



Analiza strategii robotów mini-sumo na podstawie obserwacji ich ruchu *

Jan Kędzierski
Edgar Ostrowski

Wrocław 2008

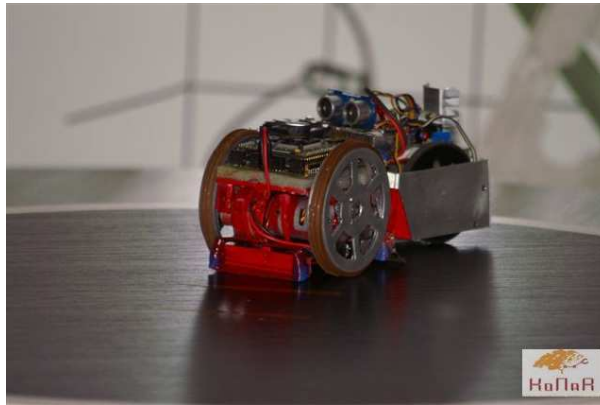
*Dokument powstał w ramach zajęć projektowych z przedmiotu *Komputerowe przetwarzanie wiedzy*, prowadzonych przez dra inż. Tomasza Kubika na Wydziale Elektroniki, Politechniki Wrocławskiej w semestrze letnim roku akademickiego 2007/2008.

Spis treści

1	Wprowadzenie	2
1.1	Cel projektu	2
1.2	Walki robotów mini-sumo	2
2	Przetwarzanie obrazów	3
2.1	Przechwytywanie pojedynczej klatki	3
2.2	Odszumianie obrazu wejściowego	3
2.3	Detekcja kolorów	3
2.4	Konwersja z RGB na skalę szrości	4
2.5	Progowanie binarne	4
2.6	Operacje morfologiczne	5
2.7	Wyliczanie momentów centralnych	5
3	Analiza ruchu	6
3.1	Skalowanie wielkości fizycznych	6
3.2	Wyznaczanie zwrotności robota	6
3.3	Wykrywanie ataków	7
3.4	Wykrywanie falstartów	8
3.5	Ocena walki	8
4	Obsługa programu	9
5	Wnioski	11

1 Wprowadzenie

Koło Naukowe Robotyków „KoNaR”[†] działające przy Wydziale Elektroniki, Politechniki Wrocławskiej od początku swoje działalności prowadzi szerokie badania między innymi w zakresie budowy robotów mini-sumo. Często spotykanym problemem przy obserwacji walk jest brak jednoznacznego rozstrzygnięcia pojedynków. Konfliktowe sytuacje wymagają ingerencji arbitra lub powtórek.



Rysunek 1: Walczące roboty podczas „V Otwartych Zawodów Robotów Mini-sumo”

1.1 Cel projektu

W niniejszym dokumencie przedstawiono komputerowy system wizyjny do analizy walk robotów minisumo, który na podstawie obserwacji ruchu robotów potrafi w krytyczny sposób ocenić przebieg walki oraz wskazać zwycięzce.

Do realizacji projektu wykorzystano język *C++* w środowisku *Microsoft Visual Studio 2006* oraz bibliotkę do przetwarzania obrazów *OpenCV* [2]. Jako układ optyczny do akwizycji danych zastosowano komputerową kamerkę internetową oraz cyfrowy aparat fotograficzny.

1.2 Walki robotów mini-sumo

Roboty klasy mini-sumo, aby brać udział w rozgrywkach powinny spełniać następujące podstawowe kryteria:

- całkowita waga robota nie może przekraczać 500g,
- wymiary robota są ograniczone do 10cm x 10cm (brak ograniczeń co do wysokości),
- roboty muszą być całkowicie autonomiczne, komunikacja bezprzewodowa jest niedozwolona,

Walki dwóch drużyn odbywają się okrągłym ringu zwanym *Dohyo* o 77cm średnicy oraz o czarnym kolorze nawierzchni. Na krawędzi ringu znajduje się 2.5cm biała linia, która umożliwia wykrycie krawędzi przez roboty.

Uczestnicy umieszczają roboty na przeciwległych połówkach ringu. Po ogłoszeniu startu przez sędziego roboty muszą pozostać nieruchome przez 5 sekund. Po upływie tego czasu rozpoczyna się walka. Roboty mają za zadanie zlokalizować przeciwnika oraz wypchać go za ring. Każdy mecz składa się z 3 pojedynków. Wygrane pojedynki nagradzane są punktami *Yuko*. Mecz wygrywa drużyna z przewagą punktową. Szczegółowy opis zasad walk zawiera *Regulamin walk robotów minisumo*[1].

[†]www.konar.pwr.wroc.pl

2 Przetwarzanie obrazów

W przedstawionej aplikacji przewidziano możliwość przetwarzania obrazu na żywo z kamery lub z pliku video. Do prawidłowej analizy walki konieczne było przeprowadzenie następujących operacji na pojedynczej klatce.

2.1 Przechwytywanie pojedynczej klatki

Pierwszym krokiem analizy jest przechwycenie pojedynczej klatki. Dokonuje się to następującym poleceniem:

```
IplImage* frame1 = cvQueryFrame(m_capture);
```

Gdzie *m_capture* to wskaźnik na strumień video.

2.2 Odszumianie obrazu wejściowego

Operacja ta ma a celu usuwanie punktów izolowanych. Dzięki temu obraz staje się jednolity co ułatwia dalszą analizę.

```
cvSmooth(frame1, frame1, CV_GAUSSIAN, 7, 7, 1., 1.);
```



Rysunek 2: Obraz przed odszumieniem i po odszumieniu

2.3 Detekcja kolorów

Algorytm detekcji kolorów analizuje każdy piksel w klatce. Wyliczane są stosunki kolorów pojedynczego piksela do kolorów RGB. Np. dla koloru czerwonego wyznacza się stosunek ilości czerwieni do ilości koloru zielonego oraz ilości czerwieni do koloru niebieskiego. Następnie porównywane są z parametrami zadanymi przez użytkownika w programie. Jeżeli zostanie wykryty żądany kolor pozostaje wcześniejsza wartość piksela jeżeli nie to przypisuje się wartość 0 (kolor czarny).

```
double tmp1=red_point / (green_point+0.001);
double tmp2=red_point / (blue_point+0.001);
if ((tmp1>parm_red1) && (tmp2>parm_red2) && (red_point>parm_red3))
{
    ptr_dest[0]=ptr_src[0];
    ptr_dest[1]=ptr_src[1];
    ptr_dest[2]=ptr_src[2];
}
else
```

```

{
    ptr_dest[0]=0;
    ptr_dest[1]=0;
    ptr_dest[2]=0;
}

```

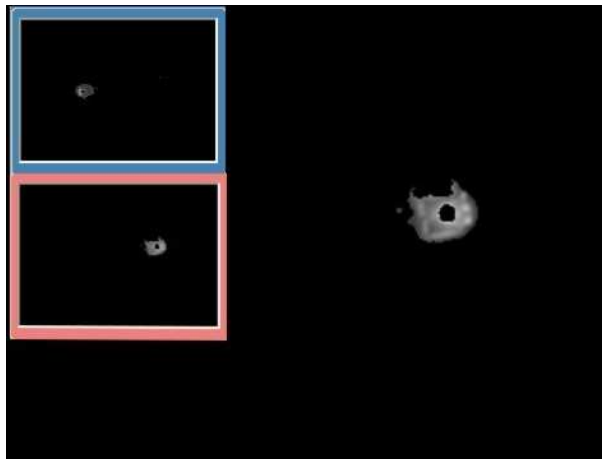


Rysunek 3: Przykład wykrycia kolorów

2.4 Konwersja z RGB na skalę szarości

Dalsze operacje są przeprowadzane 8 bitowej skali szarości. Poniższy listing przedstawia konwersję pojedynczej klatki z RGB na GRAYSCALE.

```
cvCvtColor(frame_red,frame3,CV_RGB2GRAY);
```

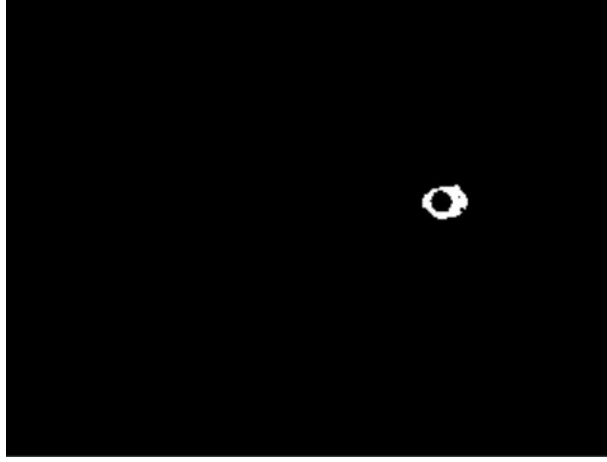


Rysunek 4: Pojedyncza klatka w odcieniach szarości

2.5 Progowanie binarne

W kolejnym kroku obraz jest zamieniany na dwuwartościowy. 1 - kolor biały, obecność obiektu, 0 - kolor czarny, brak obiektu.

```
cvThreshold( frame3, frame3, 1, 255, CV_THRESH_BINARY);
```



Rysunek 5: Obraz po sprogowaniu binarnym

2.6 Operacje morfologiczne

Operacje te mają na celu wypełnienie pikseli, które nie zostały zakwalifikowane jako obiekt na scenie. Dzięki temu uzyskujemy wypełnione sylwetki obiektów.

Zwężanie (erosion) sylwetki

$$U \ominus B = \{u \in \Omega \mid \tilde{B}_u \subset U\} \quad (1)$$

\tilde{B} - symetryczny obraz B względem punktu środkowego Rozszerzanie (dilation) sylwetki:

$$U \oplus B = \{u \in \Omega \mid \tilde{B} \cap U \neq \emptyset\} \quad (2)$$

Dualność erozji i dylatacji:

$$(U \oplus B)^C = \{u \in \Omega \mid \tilde{B} \cap U = \emptyset\} = \{u \in \Omega \mid \tilde{B}_u \in U^C\} = U^C \ominus B \quad (3)$$

Otwieranie (opening) sylwetek:

$$U_B = (U \ominus \tilde{B}) \oplus B \quad (4)$$

Domykanie (closing) sylwetek:

$$U^B = (U \oplus \tilde{B}) \ominus B \quad (5)$$

```
// otwarcie
cvDilate(frame3, frame3, 0, 3);
cvErode(frame3, frame3, 0, 3);

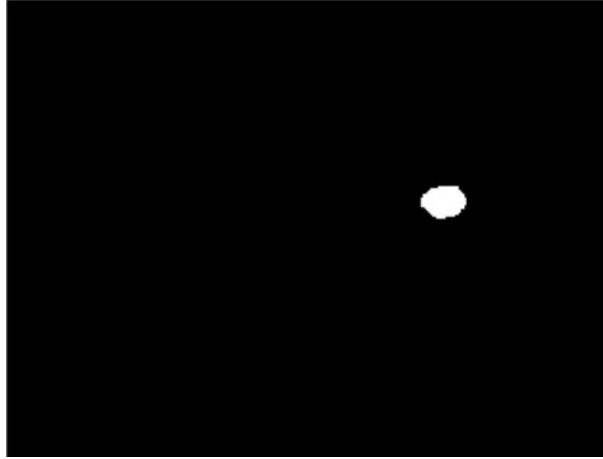
//zamknięcie
cvErode(frame3, frame3, 0, 3);
cvDilate(frame3, frame3, 0, 3);
```

2.7 Wyliczanie momentów centralnych

Operacje te umożliwiają określenie środka ciężkości wykrytych obiektów. Otrzymane współrzędne służą w dalszej części do analizy ruchliwości robotów.

Moment rzędu $p + q$

$$m_{p,q} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \chi U(x,y) x^p y^q \quad (6)$$



Rysunek 6: Obraz po operacjach morfologicznych

Znormalizowane momenty rzędu 1:

$$x_s = \frac{m_{1,0}}{m_{0,0}} \quad (7)$$

$$y_s = \frac{m_{0,1}}{m_{0,0}} \quad (8)$$

```
cvMoments(frame3, &moments, 0);
// wyliczenie srodka
center_red.x =(int)(moments.m10 / moments.m00);
center_red.y =height-(int)(moments.m01 / moments.m00);
```

3 Analiza ruchu

Mając informacje o zmianie położenia środka ciężkości wykrytych obiektów możliwe jest wyznaczenie następujących parametrów ruchu robotów:

- średnia prędkość [cm/s],
- maksymalna prędkość [cm/s],
- dystans [cm],
- zwrotność,
- ataki,
- ataki celne.

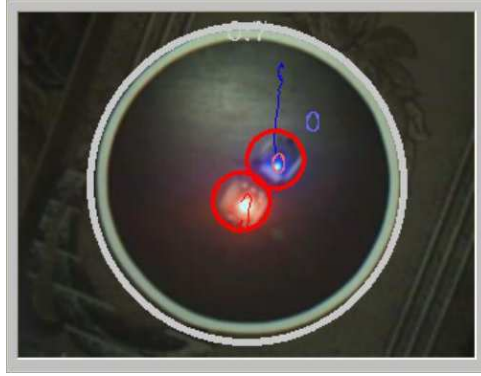
W dalszej części przedstawiono wyznaczanie bardziej skomplikowanych parametrów.

3.1 Skalowanie wielkości fizycznych

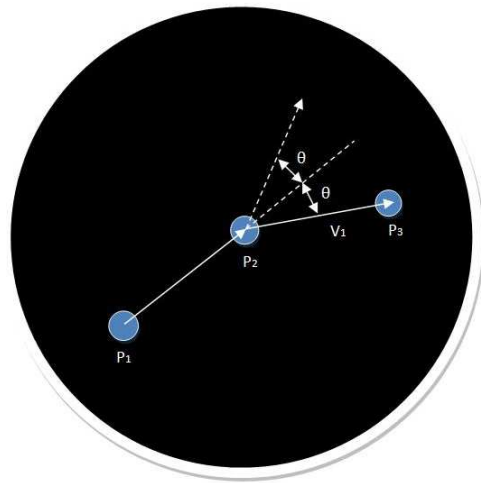
Do oszacowania prędkości oraz przebytego dystansu poruszających się robotów znana średnicę (ϕ 77cm) ringu. W programie istnieje możliwość wyświetlenia ringu za pomocą, które można prawidłowo ustawić kamerę do przechwytywania obrazów.

3.2 Wyznaczanie zwrotności robota

Aby prawidłowo wyznaczyć zwrotność robota konieczna jest znajomość aktualnej pozycji oraz 2 wcześniejszych. Dzięki temu możliwe jest wyznaczenia zmiany kierunku poruszania się obiektu.



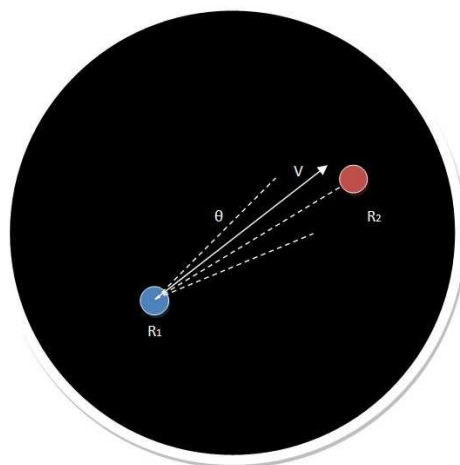
Rysunek 7: Przykład niedokładnego ustawienia kamery



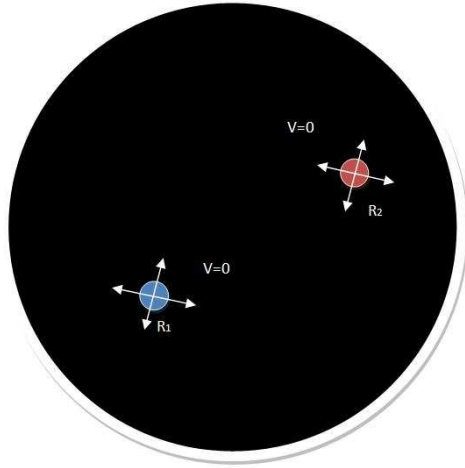
Rysunek 8: Wyznaczanie zwrotności

3.3 Wykrywanie ataków

Wykrycie ataku polega na badaniu prędkości i kierunku poruszających się robotów względem siebie. Zakwalifikowanie ruchu jako atak następuje wtedy gdy jeden robot gwałtownie przemieści się w kierunku przeciwnika. Atak celny następuje wtedy gdy w opisanej sytuacji roboty się zderzą.



Rysunek 9: Wykrywanie ataków



Rysunek 10: Wykrywanie fałstartów

3.4 Wykrywanie fałstartów

Zgodnie z regulaminem walk robotów mini-sumo [1] po ogłoszeniu startu przez sędziego roboty muszą odczekać 5 sekund nieruchomo. Jeżeli któryś z zawodników się poruszy następuje dyskwalifikacja.

3.5 Ocena walki

Program automatycznie punktu przebieg walki w czasie rzeczywistym poprzez nadawanie odpowiednich wag wcześniej wyznaczonym parametrom.

$$Pkt = V_{mean} * a_1 + V_{max} * a_2 + Dist * a_3 + Agility * a_4 + Attack * a_5 + AccurateAttack * a_6$$

gdzie:

- $a_1 = 1$
- $a_2 = 1$
- $a_3 = 1$
- $a_4 = 0, 1$
- $a_5 = 100$
- $a_6 = 1000$

4 Obsługa programu

Po uruchomieniu aplikacji pojawia się główne okno, z tego poziomu jest dostęp do wszystkich ustawień programu.



Rysunek 11: Widok okna głównego

Zgodnie z powyższym rysunkiem:

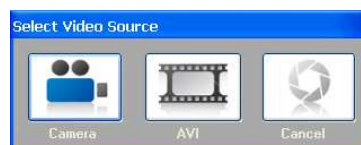
- **1 - Widok sceny**

W oknie tym istnieje możliwość wyświetlania ścieżek robotów, punktacji, ringu oraz markerów.



Rysunek 12: Wyświetlanie dodatkowych informacji

- **2 - Wybór źródła**



Rysunek 13: Wybór źródła

- **3 - Panel sterowania**

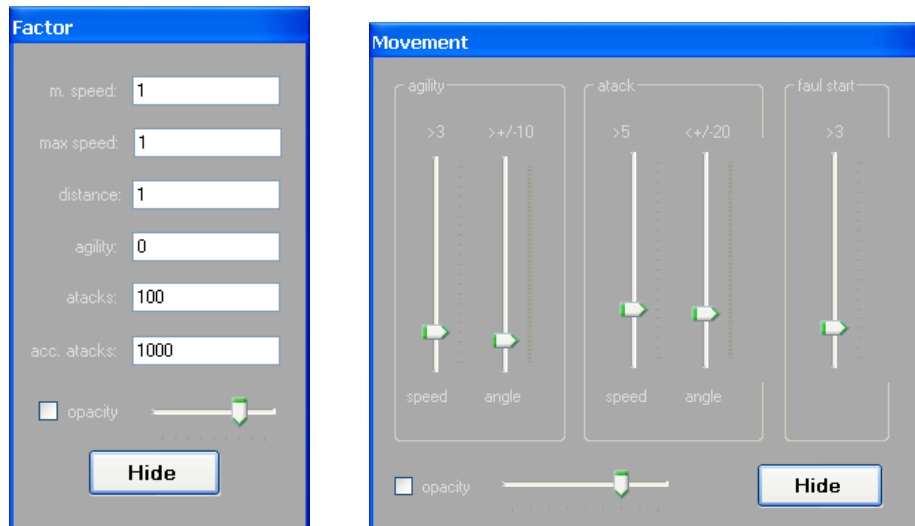
Uruchamianie oraz zatrzymywanie systemu sędziującego.

- **4, 5 - Wyświetlanie pozycji czerwonego i niebieskiego zawodnika**

Wyświetlanie wykrytych obiektów danego koloru.

- **6 - Otwieranie okna z ustawieniami parametrów**

Okno faktor służy do ustawianie parametrów przy analizie punktacji walki.

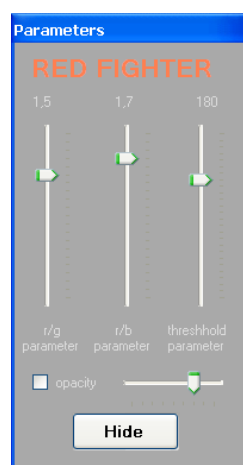


Rysunek 14: Okna *Factor* i *Movement*

- **agility, speed** - minimalna prędkość przy wykrywaniu zwrotności
- **agility, angle** - minimalny kąt przy wykrywaniu zwrotności
- **atack, speed** - minimalna prędkość przy wykrywaniu ataków
- **atack, angle** - minimalny kąt przy wykrywaniu ataków
- **faul start** - prędkość progowa

- **7 - Otwieranie okna z ustawieniami detekcji kolorów**

Ustawianie parametrów detekcji kolorów.



Rysunek 15: Okno *Parameters*

Dla czerwonego:

- **r/b parameter** - próg stosunku kolorów czerwonego do niebieskiego
- **r/g parameter** - próg stosunku kolorów czerwonego do zielonego
- **thershold parameter** - progowanie
- **8 - Ustawienia widoku sceny**
 - **show dohyo** - wyświetlaj ring
 - **show paths** - wyświetlaj ścieżki
 - **show markers** - wyświetlanie markerów
 - **show info** - wyświetlanie informacji dodatkowych
- **9 - Wyświetlanie parametrów analizy walki dla obu zawodników**

5 Wnioski

Stworzona aplikacja pozwala na wspomaganie sędziego przy ocenianiu walk. Często dochodzi do sytuacji kiedy to sędzia nie może w sposób jednoznaczny rozstrzygnąć walki. Dzieje się tak w wtedy kiedy dochodzi do zderzenia robotów, które wykazują podobną przyczepność. Po czasie 3 minut walka jest przerywana bez przyznania punktów. Program ten z pewnością pozwoliłby na krytyczną ocenę przebiegu walki.

Niestety w trakcie realizacji projektu napotkano na wiele problemów związanych głównie z słabą jakością posiadanego sprzętu rejestrującego. Internetowe kamery w zasadzie nie nadają się do tego typu zastosowań. Nie przewidziano w nich możliwości wyłączenia automatyki. Powodowało to, że kamery nie zawsze poprawnie rozpoznawały kolor związany z danym zawodnikiem. Wskazane jest używanie kamer o lepszych parametrach. W programie przewidziano zmianę wszystkich istotnych nastaw parametrów związanych z rozpoznawaniem, detekcją i punktacją.

Literatura

- [1] *Regulamin zawodów robotów minisum*, Wrocław 2004
- [2] *Intel ®Open Source Computer Vision Library, Reference Manual*, Intel Corporation, 2001
- [3] Tadeusiewicz R., Korohada P., *Komputerowa analiza i przetwarzanie obrazów*, FPT, Kraków 1997