AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

KATEDRA AUTOMATYKI I INŻYNIERII BIOMEDYCZNEJ

Praca dyplomowa inżynierska

System wizyjny śledzący obiekty wykorzystujący ruchomą kamerę zrealizowany w oparciu o heterogeniczny układ Zynq.

An object tracking vision system using a moving camera implemented in a Zynq heterogeneous device.

Autor:	*Marcin Kowalczyk*
Kierunek studiów:	*Automatyka i Robotyka*
Opiekun pracy:	*dr inż. Tomasz Kryjak*

Kraków, 2016

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

Serdecznie dziękuję . . . tu ciąg dalszych podziękowań np. dla promotora, żony, sąsiada itp.

Spis treści

1. Wstęp

1.1. Cel pracy

Celem pracy jest stworzenie demonstratora systemu wizyjnego do śledzenia obiektów, przy czym zakłada się, że kamera zamontowana jest na głowicy obrotowej. Praca inżynierska obejmuje skompletowanie, we współpracy z opiekunem, stanowiska testowego składającego się z głowicy ruchomej, kamery, platformy obliczeniowej oraz wskaźnika. Należy przeprowadzić analizę oraz weryfikację różnych koncepcji śledzenia, biorąc pod uwagę skuteczność oraz możliwość implementacji sprzętowej lub sprzętowo-programowej. Wybrane rozwiązanie zostało zaimplementowane, uruchomione i przetestowane w sprzęcie. Wyjście z modułu śledzenia stanowi podstawę do wypracowania pozycjonowania głowicy obrotowej, takiego, aby utrzymać obiekt w środku kadru oraz oznaczyć go wskaźnikiem.

1.2. Wprowadzenie

Automatyczne śledzenie z pomocą kamery jest wykorzystywane m.in. w zagadnieniach związanych z bezpieczeństwem, inwigilacją lub w zastosowaniach wojskowych. Jest ściśle powiązane z wykrywaniem oraz identyfikacją obiektów. Świadczy to o złożoności tego zagadnienia. Śledzenie różni się od innych algorytmów przetwarzania obrazów głównie tym, że musimy wykorzystywać informacje pochodzące z więcej niż jednego kadru. Uwydatnia się to szczególnie, kiedy kamera rejestruje kilka obiektów podobnych do śledzonego. Jeśli zadaniem jest obserwacja jednego konkretnego obiektu, to aby go wyróżnić musimy wykorzystać położenie śledzonego obiektu w poprzednich ramkach [1]. Znaczące problemy w śledzeniu obiektów mogą być spowodowane zmianą orientacji celu oraz dużą jego szybkością w porównaniu do częstotliwości rejestrowania klatek. W przypadku kamery umieszczonej na głowicy błędy mogą być dodatkowo spowodowane rozmyciem obrazu w trakcie poruszania układu. Istnieją liczne algorytmy służące śledzeniu elementu na obrazach z kamery. Są to m.in:

- Śledzenie przez detekcję
- Mean-shift
- Filtr cząsteczkowy
- KLT

Algorytmy te zostaną dokładniej omówione w kolejnym rozdziale pracy.

1.3. Zawartość pracy

W rozdziale ...

2. Algorytmy

W rozdziale tym omówione zostaną algorytmy śledzenia. Rozważymy również możliwość oraz skuteczność ich implementacji sprzętowej i sprzętowo-programowej.

2.1. Śledzenie przez detekcję

Jest to najłatwiejszy możliwy algorytm śledzenia. Polega on na detekcji obiektu i określeniu jego położenia (np. poprzez wyznaczenie środka ciężkości). Algorytm ten jest efektywny tylko, gdy w kadrze znajduje się maksymalnie jeden wyznaczony obiekt. Ograniczenie to możemy obejść przez zastosowanie dodatkowo algorytmu indeksacji. Wtedy musimy zdecydować który z wyznaczonych obiektów jest tym właściwym. Możemy np. śledzić element znajdujący się najbliżej (mający największą powierzchnię w kadrze). Efektywność implementacji programowo-sprzętowej zależy w większości od wykorzystanego algorytmu detekcji, i/lub segmentacji. Algorytm detekcji na podstawie koloru będzie łatwy i szybki do zaimplementowania,a algorytm bazujący na współczynnikach kształtu wymagać będzie dużo więcej pracy i zasobów.

2.2. Mean-shift

Algorytm ten bazuje na statystycznej metodzie poszukiwania lokalnego maksimum rozkładu prawdopodobieństwa. Polega on na wyznaczeniu maksymalnej wartości prawdopodobieństwa w aktualnym oknie wokół punktu startowego. W celu zastosowania tego algorytmu do śledzenia obiektu należy przedstawić obraz jako rozkład prawdopodobieństwa. W tym celu każdemu pikselowi przypisuje się wartość prawdopodobieństwa. Można to zrobić np. na podstawie koloru, przypisując kolorom wartość prawdopodobieństwa, lub histogramu, porównując histogram otoczenia piksela z histogramem obiektu [2]. Jeśli prawdopodobieństwo obliczone zostaje tylko na podstawie koloru, dobre wyniki możemy uzyskać, gdy:

- Obiekt ma jednolitą barwę.
- Występują bardzo małe zmiany oświetlenia.
- W kadrze nie ma obiektów podobnych do śledzonego.
- Kolor tła znacznie różni się od koloru obiektu.

– Obiekt nie może zostać całkowicie zasłonięty.

Algorytm ten ma następujący przebieg:

1. Wybierz rozmiar okna.
2. Wybierz początkowe położenie(środek) okna.
3. Wyznacz położenie maksimum wartości prawdopodobieństwa w oknie.
4. Przesuń okno, aby wyznaczone maksimum było jego środkiem.
5. Powtarzaj kroki 3 i 4, aż algorytm będzie zbieżny.

Punktem startowym dla każdego kolejnego obrazu z kamery jest pozycja obiektu w poprzednim obrazie. Poszukiwanie maksymalnej wartości wartości prawdopodobieństwa odbywa się w następujący sposób:

– Obliczamy moment zerowy.

$$M_{00} = \sum_x \sum_y l(x, y) \tag{2.2.1}$$

– Obliczamy moment pierwszy dla osi poziomej.

$$M_{10} = \sum_x \sum_y x \cdot l(x, y) \tag{2.2.2}$$

– Obliczamy moment pierwszy dla osi pionowej.

$$M_{01} = \sum_x \sum_y y \cdot l(x, y) \tag{2.2.3}$$

– Obliczamy środek rozkładu prawdopodobieństwa w danym oknie.

$$x_c = \frac{M_{10}}{M_{00}} \tag{2.2.4}$$

$$y_c = \frac{M_{01}}{M_{00}} \tag{2.2.5}$$

Gdzie $l(x, y)$ jest wartością prawdopodobieństwa dla piksela (x, y)[3].

2.3. Filtr cząsteczkowy

2.4. KLT

Algorytm opisany przez Kanade'a, Lucas'a i Tomasi'a może posłużyć do śledzenie obiektu na obrazie w skali szarości. Polega on na poszukiwaniu najlepszego dopasowania obrazu referencyjnego $T(x)$

do aktualnej ramki otrzymanej z kamery $I(x)$ [4]. Obrazy te są przesuwane względem siebie o wektor p:

$$W(x,p) = \begin{bmatrix} x_1 + p_1 \\ x_2 + p_2 \end{bmatrix} \tag{2.4.1}$$

Poszukiwane jest więc minimum następującej funkcji:

$$f(x,p) = \sum_x ((I(W(x,p)) - T(x))^2 \tag{2.4.2}$$

Zakłada się, że początkowa wartość przesunięcia p jest znana, a poszukiwana jest najlepsza modyfikacja tego przesunięcia Δp.

$$f(x,\Delta p) = \sum_x ((I(W(x,p+\Delta p)) - T(x))^2 \tag{2.4.3}$$

Po znalezieniu optymalnego przesunięcia Δp aktualizowana jest wartość przesunięcia początkowego:

$$p \leftarrow p + \Delta p \tag{2.4.4}$$

Funkcja $I(x, p + \Delta p)$ linearyzowana jest w otoczeniu punktu (x, p) ze względu na Δp poprzez rozwinięcie w szereg Taylora pierwszego rzędu.

$$f(x,\Delta p) = \sum_x (I(W(x,p)) + \nabla I \cdot \frac{\partial W}{\partial p} \cdot \Delta p - T(x))^2 \tag{2.4.5}$$

$\nabla I = [\frac{\partial I}{\partial x_1}, \frac{\partial I}{\partial x_2}]$ jest transponowanym gradientem obrazu wejściowego w punkcie $W(x, p)$.
$\frac{\partial W}{\partial p}$ jest Jakobianem przesunięcia obrazów.

Obliczamy pochodną funkcji dopasowania $f(x, \Delta p)$ po Δp i przyrównujemy ją do zera.

$$\frac{\partial f(x,\Delta p)}{\partial \Delta p} = 2 \cdot \sum_x (\nabla I \cdot \frac{\partial W}{\partial p})^T \cdot ((I(W(x,p)) + \nabla I \cdot \frac{\partial W}{\partial p} \cdot \Delta p - T(x)) = 0 \tag{2.4.6}$$

Przekształcając powyższy wzór otrzymuje się:

$$\Delta p = H^{-1} \cdot \sum_x (\nabla I \cdot \frac{\partial W}{\partial p})^T \cdot (T(x) - I(W(x,p))) \tag{2.4.7}$$

H jest Hesjanem:

$$H = \sum_x (\nabla I \cdot \frac{\partial W}{\partial p})^T \cdot (\nabla I \cdot \frac{\partial W}{\partial p}) \tag{2.4.8}$$

Aby algorytm dobrze działał musimy odpowiednio wybrać obraz referencyjny. Zauważmy, że we wzorze na Δp jest odwrotność Hesjanu. Odwracanie macierzy może powodować duże błędy numeryczne, gdy macierz ta jest źle uwarunkowana. Oznacza to, że Hesjan musi posiadać odpowiednio duże wartości własne.

$$\lambda(H) > \lambda_{thr} \tag{2.4.9}$$

W praktyce powyższy warunek oznacza, że obraz referencyjny nie może być jednolity. W najlepszym wypadku zawierał będzie on krawędzie śledzonego obiektu.

Algorytm ma następujący przebieg [5]:

1. Znajdź obszary spełniające warunek na wartości własne hesjanu.
2. Wyznacz obraz przesunięty o p.
3. Wyznacz gradient ∇I.
4. Oblicz Jakobian $\frac{\partial W}{\partial p}$ oraz iloczyn $\nabla I \cdot \frac{\partial W}{\partial p}$.
5. Oblicz Hesjan $H = \sum_x (\nabla I \cdot \frac{\partial W}{\partial p})^T \cdot (\nabla I \cdot \frac{\partial W}{\partial p})$.
6. Wyznacz przesunięcie $\Delta p = H^{-1} \cdot \sum_x (\nabla I \cdot \frac{\partial W}{\partial p})^T \cdot (T(x) - I(W(x,p)))$.
7. Zaktualizuj parametr $p \leftarrow p + \Delta p$.
8. Powtarzaj kroki od 2 do 7 do czasu, gdy algorytm będzie zbiegał do punktu.

3. Stanowisko demonstracyjne

Pierwszą częścią pracy jest skompletowanie stanowiska demonstracyjnego, w skład którego wchodzą:

- Kamera
- Platforma obliczeniowa
- Serwomechanizmy
- Sterownik serwomechanizmów
- Zasilanie
- Wskaźnik

3.1. Kamera

Postanowiono użyć kamery... (Tutaj nazwa kamery, specyfikacja, krótki opis).

3.2. Platforma obliczeniowa

Rys. 3.1. Płytka ZYBO Zynq-7000.
Źródło: [6]

Zdecydowano, że platformę obliczeniową stanowić będzie płytka ZYBO. Jest to bogato wyposażone narzędzie zawierające układ programowalny z rodziny Xilinx Zynq-7000. Układ ten oparty jest na architekturze Xilinx All Programable System-on-Chip, w której zintegrowany został dwurdzeniowy procesor ARM Cortex-A9 i układ programowalny FPGA z serii Xilinx 7. Zawiera ona m.in. port HDMI potrzebne do odbieranie obrazu z kamery oraz port VGA, który umożliwia nam wysyłanie obrazu do monitora [6]. W układzie programowalnym zaimplementowany zostanie tor wizyjny przetwarzający potokowo dane przesyłane z kamery oraz wyznaczający położenie obiektu w danej ramce. Oprócz tego musi on komunikować się ze sterownikiem serwomechanizmów w celu pozycjonowania głowicy w wyznaczonym punkcie oraz z komputerem klasy PC, aby otrzymywać parametry pracy oraz ustawiać początkową pozycję głowicy.

3.3. Serwomechanizmy

Ruchomą głowicę postanowiono skonstruować wykorzystując serwomechanizmy dostępne na rynku. Przy wyborze brano pod uwagę serwomechanizmy analogowe, serwomechanizmy cyfrowe oraz serwomechanizmy smart.

3.3.1. Serwomechanizm analogowy

Serwomechanizm ten sterowany jest impulsami podawanymi z częstotliwością 50 Hz. Podawanie impulsów z większą częstotliwością może doprowadzić do uszkodzenia urządzenia. W zależności od czasu trwania impulsu serwomechanizm ustawia się w odpowiedniej pozycji i ją utrzymuje. Kontrola położenia odbywa się za pomocą potencjometru sprzężonego z kołem zębatym. Serwomechanizm tego typu nie reagują szybko i nie produkują wystarczającego momentu kiedy zadajemy małe przesunięcie.

3.3.2. Serwomechanizm cyfrowy

Jest on, podobnie jak serwomechanizm analogowy, sterowany impulsami. Podawać można je jednak z maksymalną częstotliwością powyżej 300 Hz. W porównaniu do serwomechanizmu analogowego przyspiesza ono szybciej i produkuje stabilniejszy moment obrotowy. Reagują również lepiej na polecenia małych zmian pozycji. Z drugiej strony potrzebują one dużo większej mocy. Prąd, który pobierają w trakcie pracy, może sięgać kilku amperów.

3.3.3. Serwomechanizm smart

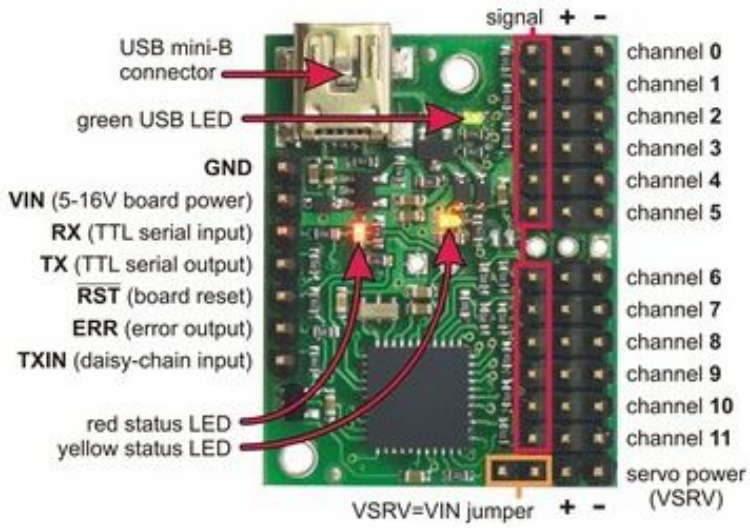
Główną różnicą pomiędzy serwomechanizmami standardowymi, a smart, jest sposób komunikacji w celu podania wartości zadanej położenia. Zamiast impulsów o różnej szerokości wykorzystuje się komunikację szeregową. Ich obsługa jest więc bardziej skomplikowana. Najpopularniejszymi protokołami są TTL Half-Duplex, TTL Full-Duplex i RS-485. Za pomocą komunikacji tego typu można zmieniać dużo więcej parametrów niż tylko pozycję zadaną. Można zmieniać nastawy regulatora PID, maksymalną prędkość kątową lub maksymalny moment. W przeciwieństwie do serwomechanizmów klasycznych, możemy również odczytywać aktualną pozycję serwomechanizmu. Minusem tych urządzeń jest ich cena oraz dostępność. Są one kilka razy droższe od cyfrowych serwomechanizmów o podobnych parametrach.

Postanowiono użyć dwóch serwomechanizmów cyfrowych PowerHD D-21HV. Są to serwomechanizmy typu standard o dużym momencie obrotowym i wysokiej maksymalnej prędkości obrotowej. Wyposażone jest w łożyska kulkowe oraz tytanowe tryby w celu zwiększenia jego wytrzymałości. Specyfikacja serwomechanizmów:

- Napięcie zasilania: $6.0 - 7.4$ V
- Zakres ruchu: 155°
- Masa: 75 g
- Moment: 21 kg·cm dla napięcie 7.4 V
- Prędkość: 0.12s/60° dla napięcia 7.4 V

Urządzenia te mogą pobierać impulsowy prąd o wartości nawet powyżej 3 A. Sterowane są za pomocą impulsów podawanych z częstotliwością 333 Hz. Pozycję naturalną zadajemy podając impulsy o czasie trwania 1500μs. Serwa zostały połączone w głowicę za pomocą metalowych uchwytów dedykowanych do łączenia serwomechanizmów typu standard w konfigurację PT.

3.4. Sterownik serwomechanizmów



Rys. 3.2. Sterownik serwomechanizmów Pololu Mini Maestro.
Źródło: [7]

Do sterowania serwomechanizmami postanowiono użyć gotowego sterownika produkowanego przez firmę Pololu. 12-kanałowy sterownik Mini Maestro pozwala na równoczesną obsługę obu serwomechanizmów. Oprócz podawania wartości zadanej w postaci impulsów pozwala on również na zmianę ustawianie maksymalnej prędkości obrotowej i maksymalnego momentu obrotowego. Dzięki temu możemy dużo lepiej kontrolować urządzenia wykonawcze i zapewnić bardziej ciągły ruch głowicy. Komunikujemy się z nim za pośrednictwem interfejsu szeregowego UART z prędkością 115200 bitów na sekundę.

3.5. Zasilanie

Do zasilania głowicy użyty został zasilacz impulsowy Redox, który na wyjściu daje napięcie 12 V i maksymalny prąd 5 A. Jako, że serwomechanizmy potrzebują napięcia 7.4 V używamy przetwornicy step-down XL4005E1 o regulowanym napięciu wyjściowym. Maksymalny prąd wyjściowy użytej przetwornicy wynosi 5 A. Zdecydowanie wystarczy to do zasilenia użytych serwomechanizmów.

3.6. Wskaźnik

Postanowiono użyć zwykłego wskaźnika laserowego do wskazywania miejsca, w stronę którego zwrócona jest głowica.

4. Komunikacja

Po zbudowaniu stanowiska postanowiono w pierwszej kolejności na implementację komunikacji między wszystkimi elementami systemu. Można ją podzielić na część PC-ZYNQ oraz ZYNQ-Maestro.

4.1. PC-ZYNQ

Postanowiono wykorzystać UART. Argumentem przemawiającym za tym rozwiązaniem był fakt wyprowadzenia dwóch pinów MIO (48 i 49) procesora płytki ZYBO do złącza mikro USB typu B. Piny te podłączone zostały do jednego z układów procesora odpowiedzialnych za komunikację szeregową - UART1. Złącze USB służy również do programowania układu przez JTAG. Dzięki temu używa się jednego przewodu do programowania układu oraz do komunikacji z nim. Po podłączeniu przewodu oraz włączeniu ZYBO w komputerze pojawia się dodatkowy port COM. Przez ten port można wymieniać dane z platformą obliczeniową. Do wysyłania komunikatów na PC wykorzystano program *Realterm*. Dzięki temu nie było konieczności pisania dodatkowego oprogramowania.

Rozmiar jednej komendy wysyłanej do układu ZYNQ wynosi 5 bajtów. Pierwszy informuje o typie rozkazu, a 4 kolejne są danymi do tego rozkazu. Każdy kolejny odebrany bajt umieszczany jest w buforze o rozmiarze 5 bajtów. Włączone zostało przerwanie od pełnego bufora. Po wysłaniu komendy z PC w procesorze układu ZYNQ uruchamiane jest przerwanie, w którym wysłany zostaje odpowiedni rozkaz do sterownika serwomechanizmów. Zaimplementowano obsługę następujących komend w ZYNQ:

- Zmiana pozycji serwomechanizmów: 0x00 0xHH 0xLL 0xHH 0xLL.
- Zmiana maksymalnej prędkości serwomechanizmów: 0x01 0xHH 0xLL 0xHH 0xLL.
- Zmiana maksymalnego momentu serwomechanizmów: 0x02 0xHH 0xLL 0xHH 0xLL.
- Odczyt pozycji ze sterownika: 0x03 0x– 0x– 0x– 0x–.
- Rozpoczęcie pracy autonomicznej (śledzenia): 0x04 0x– 0x– 0x– 0x–.

W pierwszych trzech rozkazach pierwsze dwa bajty danych są parametrami dla serwomechanizmu odpowiedzialnego za obrót, a kolejne dwa są parametrami dla serwomechanizmu odpowiedzialnego za

nachylenie. Myślniki w kolejnych komendach oznaczają, że wysłane bajty danych nie są istotne (komendy te nie potrzebują danych).

4.2. ZYNQ-Maestro

Komunikacja przez UART została narzucona, gdyż jest to jedyny protokół komunikacyjny w Maestro. Łączymy odpowiednie piny (MIO 14,15) złącza MIO PMOD płytki ZYBO do pinów Maestro odpowiadających za komunikację szeregową. Następnie do użytych pinów MIO podłączamy wyjścia innego układu procesora odpowiadającego za komunikację szeregową - UART0. W dokumentacji Maestro wyszukujemy listę komend i implementujemy wysyłanie potrzebnych w układzie ZYNQ [7]. Są to następujące komendy:

– Zmiana pozycji serwomechanizmów: 0x9F 0x02 0x0A 0xLL 0xHH 0xLL 0xHH.

– Zmiana maksymalnej prędkości serwomechanizmów: 0x87 0x0A/0x0B 0xLL 0xHH.

– Zmiana maksymalnego momentu serwomechanizmów: 0x89 0x0A/0x0B 0xLL 0xHH.

– Odczyt pozycji ze sterownika: 0x90 0x0A/0x0B.

W powyższych komendach A oznacza obrót a B nachylenie (są to kanały sterownika).

4.3. Testy komunikacji

Komunikacja była testowana na układzie złożonym z:

– Komputera klasy PC.

– Płytki ZYBO.

– Komputera Raspberry Pi 2 Model B.

W układzie tym Raspberry zastępowało sterownik serwomechanizmów. Zamiana ta została wykonana, by mieć możliwość sprawdzania poprawności danych odbieranych z ZYNQ. W trakcie testów okazało się, że pierwszy bajt wysyłany do ZYBO jest wpisywany do bufora i wskaźnik jest przesuwany, lecz w kontrolerze przerwań nie jest zwiększany licznik ilości bajtów w buforze. W efekcie przerwanie od pełnego bufora danych przychodzących było wywoływane o 1 bajt za późno (po otrzymaniu pierwszego bajtu nowej komendy). Naprawiono to poprzez dodatkowy test w trakcie inicjalizacji komunikacji szeregowej. Test ten polegał na włączenie trybu loopback, wysłaniu i odebraniu kilku bajtów danych, a następnie zresetowaniu bufora. W trybie loopback dane wysyłane przez UART są wysyłane na jego wejście. W ten sposób można sprawdzać zgodność danych. Oprócz tego sprawdzono, czy wysłanie komendy z PC do ZYNQ skutkuje wysłaniem poprawnej komendy z ZYNQ do Maestro. Po naprawieniu opisanego problemu tego problemu i ponownym przeprowadzeniu testów stwierdzono, że komunikacja działa poprawnie.

Bibliografia

[1] Wikipedia. *Video Tracking*. *https://en.wikipedia.org/wiki/Video_tracking*. Dostęp: 12.10.2016.

[2] Nicole M. Artner. *A Comparison of Mean Shift Tracking Methods*. 2008.

[3] Gary R. Bradski. „Computer Vision Face Tracking For Use in a Perceptual User Interface". W: *Intel Technology Journal* (1998).

[4] Tomas Svoboda. *Kanade–Lucas–Tomasi Tracking*. *http://cmp.felk.cvut.cz/cmp/courses/Y33ROV/Y33ROV_ZS20082009/Lectures/Motion/klt.pdf*. Dostęp: 19.11.2016.

[5] Kris Kitani. *KLT Tracker*. *http://www.cs.cmu.edu/~16385/lectures/Lecture23.pdf*. Dostęp: 19.11.2016.

[6] *ZYBO FPGA Board Reference Manual*. Xilinx. 2016.

[7] *Pololu Maestro Servo Controller User's Guide*. Pololu. 2014.