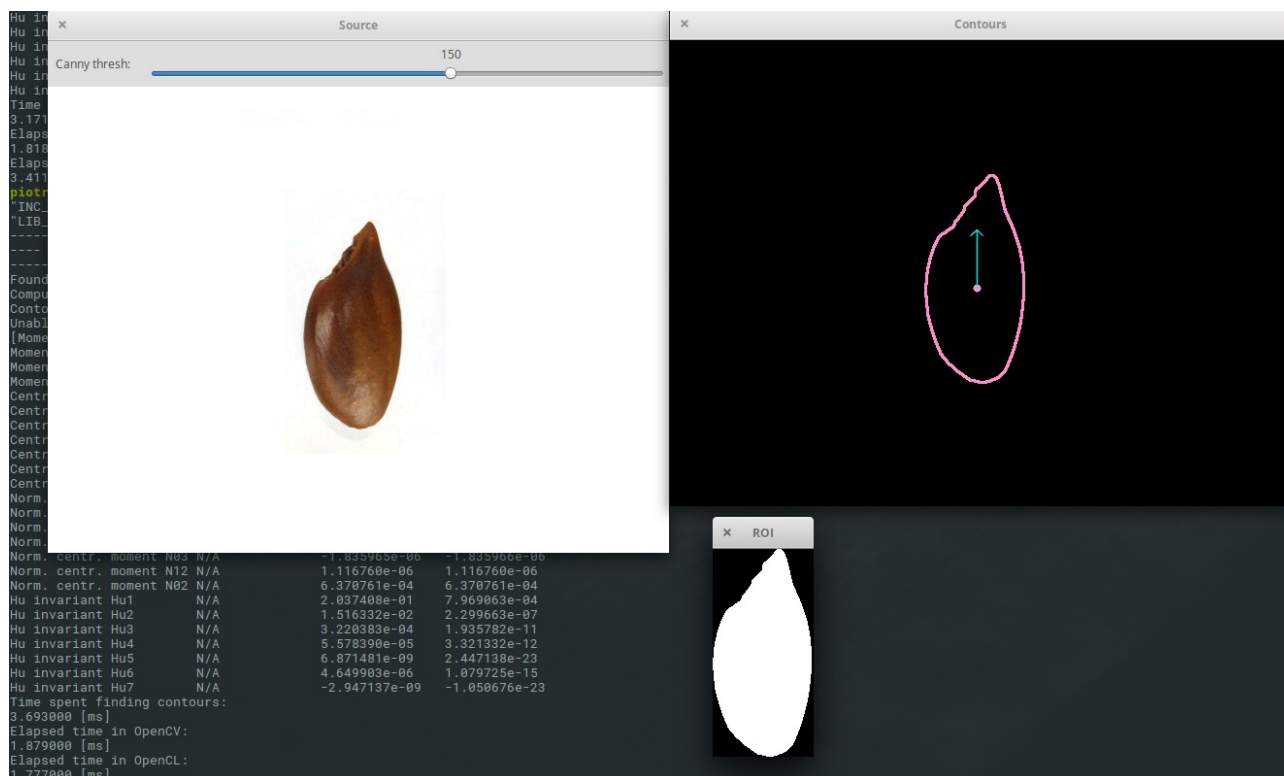
Testy wydajnościowe, mające przynieść informacje dokładniejszą niż uzyskana tutaj (orientacyjna), powinny zostać przeprowadzone przy użyciu toru wizyjnego na platformie ODROID z przetwarzaniem obrazu zrealizowanym w OpenCL. Uruchomienie OpenCL na platformie zgodnie z instrukcjami wspomnianymi w Dodatku B pozostaje zadaniem do zrealizowania na przyszłość.

6.2 Akceleracja obliczeń momentów z wykorzystaniem ROI

Niewystarczająco zadowalające przyspieszenie obliczeń po ich zrównolegleniu skłoniły autorów do podjęcia działań mających na celu optymalizację tej części analizy obrazu. Aby to osiągnąć zdecydowano, aby do dalszej analizy przekazywać jedynie najbardziej interesujący fragment obrazu zawierający obiekt, tzw. Region of interest (ROI). ROI wyliczane jest jako najmniejszy nieobrócony prostokąt, w którym mieści się znaleziony obiekt. Prostokąt jest znajdowany funkcjami wbudowanymi w OpenCV.

Pozwoliło to uzyskać przyspieszenie względem poprzednich obliczeń. Niestety testy wydajnościowe pokazały, że najwolniejszym fragmentem tak zaprojektowanego toru wizyjnego jest funkcja znajdująca kontury obiektu. Aby nieco je przyspieszyć zdecydowano by do dalszych wyliczeń używać tylko konturu zewnętrznego. W tym miejscu nastąpiło też ujednoznacznienie, na jakiej podstawie są obliczane momenty – w przesłanym ROI zewnętrzne kontury obiektu zostają wypełnione i momenty geometryczne są liczone na podstawie całego, wypełnionego kształtu.

Poniżej przedstawiono efekt działania aplikacji z wyekstrahowanym ROI, które jest przesłane na dalszy etap przetwarzania obrazu:



Rys. 7: Działanie aplikacji liczącej momenty przy użyciu ROI

6.2.1 Struktura testów

Dla przetwarzania obrazu znajdującego się w ROI powtórzono pięciokrotnie pierwszy test z poprzedniego punktu uzyskując następujące efekty (czasy to średnia arytmetyczna z wykonań):

Tabela 5. Wydajność i poprawność obliczeń momentów geometrycznych dla obrazu 1 z użyciem ROI

Parametr	test1.png
Rozdzielczość	640x480
Próg binaryzacji	150
Znajdywania konturów [ms]	3.07
Czas OpenCV[ms]	1.57
Czas OpenCL[ms]	2.70
η_{11} OpenCV	-2.346642e-05
η_{11} OpenCL	-2.346642e-05
η_{20} OpenCV	1.598302e-04
η_{20} OpenCL	1.598302e-04

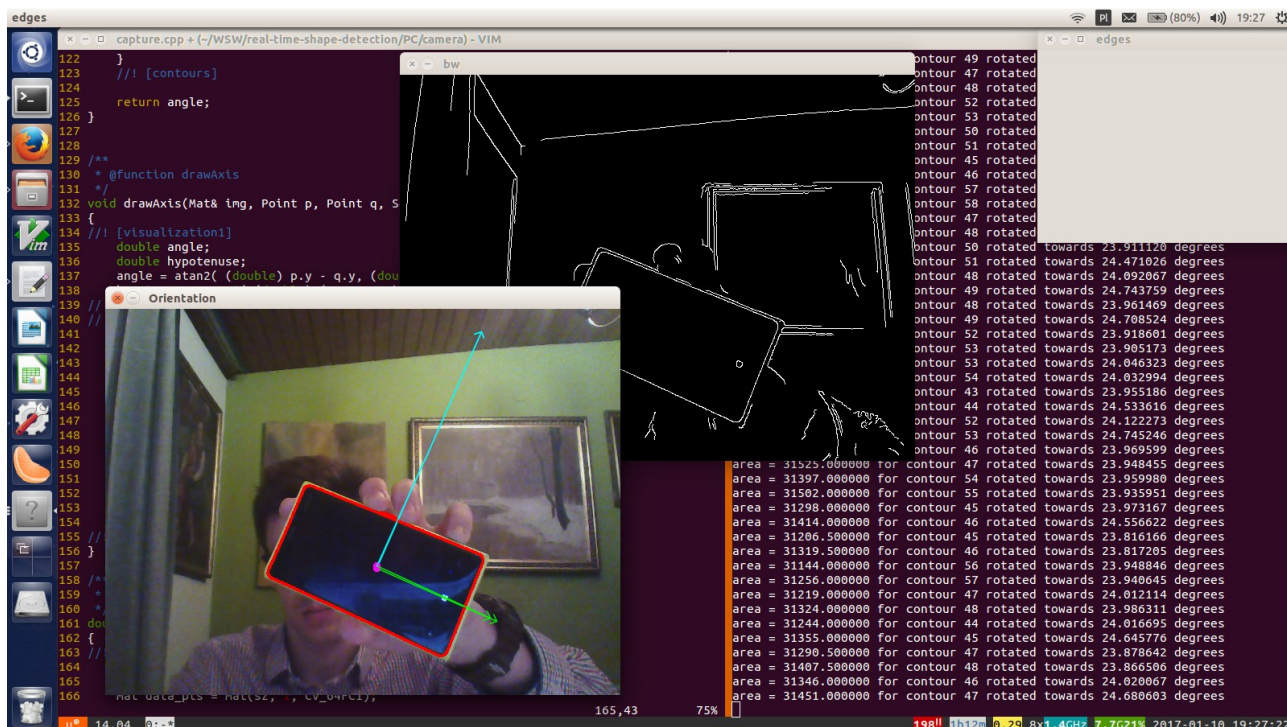
η_{30} OpenCV	-3.044919e-07
η_{30} OpenCL	-3.044918e-07

6.3 Określenie orientacji z wykorzystaniem PCA

Testy obliczania orientacji obiektów przeprowadzono na komputerze osobistym z wykorzystaniem wbudowanej kamery internetowej. Ich uruchomienie zostało wykonane automatycznie, z użyciem skryptu bash **PC/pca/run_pca**. Pod uwagę zostało wzięte przetwarzanie obrazu wraz z binaryzacją i śledzeniem konturów. Sama analiza PCA zachodziła szybko, trwając najwyżej 100 mikrosekund, jednak aby określić szybkość przetwarzania ramek (fps) należy rozpatrywać te operacje łącznie.

6.3.1 Struktura testów

Do wykonania testów wydajności użyto timerów OpenCV, funkcji `cv::getTickCount()` oraz `cv::getTickFrequency()`. Testy przeprowadzono na tych samych wektorach testowych co w sekcji 6.1, o rozdzielczościach 640 x 480, 320 x 240 oraz 1440 x 1080. Wyniki czasowe zostały dołączone do tabeli 2, 3 i 4 z opisem "Czas orientacji PCA" w milisekundach.

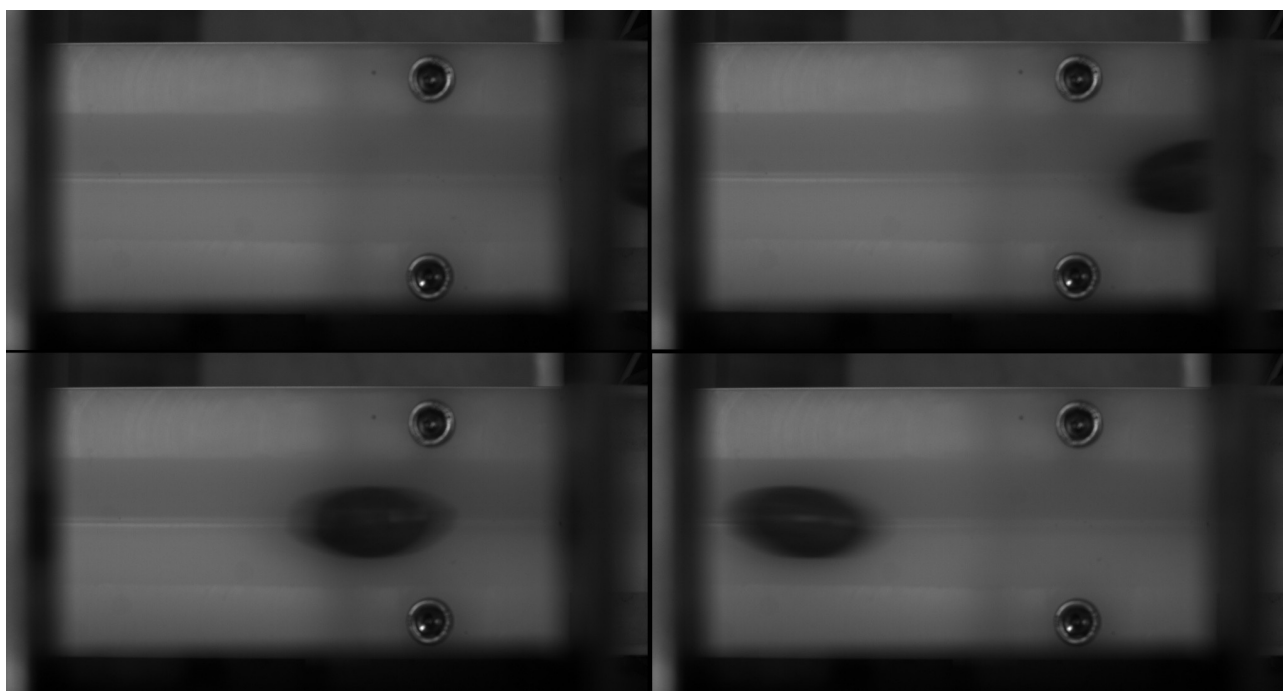


Rys. 8: Testy automatycznego określania orientacji wykrytego obiektu

6.4 Dane rzeczywiste z kamery Basler

Z użyciem kamery Basler zostały przeprowadzone próby w docelowym środowisku, to znaczy na maszynie służącej do chwytania, nacinania i segregacji nasion na zdrowe i nieprzydatne. Kamera jest ustawiona nad pochylnią, po której zjeżdżają nasiona, zanim zostaną uchwycone i nacięte. Obiekty są oświetlone oświetlaczem umieszczonym pomiędzy kamerą a pochylnią.

Przykładową sekwencję obrazów poruszającego się obiektu przedstawiono poniżej.

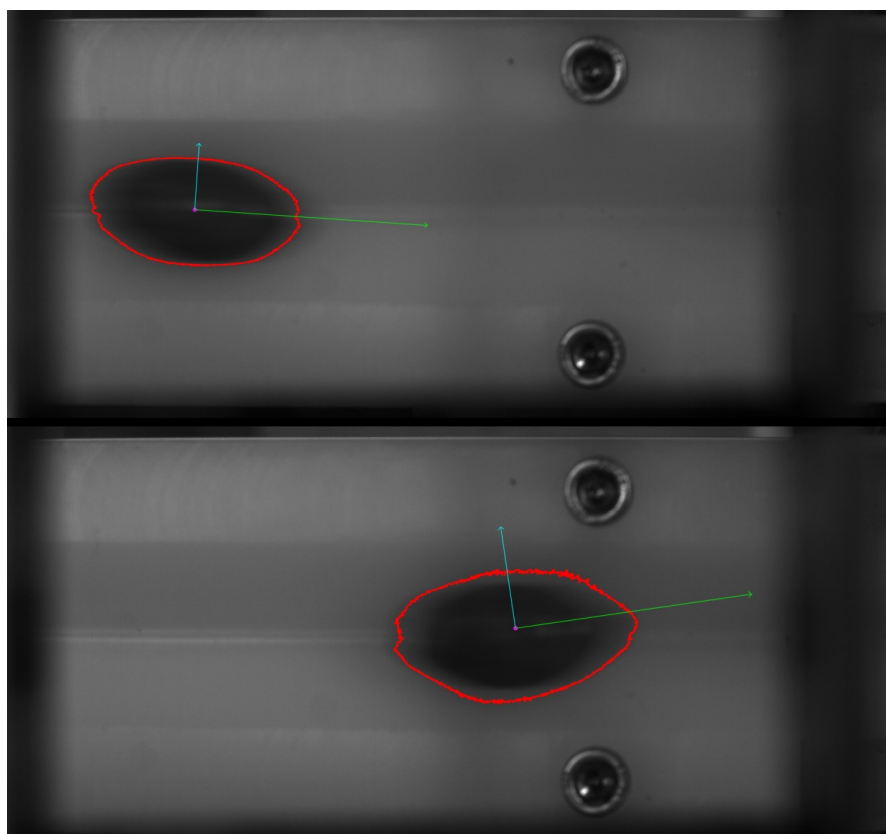


Rys. 9: Sekwencja obrazów testowych przedstawiająca ruch nasiona

Wykrycie konturów i detekcja orientacji zostały przeprowadzone dla obrazów poruszających się obiektów. Wykryte kontury są postrzępione, jednak można wyraźnie określić położenie obiektu.

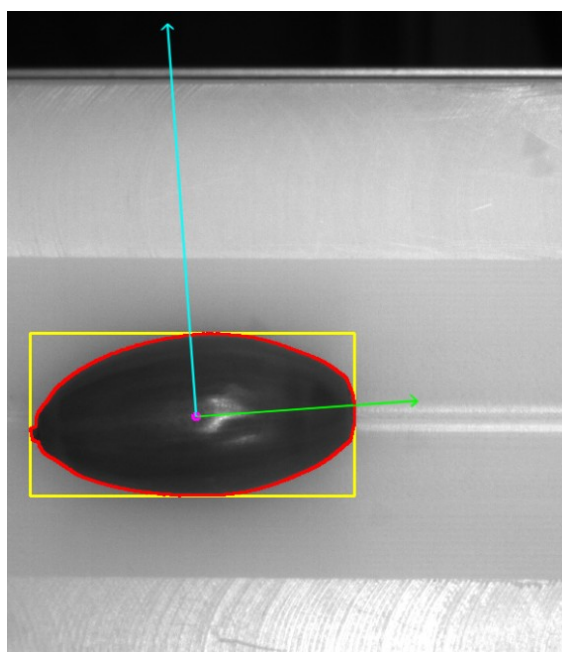
W testach, gdzie był uśredniany czas akwizycji i przetwarzania obrazu z 400 ramek, otrzymano następujące szybkości przetwarzania obrazu:

- 80 FPS jeśli nie wykryto obiektu dla rozdzielczości 600x800
- 80 FPS jeśli wykryto obiekt – razem z PCA, wyrysowaniem konturu i strzałek
- 80 FPS jeśli wykryto obiekt, wyrysowano Bounding Box, oddzielono ROI
- 71 FPS jeśli dodatkowo obliczono momenty Hu
- 66 FPS jeśli wyrysowano zwrot 7-go momentu



Rys. 10: Poruszające się obiekty z wykrytymi konturami i orientacją

Widać, że w tym czynnikiem ograniczającym jest szybkość zapisu na dysk, realizowanego po każdej ramce. Wyjściem, które należy wprowadzić, jest zapis obrazów do pamięci RAM i przesłanie na dysk dopiero po zakończeniu pracy z kamerą.



Rys. 11: Obiekt na pochylni z wyrysowanym ROI (prostokątem)

7. Podsumowanie

Podczas projektu uruchomiono minikomputer ODROID-XU4 z systemem Debian, który wykorzystano do przetwarzania obrazów w torze wizyjnym. Zaproponowano metody określania kształtu i orientacji obiektów z wykorzystaniem biblioteki OpenCV, a następnie wykonano implementację równoległego obliczania momentów geometrycznych z wykorzystaniem OpenCL.

Zdecydowano się zmniejszyć rozdzielczość przesyłanych obrazów do 600x800px, mimo że matryca kamery tworzy zdjęcia o rozmiarze 2MPx. Użycie ramek pełnej wielkości wymaga przesłania z kamery do minikomputera do 200MB danych na sekundę. Wymiana następuje przez interfejs USB 3.0, którego standard określa możliwość przesyłu danych nawet z szybkością 640MB/s, jednak obecnie rzeczywista osiągnięta szybkość przesyłu to zazwyczaj około 100MB/s.

Aby przygotować system zapewniający zawsze rzeczywisty czas reakcji, proponujemy wykorzystanie w przyszłości systemu Linux z jądrem przygotowanym do pracy w czasie rzeczywistym (LinuxRT). O możliwości użycia takiego jądra autorzy dowiedzieli się dopiero po zakończeniu pracy nad projektem. System z jądrem RT zapewnia odpowiednie planowanie zadań i dotrzymanie wymagań czasowych, co jest kluczowe przy pracy z szybko poruszającymi się obiektami.

Planuje się zmianę sposobu działania systemu przez umieszczenie akwizycji w jednym wątku, a przetwarzania obrazu w drugim. Umożliwi to czytanie danych z kamery jednocześnie z przetwarzaniem poprzedniej ramki. Przy takim rozwiązaniu jeszcze wyraźniej widać potrzebę użycia jądra RT, które będzie w stanie odpowiednio przydzielać czas wątkom i przerywać ich wykonanie (*preemptive kernel*), gdyby miały zostać przekroczone wymagania czasowe.

8. Literatura

- [1] HardKernel „Odroid XU4 User Manual”, rev. 20151207
- [2] Hu, M.K. „Visual Pattern Recognition by Moment Invariants”, IRE Transaction of Information Theory IT-8, 1962
- [3] Aldababat, Wafa Khader, Mohammad Hjoui Btoush, and Adnan I. Alrabea. "A PARALLEL ALGORITHM FOR COMPUTATION OF 2D IMAGE MOMENTS." *Journal of Theoretical and Applied Information Technology (JATIT)* 22.1 (2005).
- [4] Sewarwain M. „OpenCL. Akceleracja GPU w praktyce”, Wydawnictwo Naukowe PWN, 2014
- [5] „Getting Started with Pylon and OpenCV”, Application Note
http://s.baslerweb.com/media/documents/AW00136801000_Getting_Started_with_pylon5_and_OpenCV.pdf
- [6] OpenCV Reference Manual. „Structural Analysis and Shape Descriptors”
http://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=cvmatchshapes#humoments
- [7] Jan Flusser, Tom Suk „Construction of Complete and Independent Systems of Rotation Moment Invariants”, *Computer Analysis of Images and Patterns: 10th International Conference, CAIP 2003, Groningen, The Netherlands, August 25-27, 2003. Proceedings*

9. DODATEK A: Szczegółowy opis zadania

Specyfikacja projektu

Celem projektu było przygotowanie toru wizyjnego do określania orientacji i kształtu nasion.

W zakres projektu wchodzi:

- przygotowanie algorytmów obliczania momentów geometrycznych i momentów niezmienniczych Hu
- analiza orientacji obiektu w czasie rzeczywistym
- akceleracja obliczania momentów na karcie GPU z użyciem OpenCL
- uruchomienie minikomputera ODROID-XU4 z systemem Linux, instalacja bibliotek OpenCV i sterowników kamery Basler
- wykonanie testów wydajności zaproponowanych algorytmów

Dokładny opis algorytmów i użytych struktur danych został dostarczony w rozdziale 4.

10. DODATEK B: Dokumentacja techniczna

Dokumentacja oprogramowania

OpenCV w wersji (sprawdzenie - `dpkg -l | grep libopencv`) : **2.4.9.1**

OpenCL w wersji(sprawdzenie - `OpenCL/clinfo`) : **OpenCL C 1.2**

Sterowniki kamery Basler w wersji: **Pylon 5.0.1**

Dokumentacja sprzętowa

Odroid manual – referencja w [1]

Kamera Basler – referencja pod adresem <http://www.baslerweb.com/en>

Procedura symulacji i testowania:

1. Budowa programu do wyliczania momentów geometrycznych i określania orientacji dla obrazu przechwyconego z kamery internetowej w laptopie:

```
$ cd PC/camera
```

```
$ make
```

Uruchomienie programu

```
$ ./capture
```

2. Budowa programu do wyliczania momentów geometrycznych w OpenCL z OpenCV:

Aby możliwe było zbudowanie i uruchomienie aplikacji niezbędne będzie zainstalowanie OpenCV oraz OpenCL. Do kompilacji użyto kompilatora gcc w wersji **5.4.0**

- przejść do katalogu PC/moments
- zbudować aplikację komendą "make"
- uruchomienie aplikacji komendą: **./moments obraz.png** (przykład użycia w run_Hu_moments)

Procedura uruchomieniowa:

Uruchomienie komputera Odroid i przystosowanie do pracy z obrazem wymaga od użytkownika następujących kroków:

- Flashowanie obrazu

Obraz można pobrać z oficjalnej strony producenta Odroida:

<http://odroid.com/dokuwiki/doku.php?id=en:odroid-xu3>

Instrukcję flashowania obrazu znaleźć można pod adresem:

http://odroid.com/dokuwiki/doku.php?id=en:odroid_flashing_tools

- Uruchomienie i wstępna konfiguracja systemu

Po załączeniu zasilania i podpięciu niezbędnych peryferiów (mysz, klawiatura – warto użyć aktywnego HUBa rozdzielającego, monitor) należy zalogować się używając następujących danych:

user: **odroid**

password: **odroid**

UWAGA: W razie niepowodzenia którejs z poniższych komend, warto spróbować z prawami użytkownika root, przez użycie komendy **sudo**, lub przełączyć się na roota komendą **su** – hasło i użytkownik jak dla zwykłego użytkownika o ile nie skonfigurowano inaczej).

- Konfiguracja interfejsów sieciowych

W zainstalowanym systemie konieczna może być korekta ustawień interfejsów sieciowych. Dokonuje się ją edytując plik **/etc/network/interfaces**. Najbezpieczniej ustawić interfejs w tryb automatyczny z włączonym uzyskiwaniem adresu za pośrednictwem DHCP:


```
auto eth0
```

```
iface eth0 inet dhcp
```

w przypadku konieczności nadania komputerowi ręcznie ustalonego adresu IP można posłużyć się następującym przykładem:

```
iface eth0 inet static
```

```
address 192.168.1.5
```

```
netmask 255.255.255.0
```

```
gateway 192.168.1.254
```

- Instalacja środowiska graficznego

Lekkie środowisko graficzne znacznie ułatwi pracę i orientację w systemie oraz umożliwi testowanie przetwarzanego z kamery w czasie rzeczywistym. Przykładowym środowiskiem graficznym może być **LXDE**. Aby zainstalować go wraz z dependencjami należy wpisać:

```
apt-get install x11
```

```
apt-get install lightdm
```

```
apt-get install lxde
```

- Instalacja kompilatora **gcc**

```
apt-get install build-essentials
```

```
apt-get install gcc
```

- Instalacja pakietu OpenCV

```
apt-get install libopencv-dev
```

- Instalacja sterowników do kamery firmy Pylon

Sterowniki do kamery USB mogą zostać pobrane ze strony producenta kamery:

<http://www.baslerweb.com/en/products/software/pylon-linux-arm>

Procedura instalacji opisana jest w pliku `INSTALL` znajdującym się w paczce sterowników. Sprowadza się ona do skopiowania sterowników do systemu oraz dodaniu reguł dla kamery (`udev-rules`) w systemie.

- Instalacja środowiska OpenCL

Większość udostępnionych systemów będzie posiadało zainstalowane biblioteki OpenCL. Gdyby było inaczej pomocy należy szukać na stronie producenta układu graficznego w Odroidzie:

<http://malideveloper.arm.com/resources/sdks/mali-opencl-sdk/>

11. DODATEK C: Spis zawartości dołączonego nośnika (płyta CD ROM)

W poszczególnych folderach nośnika CD znajdują się:

- **bib** – Zebrane prace naukowe w formacie pdf oraz instrukcje użytkowania sprzętu użytego w projekcie
- **OpenCL** - Folder zawierający kod programów napisanych z użyciem OpenCL, kernele obliczeniowe oraz pliki wykonywalne
- **PC** – modele programowe przeznaczone do kompilacji i uruchamiania na komputerze PC z systemem Linux. Wymagana biblioteka OpenCV 2.4 oraz kompilator g++. Moduły wymagane do kompilacji można zainstalować z użyciem skryptu INSTALL.sh w głównym folderze. Programy buduje się poleceniem 'make'
 1. **PC/camera** – dokładny model końcowej aplikacji, przeznaczony do uruchomienia z kamerką internetową w laptopie
 2. **PC/img** – folder zawierający wektory testowe, w tym obrazy nasion 'test_nutX.jpg' użyte do porównania obliczeń momentów Hu
- **pylonCV** – programy napisane dla platformy ODROID-XU4 współpracującej z kamerą Basler. Obrazy są pobierane z kamery za pomocą sterownika Pylon 5, a następnie przetwarzane z użyciem algorytmów OpenCV, aby pokazać na ekranie monitora (HDMI) kontury wykrytego obiektu i jego orientację
- **report** - aktualna wersja niniejszego raportu w postaci elektronicznej (MS WORD i PDF) oraz pliki diagramów .ddd przeznaczone do edycji w programie Diagram Designer

12. DODATEK D: Historia zmian

Tabela 1 Historia zmian.

Autor	Data	Opis zmian	Wersja
Piotr Pałucki	18.12.2016	Utworzono pierwszą wersję	1.0
Filip Kubicz			
Piotr Pałucki	11.01.2017	Wersja uzupełniona o wyniki testów i opis uruchomienia programów	2.0
Filip Kubicz			
Piotr Pałucki	24.01.2017	Poprawione błędy, dodane testy na rzeczywistych obiektach z kamery Basler, przekazywanie do funkcji wyliczającej momenty w OpenCL tylko ROI	2.7
Filip Kubicz			

pr16wsw503			Karta oceny projektu
Data	Ocena	Podpis	Uwagi