Modelowanie, sterowanie i wizualizacja quadrocoptera

Tomasz Liecau, Michał Warkocz, Krzysztof Wąsik, Michał Grochowski
Wydział Elektrotechniki i Automatyki, Politechnika Gdańska

Streszczenie: Artykuł przedstawia podejście do budowy modelu matematycznego quadrocoptera. Głównym celem budowy modelu było zaprojektowanie odpowiedniego sterowania obiektu oraz analiza jego zachowania się w różnych sytuacjach. Jako założenie przyjęto budowę modelu, systemu sterowania oraz wszelkich towarzyszących algorytmów w otwartym środowisku programistycznym, co pozwoli na późniejszą ich implementację w rzeczywistym obiekcie, bez konieczności stosowania drogiego oprogramowania. Sterowanie quadrocopterem przez operatora odbywa się przy pomocy ruchów dłoni sczytywanych przez kamerę i odpowiednio interpretowanych przy pomocy zaawansowanych metod przetwarzania obrazów. Całość systemu zwizualizowana jest i osadzona w trójwymiarowym środowisku symulacyjnym.

Słowa kluczowe: modelowanie, sterowanie, przetwarzanie obrazów, wizualizacja, quadrocopter

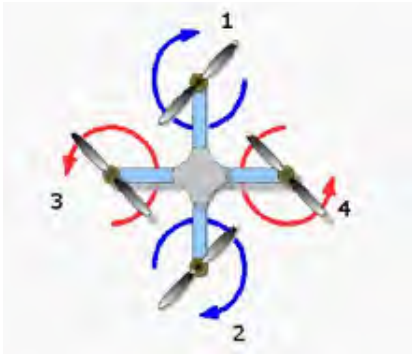
1. Wprowadzenie

Obecny stan techniki powoduje coraz większe zainteresowanie technologiami i urządzeniami, które z racji swego zaawansowania technologicznego były do tej pory, głównie ze względów finansowych, nieopłacalne do wdrożenia. Jednym z takich urządzeń jest quadrocopter. Jego wykorzystanie rozciąga się od zastosowań hobbistycznych, poprzez dydaktykę (modelowanie matematyczne, przetwarzania sygnałów, sterowanie, elektronika i wiele innych), kończąc na zastosowaniach komercyjnych (zbieranie danych meteorologicznych, fotografowanie terenu, odnajdywanie – śledzenie obiektów). Dobór efektywnych algorytmów sterowania quadrocopterem wymaga budowy jego modelu osadzonego w odpowiednim dla założonych celów środowisku symulacyjnym. Środowisko takie powinno umożliwiać stosunkowo szybką budowę modelu, implementację algorytmów sterowania oraz symulację w różnych warunkach. Ze względów komercyjnych zastosowane oprogramowanie powinno być to typu open source, bezpłatne, szybkie i niezawodne, umożliwiające ewentualną późniejszą komercjalizację. W opisywanym projekcie przyjęto, iż sterowanie quadrocopterem przez operatora będzie odbywało się za pomocą predefiniowanych gestów dłoni sczytywanych przez kamerę. Prototypowanie modelu oraz systemu jego sterowania zostało wykonane w środowisku Matlab, natomiast końcowa postać całego środowiska została napisana w C++ (model, sterowanie), przy użyciu bibliotek OpenCV (przetwarzanie obrazu) oraz OpenGL, MySQL, winAPI (wizualizacja, archiwizacja danych). W sekcji 2 przedstawiono najważniejsze aspekty modelu matematycznego quadrocoptera, w sekcji 3 przedstawiono założenia i podstawowe algorytmy sterowania położeniem oraz kątami quadrocoptera, w sekcji 4 pokazano sposób sterowania modelem przy pomocy dłoni operatora, natomiast w sekcji 5 opisano użyte narzędzia programistyczne. Sekcja 6 podsumowuje całość artykułu.

Projekt jest efektem realizacji przez Autorów pracy inżynierskiej na Wydziale Elektrotechniki i Automatyki na Politechnice Gdańskiej.

2. Model matematyczny quadrocoptera

Modelowany quadrokopter posiada 4 silniki prądu stałego o stałym wzbudzeniu magnesami trwałymi, każdy o ciągu przekraczającym 1 kg, wymiary: długość 50 cm, wysokość 15 cm, waga 1,25 kg. Rys. 1 przedstawia szkic poglądowy quadrocoptera.



Rys. 1. Szkic poglądowy quadrocoptera
Fig. 1. An illustrative sketch of quadrocopter

Model matematyczny każdego z silników opisują równania (1-2):

$$i(kT) = i((k-1)T) + T \cdot \{(u_t(kT) - i((k-1)T)R + c \cdot \phi \cdot \omega((k-1)T)) / L \quad (1)$$

$$\omega(kT) = \omega((k-1)T) + T \cdot \{(c \cdot \phi \cdot i(kT) + (m_{const} + m_{pow}) \cdot \omega((k-1)T)) / J \quad (2)$$

gdzie:

m_{const} – moment oporowy konstrukcyjny,
m_{pow} – moment oporowy powietrza,
c – stała konstrukcyjna,
ϕ – wartość strumienia wzbudzenia,
ω – prędkość kątowa wirnika,
i – prąd twornika,
J – bezwładność wirnika,
u_t – napięcie twornika,
R – rezystancja zastępcza twornika,
L – indukcyjność obwody twornika,
K – kolejny krok,
T – krok dyskretyzacji.

Przybliżony model stanu akumulatora opisuje (3):

$$B = \max\left(\frac{100}{u_x p * 3600} \cdot \sum_{k=0}^{n}\{u_z \cdot p \cdot 3600 - \sum_{j=1}^{j=4}(u_j(kT) \cdot i_j(kT)\}, 0\right) \quad (3)$$

gdzie:
B – stan akumulatora [%],
u_z – nominalne napięcie zasilania akumulatora,
p – pojemność akumulatora [Ah],
u_j – napięcie podawane na silnik z indeksem j,
i_j – prąd twornika silnika z indeksem j,

W modelu przyjęto sprawność akumulatora równą 100 % oraz założono stałe napięcie zasilania podawane przez akumulator.

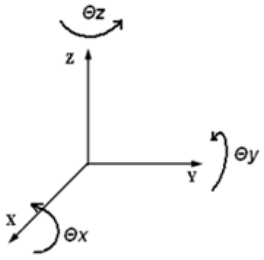
Model siły nośnej quadrocoptera [1] przedstawia (4):

$$P_z(kT) = C_z \cdot \rho \cdot skok \cdot (R \cdot \omega(kT))^2 / 2 \quad (4)$$

gdzie:
p_z – siła nośna,
c_z – współczynnik siły nośnej,
ρ – gęstość powietrza,
R – promień śmigła,
ω – prędkość wirnika,
skok – parametr śmigła – jego skok.

Obliczanie kątów obrotu modelu

W projekcie zdecydowano się na orientację kątów względem osi wokół której następuje obrót zgodnie z rys. 2.



Rys. 2. Kąty obrotu względem osi układu odniesienia
Fig. 2. Angles of rotation relative to the axis reference system

Równania pozwalające na wyznaczenie obrotu względem osi Z dla modelu dyskretnego przedstawiają (5-6):

$$\omega_z(kT) = \omega_z((k-1)T) + (\varepsilon_1(kT)J_s + \varepsilon_2(kT)J_s - \varepsilon_3(kT)J_s - \varepsilon_4(kT)J / J_z \cdot T \quad (5)$$

$$\theta_z(kT) = \theta_z((k-1)T) + \omega_z(kT) \cdot T \quad (6)$$

gdzie:
$\theta_z(kT)$ –kąt obrotu względem osi Z w chwili kT,
$\omega_z(kT)$ – prędkość obrotu względem osi Z w chwili kT,
J_s – moment bezwładności zespołu śmigło – wirnik,
J_z – moment bezwładności układu względem osi Z,
$\varepsilon_z(kT)$ – przyspieszenie obrotowe silnika z indeksem j

Obrót względem osi Y opisuje (7-8):

$$\omega_y(kT) = \omega_y((k-1)T) + \{[(f_1(kT) - f_2(kT)l - D \cdot m_0 \cdot g \cdot \sin(\theta_x(kT)) - T_{pow} \cdot \omega_y(kT) \cdot l] / J_y\} \cdot T \quad (7)$$

$$\theta_y(kT) = \theta_y((k-1)T) + \omega(kT) \cdot T \quad (8)$$

gdzie:
J_y - moment bezwładności modelu względem osi Y,
$f_1(kT), f_2(kT)$ - ciągi generowane przez silniki 1 i 2,
l – długość ramienia,
D – odległość środka ciężkości od środka aerodynamicznego,
m_0 – masa zastępcza modelu,
g – przyspieszenie grawitacyjne,
$\omega_y(kT)$ – prędkość kątowa wokół osi Y w chwili kT,
$\theta_x(kT)$ – kąt obrotu wokół osi X w chwili kT,
T_{pow} – opór aerodynamiczny powietrza.

Obrót względem osi X przedstawia (9-10):

$$\omega_x(kT) = \omega_x((k-1)T) + \{[(f_3(kT) - f_4(kT)l - D \cdot m_0 \cdot g \cdot \sin(\theta_x(kT) \cdot \cos(\theta_y(kT)) - T_{pow} \cdot \omega_x(kT) \cdot l] / J_x\} \cdot T$$
$$\theta_x(kT) = \theta_x((k-1)T) + \omega_x(kT) \cdot T \quad (9\text{-}10)$$

Wyznaczenie proporcji ciągów względem osi X, Y, Z

Równanie (11) pokazuje sposób wyznaczenia współczynników, oznaczających jaka część całkowitego ciągu działa na wybraną oś ruchu, na podstawie macierzy transformacji wyznaczonych z macierzy rotacji [4]:

$$F^{rzecz} = \begin{bmatrix} \cos\theta_z \sin\theta_y \cos\theta_x + \sin\theta_z \sin\theta_x \\ \sin\theta_z \sin\theta_y \cos\theta_x - \sin\theta_x \cos\theta_z \\ \cos\theta_y \cos\theta_x \end{bmatrix} F_s \vec{v} = \begin{bmatrix} K_x \\ K_y \\ K_z \end{bmatrix} F_s \vec{v} \quad (11)$$

gdzie:
K_x, K_y, K_z są współczynnikami podającymi procentowo jaka część całkowitego ciągu oddziałuje w danej osi.

Wyznaczenie położenia quadrocoptera w przestrzeni trójwymiarowej

Siły działające w osi Z:
– ciąg silników: $K_z(f_1(kT) + f_2(kT) + f_3(kT) + f_4(kT))$
– przyciąganie ziemskie: $g \cdot m_c$
– siła bezwładności: $a_z(kT) \cdot m_c$
– opory powietrza: $T_{pow} \cdot v_z(kT)$
– siła wiatru: $F_{wz} \cdot (kT)$

Na podstawie powyższych sił można wyznaczyć położenie i prędkość względem osi Z (12-13):

$$v_z(kT) = v_z((k-1)T) + \{(K_z(f_1(kT) + f_2(kT) + f_3(kT) + f_4(kT)) - T_{pow} v_z((k-1)T) - F_{wz}(kT)) / m_c - g\} \cdot T \quad (12)$$

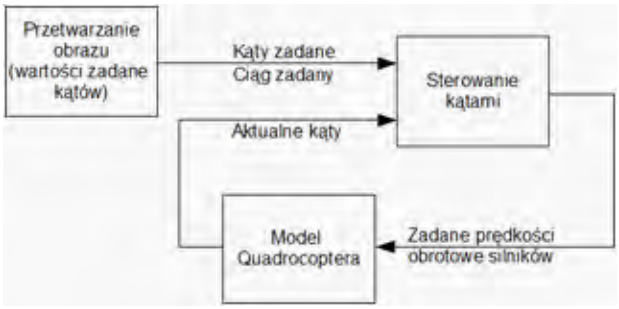
$$z(kT) = z((k-1)T) + v_z(kT) \cdot T \quad (13)$$

gdzie:
$v_z(kT)$ – prędkość względem osi Z w chwili kT,
$z(kT)$ – położenie w osi Z w chwili kT,
$f_{1-4}(kT)$ – siły ciągów silników od 1 do 4 w chwili kT,
m – całkowita masa modelu,
$F_{wz}(kT)$ – siła wiatru w osi Z.

Siły działające w osi X są analogiczne jak w osi Y (odpowiednio indeksy z należy zamienić na x), z wyjątkiem występującej w osi Z przyspieszenia ziemskiego g.

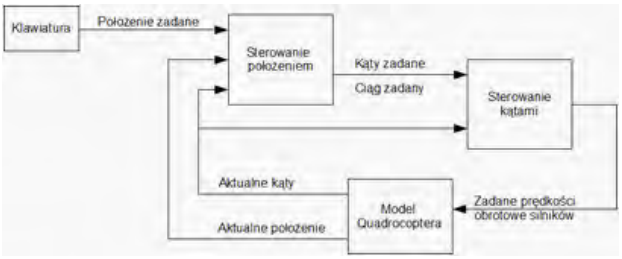
3. Sterowanie

W artykule rozważane są dwa tryby sterowania. Tryb pierwszy, w którym operator przy pomocy ruchów dłoni zadaje kąty obrotu we wszystkich osiach oraz ciąg. Schemat działania systemu sterowania w tym trybie pokazano na Rys. 3.



Rys. 3. Schemat sterowania kątami obrotu
Fig. 3. Scheme of circulation angles control

W drugim trybie quadrocopter przeprowadzany jest z aktualnej pozycji w pozycję zadaną, wprowadzaną przez operatora z klawiatury, a system wykorzystuje tryb pierwszy sterowania. Schemat działania systemu sterowania w tym trybie pokazano na rys. 4.



Rys. 4. Schemat sterowania położeniem
Fig. 4. Scheme of position control

3.1. Sterowanie kątami obrotu (tryb I)

Podstawowym założeniem układu sterowania kątami jest szybkie osiąganie przez model zadanych kątów obrotu względem osi X, Y, Z zgodnie z rys. 2.

Sygnały wejściowe:

- aktualne i zadane kąty obrotu względem osi X, Y, Z,
- zadany ciąg.

Sygnały wyjściowe:

- zadana prędkość obrotowa silników 1,2,3,4.

Podstawowe problemy sterownia kątami obrotu quadrocoptera to bardzo wysoka nieliniowość, wysoki rząd inercji oraz duża dynamika obiektu.

W projekcie systemu sterowania zdecydowano się na sterowanie za pomocą aktualnego uchybu regulacji w każdej z osi. Pominięto informacje o aktualnych wartościach kątów obrotu względem układu odniesienia. Równania (14-17) opisują sposób sterowania kątami obrotu modelu:

$$(obroty_2)^2 - (obroty_1)^2 = K_1 \int_0^t uchyb_y dt + K_5 \frac{d\ uchyb_y}{dt} + K_6 uchyb_y \tag{14}$$

Jak wynika z (4), siła nośna rośnie z kwadratem prędkość obrotowej śmigieł, natomiast z (7) można wywnioskować, że główną siłą, która odpowiada za obrót względem osi X jest różnica sił nośnych silników 1 i 2. Charakterystyka sterowania różnicą kwadratów prędkości obrotowych pozwala na zastosowanie regulatorów PID.

Analogicznie postępujemy z osią X (15):

$$(obroty_4)^2 - (obroty_3)^2 = K_1 \int_0^t uchyb_x dt + K_5 \frac{d\ uchyb_x}{dt} + K_6 uchyb_x \tag{15}$$

$$obroty_1 + obroty_2 - obroty_3 - obroty_4 = K_3 uchyb_z \tag{16}$$

gdzie:
$obroty_{1-4}$ – prędkości obrotowe zadane dla silników 1-4,
$uchyb_{x,y,z}$ – uchyb regulacji kąta obrotu w osi X, Y, Z,
K_{1-6} – współczynniki wyznaczane osobno dla danego modelu.

Jak wynika z (5), obrót względem osi Z jest zależny od sumy prędkości obrotowych silników 1 i 2 i różnicy prędkości obrotowych silników 3 i 4. Ponieważ wymuszenie równe zero dla obrotu względem osi Z powoduje utrzymanie zadanego kąta wystarczający do sterowania kątem obrotu względem osi Z jest regulator P.

$$obroty_1 + obroty_2 + obroty_3 + obroty_4 = K_4 ciag_{zadany} \tag{17}$$

gdzie: $ciag_{zadany}$ – ciąg zadany.

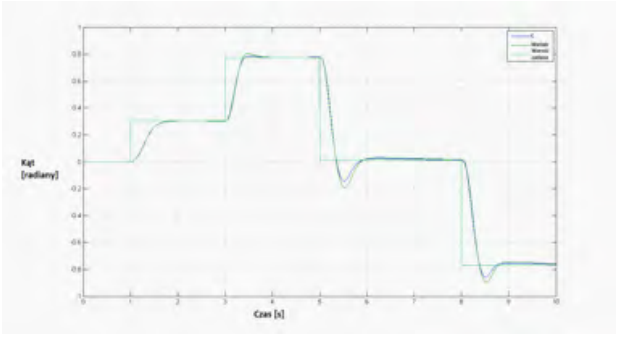
Równania (14-16) pozwalają na wyznaczenie proporcji pomiędzy zadanymi prędkościami obrotowymi dla silników 1-4. Równanie (17) pozwala na uzależnienie tych proporcji od całkowitego zadanego ciągu, przez co pozwala na końcowe wyznaczenie konkretnych wartości dla zadanych prędkości obrotowych, które są wejściami do modelu quadrokoptera. Przyjmując (18):

$$\begin{aligned} a &= K_1 \int_0^t uchyb_x dt + K_5 \frac{d\ uchyb_x}{dt} + K_6 uchyb_x \\ b &= K_1 \int_0^t uchyb_x dt + K_5 \frac{d\ uchyb_x}{dt} + K_6 uchyb_x \\ c &= K_3 uchyb_z \\ d &= K_4 ciag_{zadany} \end{aligned} \tag{18}$$

i rozwiązując układ równań (14-17), otrzymujemy równania (19) pozwalające na sterowania prędkością obrotową silników prowadzące do odpowiedniej zmiany kątów obrotu quadrocoptera:

$$\begin{aligned} obroty_1 &= \frac{c^2 + 2cd + d^2 + 4a}{4(c+d)} \\ obroty_3 &= -\frac{c^2 - 2cd + d^2 + 4b}{4(c-d)} \\ obroty_3 &= -\frac{c^2 - 2cd + d^2 - 4b}{4(c-d)} \end{aligned} \tag{19}$$

Stabilizacja prędkości obrotowych przez silniki modelu wykorzystuje regulatory PI z filtrem przeciwnasyceniowym (antywindup). Wejściem do regulatora są uchyby prędkości obrotowych silników a jego wyjście stanowią napięcia sterujące odpowiednimi silnikami. Na rys. 5 przedstawiono jakość sterowania kątami obrotu przez model quadrocoptera zaimplementowany zarówno w Matlabie, jak i bezpośrednio w C. Wyniki są porównywalne.



Rys. 5. Nadążanie modelu za zmianami wartości zadanej kąta
Fig. 5. Following the changing angle set point by the model

3.2. Sterowanie położeniem (tryb II)

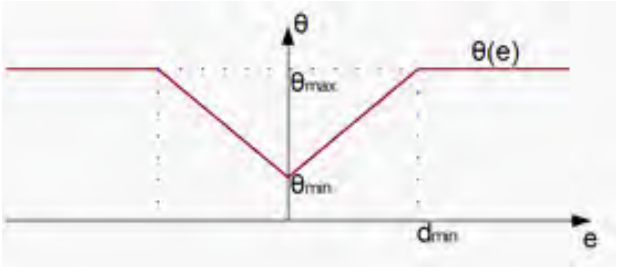
Przyjętym założeniem sterowania położeniem jest jak najszybsze osiąganie przez model zadanej wartości zadanej położenia w przestrzeni trójwymiarowej.

Wejścia:
- aktualne i zadane położenia w osi X, Y, Z,
- aktualne kąty obrotu względem osi X, Y, Z.

Wyjścia:
- zadane kąty obrotu względem osi X,Y,
- zadany ciąg.

Podstawowe problemy sterowania położeniem quadrokoptera:
- stabilizacja w punkcie zadanym,
- utrzymanie stałej wysokości,
- niwelacja zakłóceń (wiatru).

W projekcie systemu sterowania zdecydowano się na sterowanie za pomocą aktualnego uchybu regulacji w każdej z osi. W celu uniknięcia stanów niepożądanych wprowadzono saturacje dla maksymalnych kątów zadanych w funkcji uchybu (rys. 6):



Rys. 6. Funkcja saturacji
Fig. 6. Saturation function

Równania sterujące dla osi X (20-21):

$$kat\_zadany_x = \cos(\theta_z) * (P_3 uchyb_y + P_4 \frac{d\ \ uchyb_y}{dt})$$

$$+\sin(\theta_z) * (P_3 uchyb_x + P_4 \frac{d\ \ uchyb_x}{dt}) \quad (20)$$

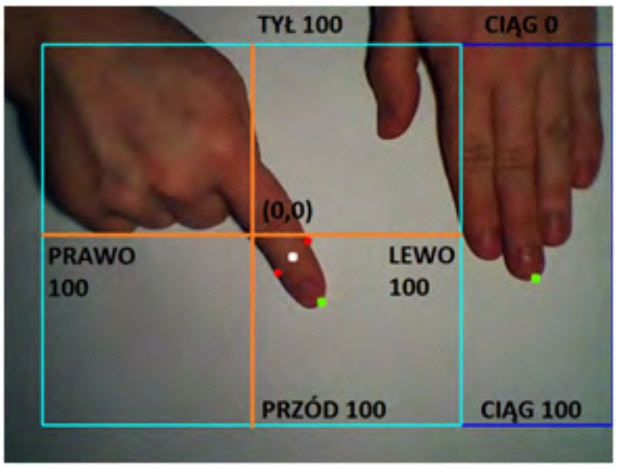
$$ciag_{zadany} = \frac{stala}{\cos(\theta_x)\cos(\theta_y)} + P_1 uchyb_z + P_2 \frac{d\ \ uchyb_z}{dt} \quad (21)$$

Regulacja w osi Y analogicznie do osi X, z tym że indeksy x zastąpione są poprzez indeksy y, a indeksy y na x.

Część proporcjonalna regulatora odpowiada za rozpędzenie modelu w danej osi, natomiast sprzężenie zwrotne od prędkości pozwala na wyhamowanie modelu przed osiągnięciem wartości zadanej. Część związana z aktualnymi kątami obrotu względem osi X i Y pozwala na utrzymanie quadrocoptera na stałej wysokości. P1-P4 z powodu wysokiej nieliniowość obiektu są dobierane w czasie rzeczywistym za pomocą *gain scheduling*.

4. Sterowanie przy pomocy gestów dłoni

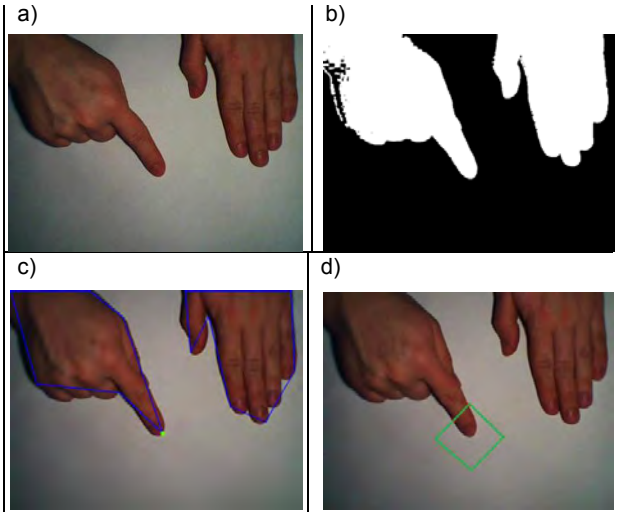
Obecnie można zaobserwować coraz większe zainteresowanie sterowania obiektami przy pomocy gestów lub ruchów np. dłoni. W opisywanym systemie zadawanie przez operatora wartości przechyłu oraz ciągu quadrocoptera jest realizowane poprzez analizę ruchów jego dłoni przy pomocy kamery i zaawansowanych metod przetwarzania i analizy obrazów. Moduł przetwarzania obrazów został napisany przy wykorzystaniu darmowej biblioteki OpenCV [2]. Podczas przesuwania lewą ręką w przód/tył w obszarze widzenia kamery (rys. 7), system zmienia wartości zadane ciągu od zera do stu. Jeśli zmienić położenie prawej dłoni na wyznaczonej płaszczyźnie, system zmienia wartości przechyłu w osiach; zmienimy kąt ułożenia dłoni względem kamery – zmienia się wymuszenie skrętu obiektu.



Rys. 7. Ilustracja sterowania przy pomocy dłoni
Fig. 7. Illustration of control by hand

Założono, ze oświetlenie jest stałe w czasie pracy programu. Cały proces bazuje na przestrzeni kolorów HSV, przestrzeń RGB służy jedynie do wyświetlania i pobierania obrazu. Podstawowe etapy działania programu to [2,5]:
- pobranie obrazy z kamery,
- konwersja z przestrzeni kolorów RGB na HSV,
- ustawienie ograniczeń na H, S i V,
- filtracja,
- wyrównywanie histogramem,
- określanie konturów,
- wyznaczanie wartości zadanych.

Na rys. 8 przedstawiono główne etapy przetwarzania.



Rys. 8. Najbardziej istotne etapy przetwarzania obrazu
Fig. 8. The most important stages of image processing

Na rys. 8a przedstawiono obraz bazowy pobrany z kamery do programu, a na rys. 8b reprezentację binarną otrzymaną przez konwersję obrazu z przestrzeni RGB do HSV, następnie nałożenie na obraz ograniczeń w postaci przedziałów dla Hue, Saturation i Value oraz filtracji medianowej, erozją oraz dylatacji. Tak otrzymany obraz stanowił bazę do określania konturów (rys. 8b). Na ich podstawie wyliczono maksymalnie wychylony punkt dłoni, reprezentowany przez zieloną kropkę. Położenie tego punktu w przestrzeniach roboczych pokazanych na rys. 1, określa jaki jest zadany ciąg oraz jak ma być położony obiekt względem osi X i Y. Aby określić pod jakim kątem należy skręcać, potrzebna jest informacja o ułożeniu dłoni względem kamery. W tym celu obraz binarny przeszukano wokół punktu reprezentującego maksymalne wychylenie, szukając zmian wartości pikseli, dzięki czemu znaleziono dwa kolejne punkty reprezentowane kolorem czerwonym: są to punkty graniczne między tłem a dłonią. Zielonym kwadratem na rys. 8d zaznaczono obszar poszukiwań. Środek odcinka łączącego je jest kolejnym istotnym punktem, oznaczonym białym kolorem. Na jego podstawie oraz na podstawie wcześniej znalezionego punktu zielonego, wyznaczono równanie prostej przecinającej oś kamery. Podstawiając do (22), uzyskano kąt, pod jakim ułożona jest dłoń:

$$\alpha = arctg\left(\frac{y_2 - y_1}{x_2 - x_1}\right) \qquad (22)$$

gdzie:
x,y – położenie w osi punktu zielonego i białego,
α – kąt ułożenia dłoni względem kamery.

Informacje o zadanych kątach oraz ciągu przekazywane są do modułu sterowania.

5. Wizualizacja i środowisko symulacji

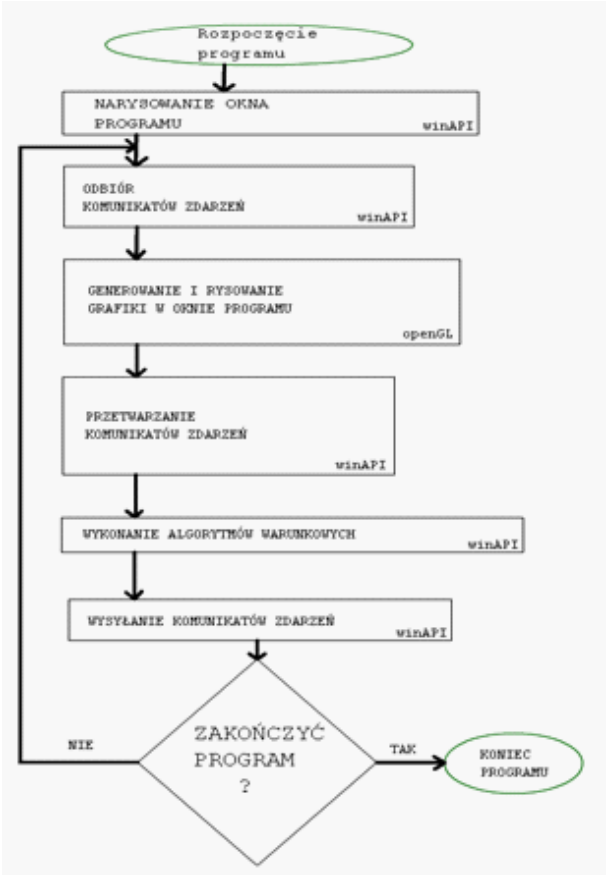
Środowisko wizualizacji i symulacji łączące i spinające całość systemu quadrocoptera zostało przygotowane przy użyciu środowiska programistycznego Microsoft Visual 2010 Ultimate. W skład modułów zintegrowanych w wizualizacji i symulacji wchodzą model matematyczny, moduły sterowania, zadawanie wartości zadanych oraz archiwizacja danych. Środowisko integracji od strony użytkowej zapewnia graficzne przedstawienie lotu quadrocoptera w oparciu o dane otrzymywane z systemu.

Kompletne środowisko zostało przygotowane w oparciu o następujące interfejsy programistyczne: winAPI, openGL, MySQL Connector C, openCV. Każdy z wymienionych interfejsów odpowiadał za przygotowanie określonej części składowej całego środowiska.

5.3. Wykorzystanie interfejsu winAPI do budowy szkieletu środowiska

WinAPI jest programistycznym interfejsem dedykowanym systemom Windows dla budowy aplikacji opartych o wygląd „okienkowy". Interfejs zapewnia obsługę wszystkich aspektów pracy w środowisku systemów Windows, m.in. obsługę drukarek, interfejsów sieciowych, urządzeń peryferyjnych i komunikacji z innymi programami. Z uwagi na dedykację winAPI tylko dla systemów Windows, stanowi on ograniczenie w konwersji środowiska wizualizacji na inne platformy systemowe. Alternatywę stanowi wieloplatformowy interfejs do obsługi aplikacji „okienkowych" – interfejs Qt

WinAPI umożliwia zaprojektowanie wyglądu całego okna programu, jego interfejsu użytkownika według zamierzeń programisty. Dodatkowo zapewnia mechanizmy komunikacji stworzonego środowiska wizualizacji tak z innymi programami, jak i z systemem. Zasada działania omawianej apli-



Rys. 9. Schemat blokowy przedstawiający cykl życia i pracy programu w systemie Windows
Fig. 9. Block diagram showing the cycle of life and work of the program in Windows

kacji, zbudowanej w oparciu o interfejs winAPI, dzieli się na 3 główne etapy:

- utworzenie i wyrysowanie okna programu wraz z inicjalizacją zmiennych oraz obiektów,
- cykliczne przetwarzanie komunikatów zdarzeń wraz z wykonywaniem powiązanych algorytmów,
- zamknięcie programu w momencie otrzymania komunikatu zdarzenia informującego o zakończeniu działania programu wraz ze zwolnieniem wszystkich używanych zasobów systemowych.

Na rys. 9 przedstawiono uproszczony schemat blokowy cyklu życia i pracy programu w systemie Windows.

Blok „wykonanie algorytmów warunkowych" zawiera implementacje dyskretną modelu quadrocoptera, obydwa tryby sterowania oraz mechanizmy modułu archiwizacji wszystkich danych pomiarowych, do których należą sygnały generowane z modułów sterowań quadrocoptera.

5.4. Wykorzystanie interfejsu openGL

Drugim z użytych interfejsów jest openGL, który jest odpowiedzialny za tworzenie złożonej grafiki 2D i 3D na potrzeby wizualizacji lotu quadrocoptera. Za pomocą openGL przygotowana została scena (scenografia), główny obiekt wizualizacji – quadrocopter, siatka informacji bieżących zapewniająca wszystkie bieżące informacje, które mogą być potrzebne użytkownikowi. Sam model quadrocoptera został zaprojektowany i wykonany w programie do tworzenia grafiki 3D – blender 3D. Następnie model graficzny quadrocoptera został wyeksportowany do postaci współrzędnych, umożliwiających import i wykorzystanie przez interfejs openGL

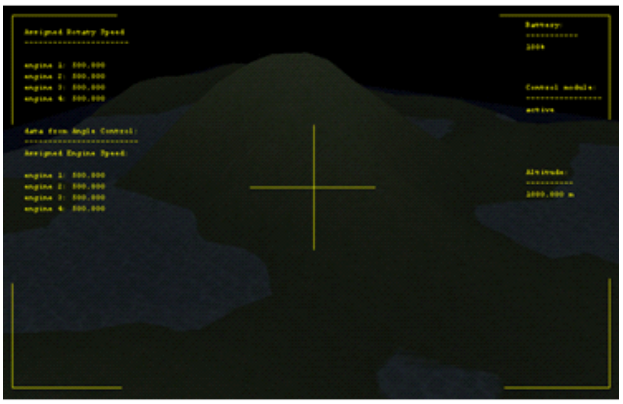
Interfejs openGL, poza możliwością rysowania grafiki, pozwala również na operacje manipulacji położeniem obiektów, kątami obrotu obiektów, położeniem punktu obserwatora, co w połączeniu z technikami odmalowywania obszarów niezajmowanych przez obiekt wizualizacji pozwoliło na osiągnięcie złudzenia dynamiki ruchu w *de facto* statycznie odmalowanym obrazie.

Tworzenie kompletnej wizualizacji lotu opiera się na czterech etapach rysowania grafiki, mianowicie na:

- narysowanie/odrysowanie siatki informacji bieżących,
- narysowanie/odrysowanie głównego obiektu wizualizacji – quadrocoptera,
- ustawienie wymaganego punktu dla obserwatora,
- narysowanie/odrysowanie sceny.

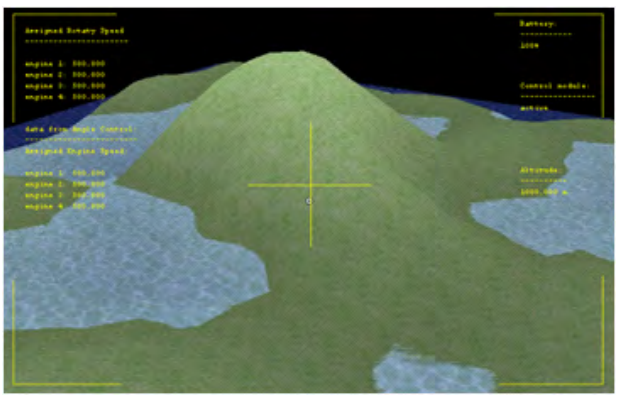
Na rys. 10-13 przedstawione zostały kolejne etapy generowania grafiki opisywan

Kolejnym użytym interfejsem/sterownikiem jest MySQL Connector C, który jest zbiorem metod i funkcji pozwalających na komunikowanie się z bazą danych MySQL. Baza danych wykorzystana została do archiwizacji wszystkich istotnych parametrów z przebiegającej symulacji. Dzięki archiwizacji danych z modelu, sterowań zyskuje się możliwość analizy



Rys. 10. Etap 1 tworzenia siatki informacji bieżących, czyli wygenerowanie siatki w sposób graficzny

Fig. 10. Stage 1 – creating the grid of the current information, generating the grid in a graphical way

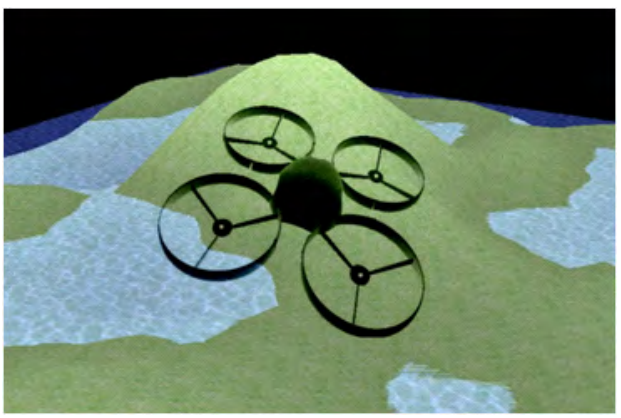


Rys .11. Nałożenie siatki informacji bieżących na pozostały świat graficzny wizualizacji

Fig .11. The imposition of the information grid onto the rest of the graphics

Rys. 12. Wygenerowanie obiektu głównego symulacji w postaci graficznej wraz z ustawieniem położenie obserwatora

Fig. 12. Generating the main simulation object in the graphical form, along with setting of the observer position



Rys.13. Wygenerowanie scenerii dla symulacji wizualnej

Fig. 13. Generating the landscape for visual simulation

zachowania quadrocoptera w zależności od wybranego sterowania i parametrów samej symulacji.

5.5. Zapewnienia integralności środowiska wizualizacji i symulacji

Integralność środowiska wizualizacji i symulacji wymaga powiązania wszystkich użytych modułów obecnych w budowie. Dodatkowo wymienione interfejsy, składające się na architekturę szkieletu środowiska, komunikują się ze sobą za pomocą wspomnianych wcześniej komunikatów zdarzeń. Moduły funkcjonalne środowiska zostały powiązane ze sobą za pomocą zmiennych decyzyjnych i operacyjnych. W celu zapewnienia symulacji lotu w czasie rzeczywistym została wykorzystana wielowątkowość wspierana przez systemy Windows i interfejs winAPI. Wielowątkowość została wykorzystana do przesyłania i przetwarzania danych z poszczególnych bloków, co zapewniło brak przestojów w oczekiwaniu na dane sterujące pomiędzy modułami funkcjonalnymi środowiska, jak i siatką informacji bieżących. Dodatkowo wielowątkowość zapewnia priorytetyzację, która skupiona jest na zapewnieniu zasobów operacyjnych dla najważniejszych elementów środowiska, takich jak komunikacja i wizualizacja. Do niewątpliwych korzyści zastosowania wielowątkowości należą przyśpieszenia wykonywania określonych operacji związanych z grafiką, możliwość operowania środowiska w czasie rzeczywistym i kontrola nad priorytetyzacją zadań. Do wad wielowątkowości zalicza się zwiększone zużycie zasobów, gdyż każdy dodatkowy wątek wymaga dodatkowych zasobów operacyjnych, dodatkowo obsługa wielu wątków wymusza na procesorze przełączanie się między nimi, co zwiększa zużycie pracy i czasu procesora. Ostatnią wadą wykorzystania wielu wątków jest skomplikowanie kodu i mechanizmów zarządzania środowiskiem w czasie jego pracy.

6. Podsumowanie

W artykule przedstawiono model matematyczny quadrocoptera oraz opisano dwa sposoby sterowania quadrocopterem. Algorytmy sterowania oparte są o umiejętnie wykorzystane i połączone regulatory regułowe oraz PI. Wielkości zadane są generowane przez operatora przy pomocy ruchów dłoni odczytywanych przez kamerę i następnie interpretowanych przez zaawansowany system przetwarzania obrazu. Model osadzony jest w zaprojektowanym środowisku symulacyjnym pozwalającym na analizę zachowania się quadrocoptera w przestrzeni trójwymiarowej, w zaprojektowanej scenerii, pod wpływem przyłożonego sterowania. Całość systemu umożliwia efektywne projektowanie i testowanie algorytmów sterowania, przetwarzania obrazów oraz tworzenia grafiki trójwymiarowej. Dotychczasowe prace symulacyjne skłaniają autorów do kontynuowania badań oraz testowania systemu na obiekcie rzeczywistym.

Bibliografia

1. Abłamowicz A.: *Podręcznik pilota szybowców*. Wydawnictwo komunikacji i łączności, 1967.
2. Bradski G. Kaehler A.: *Learning OpenCV: Computer Vision with the OpenCV Library*, O'Reilly, 2008.
3. Buca, D., Cymanowski M.: *Projekt autonomicznej platformy latającej*. Praca magisterska, Politechnika Gdańska, 2010.
4. Koprowski P.: *Okruchy geometrii komputerowej*. 29 lipca 2003 – 13 grudnia 2005.
5. Wojas S.: *Metody przetwarzania obrazów z wykorzystaniem biblioteki OpenCV*. Praca magisterska, AGH, 2010.

Modeling, control and visualisation of a quadrocopter

Abstract: The paper presents an approach for building a mathematical model of a quadrocopter. The main objective was to design the appropriate quadrocopter control and analysis of its behavior in different situations. As the assumption was that the model, control system and all the accompanying algorithms have to be realized in an open source and free programs, to allow for their later implementationin plant, without the need for expensive software. Quadrocopter control by the operator is carried outusing hand movements read by the camera and then properly interpreter with the great help of advanced methods of image processing techniques. The whole system is visualized and embedded in three-dimensional simulation environment.

Keywords: modeling, control, image processing, visualization, quadrocopter.

Tomasz Liecau
Student IV roku studiów inżynierskich na wydziale Automatyki i Robotyki, o specjalności Automatyka i Systemy sterowania. Jego zainteresowania to, systemy przetwarzania obrazu, ogólne programowanie wysoko poziomowe, tworzenie interfejsów GUI.
e-mail: tliecau@gmail.com

Michał Warkocz
Student IV roku studiów inżynierskich na wydziale Automatyki i Robotyki, o specjalności Automatyka i Systemy sterowania. Pasjonat tworzenia algorytmów i heurystyk. Zainteresowania naukowe: języki programowania wysokiego poziomu, bazy danych.
e-mail:michal.warkocz@gmail.com

Krzysztof Wąsik
Student IV roku studiów inżynierskich na wydziale Automatyki i Robotyki, o specjalności Automatyka i Systemy sterowania. Główne dziedziny zainteresowania to języki wysokiego poziomu, nowoczesne technologie CGI wspomagające tworzenie interfejsów GUI, algorytmy i bazy danych.
e-mail: krzysiek.wasik.pl@gmail.com

dr. inż. Michał Grochowski
Adiunkt w Katedrze Inżynierii Systemów Sterowania na Wydziale Elektrotechniki i Automatyki Politechniki Gdańskiej. Jego zainteresowania naukowe obejmują metody detekcji i lokalizacji uszkodzeń, diagnostyki procesów, sterowania optymalizującego oraz inteligencji obliczeniowej.
e-mail: m.grochowski@ely.pg.gda.pl