AUTOMATYKA • 2008 • Tom 12 • Zeszyt 2

Artur Zawadzki*

Algorytm sterowizyjny wykorzystany do wyznaczenia orientacji palców dłoni i możliwości jego implementacji w układzie FPGA

1. Wprowadzenie

Poznanie orientacji, położenia i konfiguracji dłoni w przestrzeni trójwymiarowej wydaje się wielce pożądane. W wielu zastosowaniach możliwość sterowania procesem w sposób najbardziej dla człowieka naturalny, naśladujący wrodzone mechanizmy i sposoby manipulacji, jest sposobem najlepszym.

Jako jeden z przykładów można sobie wyobrazić zadanie sterowania ramieniem manipulatora. Do tego celu można wykorzystać sztuczne urządzenia sterujące, takie jak przeróżne dżojstiki, klawiatury oraz manetki. Możliwe jest również wyposażenie operatora w urządzenia skanujące wybrane jego ruchy, np. oczujnikowane rękawice czy ramiona odtwarzające ruch końcówki roboczej. Wszystkie te sposoby są jednak obarczone zasadniczą wadą – wymagają dużej ilości urządzeń wspomagających, czasami skomplikowanych mechanicznie, trudnych w obsłudze, nierzadko nieintuicyjnych. Nieomal zawsze wykorzystują jedynie część możliwości manualnych człowieka. Operator musi się dostosowywać do sterowanego urządzenia, a nie odwrotnie.

Lekarstwem na to może być zastosowanie bezdotykowej metody badania położenia istotnych fragmentów ciała przy wykorzystaniu informacji wizyjnej. Często spotykaną praktyczną realizacją tej idei jest analiza gestów. Zadanie to opiera się najczęściej na porównywaniu zarejestrowanego obrazu z obrazem generowanym na podstawie modelu. Możliwe jest wprost porównywanie obrazów jako map bitowych – wyznaczanie skal podobieństwa, porównywanie wyizolowanych fragmentów z modelem [2]. Spotykanym rozwiązaniem jest generowanie mapy bitowej na podstawie modelu, albo sporządzenie bazy kilkuset możliwych gestów. Nieco innym sposobem jest wyznaczenie cech zarejestrowanego obrazu obiektu i porównanie ich z cechami przewidywanymi przez model [4, 8].

Oba podejścia są jednak dość silnie zorientowane na konkretną analizowaną część ciała (np. dłoń) i nie posiadają zbyt wielu uniwersalnych metod analizy. Zastosowanie ich do

* Katedra Automatyki, Akademia Górniczo-Hutnicza w Krakowie

większego obszaru (np. ręki wraz z dłonią) powoduje znaczną komplikację metody, jeżeli chce się zachować funkcjonalność taką jak przy osobnej analizie każdej części.

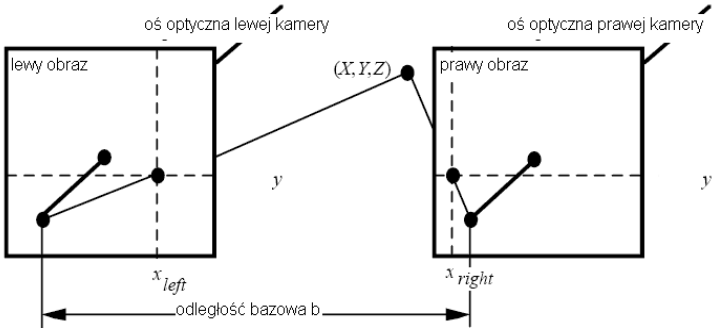
Z kolei uproszczenie metody skutkuje zazwyczaj koniecznością zmniejszenia ilości rozpatrywanych wariantów konfiguracji [9].

Zdecydowano się zatem przedstawić zastosowanie nieco innego sposobu pozyskania informacji o konfiguracji części ciała. Skupiono się na dłoni, jako na obiekcie posiadającym największe zastosowanie praktyczne w zadaniach manipulacyjnych. Postanowiono zastosować metody stereowizyjne, których wyniki działania, wśród szeregu zastosowań, można wykorzystać również w tym celu. Ponadto sposób akwizycji obrazu, opierający się na dwóch kamerach ustawionych w bliskim siebie sąsiedztwie, bez tworzenia specjalnie wydzielonego stanowiska, wydaje się najbardziej naturalnym sposobem odbioru informacji o pozycji użytkownika. Po programowej weryfikacji poprawności działania stworzonego algorytmu, zbadano możliwości implementacji sprzętowej w układzie FPGA.

2. Algorytm

2.1. Algorytm stereowizyjny Shirai

Stereowizja jest działem techniki zajmującym się przetwarzaniem obrazów rejestrowanych równocześnie przez dwie kamery w taki sposób, aby możliwe było uzyskanie informacji o głębi obrazu. Przez głębię obrazu należy rozumieć odległość obserwowanych obiektów od urządzeń rejestrujących obraz, a więc kamer systemu. Jako najprostsze wskazane jest ustawienie kamer w układzie kanonicznym (zwanym również standardowym). Cechą charakterystyczną tego układu jest to, że osie optyczne kamer są do siebie równoległe, a linie skanowania obu płaszczyzn obrazowych leżą na tych samych prostych. Standardowa stereogeometria dla dwóch kamer charakteryzuje się ponadto jednakowymi ogniskowymi obu kamer, w modelu kamery otworkowej (*pinhole camera*) [5].



Rys. 1. Układ kanoniczny

Zakłada się, że z lewą kamerą jest związany lewoskrętny, ortogonalny układ współrzędnych *XYZ*. Oś Z tego układu jest równoważna osi optycznej tej kamery. Punkt ogniskowy kamery leży na płaszczyźnie *XY*, płaszczyzna obrazowa *xy* leży równolegle do płasz-

czyzny *XY*. Ogniskowa jest w tym wypadku definiowana jako odległość pomiędzy punktem ogniskowym a płaszczyzną obrazową. Układ współrzędnych związany z prawą kamerą powstaje poprzez transpozycję o wektor (b, 0, 0), gdzie $b > 0$, zwany wektorem bazowym (*base distance*) (rys. 1).

Kamera otworkowa rzutuje punkt sceny P na płaszczyznę obrazową jako punkt p. Ponieważ wszystkie linie optyczne wychodzące z punktu P są dzielone przez dowolne dwie równoległe linie w tym samym stosunku (twierdzenie Talesa), współrzędne punktu p można wyrazić następująco:

$$p = (x, y) = (\frac{f \cdot X}{Z}, \frac{f \cdot Y}{Z}) \tag{1}$$

gdzie:

f – ogniskowa kamery,
x, y – współrzędne punktu p na płaszczyźnie obrazowej,
X, Y, Z – współrzędne punktu P.

Punkt przestrzeni $P(X, Y, Z)$ na płaszczyznach obrazowych kamer w standardowym stereowizyjnym układzie akwizycji ma współrzędne odpowiednio $P_{left} = (x_{left}, y_{left})$ oraz $P_{right} = (x_{right}, y_{right})$. Oczywiste jest, że punkt P musi być widoczny dla obu kamer, czyli nie może być przesłonięty przez inny obiekt (co jest dość częste w zagadnieniach stereowizyjnych). Jeżeli jako punkt odniesienia przyjmie się punkt obrazowy lewy $p_{left} = (x_{left}, y_{left})$, mianem dysparycji określać się będzie wektor:

$$\Delta(x_{left}, y_{left}) = (x_{left} - x_{right}, y_{left} - y_{right})^T \tag{2}$$

pomiędzy dwoma korespondującymi punktami obrazowymi $p_{left} = (x_{left}, y_{left})$ na lewym obrazie oraz $p_{right} = (x_{right}, y_{right})$ na prawym obrazie (jeśli istnieje).

W standardowej stereogeometrii można sformułować skalarną dysparycję:

$$\Delta ssg(x_{left}, y_{left}) = \sqrt{(x_{left} - x_{right})^2 + (y_{left} - y_{right})^2} \tag{3}$$

Równość współrzędnych $y_{left} = y_{right} = y$ wynika z cechy kanonicznego układu współrzędnych. Wówczas dla lewej kamery bezpośrednio z wzoru (1) [6] wynika:

$$x_{left} = \frac{f \cdot X}{Z} \tag{4}$$

$$y = \frac{f \cdot Y}{Z} \tag{5}$$

Dla prawej kamery, po uwzględnieniu przesunięcia o wektor bazowy *b* wzdłuż osi X, wzory przedstawiają się analogicznie:

$$x_{right} = \frac{f \cdot (X - b)}{Z} \tag{6}$$

$$y = \frac{f \cdot Y}{Z} \tag{7}$$

Z powodu identyczności współrzędnych *y* skalarna dysparycja (zwana również skalarną paralaksą – *scalar parallax*) dla pary korespondujących punktów jest równa:

$$\Delta ssg(x_{left}, y_{left}) = \mid x_{left} - x_{right} \mid = x_{left} - x_{right} \tag{8}$$

Warunek $x_{left} > x_{right}$ jest zawsze spełniony przy odpowiedniej aranżacji kamer (lewa-prawa). Różnica $x_{left} - x_{right}$, czyli miara dysparycji dla korespondujących punktów, pozwala zrekonstruować współrzędne punktu *P* w trójwymiarowym układzie współrzędnych *XYZ*, jeżeli parametry *f* oraz *b* są znane.

Dysparycja jest obliczana dla każdej korespondującej pary punktów i pozwala ocenić trójwymiarowe koordynaty rzutowanego punktu. Ponieważ współrzędne obrazowe są liczbami całkowitymi, więc i dysparycja jest liczbą całkowitą. W praktyce oznacza to, że dysparycja jest określona w relatywnie niewielkim zakresie liczb całkowitych. Warto zauważyć, że dla punktu *P* leżącego blisko dwóch kamer dysparycja jest duża i współrzędne przestrzenne *X*, *Y* i *Z* mogą być określone dokładnie. Zwiększenie dokładności wyznaczenia dystansu jest możliwe również poprzez zwiększenie wartości *b*. Jednakowoż zwiększenie *b* niesie za sobą czasem istotne zmniejszenie ilości punktów, które są widoczne przez obie kamery.

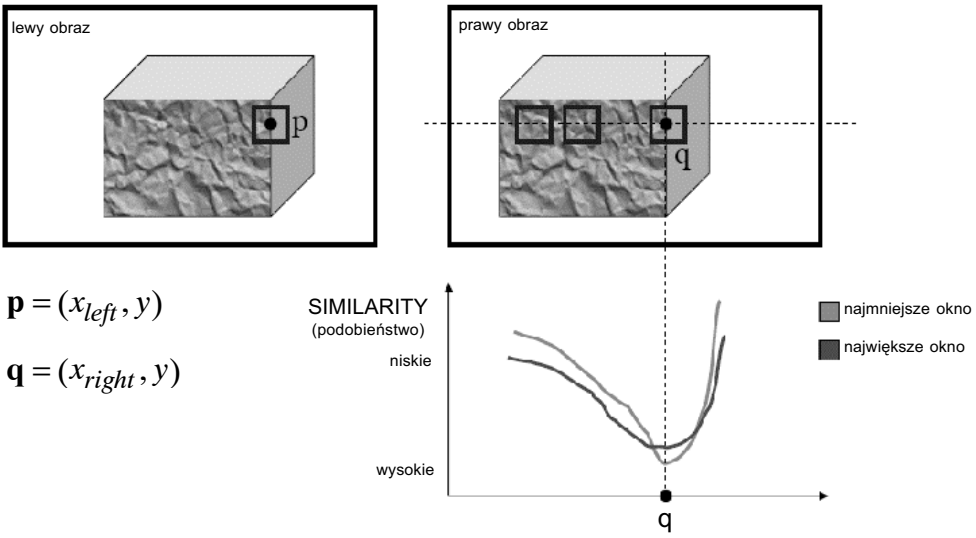
Ogniskowa *f* oraz wartość *b* mogą być wyrażone w takich jednostkach, w jakich jest rejestrowany obraz przez kamery. W efekcie wszystkie wymiary można wyrażać w szerokości piksela, co jest wygodne obliczeniowo.

Oczywiście, tylko te punkty, które są widoczne dla obu kamer, mogą być skorelowane i może być dla nich obliczona dysparycja. Drugim problemem jest tzw. problem korespondencji (*correspondence problem*), który ma miejsce w pewnych specyficznych okolicznościach. Najprostszym przykładem może być biała kartka papieru widziana przez lewą i prawą kamerę. Punkty korespondujące ze sobą mogą być określone wyłącznie dla krawędzi kartki, natomiast nie jest możliwe ich znalezienie w obszarze jednolitym.

Algorytm Shirai jest jednym z wielu algorytmów stereowizyjnych, służących do określenia tzw. rzadkiej mapy dysparycji, gdyż jako punkty dopasowania wybiera on wyłącznie punkty krawędziowe. Konieczne jest zatem wcześniejsze wydobycie krawędzi z jednego obrazu innym algorytmem.

Algorytm ten pracuje w kanonicznym (standardowym) układzie stereowizyjnym. Z tego powodu możliwe jest szukanie korespondujących ze sobą punktów w tych samych

wierszach $y = y_{left} = y_{right}$ (rys. 2). Dysparycja wyliczona przez ten algorytm może służyć do określenia współrzędnych X, Y i Z skorespondowanych punktów, pod warunkiem znajomości ogniskowej f kamer oraz odległości bazowej b (rys. 1).

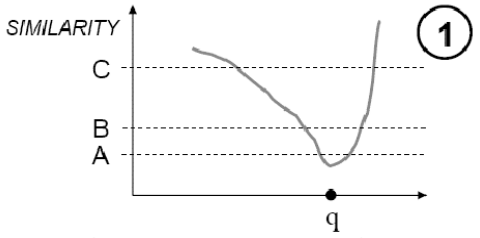


Rys. 2. Algorytm Shirai – wskaźnik *SIMILARITY* [7]

Przeszukiwanie obrazów odbywa się wzdłuż linii epipolarnych, w pewnym oknie, jak to przedstawia rysunek 2. Dla okien wyznaczane jest podobieństwo (*SIMILARITY*), pozwalające dobrać korespondujące ze sobą punkty.

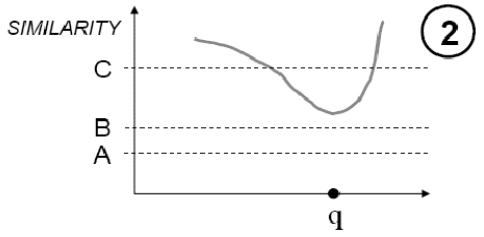
Algorytm (*Schlüns*) [7] zebrano w punktach:

1. Wybierz punkty w lewym obrazie, dla których chcemy znaleźć punkty korespondujące w prawym obrazie. Można zastosować np. wykrycie krawędzi algorytmem Canny.
2. Znajdź dla każdego punktu krawędziowego (znacznikowego) korespondujący z nim punkt w prawym obrazie:
   a) zdefiniuj największe okno wielkości $n = 2k + 1$ (początek od $k = 1$) wokół punktu p w lewym obrazie;
   b) zdefiniuj okno o wielkości n wokół każdego punktu kandydata z prawego obrazu q;
   c) oblicz dla każdej pary powyższych okien skalarną wielkość podobieństwa *SIMILARITY(p, q)*.
3. Jeżeli minimalna wartość podobieństwa jest mniejsza lub równa A (rys. 3), to odpowiadająca temu minimum współrzędna q koresponduje z punktem p. Przyporządkowana jest wartość względnej głębi. Idź do kroku 2.



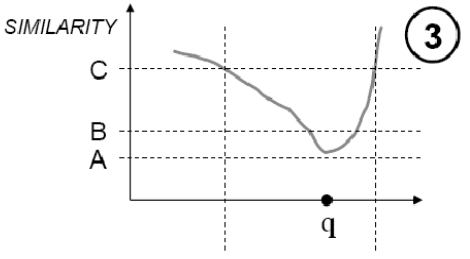
Rys. 3. Algorytm Shirai, przypadek 1 [7]

4. Jeżeli minimalna wartość podobieństwa jest większa od progu B (rys. 4), oznacza to, że nie można dla współrzędnej p znaleźć żadnego korespondującego punktu q. Idź do kroku 2.



Rys. 4. Algorytm Shirai, przypadek 2 [7]

5. Jeżeli zostanie osiągnięta maksymalna wielkość okna bez osiągnięcia warunków poprzednich, to nie można znaleźć punktów korespondujących. Idź do kroku 2.
6. Wylicz $k = k + 1$. Przejdź do kroku 2, uprzednio zmniejszając przeszukiwane obszary do wyznaczonego przez poziomy podobieństwa, które są mniejsze od C (rys. 5).



Rys. 5. Algorytm Shirai, przypadek 3 [7]

Można wyznaczyć wiele różnych miar podobieństwa. Y. Shirai w 1989 roku zaproponował dalej przedstawione wzory określające podobieństwo (ściślej, wzory te określają „niepodobieństwo", lecz ze względów historycznych pozostaje się przy nazwie *SIMILARITY*).

Dla okien o rozmiarze k, których środki określają współrzędne (x_{left}, y) oraz (x_{right}, y), liczony jest błąd kwadratowy (*square error*):

$$SE(p,q) = \sum_{i=-k}^{k} \sum_{h=-k}^{k} (E_{left}(x_{left}+i, y+j) - E_{right}(x_{right}+i, y+j))^2 \tag{9}$$

gdzie: $E_{left}(x, y)$, $E_{right}(x, y)$ – wartość piksela o współrzędnych (x, y) odpowiednio dla lewego i prawego obrazu.

Ponadto liczona jest estymowana wartość wariancji dla stałego (w danym kroku) lewego okna, oznaczonego przez p:

$$VAR(p) = \frac{1}{(2k+1)(2k+1)} \sum_{i=-k}^{k} \sum_{j=-k}^{k} [E_{left}(x_{left}+i, y+j) - AV(p)]^2 \tag{10}$$

co często zastępuje się prostszą formą:

$$VAR(p) = \frac{1}{(2k+1)(2k+1)} \sum_{i=-k}^{k} \sum_{j=-k}^{k} E_{left}(x_{left}+i, y+j)^2 - AV(p)^2 \tag{11}$$

Wartość $AV(p)$ określa średnią arytmetyczną dla okna lewego, czyli, w konsekwencji, estymuje oczekiwaną jasność w tym oknie.

Ostatecznie, „podobieństwo" Shirai jest definiowane następująco:

$$SIMILARITY(p,q) = \frac{SE(p,q)}{VAR(p)+1} \tag{12}$$

Jeżeli oba okna, z obrazu lewego i prawego, są identyczne, to wartość błędu kwadratowego wynosi zero. W każdym innym przypadku wartość powyższej funkcji jest dodatnia.

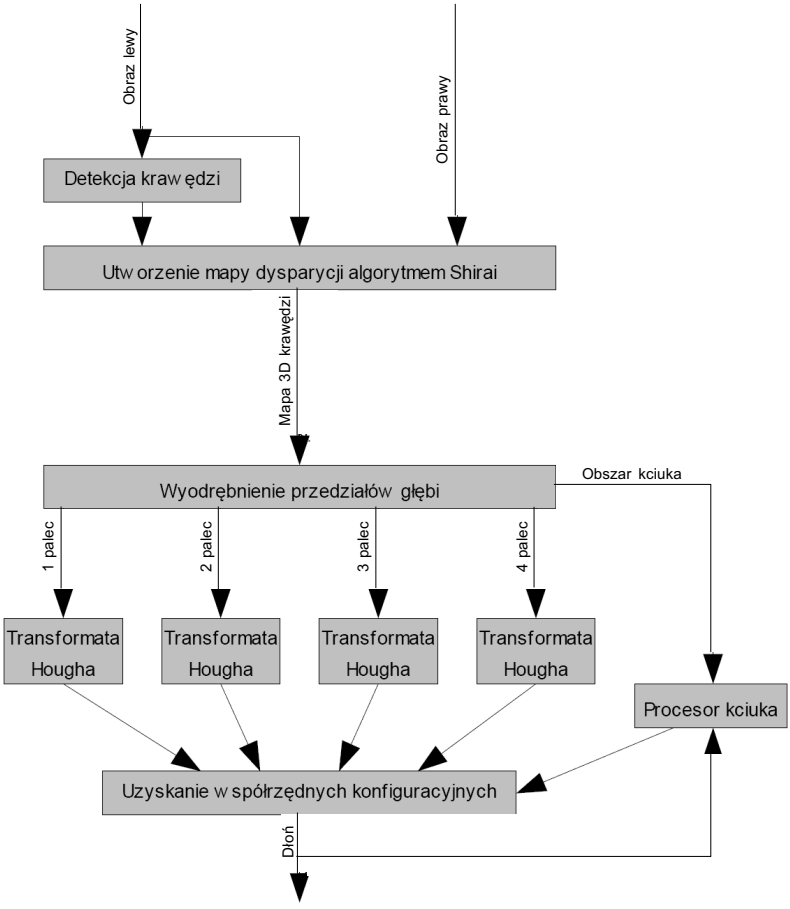
Pewną trudność w stosowaniu tego algorytmu stanowi znaczna liczba parametrów, które mają znaczący wpływ na uzyskiwaną w wyniku mapę dysparycji.

2.2. Algorytm wyznaczający orientację palców

Podstawowym założeniem, którego spełnienie jest wymagane do poprawnego działania algorytmu analizy położenia palców, przedstawionego dalej, jest zapewnienie właściwego ułożenia dłoni. Otóż dłoń powinna być ułożona w taki sposób, aby wszystkie palce z wyjątkiem kciuka pozostawały równoległe. Nie jest to trudne do osiągnięcia, gdyż w zasadzie inne ułożenie palców ze względów anatomicznych jest mało możliwe. Płaszczyzny tworzone przez zginające się palce powinny być równoległe do płaszczyzny obrazowej, jak to pokazano na rysunku 6.

Algorytm realizujący funkcje analizy dłoni został przedstawiony na rysunku 7 w formie schematu blokowego.

Rys. 6. Poprawne ułożenie dłoni



Rys. 7. Schemat blokowy algorytmu analizy dłoni

Algorytm główny składa się z kilku etapów. Najważniejszym z nich jest algorytm stereowizyjny Shirai. Operuje on na danych opisujących lewy i prawy obraz oraz wykrytych krawędziach obrazu lewego. Stąd też na schemacie blokowym obecny jest blok detekcji krawędzi.

Algorytm Shirai „produkuje" rzadką mapę dysparycji. Na wyjściu tego bloku pojawia się obraz o rozdzielczości obrazów wejściowych, w którym większość wykrytych punktów krawędziowych ma przyporządkowaną głębię. To, jak wiele punktów krawędziowych będzie posiadało przyporządkowaną wartość liczbową odpowiadającą głębi, zależy od skuteczności działania algorytmu Shirai. Uzasadnione jest przypuszczenie, że w przypadku przedstawienia do rejestracji samej dłoni na jednolitym tle, wykryte krawędzie będą przedstawiały krawędzie palców i korpusu dłoni. Przyporządkowanie im głębi pozwoli rozpoznać, które krawędzie dotyczą którego palca, lub szerzej – rejonu dłoni.

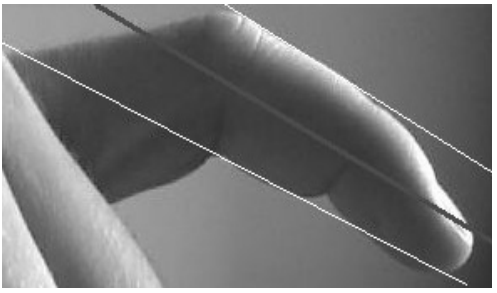
Wobec tego kolejnym etapem jest podział otrzymanej głębi na podprzedziały odpowiadające poszczególnym palcom. Dokonuje się tego na drodze analizy histogramu obrazu głębi. Po tym etapie otrzymuje się cztery podprzedziały odpowiadające dominującym głębiom.

Taki tok postępowania wynika z faktu, że krawędzie palców, zgodnie z założeniem, zorientowane są równolegle do płaszczyzn obrazowych kamer. Wobec tego dla pewnych głębi obserwuje się znaczną ilość punktów krawędziowych, odpowiadających poszczególnym palcom (i granicom korpusu dłoni). Dominujących maksimów, przy założeniu dobrego dopasowania dla każdego z palców, obserwuje się cztery, gdyż założone ustawienie dłoni wymusza granulację dysparycji.

Do wyznaczenia równania prostej pokrywającej się z wykrytą na obrazie krawędzią stosuje się transformatę Hougha [1, 10]. Metoda ta opiera swe działanie na transformacji pomiędzy przestrzenią kartezjańską a przestrzenią parametrów opisujących prostą (lub dowolną inną krzywą). Główną zaletą transformaty Hougha jest fakt, że punkty krawędziowe nie muszą tworzyć ciągłej linii na obrazie. Ta własność jest bardzo użyteczna przy próbie detekcji linii posiadających krótkie przerwy, zaszumionych. Każdy wyodrębniony palec poddawany jest transformacie Hougha, w wyniku czego otrzymuje się dla każdego z nich zbiór równań prostych, które, w założeniu, powinny pokrywać się z krawędziami poszczególnych członów palców. Na tym etapie uzyskuje się ponadto pominięcie obszarów o głębi wprawdzie zbieżnej z danym palcem, lecz nie tworzących linii prostych. Do takich obszarów można zaliczyć np. półkoliste zakończenie palców, fragmenty kciuka, a także fragmenty korpusu dłoni.

Na podstawie analizy równań prostych, a w szczególności wykryciu dla każdego podprzedziału prostych równoległych (boków tego samego członu palca), możliwe jest określenie współrzędnych konfiguracyjnych każdego palca, a w konsekwencji dłoni. Stwierdzono, że w przypadku palca nieprzysłoniętego przez palce znajdujące się bliżej kamery zazwyczaj wykrywane są obie krawędzie każdego członu palca. Proste leżące na tych krawędziach są dla każdego członu w przybliżeniu równoległe. Przyjmuje się zmienną miarę równoległości, czyli dopuszczalną „trójkątność" poszczególnych odcinków palców. Jeżeli

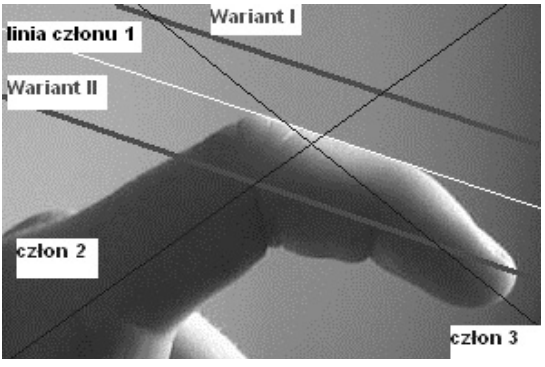
w zestawie wykrytych prostych dwie proste są równoległe, to przyjmuje się, że ograniczają one jeden człon palca. W takim wypadku tworzona jest zamiast nich trzecia prosta, równoległa do dwóch pierwotnych (w ramach skali równoległości), leżąca pomiędzy nimi w równej od nich odległości (rys. 8).



Rys. 8. Proste równoległe dla środkowego członu palca

Takich zestawów prostych równoległych powinno być trzy, gdyż palec posiada trzy odcinki. Dzięki wymaganiu dotyczącego istnienia prostych równoległych zazwyczaj pomijane są linie należące do brzegu dłoni, jeśli na etapie wyodrębniania podprzedziałów głębi nie zostały wcześniej odrzucone.

Bywa jednak, że zostanie wykryta prosta należąca jedynie do jednej krawędzi któregoś członu. W takim wypadku sprawdzane są dwa warianty – generowane są dwie proste, przesunięte równolegle w stosunku do pierwotnej o połowę średniej grubości palca (na podstawie analizy pozostałych członów lub pozostałych palców), każda w innym kierunku (rys. 9). Z dwóch wariantów wybiera się ten, który bardziej odpowiada modelowi palca. Jako kryterium przyjmuje się tutaj odległość między stawami dla członu środkowego, która powinna być o 30% dłuższa od odległości między stawami dla członu najbliższego korpusowi dłoni (dla dłoni autora).



Rys. 9. Wykryta jedna prosta dla środkowego członu palca

Ten sposób postępowania dla niektórych konfiguracji palców nie jest skuteczny. Jeżeli palec jest wyprostowany, czyli zostaną wykryte mniej niż trzy zestawy prostych (po 1 lub 2 proste równoległe w zestawie), to wykryte proste (zazwyczaj dwie równoległe) traktuje się jako konfigurację wszystkich trzech członów.

Jeżeli człon palca leżący najbliżej korpusu dłoni nie zostanie poprawnie wykryty, przeprowadzane są próby określenia jego orientacji na podstawie przewidywanej długości i położenia stawu należącego do korpusu. Przyjmuje się, że staw ten posiada w przybliżeniu te same współrzędne (x, y) dla każdego palca, co wynika z założonej pozycji dłoni względem kamer.

Może się zdarzyć, że palec zostanie częściowo lub nawet całkowicie przesłonięty przez palec leżący bliżej kamer. W takim przypadku nie zostanie wykryty poprzez analizę histogramu głębi i należy przyjąć jego konfigurację taką samą jak dla palca leżącego bliżej kamery. O braku palca (w wyniku niewykrycia lub przesłonięcia) zazwyczaj świadczy luka w histogramie.

Kciuk w toku tych operacji jest obiektem w zasadzie zbędnym. Można nawet powiedzieć, że szkodliwym z punktu widzenia możliwości uzyskania poprawnych wyników. Dlatego też traktowany jest odrębnie, a jego parametry obliczane powinny być przez osobny moduł – procesor kciuka.

W końcowym efekcie uzyskuje się dla każdego palca równania prostych pokrywających się z osiami odcinków palców oraz współrzędne (x, y) stawów ze wskazaniem, do którego odcinka należą.

3. Implementacja w FPGA

W tym etapie pracy algorytm analizy dłoni próbowano zaimplementować sprzętowo, przy użyciu programu napisanego w języku Handel-C, z wykorzystaniem bibliotek Pixel-Streams i układu FPGA, zainstalowanego na zestawie zawierającym odpowiedni zbiór wejść oraz wyjść wideo (karta Celoxica RC300).

W stosunku do imlementacji programowej dokonano kilku zmian. Przede wszystkim wszędzie, gdzie tylko było to możliwe, operacje mnożenia i dzielenia zredukowano do operacji z czynnikiem postaci 2^i, czyli do przesunięć bitowych w lewo lub prawo. Wynika to z faktu, że kompilator Handel-C dość nieefektywnie wykorzystuje wbudowane w układ Virtex II jednostki mnożące, o czym można się przekonać, analizując statystykę pokompilacyjną wykorzystania zasobów układu. Mimo obecności operacji mnożenia na dostosowanych do jednostek mnożących długościach słów (18 bitów) w pierwszych wersjach programu, jednostki te pozostawały w większości niewykorzystane, a do mnożeń tworzona była głęboka logika kombinacyjna. Operacje dzielenia możliwe są do wykonania tylko przy użyciu logiki kombinacyjnej, przy czym będzie ona wielowarstwowa [3], wykorzystując wiele bloków układu FPGA. Jest to niekorzystne ze względu na powstające duże opóźnienia. Lekarstwem jest zastąpienie mnożeń i dzieleń przesunięciem bitowym, o wiele szybszym i angażującym nieporównanie mniej zasobów sprzętowych.

W module realizującym algorytm Shirai napotkano szereg problemów imlementacyjnych. Końcowa wersja musiała być pozbawiona wielu rozwiązań przyspieszających, które miały zrównoleglać część obliczeń. Chodzi głównie o możliwości równoczesnego dostępu do wielu komórek pamięci w trakcie obliczania podobieństwa przy poszukiwaniu dopasowania. Taka konstrukcja jest możliwa do realizacji w układzie FPGA, stanowiąc jedną z głównych zalet tych układów w zagadnieniach przetwarzania obrazów. Jednak przy wielu warunkowo wykonywanych gałęziach realizujących równoczesny dostęp do pamięci budowana jest głęboka logika, uniemożliwiająca pracę z odpowiednio szybkim zegarem.

Ponadto należy zauważyć, że implementacja algorytmu w postaci tzw. filtru odbiega od standardów narzuconych przez bibliotekę PixelStreams. Przede wszystkim działa asynchronicznie – nie przetwarza jednego piksela w jednym cyklu zegara. Podstawowe opóźnienie (*latency*) wynosi 11 linii obrazu, magazynowanych w wewnętrznym buforze. Zatem przez czas 11 linii żaden piksel nie zostanie przekazany na wyjście. Po upływie tego czasu rozpoczyna pracę algorytm Shirai. Równocześnie z obliczeniami prowadzonymi na danej paczce pikseli (piksel lewy, prawy oraz krawędzie) pobierany jest następny punkt i zapisywany w buforze pod odpowiednią lokacją adresową.

Opóźnienie pomiędzy pobraniem pikseli z bufora a uzyskaniem wyniku zależy od zawartości obrazu. Jeżeli piksel nie jest krawędziowy, opóźnienie wynosi jeden cykl – na wyjście wysyłana jest wartość 0. Jeżeli piksel jest krawędziowy, opóźnienie jest niemożliwe do oszacowania, gdyż zależy od zawartości obrazów lewego i prawego. Dopasowanie może zajść szybko, zanim wielkość okna powiększy się do maksymalnego rozmiaru, lub wolno, gdy wartość dysparycji będzie duża i zostanie w dodatku wyliczona po wielokrotnym zwiększaniu wielkości okna przeszukiwań. Wynikiem może być wartość dysparycji lub zero, jeżeli proces dopasowania nie mógł zajść.

Przeprowadzono modyfikację oryginalnych dzielników w funkcjach obliczających podobieństwo. Zamiast dzielenia przez $(2k + 1)(2k + 1)$, gdzie k jest połową wielkości okna przeszukiwań, $0 < k < 6$, dzielenie następuje poprzez przesunięcia bitowe, począwszy od przesunięcia o 6 bitów dla $k = 1$. W efekcie zmieniają się dzielniki:

dla: $k = 1$ było 9, jest $2^3 = 8$;
$k = 2$ było 25, jest $2^4 = 16$;
$k = 3$ było 49, jest $2^5 = 32$;
$k = 4$ było 81, jest $2^6 = 64$;
$k = 5$ było 121, jest $2^7 = 128$.

Zmiana ta spowodowana była bardzo dużym opóźnieniem czasowym przy operacji dzielenia, wynoszącym 200 ns (przejście przez najdłuższą ścieżkę, zawierającą rozkaz dzielenia). Modyfikacje te nie wpłynęły, jak to sprawdzono w programie Matlab, na zdolność wyznaczania głębi.

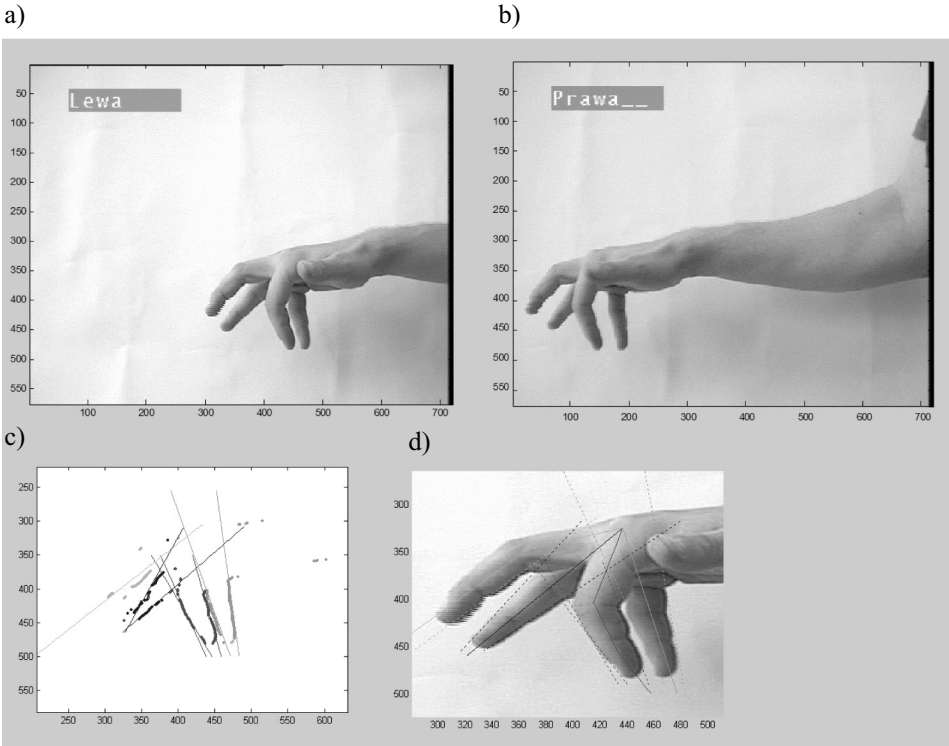
Próba implementacji sprzętowej przy użyciu języka programowania Handel-C dla układu Virtex 2 ujawniła szereg niedostosowań możliwości obecnie istniejących narzędzi programistycznych do struktury algorytmu. Złożone algorytmy, uzależniające swoje działanie od zawartości obrazu, głęboko zapętlone, nie są zbyt dobrze implementowalne w struk-

tury typu FPGA. Nadmierne opóźnienia czasowe powstawały często w zupełnie niewinnie wyglądających instrukcjach.

W wyniku szeregu zabiegów udało się zestawić poprawny tor wizyjny zawierający moduły algorytmu Shirai oraz transformaty Hougha, jednak nie osiągnięto bezwzględnej poprawności uzyskiwanych wyników. Podstawowym problemem okazało się zapewnienie powstawania na tyle krótkich ścieżek logicznych w strukturze FPGA, aby układ mógł pracować z wymaganą częstotliwością zegara. Zastosowane modyfikacje pozwoliły jednak zbliżyć się do poprawnej implementacji większości algorytmu w układzie FPGA.

4. Wyniki eksperymentalne

Przedstawione wyniki eksperymentalne dotyczą wersji algorytmu wyznaczania orientacji palców dłoni zaimlementowanej w środowisku Matlab.



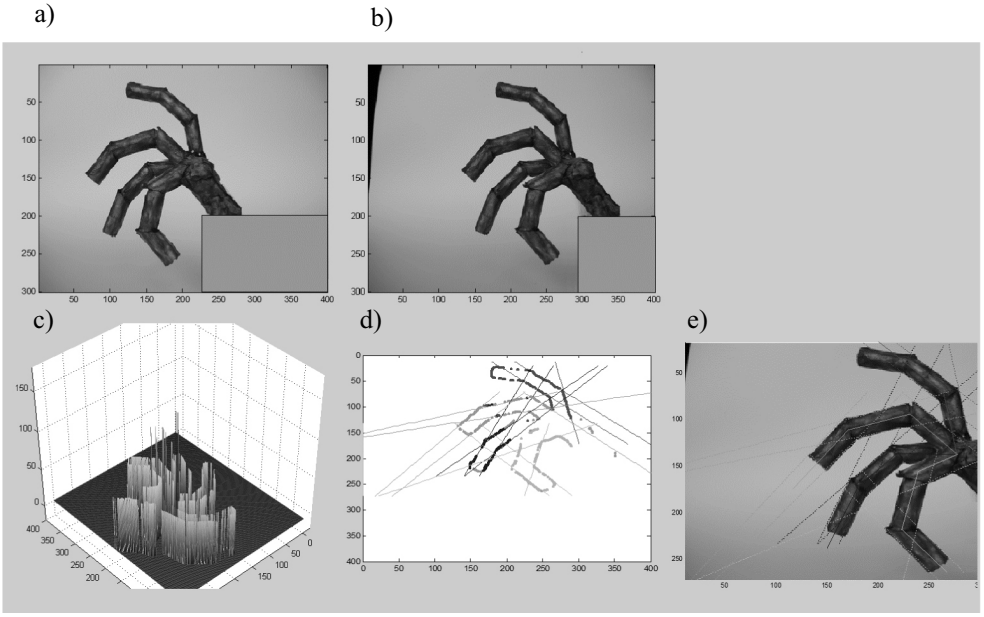
Rys. 10. Wyniki dla dłoni rzeczywistej: a) obraz z lewej kamery; b) obraz z prawej kamery; c) wykryte punkty krawędziwe z przyporządkowaną głębią; d) efekt końcowy z naniesionymi prostymi

Dla dłoni rzeczywistej (rys. 10) stosunkowo duża liczba punktów z przyporządkowaną głębią pozwoliła na właściwe oszacowanie przez program rzeczywistych konfiguracji palców. Na tym przykładzie daje się zaobserwować zarówno zastępowanie dwóch prostych

równoległych trzecią prostą (ze zmienną miarą równoległości), jak i przesuwanie niesparowanych prostych (szacowanie grubości i długości elementów palców), a także wyznaczenie punktu styku palców z korpusem dłoni (w zakładanej pozycji dłoni jest to jeden punkt). Wyznaczenie punktu wspólnego (przecięcie dwóch wyprostowanych palców) pozwoliło na wyliczenie prawdopodobnej pozycji fragmentów niedopasowanych przez algorytm Shirai. Uwzględniono tutaj spodziewaną długość członów palców w odniesieniu do ich grubości, wyznaczonej na podstawie wykrytych prostych równoległych.

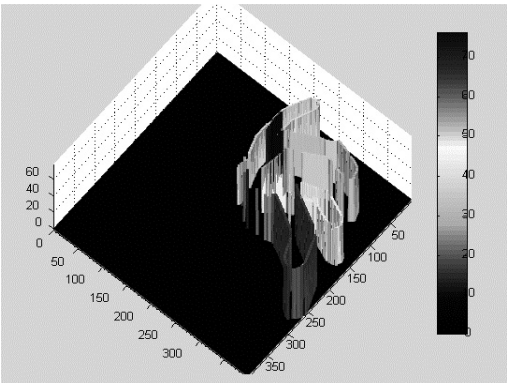
Należy zauważyć, że brak prostych równoległych uniemożliwiłby poprawne oszacowanie długości palców, a w konsekwencji również niemożliwe stałoby się poznanie pełnej konfiguracji dłoni.

Model dłoni (rys. 11) stanowi przypadek wyidealizowany, więc wykrycie poprawnej orientacji zidentyfikowanych fragmentów palców nie było tutaj trudne. Pojawiły się jednak pewne niejednoznaczności w procesie przesuwania prostych i dodawania brakujących członów. Było to spowodowane tym, że model nie spełniał w pełni anatomicznych zależności, przez co nie dało się go ustawić względem kamer w pozycji naturalnej dla prawdziwej dłoni. Wobec tego uzyskanie poprawnego efektu wymagało modyfikacji algorytmów dopasowania pozycji palców do punktu styku z korpusem dłoni. Widać jednak, że zgięcie jednego palca nie zostało poprawnie zidentyfikowane. Wynika to z faktu zbyt małej ilości dopasowanych przez algorytm Shirai punktów na ominiętym odcinku i niemożności wyznaczenia równania prostej na drodze transformaty Hougha.



Rys. 11. Wyniki dla modelu dłoni: a) obraz z lewej kamery; b) obraz z prawej kamery; c, d) wykryte punkty krawędziowe z przyporządkowaną głębią; e) efekt końcowy z naniesionymi prostymi

Jako uzupełnienie, na rysunku 12 przedstawiono trójwymiarowy wykres głębi dla jeszcze innego testowanego modelu dłoni.



Rys. 12. Przykład obliczonej głębi dla krawędzi modelu dłoni

5. Wnioski

Wykorzystanie algorytmu stereowizyjnego do rozważanego problemu jest rozwiązaniem elastycznym, stanowiącym dobrą bazę do dalszego rozwoju głównie w zagadnieniach sterowania manipulatorami. Zastosowanie układów FPGA wydaje się wielce pożądane ze względu na możliwość dużego przyspieszenia czasu obliczeń. Przeliczenie jednej stereopary przez implementację w programie Matlab zajmowało do trzech godzin dla obrazów w rozdzielczości PAL. Program napisany bezpośrednio w wybranym kompilowalnym języku programowania byłby oczywiście o wiele szybszy, lecz z pewnością nie umożliwiałby pracy w czasie rzeczywistym dla obrazów ruchomych. Tymczasem układy FPGA mogą pozwolić na zbliżenie się do tego typu pracy. Struktura algorytmu przedstawiona na rysunku 7 stwarza bowiem duże możliwości zrównoleglenia i spotokowania obliczeń. Jednakże realizacja tej idei w układach reprogramowalnych wymaga o wiele szerszych analiz, dla których ta praca stanowić może jedynie próbę częściowego uporania się z szeregiem trudności.

Wykorzystanie języka Handel-C wydaje się nie najlepszym rozwiązaniem w implementacji złożonych algorytmów, takich jak algorytm Shirai, głównie ze względu na ograniczony wpływ na powstające opóźnienia czasowe. Rozsądniejsze wydaje się wykorzystanie narzędzi dających większy wgląd w powstającą strukturę logiczną (VHDL).

Literatura

[1] Abu-Gharbieh R., Althoff K., Hamarneh G., *Project Report for the Computer VisionCourse, Automatic Line Detection*. 1999.

[2] Athitsos V., Sclaroff S., *Estimating 3D Hand Pose from a Cluttered Image*. IEEE Computer Society Conference, Computer Vision and Pattern Recognition, Boston, 2003.

[3] Celoxica Manual, *Handel-C Language Reference Manual*.
[4] Cohen I., Sung Uk Lee, *3D Hand Reconstruction from a Monocular View*. International Conference on Pattern Recognition, 2004.
[5] Cyganek B., *Komputerowe przetwarzanie obrazów trójwymiarowych*. Akademicka Oficyna Wydawnicza EXIT, 2002.
[6] Klette, Schlüns K., Koschan, *Computer Vision*. Singapur, 1998.
[7] Schlüns K., Hufnagl, *Bildverarbeitung*, *BinokulareStereoanalyse*. Wykład, rozdz. 12, 2005.
[8] Teng K., Ng J., Lim S., *Comp. Vision Based Sign Language Recognition for Numbers*. 2004.
[9] Verma S., Kofman J., Wu Xianghai, *Application of Markerless Image-Based Arm Tracking to Robot-Manipulator*. Canadian Conference Computer and Robot Vision, 2004.
[10] Żorski W., *Metody segmentacji obrazów oparte na transformacie Hougha*. Warszawa, Wojskowa Akademia Techniczna 2000.