BIULETYN INSTYTUTU AUTOMATYKI I ROBOTYKI
NR 28, 2010

Koncepcja rozpoznawania orientacji obiektów w obrazach 3D

Witold ŻORSKI

Instytut Teleinformatyki i Automatyki WAT,
ul. Gen. S. Kaliskiego 2, 00-908 Warszawa
wzorski@ita.wat.edu.pl

STRESZCZENIE: Artykuł porusza zagadnienie rozpoznawania orientacji obiektów 3D. Przystosowano opracowaną przez Ballarda metodę rozpoznawania obiektów nieregularnych. Podstawą rozpoznawania orientacji jest wyznaczanie wartości macierzy akumulatora dla kątów Eulera. Wartości akumulatora dla zadanych współrzędnych uzyskiwane są metodą zliczania voxeli[1]. Wartości kątów Eulera, dla których akumulator przyjmuje największą liczbę zliczeń, określają orientację badanego obiektu. Mechanizm zliczania voxeli został zaimplementowany i zbadany dla algorytmu bazującego na macierzy obrotu oraz algorytmu z zastosowaniem kwaternionów. Wykazano istnienie szczególnego rodzaju symetrii akumulatora, co pozwoliło na redukcję obliczeń o 50%.

SŁOWA KLUCZOWE: widzenie komputerowe, rozpoznawanie obrazów 3D, zliczanie voxeli, macierz obrotu, kwaterniony.

1. Wprowadzenie

Większość współczesnych systemów widzenia komputerowego działa w zakresie analizy i rozpoznawania obrazów 2D mimo rozpatrywania obiektów 3D [12], [18]. W ostatnim dziesięcioleciu, wraz z dokonującym się postępem w zakresie sprzętu i oprogramowania, zauważa się wzrost zainteresowania budową systemów widzenia komputerowego działających bezpośrednio na obrazach 3D [19]. Szeroki opis zagadnień odnoszących się do procesu pozyskiwania obrazów obiektów 2D i 3D można znaleźć w [17]. Szczególnie

[1] Ang. *voxel counting method* (*voxel - volumetric element, volumetric picture element*).

dużym zainteresowaniem cieszą się zastosowania związane z medycyną (np.: mammografia [6], automatyczna detekcja zmian nowotworowych [29], detekcja mikrozwapnień [30]), czego przyczynę można upatrywać w powszechności urządzeń medycznych pozyskujących obrazy 3D (MRI, SPECT, tomografia, ultrasonografia).

Odpowiednie narzędzia matematyczne, odnoszące się do obiektów w przestrzeni 3D, znane są już od bardzo dawna. Jednym z bardziej użytecznych jest (znana od 1917 roku) transformata Radona [20], której znaczenia dla współczesnej tomografii komputerowej nie sposób przecenić. Warto zauważyć, że transformata Hougha [9] dla krzywych analitycznych jest przypadkiem szczególnym [5] transformaty Radona.

Na potrzeby robotyki szeroko stosowane są macierzowe operatory przekształceń [3], bez których nie sposób wyobrazić sobie poruszania się po łańcuchu kinematycznym manipulatora czy choćby rozwiązania prostego zadania kinematyki.

Kolejnym narzędziem matematycznym są kwaterniony, znane od roku 1853, za sprawą prac Williama R. Hamiltona [8][2]. Wiele systemów współczesnej grafiki komputerowej działa przy ich użyciu. Doczekały się nawet implementacji sprzętowej we wszystkich kartach graficznych zgodnych[3] z DirectX 9, w komponencie Direct3D [28]. Podstawowe własności kwaternionów, które o tym zadecydowały są następujące:

- kwaterniony nie są czułe na zjawisko utraty stopnia swobody zwane „*gimbal lock*", ale wskazana jest ich normalizacja,
- reprezentowane są przez cztery liczby, a nie 9 jak w przypadku macierzy obrotu,
- zapewniają łatwe przejście od kątów Eulera do osi obrotu i odwrotnie,
- interpolacja dla dwóch kwaternionów jest łatwiejsza niż dla macierzy,
- normalizacja kwaternionów jest łatwiejsza niż ortogonalizacja macierzy (przypadek usuwania kumulacji narastających błędów wielu przeprowadzonych obliczeń),
- złożenie obrotów realizowane jest na drodze mnożenia (podobnie jak w macierzach).

Kwaterniony nie są jednak intuicyjne oraz są uciążliwe w wizualizacji, która wymaga konwersji do kątów Eulera lub postaci macierzowej (np. w przypadku stosowania OpenGL).

[2] Pozycja dostępna na Google Książki (Google Books).

[3] Obecnie (2010 rok) w najnowszych układach GPU nvidia i AMD/ATi jest już wspierany DirectX 11.

Kwaterniony zostały zaimplementowane w środowisku Matlab, jako element Aerospace Toolbox[4]. Zagadnienie kwaternionów nie jest szeroko i przystępnie rozpowszechnione w publikacjach książkowych i można wskazać ich zaledwie kilka: [2][2], [13], [16], [21].

W niniejszym opracowaniu rozważane jest zagadnienie rozpoznawania orientacji izolowanych obiektów 3D. Orientacja obiektu 3D definiowana jest jako wartości kątów Eulera, które jednoznacznie określają obrót układu współrzędnych przywiązany do obiektu, względem zadanego układu odniesienia. Zastosowane w artykule podejście można w pewnym sensie uznać za adaptację techniki zaproponowanej przez Ballarda [1], gdyż rzecz sprowadza się do zagadnienia zliczania głosów voxeli tworzących obiekt i wykorzystania przestrzeni parametrów zbudowanej na kątach Eulera. Analogiczne podejście stosował już autor w przypadku obiektów w obrazach 2D, w poziomach szarości i kolorowych, gdzie przestrzeń parametrów określana była dla operacji translacji, obrotu i (opcjonalnie) skalowania wzorca [22], [23], [26], [27].

Do przeprowadzenia odpowiednich badań zastosowano przekształcenia w 3D wykorzystujące:

1) macierz obrotu,
2) kwaterniony.

2. Macierz obrotu

Do prowadzenia rozważań nad wzajemnym położeniem obiektów w przestrzeni 3D stosuje się opisy układów współrzędnych przywiązywanych do tych obiektów. W celu przedstawienia podstawowych zagadnień zostanie tutaj zastosowana notacja zaczerpnięta z [3], gdzie można też znaleźć szeroką wykładnię zasygnalizowanych w tym punkcie zagadnień.

Dla jednoznacznego opisania orientacji układu współrzędnych $\{B\}$ względem układu $\{A\}$ wystarczy przedstawić wersory $\hat{X}_B, \hat{Y}_B, \hat{Z}_B$ układu $\{B\}$ w układzie $\{A\}$, co można zapisać za pomocą macierzy obrotu następująco:

$$ {}^{A}_{B}R = \left[{}^{A}\hat{X}_B \;\; {}^{A}\hat{Y}_B \;\; {}^{A}\hat{Z}_B \right] = \begin{bmatrix} \hat{X}_B\hat{X}_A & \hat{Y}_B\hat{X}_A & \hat{Z}_B\hat{X}_A \\ \hat{X}_B\hat{Y}_A & \hat{Y}_B\hat{Y}_A & \hat{Z}_B\hat{Y}_A \\ \hat{X}_B\hat{Z}_A & \hat{Y}_B\hat{Z}_A & \hat{Z}_B\hat{Z}_A \end{bmatrix} \tag{1} $$

gdzie elementami macierzy są wartości iloczynu skalarnego odpowiednich

[4] Nie jest jednak rzeczą trudną ich samodzielne zaimplementowanie w Matlabie.

wersorów. Rozważając teraz obrócone, ale współśrodkowe układy $\{A\}$ i $\{B\}$, oraz dany punkt ${}^{B}P$ z układu $\{B\}$, można łatwo wyznaczyć współrzędne punktu P w układzie $\{A\}$:

$$ {}^{A}P = {}^{A}_{B}R \cdot {}^{B}P \tag{2} $$

Wiersze macierzy obrotu ${}^{A}_{B}R$ są wersorami układu $\{A\}$ wyrażonymi w układzie $\{B\}$. Wiersze te są wektorami ortonormalnymi, a macierz obrotu jest ortogonalna:

$$ {}^{B}_{A}R = {}^{A}_{B}R^{T} = {}^{A}_{B}R^{-1}. \tag{3} $$

Do rozważań o charakterze praktycznym często stosuje się konwencję kątów Eulera $Z-Y-X$, zgodnie z którą układ $\{B\}$ wykonuje obrót względem układu $\{A\}$ kolejno o kąty: α (względem osi $\hat{Z}_A$), β (względem osi $\hat{Y}_A$) oraz γ (względem osi $\hat{X}_A$). Przyjmując tę konwencję oraz wykorzystując (1), otrzymujemy:

$$ {}^{A}_{B}R = {}^{A}_{B}R_{ZYX} = R_Z(\alpha) \cdot R_Y(\beta) \cdot R_X(\gamma) = $$

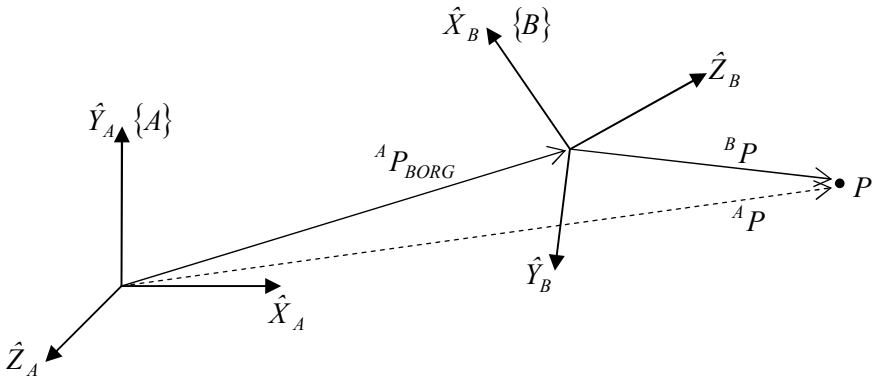
$$ = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\gamma & -\sin\gamma \\ 0 & \sin\gamma & \cos\gamma \end{bmatrix} = $$

$$ = \begin{bmatrix} \cos\alpha\cdot\cos\beta & \cos\alpha\cdot\sin\beta\cdot\sin\gamma - \sin\alpha\cdot\cos\gamma & \cos\alpha\cdot\sin\beta\cdot\cos\gamma + \sin\alpha\cdot\sin\gamma \\ \sin\alpha\cdot\cos\beta & \sin\alpha\cdot\sin\beta\cdot\sin\gamma + \cos\alpha\cdot\cos\gamma & \sin\alpha\cdot\sin\beta\cdot\cos\gamma - \cos\alpha\cdot\sin\gamma \\ -\sin\beta & \cos\beta\cdot\sin\gamma & \cos\beta\cdot\cos\gamma \end{bmatrix} \tag{4} $$

Dotychczasowe rozważania prowadzone były przy założeniu wspólnych środków układów $\{A\}$ i $\{B\}$. Włączenie do rozważań możliwości przesunięć układów współrzędnych (rys. 1) oznacza, że należy uwzględnić składnik ${}^{A}P_{BORG}$, który jest wektorem położenia początku układu $\{B\}$ w układzie $\{A\}$. Wzór (2) wymaga wówczas rozszerzenia do postaci:

$$ {}^{A}P = {}^{A}_{B}R \cdot {}^{B}P + {}^{A}P_{BORG} \tag{5} $$

Układ $\{B\}$ można zatem opisać względem układu $\{A\}$ następująco: $\{B\} = \left\{ {}^{A}_{B}R, {}^{A}P_{BORG} \right\}$.



Rys. 1. Ogólny przypadek dla obrotu i przesunięcia układu [3]

W praktyce unika się stosowania wzoru (5), a wszelkie obliczenia sprowadza się do wykorzystania liniowych operatorów przekształceń w przestrzeni o zwiększonym o 1 wymiarze. Wynik równoważny (5) uzyskuje się wykorzystując następującą macierz operatora:

$$\begin{bmatrix} {}^A P \\ 1 \end{bmatrix} = \begin{bmatrix} {}^A_B R & {}^A P_{BORG} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} {}^B P \\ 1 \end{bmatrix} \tag{6}$$

3. Kwaterniony

Kwaterniony, zaliczane do liczb hiperzespolonych, zostały wprowadzone przez Williama Hamiltona w połowie XIX wieku, na potrzeby mechaniki teoretycznej (w przestrzeni trójwymiarowej). Fundamentalne znaczenie ma (odkryta przez Hamiltona) formuła: $i^2 = j^2 = k^2 = ijk = -1$.

Stosuje się kilka postaci zapisu kwaternionu, ale najbardziej znana jest postać algebraiczna:

$$q = a + bi + cj + dk\,, \tag{7}$$

gdzie: $a,b,c,d \in R$; natomiast i, j, k są jednostkami urojonymi (kwaternionami bazowymi). Kwaternion sprzężony do (7) określany jest następująco:

$$\overline{q} = a - bi - cj - dk\,. \tag{8}$$

Moduł kwaternionu oblicza się ze wzoru:

$$|q| = \sqrt{a^2 + b^2 + c^2 + d^2}\,. \tag{9}$$

Zapis kwaternionu przy użyciu macierzowych postaci kwaternionów bazowych jest następujący:

$$q = a\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + b\begin{bmatrix} i & 0 \\ 0 & -i \end{bmatrix} + c\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} + d\begin{bmatrix} 0 & i \\ i & 0 \end{bmatrix} = \begin{bmatrix} a+bi & c+di \\ -c+di & a-bi \end{bmatrix} \tag{10}$$

Istnieje też zapis kwaternionu w postaci macierzy 4x4:

$$q = \begin{bmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{bmatrix} \tag{11}$$

Postać taką można uzyskać, rozważając mnożenie kwaternionu $q = a + bi + cj + dk$ przez kwaternion $p_1 + p_2 i + p_3 j + p_4 k$. Kwaternion $r_1 + r_2 i + r_3 j + r_4 k$, będący wynikiem takiego mnożenia, można wówczas wyznaczać następująco:

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{bmatrix} = \begin{bmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} \tag{12}$$

Poniższa tablica, zawierająca reguły mnożenia kwaternionów bazowych, jest pomocna przy mnożeniu kwaternionów w postaci algebraicznej:

	1	*i*	*j*	*k*
1	*1*	*i*	*j*	*k*
i	*i*	*-1*	*k*	*-j*
j	*j*	*-k*	*-1*	*i*
k	*k*	*j*	*-i*	*-1*

Na podstawie własności $q \cdot \overline{q} = |q|^2$ można określić kwaternion odwrotny do q następująco:

$$q^{-1} = \frac{1}{|q|^2}\overline{q}\,. \tag{13}$$

Z powyższego wzoru wynika natychmiast, że w przypadku kwaternionu jednostkowego (znormalizowanego) zachodzi $q^{-1} = \overline{q}$, co ma podstawowe znaczenie w przypadku stosowania kwaternionów do obrotów i zostanie dalej wykorzystane.

4. Kwaterniony w służbie obrotów

Niech w przestrzeni 3D zadany będzie obrót punktu (p_x, p_y, p_z), wokół osi wyznaczonej przez wektor $v = [v_x, v_y, v_z]$, prawoskrętnie o kąt φ. Dla wyznaczenia punktu uzyskiwanego w wyniku tak zadanego obrotu, w przypadku korzystania z kwaternionów, należy:

a) utworzyć kwaternion $p = 0 + p_x i + p_y j + p_z k$,

b) normalizując wektor v, wyznaczyć wektor jednostkowy $\hat{v} = [\hat{v}_x, \hat{v}_y, \hat{v}_z]$,

c) wyznaczyć kwaternion

$$q = \cos(\varphi/2) - (\hat{v}_x i + \hat{v}_y j + \hat{v}_z k)\sin(\varphi/2), \tag{14}$$

d) obliczyć

$$r = q \cdot p \cdot q^{-1}, \tag{15}$$

a z otrzymanego kwaternionu $r = 0 + r_x i + r_y j + r_z k$ odczytać wynik obrotu: (r_x, r_y, r_z).

Na podstawie przedstawionego rozwiązania można zaproponować „kwaternionową" alternatywę do obliczeń bazujących na macierzy obrotów (4) opartej o kąty Eulera. W tym celu należy:

a) potraktować wersory układu współrzędnych jako wektory dla obrotów składowych:

$$v_X = [1,0,0] \qquad v_Y = [0,1,0] \qquad v_Z = [0,0,1], \tag{16}$$

b) wyznaczyć odpowiednie kwaterniony:

$$\begin{cases} q_x = \cos(\gamma/2) - i\sin(\gamma/2) \\ q_y = \cos(\beta/2) - j\sin(\beta/2) \\ q_z = \cos(\alpha/2) - k\sin(\alpha/2) \end{cases} \tag{17}$$

c) obliczyć

$$r = q_z \cdot q_y \cdot q_x \cdot p \cdot q_x^{-1} \cdot q_y^{-1} \cdot q_z^{-1}. \tag{18}$$

Warto teraz bliżej rozpatrzyć powyższy wzór, gdyż bezpośrednia jego implementacja w programie komputerowym skutkowałaby czasochłonnymi obliczeniami. Przyjmijmy dla uproszczenia zapisu, że: $c_x = \cos(\gamma/2)$, $s_x = \sin(\gamma/2)$, $c_y = \cos(\beta/2)$, $s_y = \sin(\beta/2)$, $c_z = \cos(\alpha/2)$, $s_z = \sin(\alpha/2)$. Wykorzystamy fakt, iż kwaternion można przedstawić w postaci macierzy (11), a wyznaczenie iloczynu $q_L = q_z \cdot q_y \cdot q_x$ można przeprowadzić na drodze mnożeń odpowiednich macierzy (dla przejrzystości nie uwzględniono w nich elementów, które nie są kluczowe dla zapisu) następująco:

$$q_L = \begin{bmatrix} c_z & 0 & 0 & -s_z \\ 0 & c_z & -s_z & 0 \\ 0 & s_z & c_z & 0 \\ s_z & 0 & 0 & c_z \end{bmatrix} \begin{bmatrix} c_y & . & . & . \\ 0 & . & . & . \\ s_y & . & . & . \\ 0 & . & . & . \end{bmatrix} \begin{bmatrix} c_x & . & . & . \\ s_x & . & . & . \\ 0 & . & . & . \\ 0 & . & . & . \end{bmatrix} =$$

$$= \begin{bmatrix} c_y c_z & s_y s_z & -s_y c_z & -c_y s_z \\ -s_y s_z & c_y c_z & -c_y s_z & s_y c_z \\ s_y c_z & c_y s_z & c_y c_z & s_y s_z \\ c_y s_z & -s_y c_z & -s_y s_z & c_y c_z \end{bmatrix} \begin{bmatrix} c_x & . & . & . \\ s_x & . & . & . \\ 0 & . & . & . \\ 0 & . & . & . \end{bmatrix} = \begin{bmatrix} c_x c_y c_z + s_x s_y s_z & . & . & . \\ -c_x s_y s_z + s_x c_y c_z & . & . & . \\ c_x s_y c_z + s_x c_y s_z & . & . & . \\ c_x c_y s_z - s_x s_y c_z & . & . & . \end{bmatrix}$$

Na podstawie tego wyniku można przejść do następującej postaci algebraicznej wyniku mnożenia:

$$q_L = c_x c_y c_z + s_x s_y s_z + (-c_x s_y s_z + s_x c_y c_z) i + (c_x s_y c_z + s_x c_y s_z) j + (c_x c_y s_z - s_x s_y c_z) k \tag{19}$$

Postępując analogicznie w przypadku iloczynu $q_R = q_x^{-1} \cdot q_y^{-1} \cdot q_z^{-1}$ mamy:

$$q_R = \begin{bmatrix} c_x & s_x & 0 & 0 \\ -s_x & c_x & 0 & 0 \\ 0 & 0 & c_x & s_x \\ 0 & 0 & -s_x & c_x \end{bmatrix} \begin{bmatrix} c_y & . & . & . \\ 0 & . & . & . \\ -s_y & . & . & . \\ 0 & . & . & . \end{bmatrix} \begin{bmatrix} c_z & . & . & . \\ 0 & . & . & . \\ 0 & . & . & . \\ -s_z & . & . & . \end{bmatrix} =$$

$$= \begin{bmatrix} c_x c_y & s_x c_y & c_x s_y & -s_x s_y \\ -s_x c_y & c_x c_y & -s_x s_y & -c_x s_y \\ -c_x s_y & s_x s_y & c_x c_y & s_x c_y \\ s_x s_y & c_x s_y & -s_x c_y & c_x c_y \end{bmatrix} \begin{bmatrix} c_z & . & . & . \\ 0 & . & . & . \\ 0 & . & . & . \\ -s_z & . & . & . \end{bmatrix} = \begin{bmatrix} c_x c_y c_z + s_x s_y s_z & . & . & . \\ -s_x c_y c_z + c_x s_y s_z & . & . & . \\ -c_x s_y c_z - s_x c_y s_z & . & . & . \\ s_x s_y c_z - c_x c_y s_z & . & . & . \end{bmatrix}$$

skąd ostatecznie otrzymujemy:

$$q_R = c_x c_y c_z + s_x s_y s_z + (-s_x c_y c_z + c_x s_y s_z)i + (-c_x s_y c_z - s_x c_y s_z)j + (s_x s_y c_z - c_x c_y s_z)k \tag{20}$$

Można zauważyć, że kwaternion q_L jest nie tylko odwrotny do q_R, ale jest również do niego sprzężony, co wynika z tego, iż kwaterniony q_x, q_y, q_z są znormalizowane (widać to we wzorze 17). Daje to możliwość łatwego wyznaczenia kwaternionu q_R na podstawie q_L. Wyprowadzone wzory (19 i 20) będą stanowiły podstawę implementacji obrotów na kwaternionach, gdyż zwalniają z wykonywania mnożeń macierzy zawierających dużą liczbę elementów zerowych. Wzór (18) można teraz zapisać w postaci docelowej:

$$r = q_L \cdot p \cdot \bar{q}_L . \tag{21}$$

5. Rozpoznawanie orientacji obiektu

Jak już zostało wspomniane we wprowadzeniu, rozpoznawanie orientacji obiektu 3D będzie realizowane podobnie, jak to czynił Ballard w przypadku rozpoznawania kształtów [1], gdzie obiekty były reprezentowane przez listę pikseli tworzących krawędź, a przestrzeń parametrów zdefiniowana była dla translacji, obrotu i skalowania. Bezpośrednie uwzględnienie w przypadku obrazów 3D operacji translacji (3 współrzędne), obrotu (3 kąty Eulera) i skali (przynajmniej 1 parametr) pociągnęłoby za sobą iście horrendalne zapotrzebowanie na moc obliczeniową (czas obliczeń) i pamięć dla akumulatora. Na szczęście w wielu zagadnieniach problem skalowania nie występuje lub skala jest znana.

Założymy, że potrafimy izolować obiekt lub jego kontur w obrazie. Do izolowanego obiektu można przywiązać układ współrzędnych i następnie przystąpić do rozpoznania jego orientacji względem orientacji wzorca, do którego przywiązany jest układ odniesienia. W celu uproszczenia zagadnienia sprowadzamy rozważania do poziomu obrazów rastrowych binarnych, uznając wszystkie voxele obiektu za „aktywne" (a pozostałe za „pasywne"). Ograniczenie obiektu do jego konturu (tzn. utożsamienie obiektu z konturem) wpływa znacząco na skrócenie czasu obliczeń, bez większego uszczerbku dla wyniku końcowego.

Przed przystąpieniem do zdefiniowania przekształcenia umożliwiającego przejście od obrazu do przestrzeni parametrów wpierw przedstawimy podstawowe pojęcia. Rozważane będą **obrazy przestrzenne rastrowe binarne** tzn. takie, które są zbiorami rozróżnialnych punktów, gdzie każdy punkt jest

umownie czarny (aktywny) albo biały (pasywny). Obraz o takiej własności można przedstawić jako funkcję:

$$I : D \to \{0,1\}, \text{ gdzie } D = [-K,\ldots,K]\times[-L,\ldots,L]\times[-M,\ldots,M] \subset Z^3 . \quad (22)$$

Symbole K, L oraz M określają odpowiednio liczbę: wierszy, kolumn i warstw (dla głębokości) obrazu.

Przez **obiekt** w obrazie I będzie rozumiany zbiór tych voxeli $(x,y,z) \in D$, dla których funkcja $I(x,y,z)$ przyjmuje wartość 1, co można zapisać następująco:

$$o(I) = \{(x,y,z) \in D : I(x,y,z) = 1\} . \quad (23)$$

Przez **wzorzec** I_P rozumieć będziemy obraz przestrzenny rastrowy binarny, który został pozyskany na obiekcie rzeczywistym (ewentualnie poddany przetworzeniu) lub stworzony metodami grafiki komputerowej. Przyjmujemy, że wzorzec jest pamiętany w postaci listy voxeli aktywnych, co oznacza jego utożsamienie z $o(I_P)$.

Przed przystąpieniem do wyznaczania akumulatora należy przeliczyć współrzędne voxeli obiektu i wzorca względem układów współrzędnych przywiązanych do środków ich mas, co odpowiada ich przemieszczeniu w obrazach. W ten sposób możliwe będzie spełnienie założenia, że przywiązane układy współrzędnych mają wspólny środek i zadanie polega na rozpoznaniu orientacji układu współrzędnych przywiązanego do obiektu względem układu współrzędnych przywiązanego do wzorca, czyli wyznaczeniu kątów Eulera.

Funkcja $H(\alpha,\beta,\gamma)$, odpowiedzialna za proces akumulacji, dla obrazu $I(x,y,z)$ danego wzorem (22), jest zdefiniowana następująco:

$$H(\alpha,\beta,\gamma) = \sum_{i=1}^{n} h(x_i, y_i, z_i, \alpha, \beta, \gamma), \quad (24)$$

gdzie n jest liczbą voxeli aktywnych danego wzorca I_P, a funkcja podobieństwa ma postać:

$$h(x_i, y_i, z_i, \alpha, \beta, \gamma) = \begin{cases} 1 & dla\ (x_i', y_i', z_i') \in o(I) \\ 0 & w\ przeciwnym\ przypadku. \end{cases} \quad (25)$$

W przypadku stosowania macierzy obrotu zależność między współrzędnymi voxela $(x_i', y_i', z_i') \in I$, dla zadanych wartości parametrów α, β, γ i współrzędnymi voxela wzorca $(x_i, y_i, z_i) \in o(I_P)$, można zapisać następująco:

$$\begin{bmatrix} x_i' \\ y_i' \\ z_i' \end{bmatrix} = \begin{bmatrix} \cos\alpha\cdot\cos\beta & \cos\alpha\cdot\sin\beta\cdot\sin\gamma-\sin\alpha\cdot\cos\gamma & \cos\alpha\cdot\sin\beta\cdot\cos\gamma+\sin\alpha\cdot\sin\gamma \\ \sin\alpha\cdot\cos\beta & \sin\alpha\cdot\sin\beta\cdot\sin\gamma+\cos\alpha\cdot\cos\gamma & \sin\alpha\cdot\sin\beta\cdot\cos\gamma-\cos\alpha\cdot\sin\gamma \\ -\sin\beta & \cos\beta\cdot\sin\gamma & \cos\beta\cdot\cos\gamma \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} \tag{26}$$

W przypadku stosowania kwaternionów dla każdego voxela $(x_i, y_i, z_i) \in I_P$ należy zbudować odpowiedni kwaternion $p_i = 0 + x_i i + y_i j + z_i k$, a następnie wykorzystując (21) wyznaczyć kwaternion $r = 0 + x_i' i + y_i' j + z_i' k$, z którego można już odczytać współrzędne $(x_i', y_i', z_i') \in I$.

Zachodzi jeszcze konieczność określenia przestrzeni parametrów dla akumulatora. Oznaczając przez $\Delta\theta$ przyjęty kwant dla kątów α, β, γ, akumulator można zdefiniować następująco:

$$A : \{0, \ldots, \ell \cdot \Delta\theta\}^3 \rightarrow N , \tag{27}$$

gdzie $\ell = \dfrac{2\pi}{\Delta\theta}$.

Funkcja akumulacji $H(\alpha, \beta, \gamma)$ pozwala, dla zadanego wzorca I_P, na przeprowadzenie procesu akumulacji voxeli obiektu $o(I)$ w akumulatorze A. Po zakończeniu procesu akumulacji należy wyznaczyć współrzędne akumulatora $\alpha_0, \beta_0, \gamma_0$, dla których akumulator osiąga wartość największą[5], a tym samym pozyskać informację o orientacji danego obiektu.

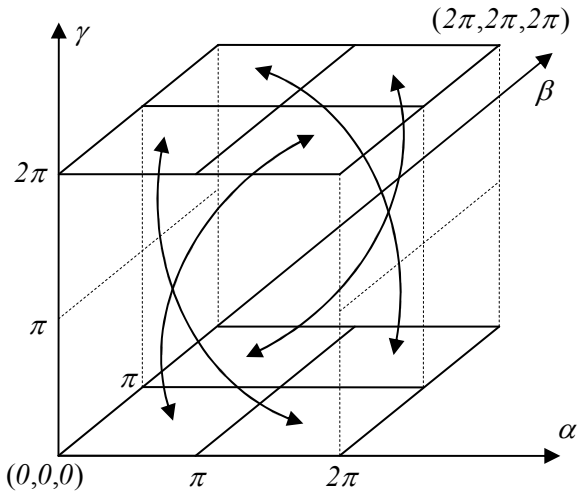
Można stosować niezależną kwantyzację dla każdego z kątów α, β, γ, ale nie ma ku temu szczególnych wskazań. Na potrzeby badań przyjęto kwant równy 2 stopniom miary kątowej, co w omawianej metodzie pozwoliło uprościć kod programu i skrócić czas obliczeń przy stosowaniu wzoru (17).

Okazuje się, że akumulator zbudowany na bazie kątów Eulera charakteryzuje się szczególnym rodzajem symetrii.. Dla rozważanego akumulatora symetrię tę można opisać następującą formułą matematyczną[6]:

$$A(\alpha, \beta, \gamma) = A(\alpha + \pi, -\beta + \pi, \gamma + \pi) , \tag{28}$$

[5] Akumulator może mieć kilka maksimów lokalnych o równych wartościach, co przykładowo może wynikać z symetrii badanego obiektu.

[6] Formułę łatwo zobrazować na przykładzie: obrót pióra względem każdej z trzech osi o kąt półpełny przywraca pióro do pozycji wyjściowej.



Rys. 2. Poglądowe wyobrażenie symetrii w akumulatorze (28)

przy czym należy pamiętać, iż wartości kątów są z przedziału $[0,2\pi)$ i wszędzie obliczane są jako *mod* 2π. Poprawność wzoru (28) można łatwo wykazać. W przypadku macierzy obrotu danej wzorem (4) wystarczą następujące podstawienia: $\cos(\alpha+\pi)=-\cos\alpha$, $\sin(\alpha+\pi)=-\sin\alpha$, $\cos(-\beta+\pi)=-\cos\beta$, $\sin(-\beta+\pi)=\sin\beta$, $\cos(\gamma+\pi)=-\cos\gamma$, $\sin(\gamma+\pi)=-\sin\gamma$, aby przekonać się, że macierz ta nie ulega zmianie. W przypadku kwaternionów, do wzorów (19) i (20) należy podstawić podobne tożsamości trygonometryczne, ale dla kątów połówkowych: $\frac{\alpha}{2}+\frac{\pi}{2}$, $\frac{-\beta}{2}+\frac{\pi}{2}$, $\frac{\gamma}{2}+\frac{\pi}{2}$, co wynika z (17).

Rys. 2 ma za zadanie ukazać wykrytą symetrię na drodze podziału całego akumulatora na 8 kubików i wskazania par, które zawierają takie same zbiory wartości (w celu zachowania przejrzystości narysowano tylko wybrane ściany kubików). Praktyczne wykorzystanie wskazanej symetrii można realizować, ograniczając wartości jednego wybranego kąta do połowy zakresu, co przyniesie 50% oszczędności na obliczeniach wartości akumulatora (drugą połowę akumulatora można ewentualnie przepisać, wykorzystując wzór (28)).

6. Uwagi do implementacji

Zwrócimy uwagę na pewne aspekty związane z implementacją omówionych metod w postaci programu komputerowego.

Pierwszą rzeczą, na którą można zwrócić uwagę w obu metodach, jest konieczność „bezustannego" liczenia wartości funkcji sinus i cosinus. Warto zatem dokonać (na etapie uruchomienia programu) tablicowania funkcji sinus i cosinus, np. co przyjęty we wzorze (27) kwant $\Delta\theta$, co znakomicie przyspieszy obliczenia.

Następną sprawą jest wybór podstawowego typu danych. Ze względu na charakter obliczeń musi to być typ rzeczywisty. Po przeprowadzeniu odpowiednich doświadczeń na typach rzeczywistych dostępnych w języku Delphi okazało się, że najkrótszy czas obliczeń uzyskano dla typu Single[7].

```
TVector = array[1..3] of Single;

function M_rotate(a,b,g:Integer; P: TVector): TVector; //a,b,g - angles
  var ca,sa,cb,sb,cg,sg: Single;                       //P=[x,y,z]
      P1,P2,P3: Single;
begin
  P1:=P[1]; P2:=P[2]; P3:=P[3];
  ca:=Tcos[a]; sa:=Tsin[a];
  cb:=Tcos[b]; sb:=Tsin[b];
  cg:=Tcos[g]; sg:=Tsin[g];
  M_rotate[1]:= ca*cb*P1 + (ca*sb*sg-sa*cg)*P2 + (ca*sb*cg+sa*sg)*P3;
  M_rotate[2]:= sa*cb*P1 + (sa*sb*sg+ca*cg)*P2 + (sa*sb*cg-ca*sg)*P3;
  M_rotate[3]:=-sb   *P1 +  cb*sg          *P2 +  cb*cg          *P3;
end;
```

Rys. 3. Przykładowa implementacja wzoru (26)

W przypadku stosowania macierzy obrotu należy skupić się na implementacji wzoru (26). Współczesne komputery PC zazwyczaj dysponują pamięcią RAM o pojemności ok. 2GB, co może zachęcić do „odważnej" próby stablicowania wszystkich możliwych macierzy obrotu dla przyjętego kwantu $\Delta\theta$. W przypadku przyjęcia $\Delta\theta = 2^{o}$ wymagana pamięć może być obliczona następująco: $180^3 \cdot 9 \cdot 4\mathrm{B} = 209952000\mathrm{B} \approx 200\mathrm{MB}$, co nie powinno sprawiać problemów związanych z zasobami współczesnego komputera PC. Okazuje się jednak, że w praktyce takie rozwiązanie nie sprawdza się, gdyż znacząco wzrasta nakład związany z adresowaniem takiej tablicy (4 wymiary) oraz należy

[7] Typ Single: zakres liczb (ze znakiem) od 1.5E-45 do 3.4E+38, 7-8 cyfr znaczących, zajmuje 4B.

pamiętać, że to system operacyjny przydziela zasoby i czasem część tak dużej tablicy może znaleźć się w pliku wymiany. Przykładowa implementacja wzoru (26) jest zamieszczona na rys. 3.

Przypadek zastosowania kwaternionów wydaje się ciekawszy. W rozważanym przypadku wystarczające jest zaimplementowanie dwóch operacji: mnożenia kwaternionów wyrażonego wzorem (12) oraz obrotu wyrażonego wzorem (21). Przykładowa implementacja tych operacji została przedstawiona na rys. 4.

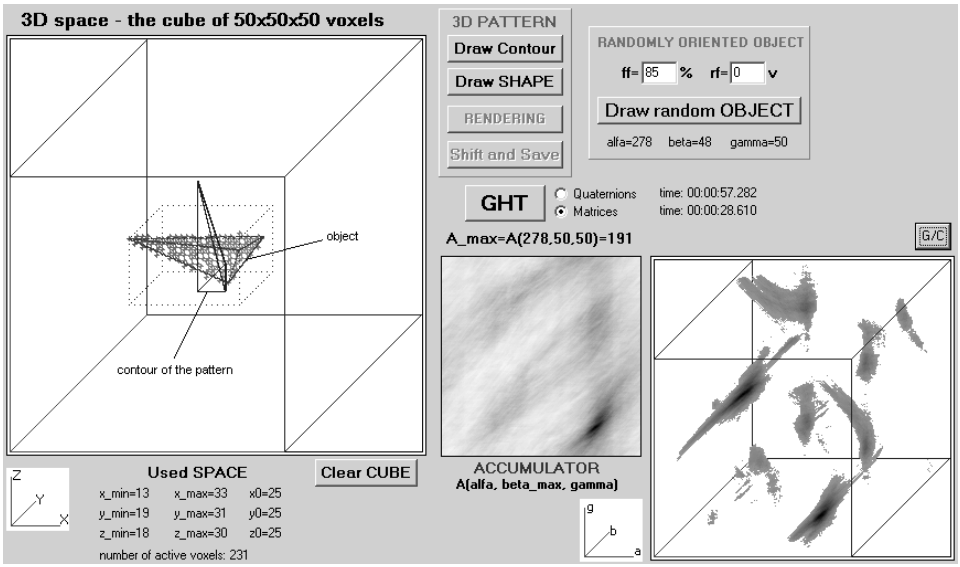
```
TQuaternion = record
                a,b,c,d: Single;
              end;

function Q_mult(p,q: TQuaternion): TQuaternion;
begin
  Q_mult.a:=p.a*q.a-p.b*q.b-p.c*q.c-p.d*q.d;
  Q_mult.b:=p.b*q.a+p.a*q.b-p.d*q.c+p.c*q.d;
  Q_mult.c:=p.c*q.a+p.d*q.b+p.a*q.c-p.b*q.d;
  Q_mult.d:=p.d*q.a-p.c*q.b+p.b*q.c+p.a*q.d;
end;

function Q_rotate(a,b,g:Integer; p: TQuaternion): TQuaternion;
  var qL,qR: TQuaternion;                    //a,b,g - angles; p=[0,x,y,z]
      cx,cy,cz,sx,sy,sz: Single;
begin
  cx:=Tcos[g]; cy:=Tcos[b]; cz:=Tcos[a];
  sx:=Tsin[g]; sy:=Tsin[b]; sz:=Tsin[a];
  qL.a:=  cx*cy*cz+sx*sy*sz;     qR.a:=qL.a;   //qR.a:= cx*cy*cz+sx*sy*sz;
  qL.b:= -cx*sy*sz+sx*cy*cz;     qR.b:=-qL.b;  //qR.b:=-sx*cy*cz+cx*sy*sz;
  qL.c:=  cx*sy*cz+sx*cy*sz;     qR.c:=-qL.c;  //qR.c:=-cx*sy*cz-sx*cy*sz;
  qL.d:=  cx*cy*sz-sx*sy*cz;     qR.d:=-qL.d;  //qR.d:= sx*sy*cz-cx*cy*sz;
  Q_rotate_FAST:=Q_mult(qL,Q_mult(p,qR));
end;
```

Rys. 4. Przykładowa implementacja przy zastosowaniu kwaternionów

7. Wyniki badań

Dla przeprowadzenia podstawowych badań, w zakresie stosowania macierzy obrotów i kwaternionów do rozpoznawania orientacji obiektów w obrazach 3D, napisane zostało odpowiednie oprogramowanie w środowisku programistycznym Borland Delphi 7 Personal. Istotne jest nadmienić, iż wykorzystaną platformą systemową był Windows XP, a platformą sprzętową komputer PC z procesorem Core 2 Duo E8400 3GHz oraz 2GB pamięci.

Stworzona aplikacja umożliwia m.in.: wrysowanie do pustego obrazu 3D (50x50x50 voxeli) obiektu (o losowej orientacji, z zadanym współczynnikiem wypełnienia i rozmycia powierzchni), przesunięcie obiektu do środka obrazu (odpowiednik przywiązania układu współrzędnych do środka masy) oraz obliczenie i zobrazowanie akumulatora dla zadanego wzorca (dla wyników obliczeń w dwóch przypadkach: z wykorzystaniem macierzy obrotu, przy zastosowaniu kwaternionów).

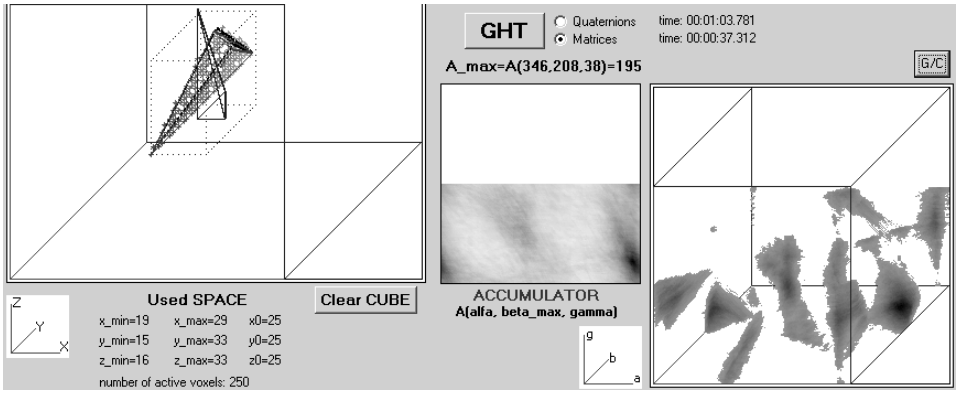


Rys. 5. Przykładowy wynik działania aplikacji

Na rys. 5 ukazano przykładowy wynik działania aplikacji. Akumulator prezentowany jest w postaci 3D (na drodze renderingu) oraz jako przekrój 2D przez maksimum[8] akumulatora (dla ustalonego kąta β_0). Wyraźnie widać charakterystyczną symetrię akumulatora, którą opisuje wzór (27), a poglądowo ilustruje rys. 2. Rys. 6 zamieszczono w celu ukazania połowy akumulatora, która jest rzeczywiście obliczana.

Czas wykonywania obliczeń w przypadku wykorzystania kwaternionów był w przybliżeniu dwukrotnie dłuższy niż w przypadku zastosowania macierzy obrotu, co jest ukazane w zaprezentowanych przykładach.

[8] Akumulator przyjmuje największą wartość dla $\alpha_0, \beta_0, \gamma_0$.



Rys. 6. Przykładowy wynik ukazujący rzeczywiście obliczaną połowę akumulatora

8. Podsumowanie

W przedstawionych algorytmach wyznaczania orientacji można stosować zarówno operacje bazujące na macierzach obrotu, jak i operacje bazujące na kwaternionach. Z punktu widzenia otrzymanych wyników rozpoznawania stosowanie kwaternionów było równoważne stosowaniu macierzy obrotu. Opracowanie i zaimplementowanie metody rozpoznawania orientacji obiektów w przestrzeni 3D z użyciem macierzy obrotu okazało się zdecydowanie łatwiejsze, niż z użyciem kwaternionów. Czas obliczeń w przypadku korzystania z macierzy obrotu był dwukrotnie krótszy niż w przypadku kwaternionów. Na taki wynik wpłynęło m.in. to, że opracowane algorytmy poszukiwały rozwiązania w postaci wartości kątów Eulera.

Podstawowym i chyba jedynym problemem zastosowania techniki Ballarda do rozpoznawania orientacji jest czas obliczeń, wynoszący kilkadziesiąt sekund, który nie jest akceptowalny w wielu systemach widzenia komputerowego. Z pomocą może przyjść tutaj technologia CUDA[9] firmy NVIDIA lub technologia Stream firmy AMD/ATi, dzięki którym można wykorzystać setki procesorów strumieniowych (ang. *SPU - Streaming Processing Unit*) współczesnych układów GPU, aby dziesiątki lub setki razy szybciej wykonać obliczenia niż w przypadku stosowania CPU. Należy tutaj zaznaczyć, że konieczna jest wówczas dekompozycja zadania na niezależne obliczeniowo fragmenty, co w omawianym przypadku jest stosunkowo łatwe.

[9] Compute Unified Device Architecture - www.nvidia.com

Literatura

[1] BALLARD D. H., *Generalizing the Hough Transform to Detect Arbitrary Shapes.* Readings in Computer Vision: Issues, Problems, Principles, and Paradigms. Los Altos, CA. 1987, pp. 714-725.

[2] CONWAY J. H., SMITH D. A., *On Quaternions and Octonions: Their Geometry, Arithmetic, and Symmetry*. Department of Mathematics University of California, 2004.

[3] CRAIG J. J., *Introduction to Robotics: Mechanics and Control*. Prentice Hall, 2005.

[4] DAVIES E. R., *Machine Vision: Theory, Algorithms, Practicalities*. Academic Press Ltd, 24/28 Oval Road, London NW1 7DX, United Kingdom 1990.

[5] DEANS S. R., *Hough transform from the Radon transform*. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vol. 3, No. 2, 1981, pp. 185-188.

[6] DZWINEL W., BORYCZKO K., ARODŹ T., KURDZIEL M., *Komputerowe metody detekcji nowotworów piersi w zdjęciach mammograficznych*. Bio-Algorithms and Med-Systems, Vol. 1, No. 1/2, 2005, pp. 287-290. http://www.bams.cm-uj.krakow.pl/bams_pdf/287-290_dzwinel.pdf

[7] FU K. S., GONZALEZ R. C., LEE C. S. G., *ROBOTICS: Control, Sensing, Vision, and Intelligence.* McGraw-Hill, New York 1987.

[8] HAMILTON W. R., Sir., *Lectures on quaternions*. Royal Irish Academy, 1853.

[9] HOUGH P. V. C., *Method and means for recognizing complex patterns*. U.S. Patent 3,069,654, Dec. 18, 1962.

[10] ILLINGWORTH J., KITTLER J., *A survey of the Hough Transform*. Computer Vision, Graphics and Image Processing 44, 1988, pp. 87-116.

[11] JAIN A. K., *Fundamentals of Digital Image Processing.* Prentice-Hall, New Jersey 1989.

[12] KANEZAKI A., NAKAYAMA H., HARADA T., KUNIYOSHI Y., *High-speed 3D Object Recognition using Additive Features in a Linear Subspace*. IEEE International Conference on Robotics and Automation (ICRA 2010), pp.3128-3134, 2010. http://www.isi.imi.i.u-tokyo.ac.jp/~kanezaki/ICRA2010WeD116.pdf

[13] KUIPERS J. B., *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*. Princeton University Press, 2002.

[14] LEAVERS V. F., *Shape Detection in Computer Vision Using the Hough Transform*. Springer, London 1992.

[15] LI H., LAVIN M. A., LEMASTER R. J., *Fast Hough transform*. Proceedings of the Third Workshop on Computer Vision: Representation and Control (Bellaire, MI, October 13-16, 1985), IEEE Publ. 85CH2248-3, pp. 75-83.

[16] MCAULAY A., *Utility of Quaternions in Physics*. London, 1893. http://www.gutenberg.org/ebooks/26262

[17] MOKRZYCKI W. S., *Wprowadzenie do przetwarzania informacji wizualnej*. Wydawnictwo EXIT, 2010.

[18] NAKAMURA K., ARIMURA K., YOSHIKAWA T., *Recognition of object orientation and shape by a rotation spreading associative neural network*. Neural Networks 2001: Proceedings of IJCNN '01, Vol.1, 2001, pp. 565-570.

[19] PEDERSEN P.C., QUARTARARO J.D., SZABO T.L., *Segmentation of speckle-reduced 3D medical ultrasound images*. IEEE International Ultrasonics Symposium (IUS), 2008, pp. 361-366.

[20] RADON J., *Uber die Bestimmung von Funktionen durch ihre Integralwerte langs gewisser Mannigfaltigkeiten*. Berichte Sachsische Akademie der Wissenschaften Leipzig, Math Phys Kl., 69, 1917, pp. 262-267.

[21] SIMON L. A., *Rotations, Quaternions, and Double Groups*. Oxford University Press, 1986.

[22] ŻORSKI W., FOXON B., BLACKLEDGE J., TURNER M., *Irregular Pattern Recognition Using the Hough Transform*. Machine Graphics & Vision, 9, 2000, pp. 609-632.

[23] ŻORSKI W., *Application of the Hough Technique for Irregular Pattern Recognition to a Robot Monitoring System*. Proceedings of the 11th IEEE International Conference MMAR 2005, pp. 725-730.

[24] ŻORSKI W., *The Hough Transform Application Including Its Hardware Implementation*. Advanced Concepts for Intelligent Vision Systems: Proceedings of the 7th International Conference, Lecture Notes in Computer Science, Springer-Verlag Vol. 3708/2005, pp. 460-467. http://www.springerlink.com/content/50yk3q0fw71x1qld

[25] ŻORSKI W., *Fast Hough Transform Based on 3D Image Space Division*. Advanced Concepts for Intelligent Vision Systems: Proceedings of the 8th International Conference, Lecture Notes in Computer Science, Springer-Verlag Vol. 4179/2006, pp. 1071-1079. http://www.springerlink.com/content/6216256332x1p166

[26] ŻORSKI W., *Unknown scale objects recognition*. Biuletyn WAT (652), 4/2008, pp. 197-207.

[27] ŻORSKI W., SAMSEL P., *Segmentacja obrazów kolorowych wzorcami nieregularnymi*. Biuletyn ITA (26), 2009, pp.45-64. http://www.ita.wat.edu.pl/

Źródła elektroniczne

[28] *Vectors, Vertices, and Quaternions (Direct3D 9)*. MSDN Library, Build date: 9/14/2010 http://msdn.microsoft.com/en-us/library/bb206327(VS.85).aspx

[29] BATOR M., *Automatyczna detekcja zmian nowotworowych w obrazach mammograficznych z wykorzystaniem dopasowania wzorców i wybranych narzędzi sztucznej inteligencji*. Rozprawa doktorska, Warszawa 2008. http://www.ippt.gov.pl/_download/doktoraty/Bator_doktorat.pdf

[30] WRÓBLEWSKA A, PRZELASKOWSKI A., *System automatycznej detekcji i klasyfikacji mikrozwapnień w cyfrowej mammografii*. Warszawa 2003. http://www.ire.pw.edu.pl/~arturp/Publikacje/Elektronizacja3_2003.pdf

A conception of object orientation recognition in 3D images

ABSTRACT: This paper considers the problem of 3D object orientation recognition. The Ballard method of arbitrary shapes detection is adopted. The basis of the orientation recognition is the mapping of an accumulator array for Euler angles. Accumulator values for given coordinates are calculated using the voxel counting method. An object orientation is determined by Euler angles with the maximum number of votes in the accumulator array. The voxel counting method was implemented and verified for an algorithm based on a rotation matrix as well as for an algorithm based on quaternions. A characteristic kind of accumulator symmetry was detected, which reduced computations by 50%.

KEYWORDS: computer vision, 3D image recognition, voxel counting, rotation matrix, quaternions

Praca wpłynęła do redakcji: 20.10.2010.