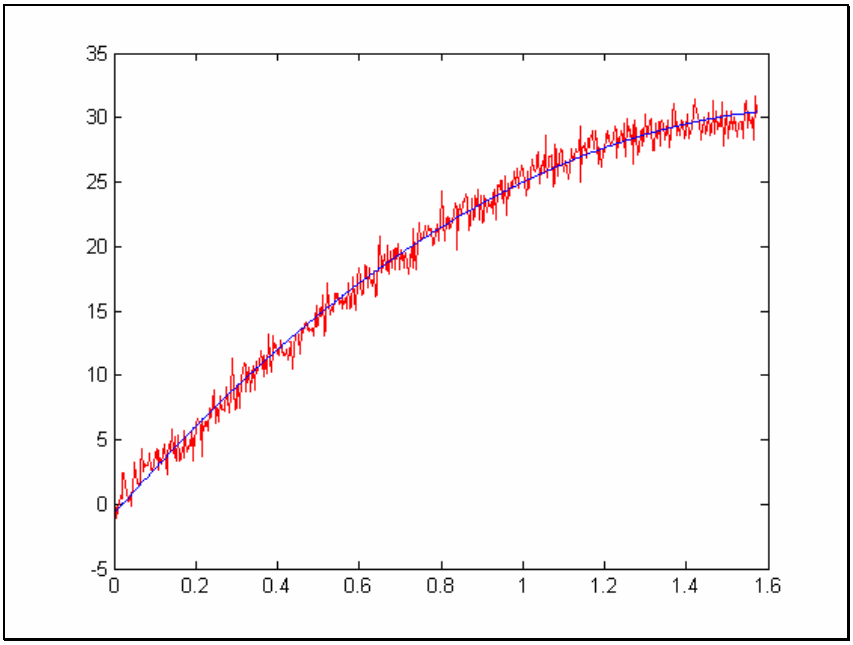
Wykład III<< zrealizowany w 2006, przedrostki-nazwy zretuszowane w Acrobat >>
z Podstaw Przetwarzania Informacji
(na danych obrazów 2D w środowisku Matlab 6.x 7.x)

W zadaniu prezentacji danych obrazów 2D, jak i wykresów 3D (*a szczególności danych rekonstrukcji powierzchni 3D na podstawie obrazów płaskich 2D*) tytułem wstępu będzie omówiona funkcja dopasowania wielomianem do przebiegu funkcji zadanej tabelarycznie.

Niech dany będzie przebieg sinusoidalny, w określonym przez użytkownika zakresie ([0,Π/2]) z nałożonym addytywnie przebiegiem szumu Gauss'owskiego. Wówczas wywołanie funkcji `polyfit` w skrypcie jak poniżej, umożliwi nam dopasowanie do przebiegu wielomianem II stopnia:

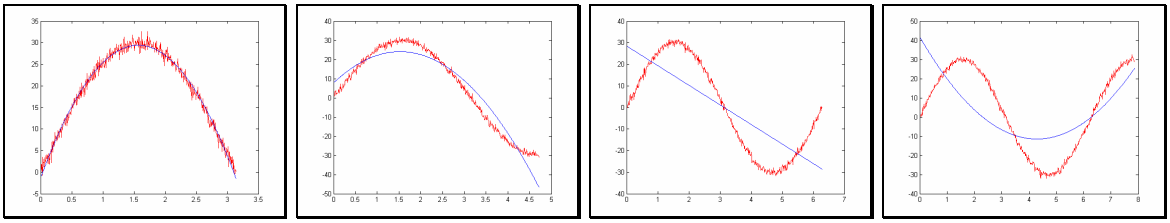
```
function [wykres]=dopasuj(zakres)% zakres => określenie przedzialu
 dziedzina=linspace(0,zakres,500);%500 PUNKTOW W DZIEDZINIE NP:<0,2*pi>
 Y=30*sin(dziedzina);              %przebieg sin liczony w zadanym przedziale
 X1=randn(500,1);                  %losowe 500 próbek szumu Gauss'owskiego
 przebieg=Y+X1';                   %przebieg wynikowy: zaszumiona sinusoida
%aproksymacja wielomianem 2-go stopnia: y=a*x^2+b*x+c
[wsp,struktura]=polyfit(dziedzina,przebieg,2);
%generacja punktów Krzywej otrzymanej w wyniku aproksymacji:
%a=wsp(1), b=wsp(2), c=wsp(3)
Krzywa=wsp(3)+wsp(2).*dziedzina+wsp(1).*dziedzina.*dziedzina;
%wykreślaj przebieg na czerwono, a przebieg wiel.II stopnia na niebiesko
plot(dziedzina,przebieg,'r',dziedzina,Krzywa,'b');
```

Rys1 Skrypt `dopasuj.m` zadaniu aproksymacji przebiegu.



Rys2 Na czerwono przebieg, na niebiesko aproksymacja wielomianem II stopnia.

Jednak, dla szerszego przedziału przebiegu zasadniczo periodycznego, aproksymacja wielomianem II stopnia jest nieefektywna:



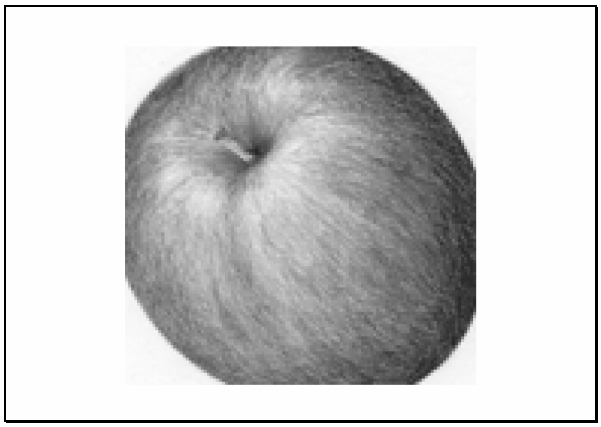
Rys3 Na czerwono przebieg, na niebiesko aproksymacja wielomianem II stopnia.
Widoczny brak dopasowania dla przedziału przebiegu okresowego w rozciągłości powyżej długości II.

Porzucając temat aproksymacji przebiegów krzywymi wielomianowymi wyższych stopni, obecnie można by się zastanowić, nad prostą rekonstrukcją zarysu mikronierówności na obrazie 2D.

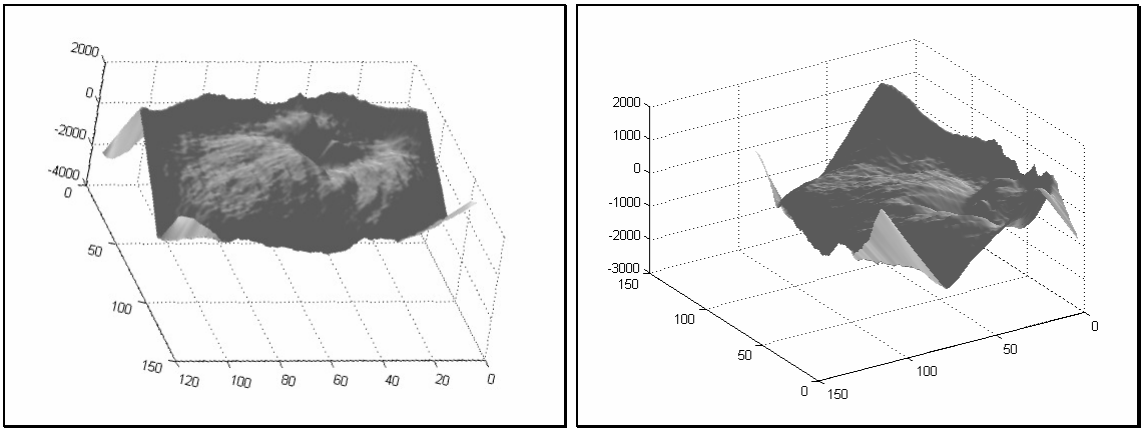
Można by, na przykład zsumowywać sukcesywnie intensywności luminancji w liniach poziomo lub pionowo. Powstała w ten sposób pochylona powierzchnia 3D danych wynikowych, mogłaby być następnie niwelowana w swoim pochyleniu, albo globalnie dla całej powierzchni, albo indywidualnie dla każdej z linii z osobna. To drugie rozwiązanie okazało się lepsze, a rezultaty choć nie są idealne, przypominają wyniki tych bardziej profesjonalnych rekonstrukcji 3D, już nie tylko dla mikronierówności, lecz również dla niektórych makroobiektów, typu twarz ludzka lub powierzchnia jabłka:

```
%skrypt naliczania sum kumulacyjnych z linii obrazu 2D
function [kcx]=mapalicz(obraz)
kx=cumsum(double(obraz),1); %naliczanie na I kierunku
for j=1:size(obraz,2);
    %szereg liczb naturalnych=>odpowiednik danych w linii
    y=1:size(obraz,1);
    %dopasowywanie wielomianem I stopnia do biezacej linii obrazu 2D
    [p,s]=polyfit(y',kx(:,j),1);
    %wyznaczanie szeregu punktów wielomianu I stopnia
    cx=p(2)+p(1)*y';
    %niwelacja pochylenia linii sum intensywności luminancji
    kcx(:,j)=kx(:,j)-cx;
end;
    %paleta kolorow 196-stopni gradacji szarosci
gr=linspace(0,1,196);
    %paleta szarosci kompletna dla kanalow R G B
grs=[gr' gr' gr'];
    %wywolanie funkcji surfl, prezentacji pow.3D z oswietleniem wirtualnym
surfl(kcx);shading interp;colormap(grs);
```

Rys4 Naliczanie sum kumulacyjnych i niwelacja pochylenia aproksymowanym wielomianem I stopnia



Rys5 Rysunek jabłka z peryferyjnymi obszarami bieli w 4 rogach



Rys6 Obiekt z rys.4 w rekonstrukcji 3D, dwa różne widoki z użyciem funkcji `surfl`.

W przypadku danych 3D, takich jak zrekonstruowane powierzchnie 3D, bardziej wskazane byłoby wykorzystanie mechanizmów generacji animacji z użyciem wywołania funkcji `getframe`, jak i możliwości cyklicznej zmiany parametrów obserwatora (lub/oraz wirtualnego oświetlenia) w prezentacji wyników z pomocą funkcji `surfl`. W tym celu możliwe jest wykorzystanie funkcji `view`, określającej kąt azymutu i elewacji dla punktu obserwatora:

```
function [M]=animuj(obraz)
kx=cumsum(double(obraz),1);
for j=1:size(obraz,2);
    y=1:size(obraz,1);
    [p,s]=polyfit(y',kx(:,j),1);
    cx=p(2)+p(1)*y';
    kcx(:,j)=kx(:,j)-cx;
end;
gr=linspace(0,1,196);
grs=[gr' gr' gr'];
surfl(kcx);shading interp;colormap(grs);
M=moviein(12); %rezerwacja klatek w animacji AVI
figure,
hold on;
for k=1:1:12,
    k*30
    view(k*30,45);%zmieniaj parametr azymutu w petli co 30 stopni
    surfl(kcx);shading interp;colormap(grs);
    M(:,k)=getframe; % lap klatku
end;
figure,movie(M);%odtworzenie probne animacji
% zapis na dysk z 4 klatkami/sek, w standardzie Indeo5 kompresji
movie2avi(M,'klatki_avi.avi','FPS',4,'COMPRESSION','Indeo5');
```

Rys7 Skrypt rekonstrukcji 3D i animacji z obrotem azymutalnym punktu obserwatora

Alternatywnym sposobem prezentacji danych wynikowych 3D, może być w niektórych sytuacjach scenariusz z nieruchomym punktem obserwacji, przy zmienianym kierunku rzutowanego światła. W skrypcie prezentowanym poniżej, dodatkowo zdecydowano się na wprowadzenie własnych arbitralnie wybranych ustawień modelu odbiciowego powierzchni 3D. Powierzchnia 3D prezentowana, będzie się wykazywać w strumieniu światła odbitego niewielkim stosunkowo współczynnikiem odbicia światła rozproszonego (k_a=0.1), dość

dużym stosunkowo współczynnikiem odbicia matowego światła kierunkowego (*k_d*=0.5) oraz istotnym, lecz nieprzeważającym w całości generowanej animacji AVI, udziałem połyskliwości (współczynniki modelu Phong'a *ks*=0.1 oraz *ksn*=0.1). Po zadeklarowaniu tych czterech powyższych parametrów ustawienia własności odbiciowych powierzchni dokonuje się wywołaniem funkcji `material`. Natomiast wirtualne źródło światła może być inicjowane(pierwsze źródło oraz kolejne jednocześnie naświetlające powierzchnię 3D) poprzez wywołanie funkcji `light`:

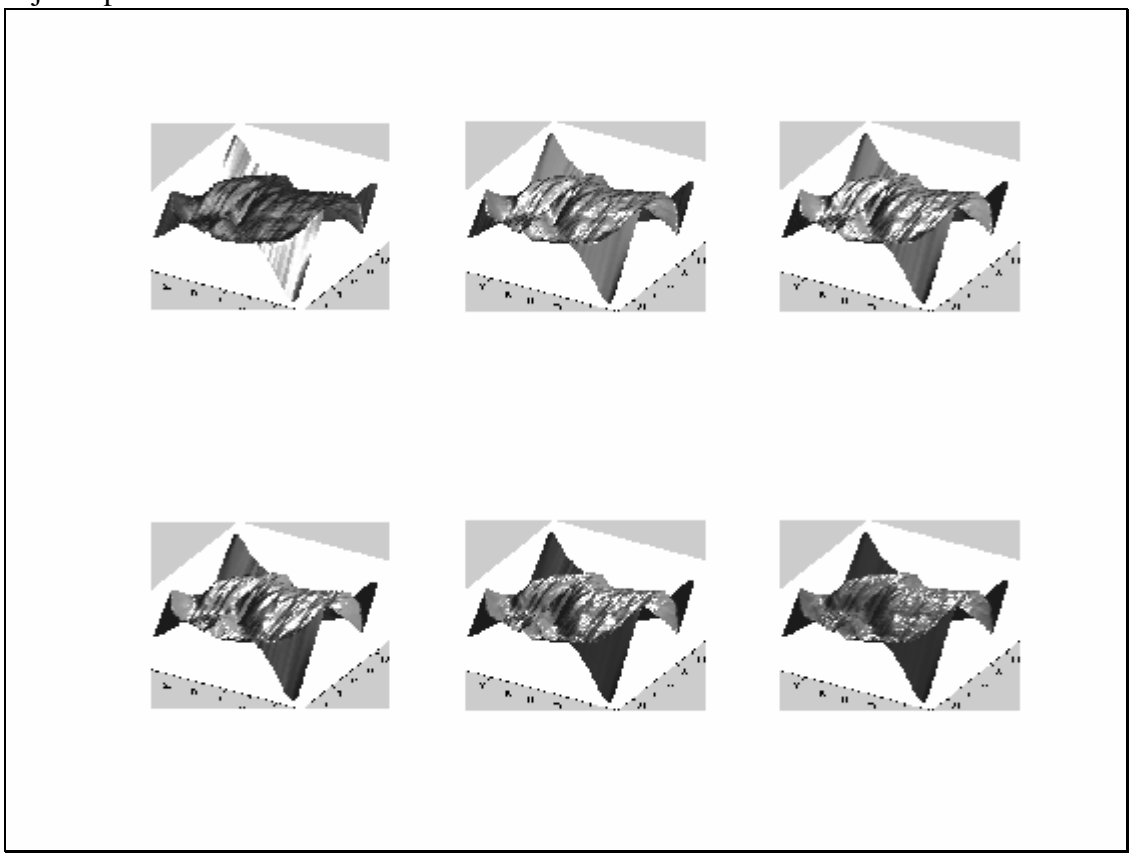
```
function [M]=animuj_kierunek_swiatla(obraz)
kx=cumsum(double(obraz),1);
for j=1:size(obraz,2);
    y=1:size(obraz,1);
    [p,s]=polyfit(y',kx(:,j),1);
    cx=p(2)+p(1)*y';
    kcx(:,j)=kx(:,j)-cx;
end;
gr=linspace(0,1,196);
grs=[gr' gr' gr'];
surfl(kcx);shading interp;colormap(grs);
M=moviein(24); %rezerwacja klatek w animacji AVI
figure,
hold on;
view(30,45);%staly punkt obserwacji k.azymutu:30,elewacji:45
k_a=0.1; %wsp. odbicia swiatla rozproszonego
k_d=0.5; %wsp. odbicia matowego swiatla kierunkowo rzutowanego
k_s=0.1; %wsp. multiplikatywny modelu Phong'a
k_n=0.1; %wsp. ekspotencjalny modelu Phong'a
material([k_a k_d k_s k_n]); %ustawienia wl. odbiciowych powierzchni
 L1=light;%stworz wirtualne swiatlo
for k=1:12,
    k*30
    lightangle(L1,k*30,45);%zmieniaj parametr azymutu w petli co 30 stopni
    surfl(kcx);shading interp;colormap(grs);
    M(:,k)=getframe; % lap klatku
end;
figure,movie(M);%odtworzenie probne animacji
% zapis na dysk z 4 klatkami/sek, w standardzie Indeo5 kompresji
movie2avi(M,'klatkiB_avi.avi','FPS',4,'COMPRESSION','Indeo5');
```

Rys8 Skrypt rekonstrukcji 3D i animacji ze zmianą cykliczną kierunku światła rzutowanego, z udziałem mieszanym odbicia matowego i połyskliwego

Powyższy skrypt tworzy 12 klatek animacji z kątem azymutalnym kierunku rzutowanego światła zmienianym w pętli `for` co 30 stopni, a powstałą animację o nazwie powiedzmy *M* można wtórnie *'rozłożyć'* na 12 obrazów indeksowanych (tj. w wydzielaną indywidualnie paletą kolorów), co uczyniono w następującej pętli:

```
for i=1:12,
 [X,Map]=frame2im(Mx(i));
 subplot(3,4,i);imshow(X);
end;
```

Jednakże, dla prezentacji w tym dokumencie wyników generacji tej animacji przytoczono jedynie sześć pierwszych klatek animacji w formie wykresu globalnego z wywołaniem funkcji subplot:



Rys9 Pierwszych 6 klatek animacji AVI z kątem azymutu rzutowanego światła zmienianym, co 30 stopni

Wracając do wykładu wstępnego II w temacie omawianym zastosowań *Image Processing Toolbox*, tworząc animację z kolejno coraz bardziej rozmywanych obiektów w polu fotografowanym, można wykorzystać wywołanie funkcji `blockproc`:

```
est4x4=blkproc(im,[4 4],'min(double(x(:)))');
est8x8=blkproc(im,[8 8],'min(double(x(:)))');
est16x16=blkproc(im,[16 16],'min(double(x(:)))');
est32x32=blkproc(im,[32 32],'min(double(x(:)))');
est64x64=blkproc(im,[64 64],'min(double(x(:)))');
```

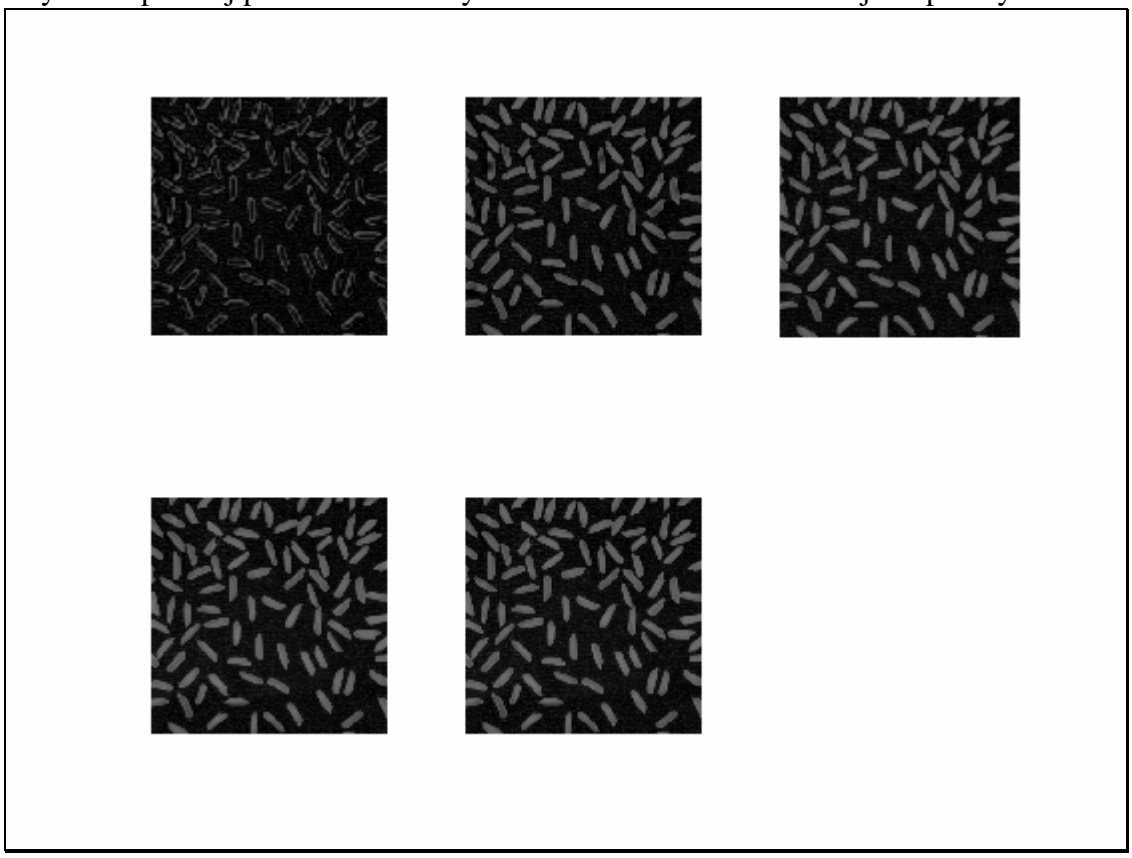
Wywołanie funkcji `blkproc` razem z funkcją oceny poszukiwanej wartości w podblokach o wielkości [4 4] lub [8 8] itp, pozwala w szczególności na określenie średniej intensywności tła w podblokach obrazu o zadanej wielkości (z wywołaniem `min` oraz pośrednio poprzez wyliczenie kolumnowe wszystkich elementów obrazu 2D). Problemem tylko pozostaje wielkość map wynikowych estymat średnich wartości tła(tj. średnich wartości minimalnych w podblokach). Należy zatem przeskalować mapy tych szacunkowych minimalnych intensywności do rozmiaru oryginalnego obrazu:

```
>> imest4x4=imresize(est4x4,[256 256],'bicubic');
>> imest8x8=imresize(est8x8,[256 256],'bicubic');
>> imest16x16=imresize(est16x16,[256 256],'bicubic');
>> imest32x32=imresize(est32x32,[256 256],'bicubic');
>> imest64x64=imresize(est32x32,[256 256],'bicubic');
```

Z użyciem funkcji `subtract` można następnie od treści oryginalnego obrazu 2D odjąć intensywności z map estymat intensywności minimalnych w podblokach, a z całości złożyć animację:

```
M=moviein(5);
figure,imshow(imsubtract(im,uint8(imest4x4)));
M(:,1)=getframe;
imshow(imsubtract(im,uint8(imest8x8)));
M(:,2)=getframe;
imshow(imsubtract(im,uint8(imest16x16)));
M(:,3)=getframe;
imshow(imsubtract(im,uint8(imest32x32)));
M(:,4)=getframe;
imshow(imsubtract(im,uint8(imest64x64)));
M(:,5)=getframe;
movie(M);
```

Na rysunku poniżej przedstawiono wyniki w formie 5 klatek animacji na podwykresach:



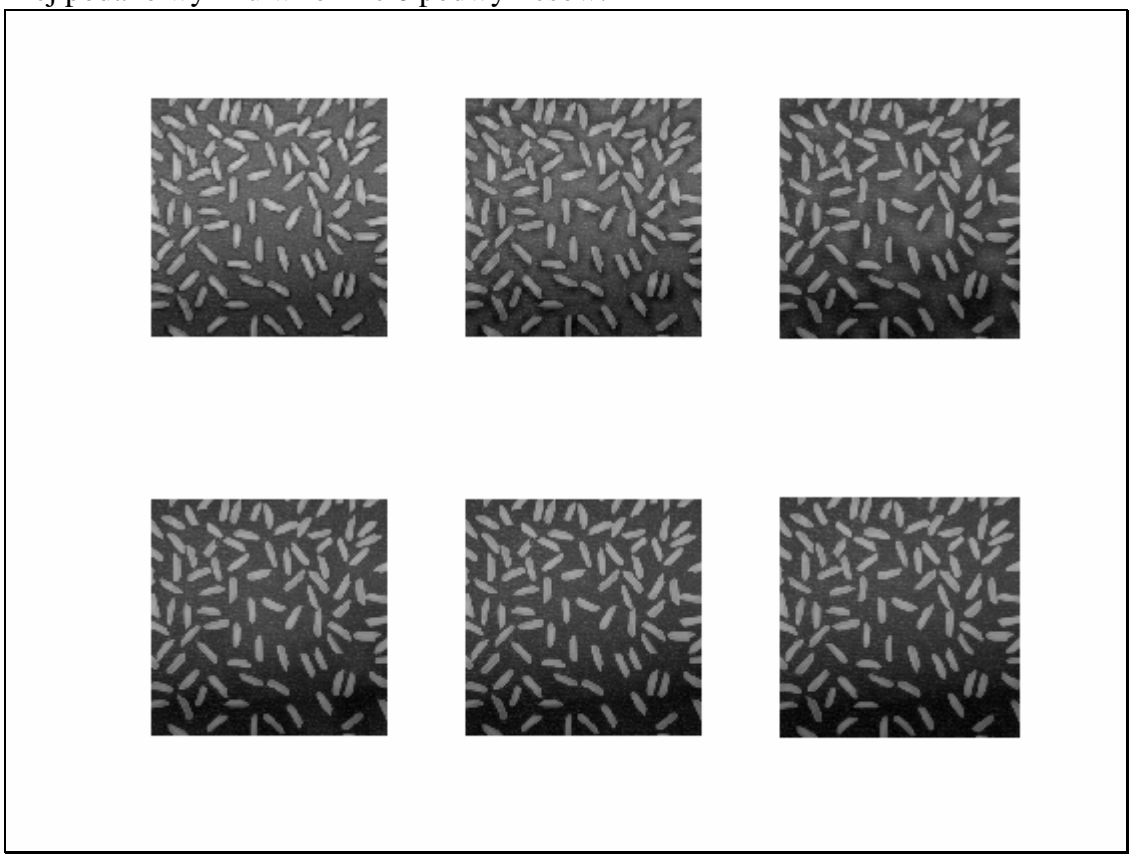
Rys10 Kolejne klatki animacji wyników oceny i niwelacji tła w podblokach na obrazie `rice.png`

Oczywiście wywołanie funkcji blkproc można zaimplementować z wywołaniami innych prostych funkcji estymacji wartości statystycznych w podblokach typu: max, std, mean, median:

```
std_est4x4=blkproc(im,[4 4],'std(double(x(:)))');
>> std_est8x8=blkproc(im,[8 8],'std(double(x(:)))');
>> std_est16x16=blkproc(im,[16 16],'std(double(x(:)))');
>> std_est32x32=blkproc(im,[32 32],'std(double(x(:)))');
>> std_est64x64=blkproc(im,[64 64],'std(double(x(:)))');
>> std_est128x128=blkproc(im,[128 128],'std(double(x(:)))');
```

```
>> imSTDest4x4=imresize(std_est4x4,[256 256],'bicubic');
>> imSTDest8x8=imresize(std_est8x8,[256 256],'bicubic');
>> imSTDest16x16=imresize(std_est16x16,[256 256],'bicubic');
>> imSTDest32x32=imresize(std_est32x32,[256 256],'bicubic');
>> imSTDest64x64=imresize(std_est64x64,[256 256],'bicubic');
>> imSTDest128x128=imresize(std_est128x128,[256 256],'bicubic');
>> M=moviein(6);
```

Poniżej podano wyniku w formie 6 podwykresów:



Rys11 Kolejne klatki animacji wyników z próbą niwelacji informacji o odchyleniu standardowym intensywności luminancji w podblokach na obrazie `rice.png`

Jak wynika z rysunku 11, być może korzystniej byłoby zaprezentować samą informację o standardowym odchyleniu intensywności luminancji pikseli w podblokach, niż dokonywać jednoczesnej niwelacji poprzez odejmowanie tej estymaty od treści obrazu zasadniczego.