Wykład wstępny (IV) *<< zrealizowany w 2006, przedrostki-nazwy zretuszowane w Acrobat >>*
z Podstaw Przetwarzania Informacji
(na danych obrazów 2D w środowisku Matlab 6.x 7.x)

Indeksacja danych obrazów kolorowych 2D, zastosowanie, praktyka

Na poprzednim wykładzie zaznaczono istotną różnicę pomiędzy prostą reprezentacją danych obrazu kolorowego 2D w formacie 24-bitowym, a postacią danych obrazu o zindeksowanej palecie kolorów. Aby zindeksować dane obrazu kolorowego 2D należy wykorzystać polecenie rgb2ind z jednoczesną deklaracją zmiennej danych obrazu zindeksowanego powiedzmy o nazwie image_indeks oraz palety kolorów o nazwie przykładowej paleta. Nadto, jako argument wejściowy należy podać liczbę docelową kolorów palety obrazy indeksowanego, ponieważ następuje tu pewnego rodzaju optymalizacja w objętości danych reprezentujących informację w treści obrazu 2D:

```
>> image=imread('autumn.tif');

>> [image_indeks,paleta]=rgb2ind(image,130);

>> whos image* paleta
```

```
  Name           Size                 Bytes  Class

  image        206x345x3             213210  uint8 array
  image_indeks   206x345              71070  uint8 array
  paleta         130x3                 3120  double array
Grand total is 284670 elements using 287400 bytes
```

W powyższym przykładzie zredukowano liczbę kolorów w indeksacji danych obrazu 2D do 130. Dane 2D obrazu kolorowego formatu 24-bitów o zajętości ok. 213kB zostały zredukowane trzykrotnie w zajętości pamięci, cała wynikowa paleta nowo-zdefiniowanych kolorów zajmuje około 3kB.

W prostym przykładzie redukcji liczności kolorów palety wynikowej w indeksacji danych napisano pewien skrypt:

```
function [M]=animuj_kolory(im);
%SKRYPT REDUKCJI PALETY KOLOROW
%Z GENERACJA ANIMACJI avik.avi
whos im
M=moviein(7);
licznosc=128;
for i=1:7,
    [imgtemp,mapaindeksowana]=rgb2ind(im,licznosc);
    licznosc=licznosc/2;
    imshow(imgtemp,mapaindeksowana);
    M(:,i)=getframe;
end;
movie(M);
movie2avi(M,'avik.avi','COMPRESSION','None','FPS',1);
```

Rys 1 Skrypt redukcji stopniowej liczności palety kolorów poprzez wartości 127,64,...1.

Poniżej podano postać obrazu reprezentowanego jedynie dla ostatnich 3 obrazów.

Rys2 Postacie obrazów 2D przy 4, 2 oraz 1-elementowej palecie kolorów.

Z indeksacją kolorów w zastosowaniach inżynierskich środowiska Matlab często wykorzystuje się łączne wywołanie funkcji find, choć jej funkcjonalność wykracza poza ramy stosowalności *Image Processing Toolbox*:

```
%depth map script for obtaining 3D map
%--copyright Artur Bernat
%from color map
%first argument- the map to be processed
%second argurment - the reference scale
%third argument - the depth on min-max peaks
function [tpdeep,deep]=deepmaps(map,sc,dpth)
%commande to index scale map
sc=sc(:,fix(size(sc,2)/2)+1,:);
[scInd,scMap]=rgb2ind(sc);
%color map indexed within scale color bar gradation
tpgrInd=rgb2ind(map,scMap);
% X,Y size of the map to be processed
Xdim=size(map,2);
Ydim=size(map,1);
for i=1:Xdim,
    for j=1:Ydim,
     temp=tpgrInd(j,i);
     tmpx=find(scInd==temp);
     if( (size(tmpx,1)>0) & (size(tmpx,2)>0) )
         tpdeep(j,i)=tmpx(1);
     else
         tpdeep(j,i)=0;
     end
 end
end
%height of depth scale color bar
%     scY=size(sc,1);
      scY=size(scMap,1);
%final depth map scaled according to given depth
    deep=dpth*(1-tpdeep/scY);
```

Rys 3 Skrypt konwersji kolorowej mapy danych 3D na ich reprezentację głębokościową.

Gdyby na przykład zachodziła potrzeba określenia głębokości punktów na mapie 3D w kolorowej reprezentacji danych, to funkcja find mogłaby kojarzyć odpowiednie głębokości z paska skali głębokości z indeksem odpowiedniego koloru palety kolorowej mapy danych 3D.

W powyższym skrypcie, w linii 10 (licząc wszystkie linie łącznie z linijkami komentarzy) pewien pasek skali głębokości mapy 3D o nazwie sc został zindeksowany ze swobodną, tj. domyślną liczbą kolorów indeksacji. Uzyskano postać zindeksowaną mapy paska głębokości scInd, oraz co najważniejsze, paletę indeksów kolorów o nazwie scMap. Następnie, w linii 12 dokonano indeksacji właściwej mapy kolorowej głębokości map z użyciem wcześniej uzyskanej palety indeksów scMap (zabieg ten składniowo i funkcjonalnie jest jeszcze niemożliwy do przeprowadzenia na bazie środowiska Matlab 6.1). Na tym etapie, należy przeanalizować zawartości uzyskanej w tej sposób zindeksowanej mapy głębokości 3D tpgrInd, pod kątem zgodności numeru indeksu z numerem indeksu na pasku skali głębokości.

Reszta zabiegów w powyższym skrypcie dotyczy przypadków występowania danych pustych w zmiennych przestrzeni roboczej, jak również drobnego zagadnienia skalowania rozpiętości w pikselach indeksowanego paska skali do rzeczywistej rozpiętości danych na mapie 3D.

Wizualizacja zjawisk falowych w ich modelowaniu 2D/3D

Zasadniczym tematem bieżącego wykładu jest omówienie technik i metod przekształceń na obrazach 2D, wraz z nieco bardziej zaawansowaną wizualizacją, w tym również animacją modelowanych prostych zjawisk falowych. Będzie to próba implementacji modelu fali poprzecznej o stałej wartości prędkości promieniowej v jej propagacji (tj. jej rozchodzenia się w sposób ciągły promieniowo i izotropowo od punktowego źródła wychyleń pionowych sinusoidalnych). Jednocześnie, będzie to ta sama wartość prędkości v rozchodzenia się czoła fali sinusoidalnej w początkowych etapach wizualizacji modelowanego zjawiska w oparciu o animację.

Niech amplituda Y wychyleń pionowych dla dowolnej z lokalizacji (x, y) na płaszczyźnie XY pierwszej ćwiartki układu kartezjańskiego współrzędnych o początku w punkcie (x_0,y_0) przyjmie postać:

$$Y(x,y,t) = A \cdot \sin(\sqrt[2]{(x/period - x_0)^2 + (y/period - y_0)^2 + (v \cdot t)^2}) \quad (1)$$

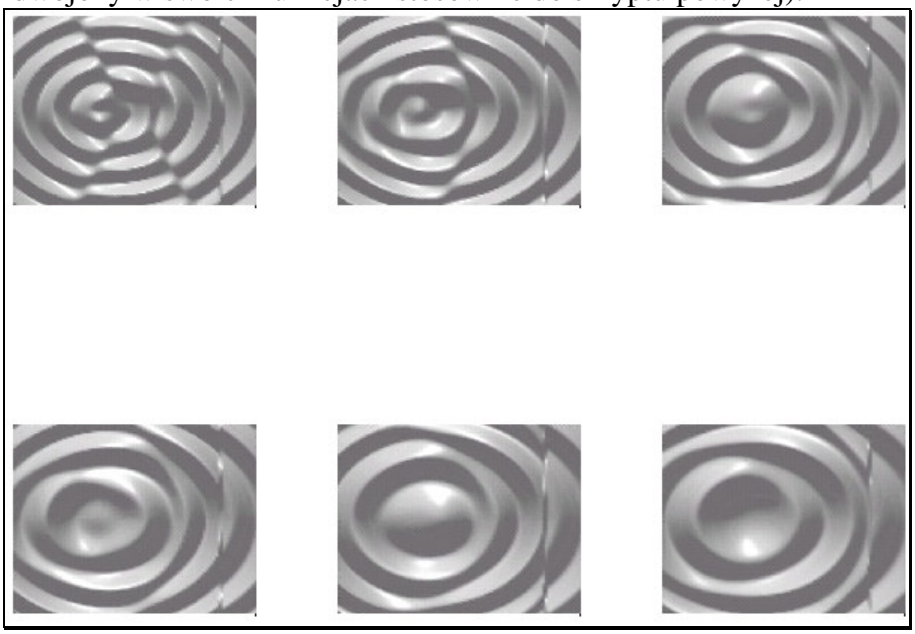
Przebieg zaburzenia z pierwszej ćwiartki na płaszczyźnie XY zostaje następnie odwzorowany przez odbicie w symetrii liniowej, punktowej oraz liniowej w II, III oraz IV ćwiartce, odpowiednio.

Nie jest to oczywiście klasyczna formuła rozchodzenia się fali ciągłej sinusoidalnej (dla przypadku dwu-wymiarowego) o poprzecznym wychyleniu chwilowym względem kierunku jej rozchodzenia się oraz o stałej promieniowej prędkości rozchodzenia. Jednakże, jak się wkrótce okaże, dla celów czysto implementacyjnych jest to dość pouczający przykład, dający emulację w pewnym stopniu zjawiska tłumienia, spowolniania i w konsekwencji wydłużania się w praktyce długości generowanej fali.

```
%Artur Bernat,  all right rights reserved
%12 April 2006
%[M]=animuj_plusk_vel(fr,sz,leng,vel,motn,stts);
%input  arguments:
%---------------------------------------
%fr   <= number of frames per full circle rotation of the camera
%sz   <= size in points of square 3D map for sinusoidal waves
%leng <= period extention
%vel  <= velocity of the propagated sinusoidal wave in animation
%stts <= status: if 1 to save on disk Avi,  0: don't save
%motn <= 0:motionless, 1:variable point of view
%---------------------------------------
%output variables:
%M    <=animation composed of fr frames
%---------------------------------------
function [M]=animuj_plusk_vel(fr,sz,leng,vel,motn,stts);
%SKRYPT REDUKCJI PALETY KOLOROW
%Z GENERACJA ANIMACJI avik.avi
whos im
mapa3D=zeros(sz);
gr=linspace(0,1,180);
grs=[gr' gr' gr'];
okrd=1/leng;
am=double(sz)/8;
ka=0.1;
kd=0.6;
ks=0.2;
ksn=0.2;
material([ka kd ks ksn]);
%figure,surfl(mapa_plusk);shading interp;colormap(grs);
%figure,
clear M;
M=moviein(fr+1);
az_step=double(360)/fr;
for k=1:fr,
    for i=1:sz,
     for j=1:sz,
       %mapa3D(i,j)= am.* sin( sqrt  ( (okrd.*i+vel.*k).^2 + (okrd.*j+vel.*k).^2 ) );
       mapa3D(i,j)=am.*sin(sqrt((okrd.*i).^2+(okrd.*j).^2+(vel.*k).^2));
       %mapa3D=imresize(mapa3D,2,'bilinear');
       mapa_plusk=[flipud(fliplr(mapa3D)) flipud(mapa3D);fliplr(mapa3D) mapa3D];
     end;
    end;
    mapa_plusk(1,1)=sz;
   surfl(mapa_plusk);shading interp;colormap(grs);
   %---------------
   if motn==1
    view(k*az_step,89.5);%axis equal;
   else
    view(0,89.5);
   end;
   %---------------
   k
   M(:,k)=getframe;
end;
movie(M);
clear mapa_plusk;
%movie2avi(M,'vs.avi','FPS',1,'COMPRESSION','None');
if stts==1
 %close('klt.avi');
 movie2avi(M,'klt.avi','fps',16,'compression','Indeo5');
 close;
end;
```

Rys 4 Skrypt animuj_plusk_vel.m dla pojedynczej ciągłej fali sinusoidalnej

Powyższy skrypt animuj_plusk_vel.m dość dobrze emuluje rozchodzenie się fali z pojedynczego punktowego źródła sinusoidalnych zaburzeń. Możliwa jest również emulacja zjawiska rozchodzenia się fali od dwóch punktowych źródeł sinusoidalnych zaburzeń powierzchni, a znajdujących się w niedalekiej odległości od siebie. W takim razie na kolejnych klatkach generowanej w modelowaniu animacji zauważyć będzie można zjawisko interferencji, tj. wzajemnego wyciszania się i wzmacniania się chwilowych amplitud zaburzeń na powierzchni ośrodka płynnego (według skryptu uruchomieniowego animuj_plusk_velx2.m, który został zdwojony w swoich funkcjach stosownie do skryptu powyżej):



Rys 5 Rezultaty wykonania skryptu animuj_plusk_velx2.m generacji fali sinusoidalnej rozchodzącej się promieniowo, w sposób ciągły z dwóch punktowych źródeł drgań. Długości fali wynoszą 2 i 3 jednostki, prędkości rozchodzenia się fal wynoszą:1.5 i 0.7 jednostki na jedną klatkę animacji. Rozstęp poziomy pomiędzy źródłami to 15 jednostek przy wymiarach wzbudzanej powierzchni fal 100x100. Całość animacji zawiera 30 klatek. Na rysunku zaprezentowano wyciąg jedynie co piątej klatki poczynając od piątej włącznie. Widoczne pewne przekłamania algorytmu z prawej strony zawartości obrazów

$$Y(x,y,t) = A\cdot[\sin(\sqrt[2]{(x/\,periodA - x_{0A})^2 + (y/\,periodA - y_{0A})^2} + (v_A\cdot t)^2) + \sin(\sqrt[2]{(x/\,periodB - x_{0B})^2 + (y/\,periodB - y_{0B})^2} + (v_B\cdot t)^2)] \quad (2)$$

Po dłuższym namyśle zdecydowano się na korekcję powyższych zależności ((1) oraz (2)), celem uzyskania efektu w modelowaniu generacji fali sinusoidalnej w oparciu o zależnośći:

$$Y(x,y,t) = A\cdot[\sin(\sqrt[2]{(x/\,period - x_0)^2 + (y/\,period - y_0)^2} + v\cdot t] \quad (3)$$

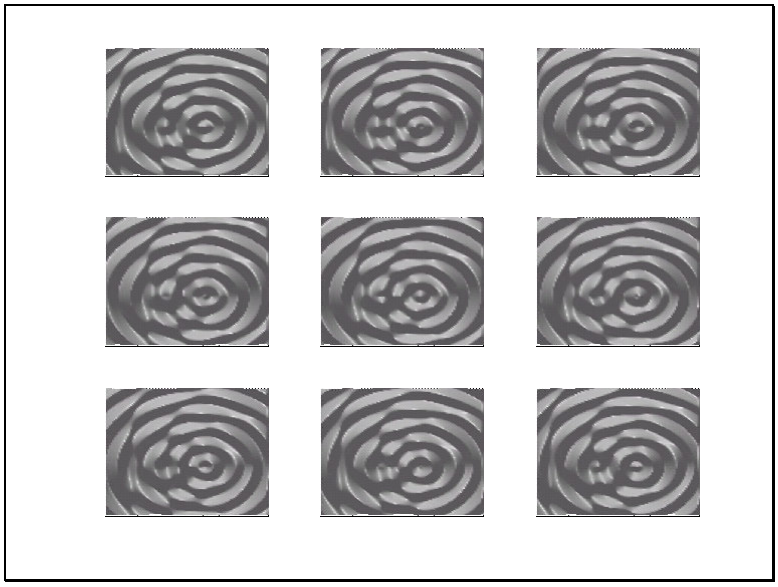
oraz

$$Y(x,y,t) = A\cdot[\sin(\sqrt[2]{(x/\,periodA - x_{0A})^2 + (y/\,periodA - y_{0A})^2} + v_A\cdot t] + A\cdot[\sin(\sqrt[2]{(x/\,periodB - x_{0B})^2 + (y/\,periodB - y_{0B})^2} + v_B\cdot t] \quad (4)$$

W tym przypadku przebieg chwilowych zmian na wzbudzanej powierzchni 3D jest bardziej wiarygodny dla oka, zwłaszcza przy dwóch źródłach ciągłej generacji fal. Poniżej przykład wizualizacji opartej na animacji 90-klatkowej z przestrzenią 100x100 punktów powierzchni 3D. Odstęp w poziomie pomiędzy dwoma źródłami fali sinusoidalnej wynosi 25 jednostek. Długości fal są zgodne z tymi podanymi powyżej na rysunku 5 tj. 2 i 3 jednostki odpowiednio, przy prędkościach rozchodzenia się tych fal wynoszących odpowiednio 1.5 i 0.7. Jest to oczywiście warunek zjawiska *nadzwyczajnego*, zakładający **różne** prędkości V_A i V_B propagacji fal w jednorodnym środowisku, tylko dlatego, że pochodzą one z dwóch różnych źródeł! Nie mniej umiarkowanie prosty skrypt (taki jak plusk_velx2.m, poniżej) umożliwia modelowanie zjawisk z założenia w przyrodzie niemożliwych.

```
%Artur Bernat, all right rights reserved
%12 April 2006 CONSTANT RADIAL VELOCITY OF PROPAGATED WAVE
%[M]=plusk_velx2(fr,sz,leng,vel, leng2, vel2, ext, motn,stts);
%input  arguments:
%-------------------------------------
%fr   <= number of frames per full circle rotation of the camera
%sz   <= size in points of square 3D map for sinusoidal waves
%leng <= period extention
%leng2<= period extention for the 2nd source
%vel  <= velocity of the propagated sinusoidal wave in animation
%vel2 <= velocity of the 2nd source
%ext  <= extension span between 1st and 2nd sources on 3D surface
%stts <= status: if 1 to save on disk Avi, 0: don't save
%motn <= 0:motionless, 1:variable point of view
%-------------------------------------
%output variables:
%M    <=animation composed of fr frames
%-------------------------------------
function [M]=plusk_velx2(fr,sz,leng,vel,leng2,vel2,extens,motn,stts);
%SKRYPT REDUKCJI PALETY KOLORÓW
%Z GENERACJA ANIMACJI avik.avi
whos in
mapa3D=zeros(sz+extens/2);
mapa3DB=zeros(sz+extens/2);
gr=linspace(0,1,180);
grs=[gr' gr' gr'];
okrd=1/leng;
okrdB=1/leng2;
am=double(sz)/8;
ka=0.1;
kd=0.6;
ks=0.2;
ksn=0.2;
material([ka kd ks ksn]);
%figure,surfl(mapa_plusk);shading interp;colormap(grs);
%figure,
clear M;
M=moviein(fr+1);
az_step=double(360)/fr;
for k=1:fr,
%--SOURCE A---
    for i=1:sz+extens/2,
     for j=1:sz+extens/2,
       mapa3D(i,j)=am.*sin(sqrt((okrd.*i).^2+(okrd.*j).^2)-vel.*k);
       mapa_plusk=[flipud(fliplr(mapa3D)) flipud(mapa3D);fliplr(mapa3D) mapa3D];
     end;
    end;
    mapa_plusk(1,1)=sz;
%--SOURCE B---
    for i=1:sz+extens/2,
     for j=1:sz+extens/2,
       mapa3DB(i,j)=am.*sin(sqrt((okrdB.*i).^2+(okrdB.*j).^2)-vel2.*k);
       mapa_pluskB=[flipud(fliplr(mapa3DB)) flipud(mapa3DB);fliplr(mapa3DB) mapa3DB];
     end;
    end;
    mapa_pluskB(1,1)=sz;
    cols=zeros(sz+sz+2*floor(extens/2),2*floor(extens/2));
    whos mapa_plusk mapa_pluskB cols
    sur1=[cols mapa_plusk];
    sur2=[mapa_pluskB cols];
    sumsur=sur1+sur2;
    whos sur1 sur2 sumsur
    surwyn=sumsur(2*floor(extens/2)+1:sz+sz+2*floor(extens/2),extens+1:sz+sz+extens);
    whos surwyn
    surfl(surwyn);shading interp;colormap(grs);
   %---------------
   if motn==1 %viewer point in move
    view(k*az_step,89.5);%axis equal;
   else       %motionless point of view
    view(0,89.5);
   end;
   %---------------
   k
   M(:,k)=getframe;
end;
movie(M);
clear mapa_plusk;
clear mapa_pluskB;
%movie2avi(M,'vs.avi','FPS',1,'COMPRESSION','None');
if stts==1
 %close('klt.avi');
 movie2avi(M,'klt.avi','fps',16,'compression','Indeo5');
 close;
end;
```

Rys 6 Skrypt plusk_velx2.m dla ciągłej fali sinusoidalnej generowanej z dwóch źródeł punktowych. Istnieje możliwość definiowania niezależnie długości fal oraz ich prędkości propagacji dla źródła A i B.



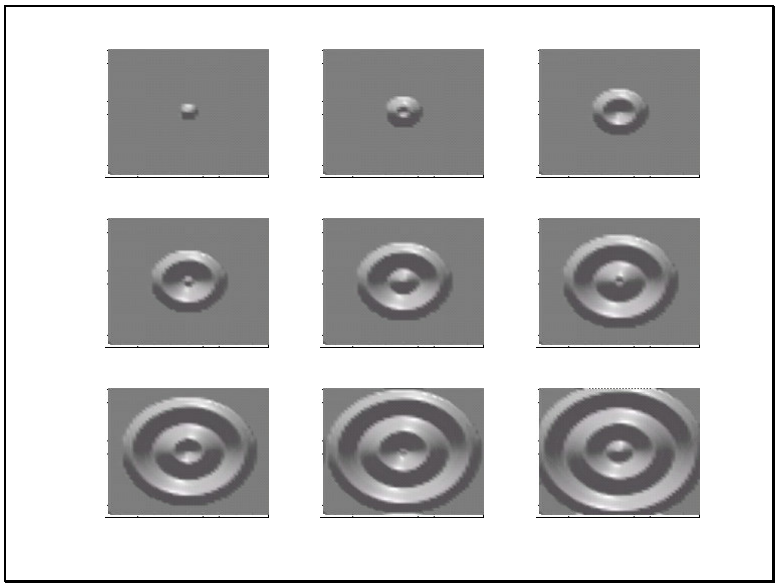
Rys 7 Rezultaty wykonania skryptu plusk_velx2.m generacji ciągłej fali sinusoidalnej rozchodzącej się promieniowo ze stałą prędkością, z dwóch punktowych źródeł drgań. Długości fali wynoszą 2 i 3 jednostki, prędkości rozchodzenia się fal wynoszą:1.5 i 0.7 jednostki na jedną klatkę animacji. Rozstęp poziomy pomiędzy źródłami to 25 jednostek przy wymiarach wzbudzanej powierzchni 100x100. Całość animacji zawiera 90 klatek. Na rysunku jedynie zaprezentowano 10 klatek

W realnych zastosowaniach modelowania propagacji fal wzbudzanych w ośrodku jednorodnym, nadto niezbędna jest implementacja rozchodzącego się czoła fali, którą zaczęto wzbudzać w ośrodku:

$$radial\_dist = \sqrt[2]{(x/period - x_0)^2 + (y/period - y_0)^2} \qquad (5)$$

$$Y(x,y,t) = A \cdot [(1 - sign(radial\_dist - v \cdot t)] \cdot [\sin(radial\_dist - v \cdot t)]$$

Prędkość propagacji czoła fali v jest tutaj w implementacji tożsama z prędkością ciągłej propagacji fali.



Rys 8 Rezultaty wykonania skryptu plusk_velfront.m generacji fali sinusoidalnej rozchodzącej się promieniowo ze stałą prędkością, z pojedynczego źródła drgań.

W zależności (5) nie podano opisu czasowego amplitudy w oparciu o uskok Heaviside'a, lecz posłużono się analogicznym w działaniu wyrażeniu zbudowanym w oparciu o wywołanie funkcji sign.

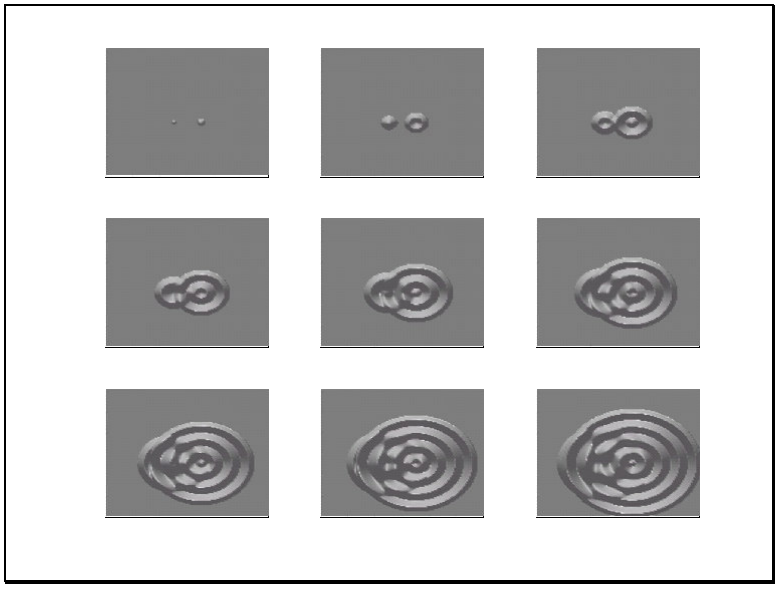
```
%Artur Bernat, all right rights reserved
%12 April 2006 with INITIAL front of wave
%[M]=plusk_velfront(fr,sz,leng,vel,motn,stts);
%input  arguments:
%-----------------------------------
%fr   <= number of frames per full circle rotation of the camera
%sz   <= size in points of square 3D map for sinusoidal waves
%leng <= period extention
%vel  <= velocity of the propagated sinusoidal wave in animation
%stts <= status: if 1 to save on disk Avi, 0: don't save
%motn <= 0:motionless, 1:variable point of view
%-----------------------------------
%output variables:
%M    <=animation composed of fr frames
%-----------------------------------
function [M]=plusk_velfront(fr,sz,leng,vel,motn,stts);
%SKRYPT REDUKCJI PALETY KOLOROW
%Z GENERACJA ANIMACJI avik.avi
whos im
mapa3D=zeros(sz);
gr=linspace(0,1,180);
grs=[gr' gr' gr'];
okrd=1/leng;
am=double(sz)/8;
ka=0.1;
kd=0.6;
ks=0.2;
ksn=0.2;
material([ka kd ks ksn]);
%figure,surfl(mapa_plusk);shading interp;colormap(grs);
%figure,
clear M;
M=moviein(fr+1);
az_step=double(360)/fr;
for k=1:fr,
    for i=1:sz,
     for j=1:sz,
           %mapa3D(i,j)=am.*sin(sqrt((okrd.*i).^2+(okrd.*j).^2+(vel.*k).^2));
           radial_dist=sqrt((okrd.*i).^2+(okrd.*j).^2);
           mapa3D(i,j)=am.*(1-sign(radial_dist-vel.*k)).*sin(radial_dist-vel.*k);
      mapa_plusk=[flipud(fliplr(mapa3D)) flipud(mapa3D);fliplr(mapa3D) mapa3D];
     end;
    end;
    mapa_plusk(1,1)=sz;
   surfl(mapa_plusk);shading interp;colormap(grs);
   %---------------
   if motn==1
    view(k*az_step,89.5);%axis equal;
   else
    view(0,89.5);
   end;
   %---------------
   k
   M(:,k)=getframe;
end;
movie(M);
clear mapa_plusk;
if stts==1
 %close('klt.avi');
 movie2avi(M,'klt.avi','fps',16,'compression','Indeo5');
 close;
end;
```

Rys 9 Skrypt plusk_velfront.m dla fali sinusoidalnej z czołem, generowanej z pojedynczego źródła

Zaimplementowano również propagację fali z dwóch źródeł drgań z czołem fali (plusk_velxfront.m):

```
%Artur Bernat, all right rights reserved
%12 April 2006 CONSTANT RADIAL VELOCITY OF PROPAGATED WAVE
% WITH IMPLEMENTED WAVE FRONT
%[M]=plusk_velx2front(fr,sz,leng,vel, leng2, vel2, ext, notn,stts);
%input  arguments:
%-------------------------------------
%fr   <= number of frames per full circle rotation of the camera
%sz   <= size in points of square 3D map for sinusoidal waves
%leng <= period extention
%leng2<= period extention for the 2nd source
%vel  <= velocity of the propagated sinusoidal wave in animation
%vel2 <= velocity of the 2nd source
%ext  <= extension span between 1st and 2nd sources on 3D surface
%stts <= status: if 1 to save on disk Avi, 0: don't save
%notn <= 0:motionless, 1:variable point of view
%-------------------------------------
%output variables:
%M    <=animation composed of fr frames
%-------------------------------------
function [M]=plusk_velx2front(fr,sz,leng,vel,leng2,vel2,extens,notn,stts);
%SKRYPT REDUKCJI PALETY KOLORÓW
%Z GENERACJA ANIMACJI avi%.avi
whos in
mapa3D=zeros(sz+extens/2);
mapa3DB=zeros(sz+extens/2);
gr=linspace(0,1,180);
grs=[gr' gr' gr'];
okrd=1/leng;
okrdB=1/leng2;
am=double(sz)/8;
ka=0.1;
kd=0.6;
ks=0.2;
ksn=0.2;
material([ka kd ks ksn]);
%figure,surfl(mapa_plusk);shading interp;colormap(grs);
%figure,
clear M;
M=moviein(fr+1);
az_step=double(360)/fr;
for k=1:fr,
%--SOURCE A---
    for i=1:sz+extens/2,
     for j=1:sz+extens/2,
       radial_dist=sqrt((okrd.*i).^2+(okrd.*j).^2);
       mapa3D(i,j)=am.*(1-sign(radial_dist-vel.*k)).*sin(radial_dist-vel.*k);
       mapa_plusk=[flipud(fliplr(mapa3D)) flipud(mapa3D);fliplr(mapa3D) mapa3D];
     end;
    end;
    mapa_plusk(1,1)=sz;
%--SOURCE B---
    for i=1:sz+extens/2,
     for j=1:sz+extens/2,
         radial_distB=sqrt((okrdB.*i).^2+(okrdB.*j).^2);
       mapa3DB(i,j)=am.*(1-sign(radial_distB-vel2.*k)).*sin(radial_distB-vel2.*k);
       mapa_pluskB=[flipud(fliplr(mapa3DB)) flipud(mapa3DB);fliplr(mapa3DB) mapa3DB];
     end;
    end;
    mapa_pluskB(1,1)=sz;
    cols=zeros(sz+sz+2*floor(extens/2),2*floor(extens/2));
    whos mapa_plusk mapa_pluskB cols
    sur1=[cols mapa_plusk];
    sur2=[mapa_pluskB cols];
    sumsur=sur1+sur2;
    whos sur1 sur2 sumsur
    surwyn=sumsur(2*floor(extens/2)+1:sz+sz+2*floor(extens/2),extens+1:sz+sz+extens);
    whos surwyn
    surfl(surwyn);shading interp;colormap(grs);
   %---------------
   if notn==1 %viewer point in move
    view(k*az_step,89.5);%axis equal;
   else       %motionless point of view
    view(0,89.5);
   end;
   %---------------
   k
   M(:,k)=getframe;
end;
movie(M);
clear mapa_plusk;
clear mapa_pluskB;
%movie2avi(M,'vs.avi','FPS',1,'COMPRESSION','None');
if stts==1
 %close('klt.avi');
 movie2avi(M,'klt.avi','fps',16,'compression','Indeo5');
```

Rys 10 Skrypt plusk_velx2front.m dla fali sinusoidalnej z czołem, generowanej z dwóch źródeł

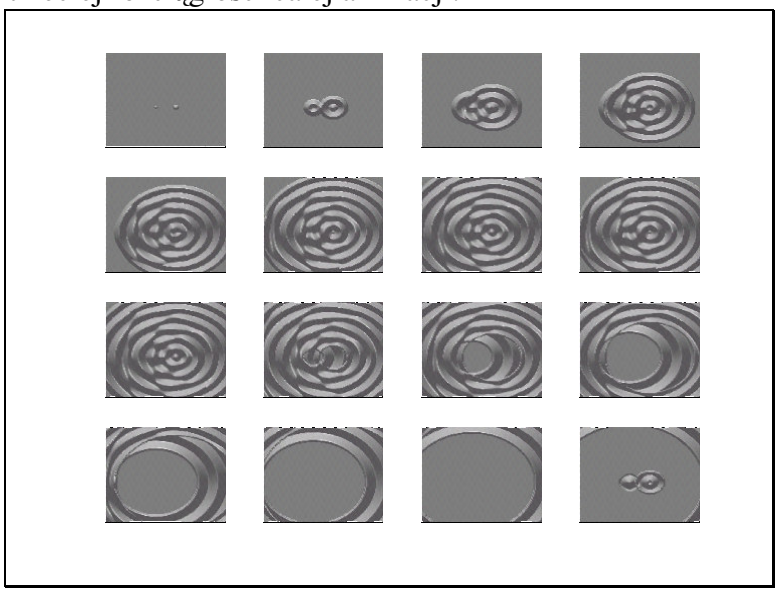


Rys 11 Rezultaty wykonania skryptu plusk_velx2front.m generacji fali sinusoidalnej rozchodzącej się promieniowo ze stałą prędkością, z dwóch źródeł drgań, z czołami fal.

Zależność zaimplementowana w tym przypadku jest przedstawiona poniżej

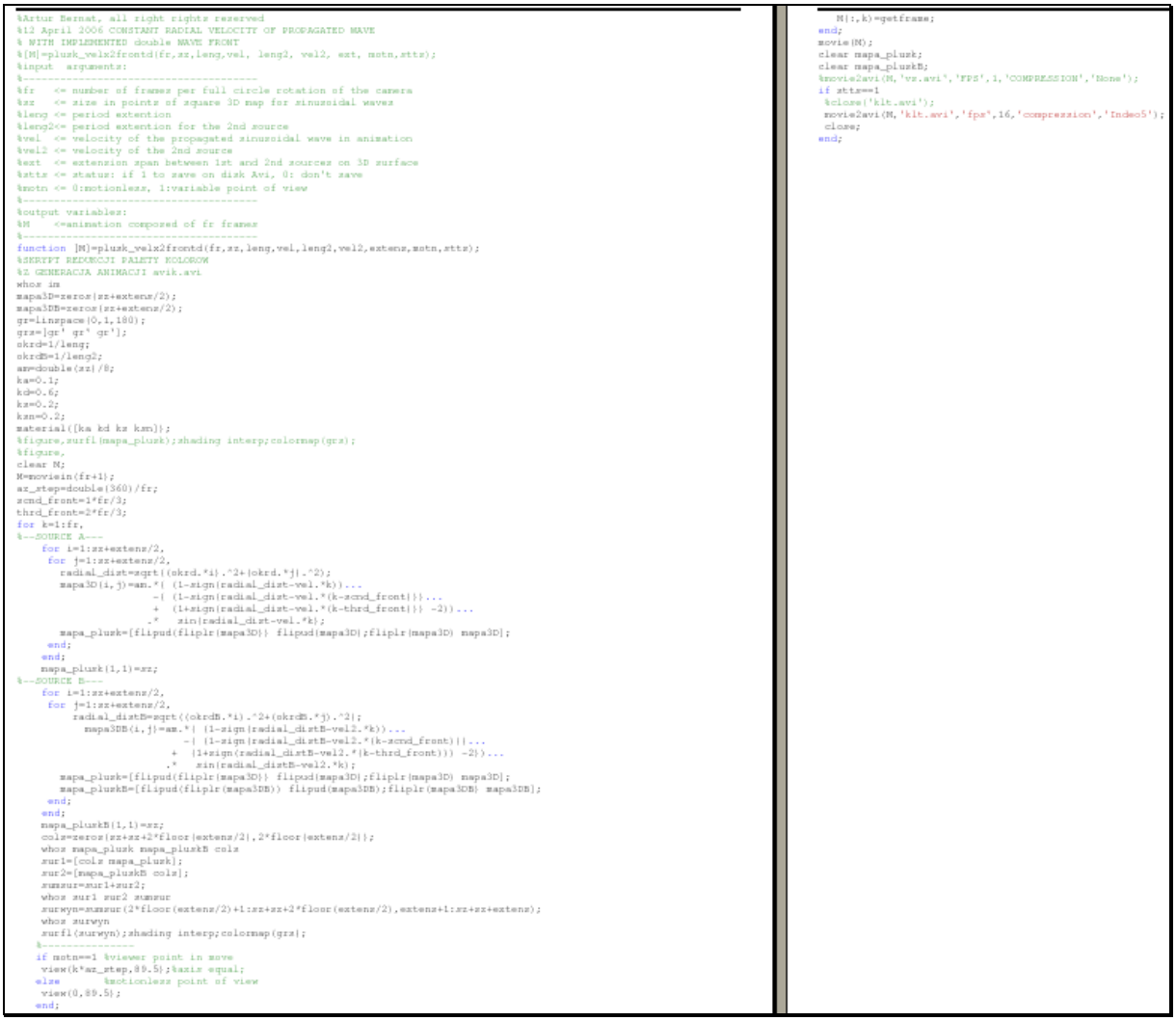
$$\begin{aligned} radial\_dist_A &= \sqrt[2]{(x/periodA - x_{0A})^2 + (y/periodA - y_{0A})^2} \\ radial\_dist_B &= \sqrt[2]{(x/periodB - x_{0B})^2 + (y/periodB - y_{0B})^2} \\ Y(x,y,t) &= A\cdot[(1-sign(radial\_dist_A - v_A\cdot t)]\cdot[\sin(radial\_dist - v_A\cdot t)] \\ &+ \quad A\cdot[(1-sign(radial\_dist_B - v_B\cdot t)]\cdot[\sin(radial\_dist_B - v_B\cdot t)] \end{aligned} \tag{5}$$

Ostatecznie, zamodelowano przebieg fali z pierwszym i drugim czołem fali generowanym na początku animacji oraz w dwóch trzecich jej rozciągłości, z wygaszeniem pierwszego ciągu fal w jednej trzeciej rozciągłości całej animacji:



Rys 12 Rezultaty wykonania skryptu plusk_velx2frontd.m generacji fali sinusoidalnej rozchodzącej się promieniowo ze stałą prędkością, z dwóch źródeł drgań, ze zdwojonymi w czasie czołami fal. Animacja składająca się z 90 klatek została tutaj przedstawiona schematycznie z wyciągiem średnio co 5 klatki.

$$\begin{aligned} radial\_dist_A &= \sqrt[2]{(x/periodA - x_{0A})^2 + (y/periodA - y_{0A})^2} \\ radial\_dist_B &= \sqrt[2]{(x/periodB - x_{0B})^2 + (y/periodB - y_{0B})^2} \\ Y(x,y,t) &= A\cdot[(1-sign(radial\_dist_A - v_A\cdot t)]\cdot[\sin(radial\_dist - v_A\cdot t)] \\ &+ \quad A\cdot[(1-sign(radial\_dist_B - v_B\cdot t)]\cdot[\sin(radial\_dist_B - v_B\cdot t)] \end{aligned} \tag{6}$$



```
%Artur Bernat, all right rightz reserved
%12 April 2006 CONSTANT RADIAL VELOCITY OF PROPAGATED WAVE
% WITH IMPLEMENTED double WAVE FRONT
%[M]=plusk_velx2frontd(fr,sz,leng,vel, leng2, vel2, ext, mstn,sttz);
%input  arguments:
%-------------------------------------
%fr   <= number of frames per full circle rotation of the camera
%sz   <= size in points of square 3D map for sinusoidal waves
%leng <= period extention
%leng2<= period extention for the 2nd source
%vel  <= velocity of the propagated sinusoidal wave in animation
%vel2 <= velocity of the 2nd source
%ext  <= extension span between 1st and 2nd sources on 3D surface
%sttz <= status: if 1 to save on disk Avi, 0: don't save
%mstn <= 0:motionless, 1:variable point of view
%-------------------------------------
%output variables:
%M    <=animation composed of fr frames
%-------------------------------------
function [M]=plusk_velx2frontd(fr,sz,leng,vel,leng2,vel2,extenz,mstn,sttz);
%SKRYPT REDUKCJI PALETY KOLOROW
%Z GENERACJA ANIMACJI avik.avi
whos in
mapa3D=zeros(sz+extenz/2);
mapa3DB=zeros(sz+extenz/2);
gr=linspace(0,1,180);
grz=[gr' gr' gr'];
okrd=1/leng;
okrdB=1/leng2;
am=double(sz)/8;
ka=0.1;
kd=0.6;
ks=0.2;
ksn=0.2;
material([ka kd ks ksn]);
%figure,surfl(mapa_plusk);shading interp;colormap(grz);
%figure,
clear M;
M=moviein(fr+1);
ax_step=double(360)/fr;
zcnd_front=1*fr/3;
thrd_front=2*fr/3;
for k=1:fr,
%--SOURCE A---
    for i=1:sz+extenz/2,
     for j=1:sz+extenz/2,
       radial_dist=sqrt((okrd.*i).^2+(okrd.*j).^2);
       mapa3D(i,j)=am.*( (1-sign(radial_dist-vel.*k))...
                      -( (1-sign(radial_dist-vel.*(k-zcnd_front)))...
                      +  (1+sign(radial_dist-vel.*(k-thrd_front))) -2))...
                    .*   sin(radial_dist-vel.*k);
       mapa_plusk=[flipud(fliplr(mapa3D)) flipud(mapa3D);fliplr(mapa3D) mapa3D];
     end;
    end;
    mapa_plusk(1,1)=sz;
%--SOURCE B---
    for i=1:sz+extenz/2,
     for j=1:sz+extenz/2,
         radial_distB=sqrt((okrdB.*i).^2+(okrdB.*j).^2);
           mapa3DB(i,j)=am.*( (1-sign(radial_distB-vel2.*k))...
                           -( (1-sign(radial_distB-vel2.*(k-zcnd_front)))...
                         +   (1+sign(radial_distB-vel2.*(k-thrd_front))) -2))...
                       .*    sin(radial_distB-vel2.*k);
       mapa_plusk=[flipud(fliplr(mapa3D)) flipud(mapa3D);fliplr(mapa3D) mapa3D];
       mapa_pluskB=[flipud(fliplr(mapa3DB)) flipud(mapa3DB);fliplr(mapa3DB) mapa3DB];
     end;
    end;
    mapa_pluskB(1,1)=sz;
    colz=zeros(sz+sz+2*floor(extenz/2),2*floor(extenz/2));
    whos mapa_plusk mapa_pluskB colz
    zur1=[colz mapa_plusk];
    zur2=[mapa_pluskB colz];
    zurzur=zur1+zur2;
    whos zur1 zur2 zurzur
    zurwyn=zurzur(2*floor(extenz/2)+1:sz+sz+2*floor(extenz/2),extenz+1:sz+sz+extenz);
    whos zurwyn
    surfl(zurwyn);shading interp;colormap(grz);
   %---------------
   if mstn==1 %viewer point in move
    view(k*ax_step,89.5);%axis equal;
   else    %motionless point of view
    view(0,89.5);
   end;
   M(:,k)=getframe;
end;
movie(M);
clear mapa_plusk;
clear mapa_pluskB;
%movie2avi(M,'vz.avi','FPS',1,'COMPRESSION','None');
if sttz==1
 %close('klt.avi');
 movie2avi(M,'klt.avi','fps',16,'compression','Indeo5');
 close;
end;
```

Rys 13 Skrypt plusk_velx2frondt.m dla fali sinusoidalnej, generowanej z dwóch źródeł, ze zdwojonymi w czasie czołami fal

Opis funkcji czasowego sterowania amplitudą *A* przebiegów sinusoidalnych nie wymaga tutaj w tym wykładzie zapisu formalnego.

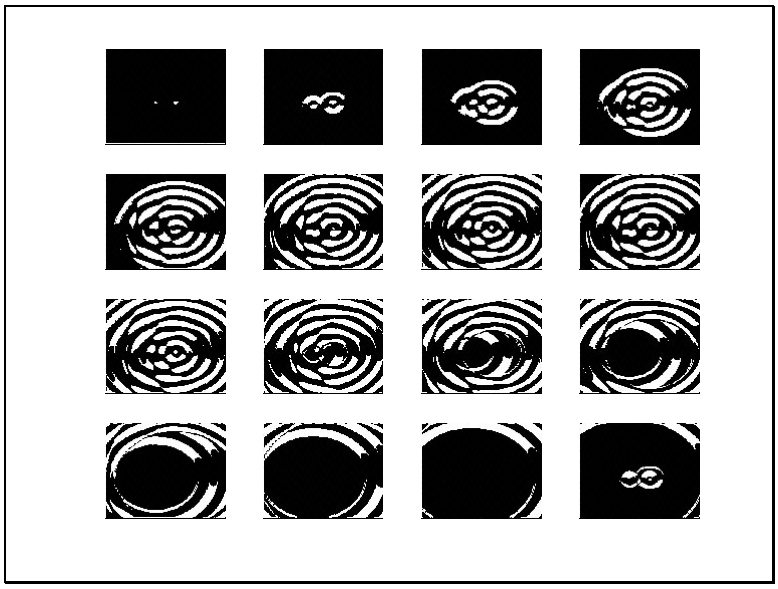
Grupa funkcji wyodrębniania cech w zawartości obrazów 2D

Obecnie, przebieg ostatni fal sinusoidalnych utrwalony na szeregu obrazach 2D w formie animacji, posłuży w wizualizacji możliwości zastosowań szeregu funkcji z *Image Processing Toolbox*. W grupie przekształceń podstawowych omówione zostaną między innymi funkcje detekcji krawędzi, rozmycia obrazu, uśredniania, binaryzacji oraz etykietowania, tj. oznaczania spójnych podobszarów danych na mapie logicznej.

W tym celu we większości poniższych skryptów animacja zadana, będzie ‘rozkładana’ na pojedyncze obrazy z użyciem funkcji frame2im oraz poddawana przekształceniom, celem złożenia ponownej animacji z przekształconą zawartością obrazów 2D.

```
%Artur Bernat all rights reserved
%a script for binarization of movie frames contents
function [Mbw]=graymovie2bw(M);
fr=size(M,2);                 %get number of frames in movie
Mbw=moviein(fr);              % frames reservation for new movie
for i=1:fr,                   % for loop
    [obr,mp]=frame2im(M(i));%get current frame from given movie
    obrgray=rgb2gray(obr);  %RGB to gray convertion
    imbw=im2bw(obrgray,0.5);% thresholding in binarization at 0.5 lvl
    imshow(imbw);           % show in current display window
    %L=bwlabel(imbw,4);
    Mbw(:,i)=getframe;      %get frame into i-cell in newly formed movie
end;
movie(Mbw);                   %just show once again movie
movie2avi(Mbw,'BWavis.avi','compression','Indeo5');
close                         %saving in
```

Rys 14 Skrypt graymovie2bw.m binaryzacji ciągu obrazów 2D z zadanej animacji



Rys 15 Rezultaty wykonania skryptu graymovie2bw.m w zadaniu binaryzacji.

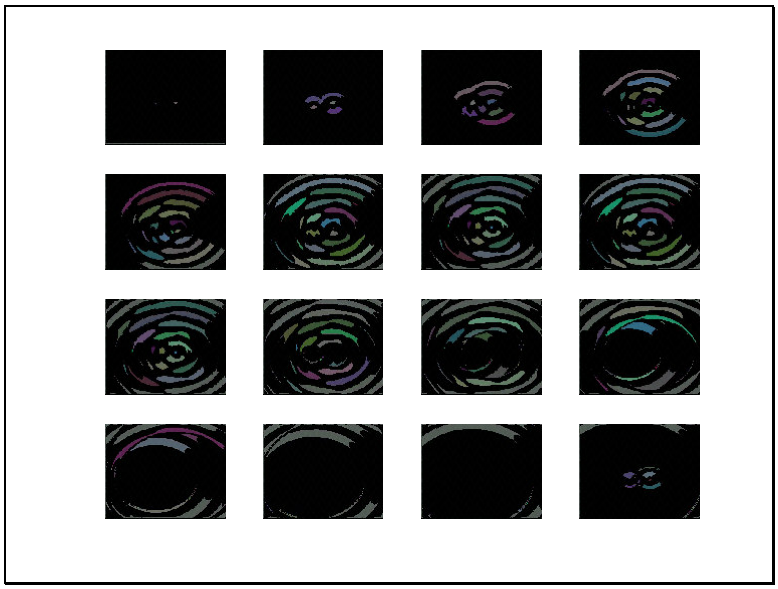
Na poziomie środowiska Matlab 6.1 nie istnieje implementacja funkcji graythresh dokonująca automatycznego doboru progu binaryzacji według kryterium Otsu. Dla prostoty i zgodności z różnymi wersjami środowiska Matlab wybrano arbitralnie próg równy 0.5.

W praktyce, często istnieje konieczność oznaczenia rejonów na wynikowej mapie, logicznie reprezentujących spójne podobszary bieli na czarnym tle. Stosuje się przy tym, kryterium sąsiedztwa równe 4 lub 8, a oznaczające liczbę pikseli, w najbliższym otoczeniu bieżąco badanego punktu, biorących udział w tworzeniu jednego spójnego podobszaru danych.

```
%Artur Bernat all rights reserved
%a script for labelling consistent regions on binarized image
function [Mlabel]=graymovie2bwlabel(M);
fr=size(M,2);                 %frame in movie M to be processed
Mlabel=moviein(fr);           % fr frame reserved in new Mlabel movie
grs=(randn(190,3)+3.3)./9.9; %generating random set of elements in palette
grs(1,:)=0;                   %1st index treated here as a background
max(grs(:))                   %just info
for i=1:fr,                   %in for loop
    [obr,mp]=frame2im(M(i)); %get current frame to be processed
    obrgray=rgb2gray(obr);    %convert RGB to gray represantion
    imbw=im2bw(obrgray,0.6); %arbitrary set level of thresholding
  % imshow(imbw);             %first show binarized image
    L=bwlabel(imbw,4);        %labelled of regions with 4 connectivity param.
    max(L(:))                 %just info
    imshow(uint8(L),double(grs)); %labelled 2D contents showed as indexed image
    whos L grs                %just info
    Mlabel(:,i)=getframe;     %get frame to new Mlabel movie of labelled contents
end;
movie(Mlabel);
movie2avi(Mlabel,'Movie_label.avi','compression','Indeo5');
close
```

Rys 16 Skrypt graymovie2bwlabel.m binaryzacji ciągu obrazów 2D z zadanej animacji

W skrypcie powyższym zastosowano losowo generowaną paletę kolorów w liczności 190. Z uwagi na to, że wywołanie funkcji bwlabel wykrywa do około 50-60 elementów na szeregu rozważanych obrazów 2D, w ten sposób przygotowana paleta kolorów jest wystarczająca w wizualizacji:



Rys 17 Rezultaty wykonania skryptu graymovie2bwlabel.m w zadaniu etykietowania podobszarów.

```
%Artur Bernat all rights reserved
%a script for edge detection
%M <= movie to be processed
%opt<=1: Prewitt's detector, 2: Canny's detector
function [Medge]=graymovie2edge(M,opt);
fr=size(M,2);
Medge=moviein(fr);
gr=[0];
grs=[gr' gr' gr'];
for i=1:fr,
    [obr,mp]=frame2im(M(i));
    obrgray=rgb2gray(obr);
    switch opt
    case 1
     edgegray=edge(obrgray,'prewitt');
    case 2
     edgegray=edge(obrgray,'canny');
 end;
    imshow(edgegray);
    Medge(:,i)=getframe;
end;
movie(Medge);
movie2avi(Medge,'Movie_edge.avi','compression','Indeo5');
close
```

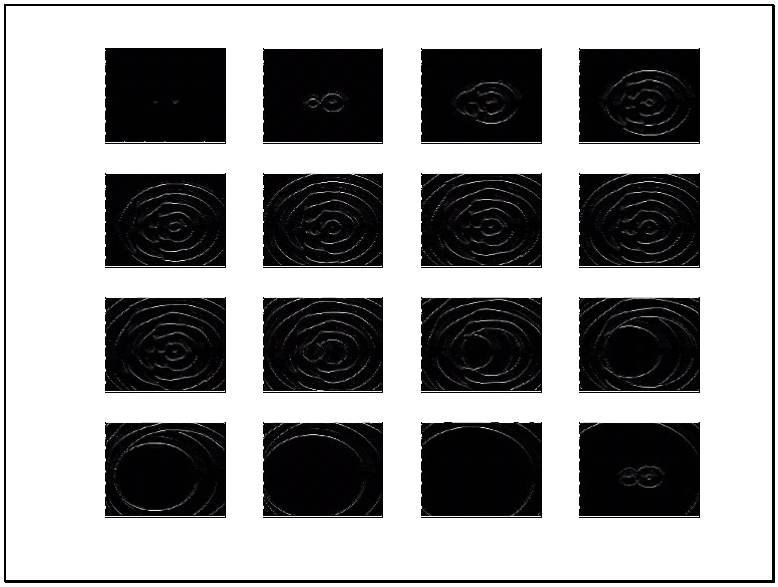
Rys 18 Skrypt graymovie2edge.m detekcji krawędzi na obrazach 2D z zadanej animacji

W detekcji krawędzi na treści obrazu 2D można wykorzystać funkcję edge, która na poziomie środowiska Matlab 6.1 posiada dwie wbudowane opcje ‘prewitt’ oraz ‘canny’. Oznaczają one metodę oraz sposób formowania maski, służącej w detekcji krawędzi, a swoje nazwy zawdzięczają wprost od nazwisk swoich autorów. W praktyce jednak w szerszym zestawie dostępnych opcji przekształceń na danych obrazów 2D, stosuje się funkcję fspecial w formowaniu maski przekształceń oraz funkcję filter2 ogólnych przekształceń wykorzystujących splot. Stąd funkcja filter2 w części korzysta z wywołania w swoim wnętrzu funkcji conv2 (splot dwuwymiarowych danych). Poniżej podano przed wynikami działania skryptu z rysunku 14 treść skryptu graymovie2filter.m, służącego wielorakim celom w przekształceniach i wyodrębnianiu cech na obrazach 2D.

```
%Artur Bernat all rights reserved
%a script for filtering of the frames contents
function [Mf]=graymovie2filter(M,op);
fr=size(M,2);                %get number of frames in movie
if op==[]
op=4;
end;
switch op
case 1
    h=fspecial('gaussian',1,0.5);
case 2
    h=fspecial('sobel');
case 3
    h=fspecial('prewitt');
case 4
    h=fspecial('laplacian');
case 5
    h=fspecial('log');
case 6
    h=fspecial('average');
case 7
    h=fspecial('unsharp');
otherwise
    h=special('gaussian',1,1);
end;
Mf=moviein(fr);             % frames reservation for new movie
for i=1:fr,                  % for loop
    [obr,mp]=frame2im(M(i));%get current frame from given movie
    obrgray=rgb2gray(obr);  %RGB to gray convertion
    if op==7
     imfltrN=imnoise(obrgray,'salt & pepper',0.02);
     imfltr=filter2(h,imfltrN)/255;
     subplot(1,2,1);imshow(imfltr);subplot(1,2,2);imshow(imfltrN);
    else
     imfltrd=filter2(h,obrgray)/255;           %mask for blurring gradually enlarged
     imshow(imfltrd);
    end;

    Mf(:,i)=getframe;       %get frame into i-cell in newly formed movie
end;
movie(Mf);                  %just show once again movie
movie2avi(Mf,'MovieFiltered.avi','compression','Indeo5');
close                        %saving in
```

Rys 19 Skrypt graymovie2filter.m detekcji krawędzi na obrazach 2D z zadanej animacji



Rys 20 Rezultaty wykonania skryptu graymovie2edge.m w detekcji krawędzi z opcją ' prewitt'.