

Przedmiot:	Wieloprocessorowe Systemy Wizyjne.			
			pr16wsw503	
Temat projektu:				
	<p align="center">Detekcja kształtu oraz orientacji obiektów za pomocą momentów geometrycznych w czasie rzeczywistym</p>			
Wykonali:				
			Automatyka i Robotyka	
Rok akademicki		Semestr zimowy		
Prowadzący		dr inż. Mirosław Jabłoński		
<p align="center">wersja 1.0</p> <p align="center">Kraków, grudzień, 2016</p>				

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 szybkoklatkowej, 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 środowiskiem obliczeń równoległych OpenCL.

Słowa kluczowe: Momenty 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 szybkoklatkowa 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.

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

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 obiektu w różnych 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 60/100 klatek na sekundę i głębią kolorów 10-bit. Używana kamera Basler acA2000-165um pozwala na akwizycję obrazu z maksymalną prędkością 165 fps, z głębią kolorów 10- 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

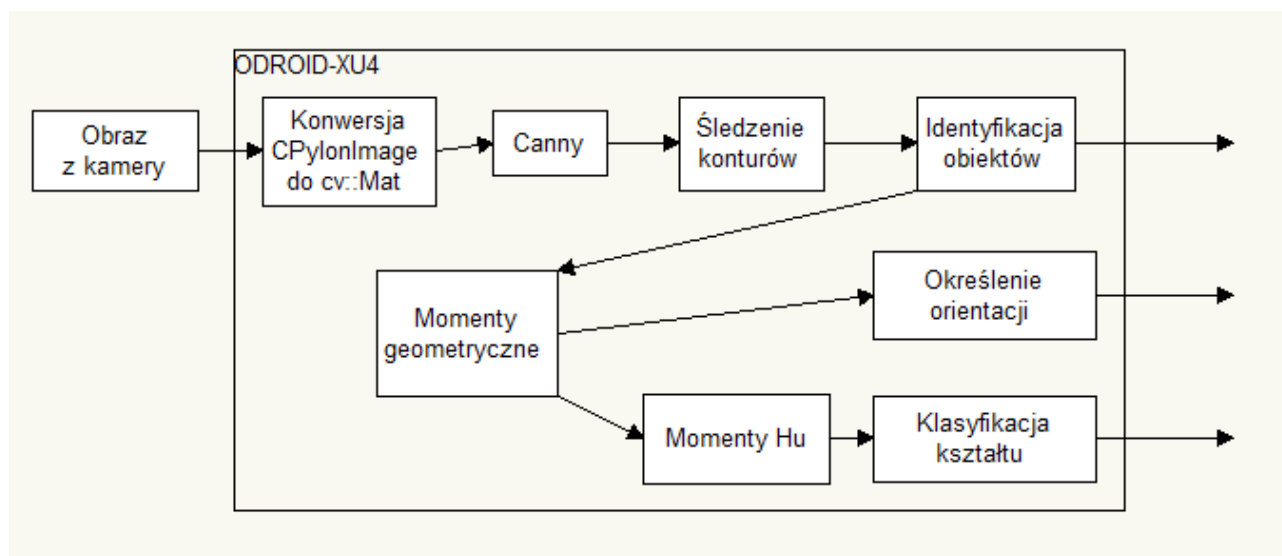
Aby uzyskać założone cele, użyto biblioteki algorytmów wizyjnych *OpenCV* z interfejsem w języku C++. Do akwizycji obrazu zostały wykorzystane funkcje udostępnione przez moduł *Pylon 5*, współpracujący z użytą kamerą Basler. Użycie tych komponentów było możliwe, ponieważ na minikomputerze ODROID zainstalowano 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 wystarczy do dokładnej analizy kształtu i orientacji, a większy (dostępny nawet 2Mpx) może wyraźnie spowolnić przetwarzanie. **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.

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

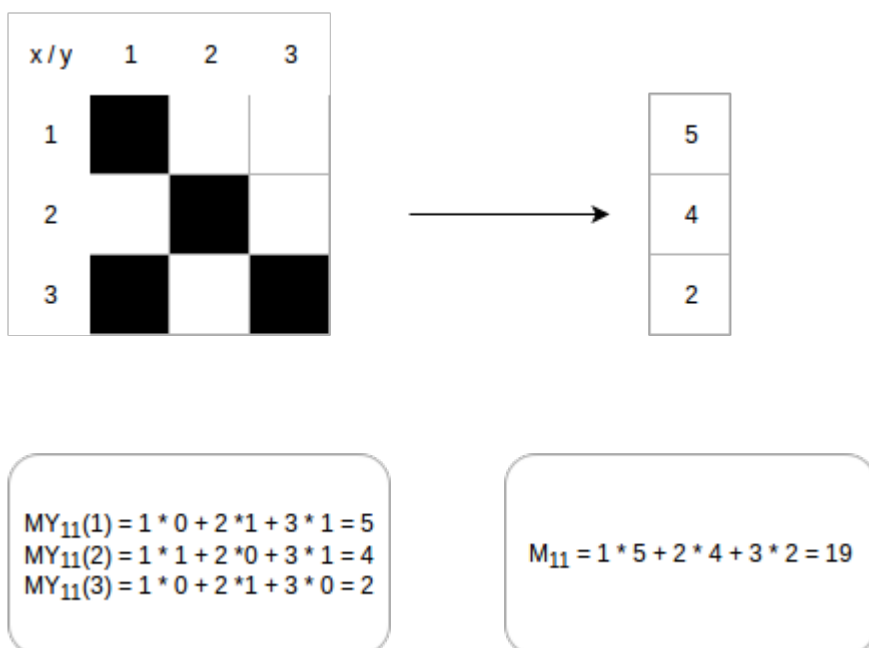
2) Wyliczenie momentów geometrycznych na podstawie 1)

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. Piksel biały reprezentuje wartość 1, piksel czarny – 0.



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 typu **float8** w OpenCL)

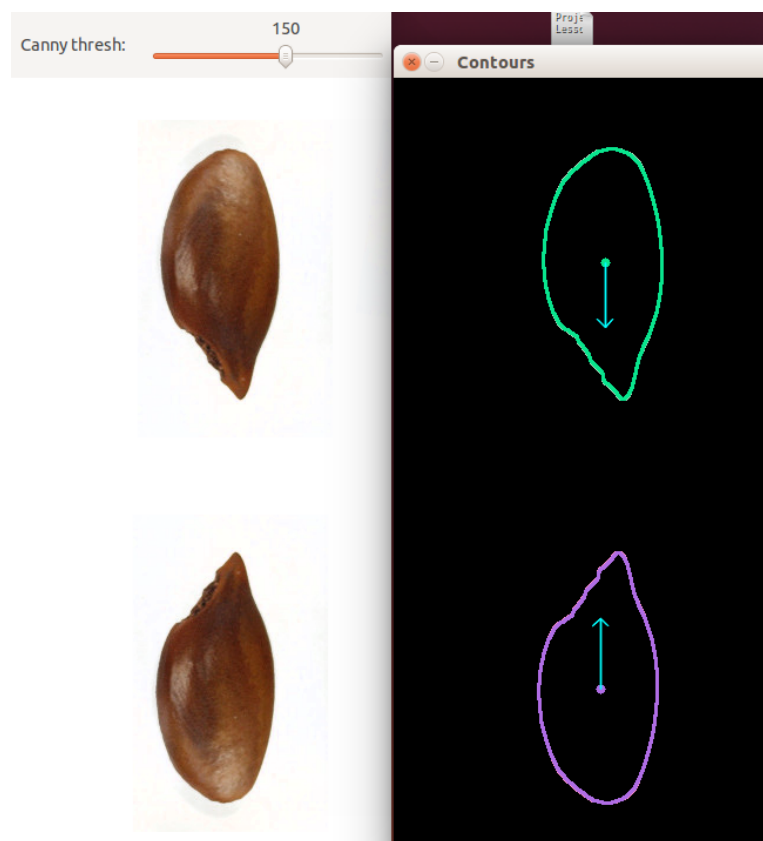
Takie podejście automatycznie nakłada ograniczenie na wielkość obrazu – jego szerokość w pikselach musi być liczbą podzielną przez 8.

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.

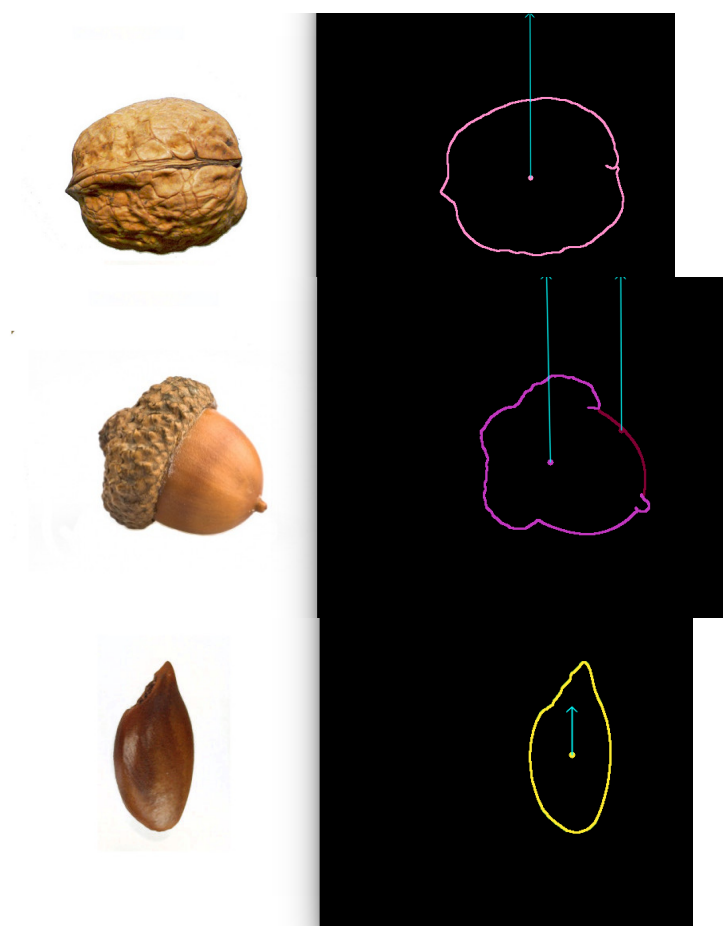
W tabeli 1 zgromadzono obliczone dla trzech wektorów testowych momenty H_u . Z kolei na rysunku 3 przedstawiono, jak siódmy moment H_u może posłużyć do odróżnienia obiektów stanowiących swoje odbicie lustrzane. Zwrot strzałki symbolizuje na rysunku znak momentu I_7 .



Rys. 3: Przykład z zaznaczonym siódmym momentem H_u

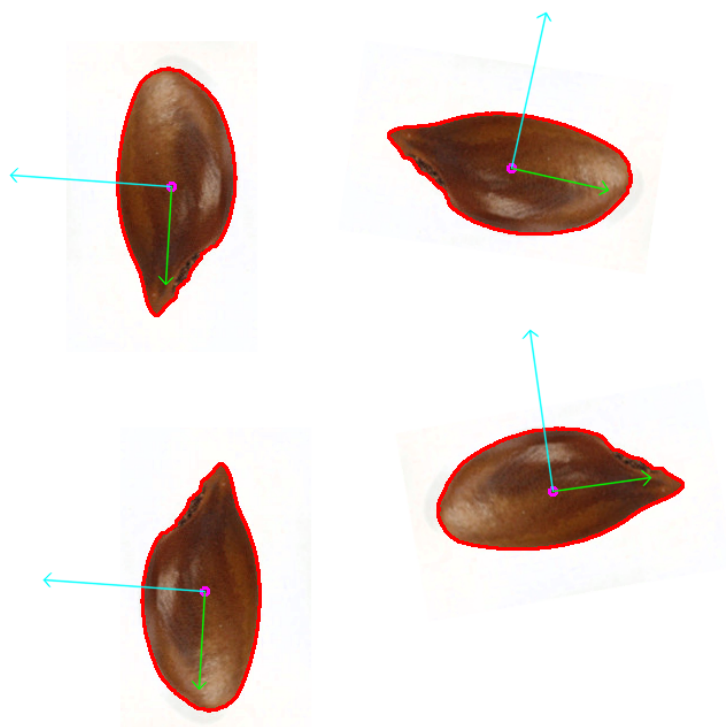
Tabela 1. Momenty Hu oraz długości konturu głównego rozpoznanego obiektu
dla wektorów testowych

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

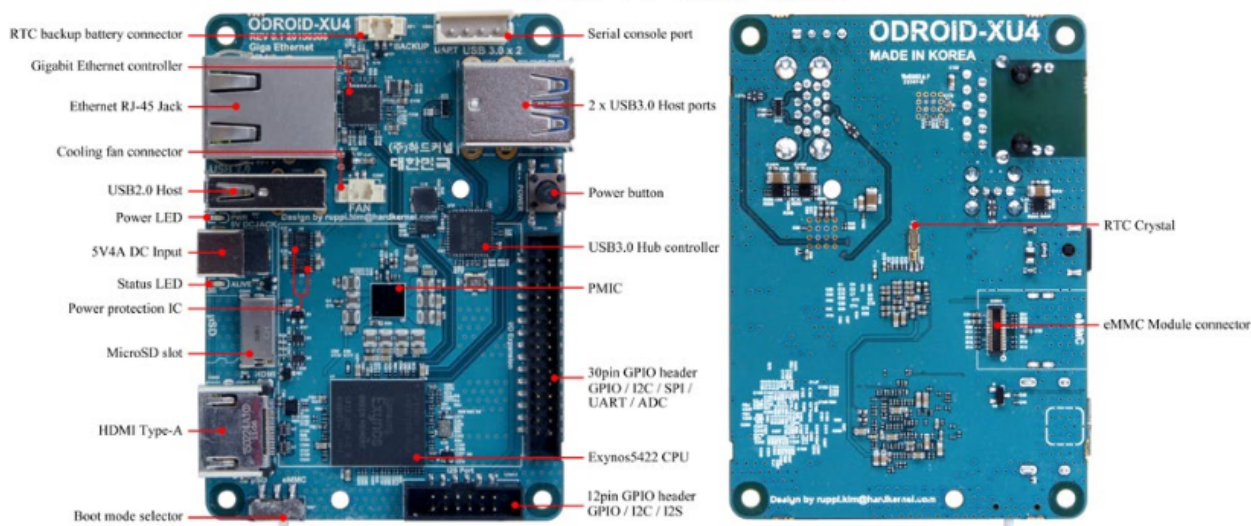
Na rysunku przedstawiono udaną próbę rozpoznania orientacji obiektów na płaszczyźnie z użyciem algorytmu PCA. Planuje się zintegrowanie tego algorytmu z algorytmem detekcji kształtu i wykorzystanie obliczanych na potrzeby kształtu momentów w celu minimalizacji ilości obliczeń.



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 odsyłam do Dodatku B.

6. Rezultaty i wnioski

7. Podsumowanie

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 Hjouj 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

Szczegółowy opis realizowanych algorytmów przetwarzania danych.

10. DODATEK B: Dokumentacja techniczna

Dokumentacja oprogramowania

Dokumentacja sprzętowa

Procedura symulacji i testowania:

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 roota, przez użycie **sudo** komenda, 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 D: Spis zawartości dołączonego nośnika (płyta CD ROM)

12. DODATEK E: Historia zmian

Tabela 1 Historia zmian.

[illegible]

pr16wsw503			Karta oceny projektu
Data	Ocena	Podpis	Uwagi