Notatki wykładowe dot. przeglądu metod postrzegania komputerowego w kontekście wykorzystania metod wieloobrazowych (stereoskopii fotometrycznej).

Na początek, rozważania zaczniemy od zwięzłego omówienia metod generowania powierzchni losowych. Zagadnienie generacji losowych powierzchni znajduje szerokie zastosowanie, w ogólności:

- w modelowaniu powierzchni losowych – rozważanych z czysto matematycznego punktu widzenia, zwłaszcza z uwzględnieniem właściwości losowych powierzchni, jako takich w ujęciu statystycznym

- w odniesieniu modelowanych powierzchni dwuwymiarowych do prezentacji, przedstawień oraz analiz rozkładów losowych dwuwymiarowych zjawisk, opisywanych, jako te o charakterze ciągłym argumentu (ciągłej funkcji gęstości prawdopodobieństwa).

Ponadto, zagadnienie generacji losowych powierzchni znajduje w szczególności zastosowanie między innymi w świecie obiektów makroskopowych (makro skali):

- w opisie właściwości trybologicznych powierzchni technicznych, zwłaszcza w modelowaniu zjawisk i skutków kontaktu sprężystego, pół-plastycznego oraz plastycznego dwóch ciał

- w opisie właściwości statystycznych, jak i dynamicznych zmian na przykład powierzchni dużych akwenów wodnych (połaci mórz i oceanów) w czasie sztormowej pogody

- w zarówno opisie matematycznym, jak i wnioskowaniu i prognozowaniu zmian topografii dna mórz i oceanów, przy ograniczonym zbiorze danych reprezentatywnych

- w geomorfologiii, to znaczy nauce pokrewnej, względem dziedziny badania topografii tworów geologicznych na powierzchni lądów i dna akwenów wodnych oraz na powierzchniach ciał stałych astronomicznych

- w ocenie stopnia poprawności i wierności odwzorowywania topografii tworów geologicznych w postaci cyfrowo zapisanych modeli powierzchni badanych, odwzorowywanych (na przykład mapowaniu satelitarnym powierzchni ciał stałych: astronomicznych, jak i samej Ziemi)

- w ocenie wielokryterialnej powierzchni średnio oraz mocno-zurbanizowanych (chociaż tutaj mogą przeważać badania wykorzystujące metodę analizy spektralnej widma strumienia światła odbitego od powierzchni badanej nad metodami rekonstrukcji 3D oraz odwzorowywania topografii powierzchni postrzeganej z pomocą środków wywiadu satelitarnego)

- w ocenie stopnia i jakości upraw (na przykład plantacji niektórych roślin uprawowych), ponadto: w ocenie stopnia i jakości zalesienia obszarów dzikich (na przykład połaci dorzecza Amazonii), które w rzeczy samej dotyczą wysokości, rozpiętości, jak i gęstości upakowania konarów drzew w obszarach zalesionych oraz ponadto dynamiki zmian czasowych górnej warstwy poszycia obszarów zalesionych, często prowadzonych z dokładnością do pojedynczych drzew-konarów

- w ocenie stopnia nasilenia ruchu komunikacyjnego na głównych szlakach komunikacji koło-asfaltowej, jak i kołowo-szynowej, z uwzględnieniem na przykład oceny zmian pobocza pasów komunikacyjnych

- w ocenie topografii terenów zurbanizowanych, dotyczących bezpośrednio miejsc występowania szlaków komunikacyjnych, skrzyżowań, jak i samych budynków
- w monitorowaniu stanu wód, często realizowanych w połączeniu z oceną i monitoringiem terenów silnie zurbanizowanych, na przykład w deltach ujść rzek.

Ponadto opis właściwości topografii powierzchni badanych, połączony, lub też nie, z wykorzystaniem metod postrzegania komputerowego może dotyczyć również obiektów mikro- mezo- i nano-skali.

Według kategoryzacji autora:

- obiekty makroskali to obiekty, które rozmiarami mikro-chropowatości, chropowatości, jak i zarysu nierówności i konturów powierzchni postrzeganych mogą znacznie przekraczać rozmiary ludzkiego obserwatora. To znaczy są to z reguły obiekty w postaci dróg, szlaków komunikacyjnych, budynków i ich elewacji oraz znacznie większe, dotyczące obiektów rozważanych w geomorfologii – twory geologiczne – łańcuchy, przełęcze, uskoki, doliny i kotliny górskie oraz powierzhni ciał stałych astronomicznych.

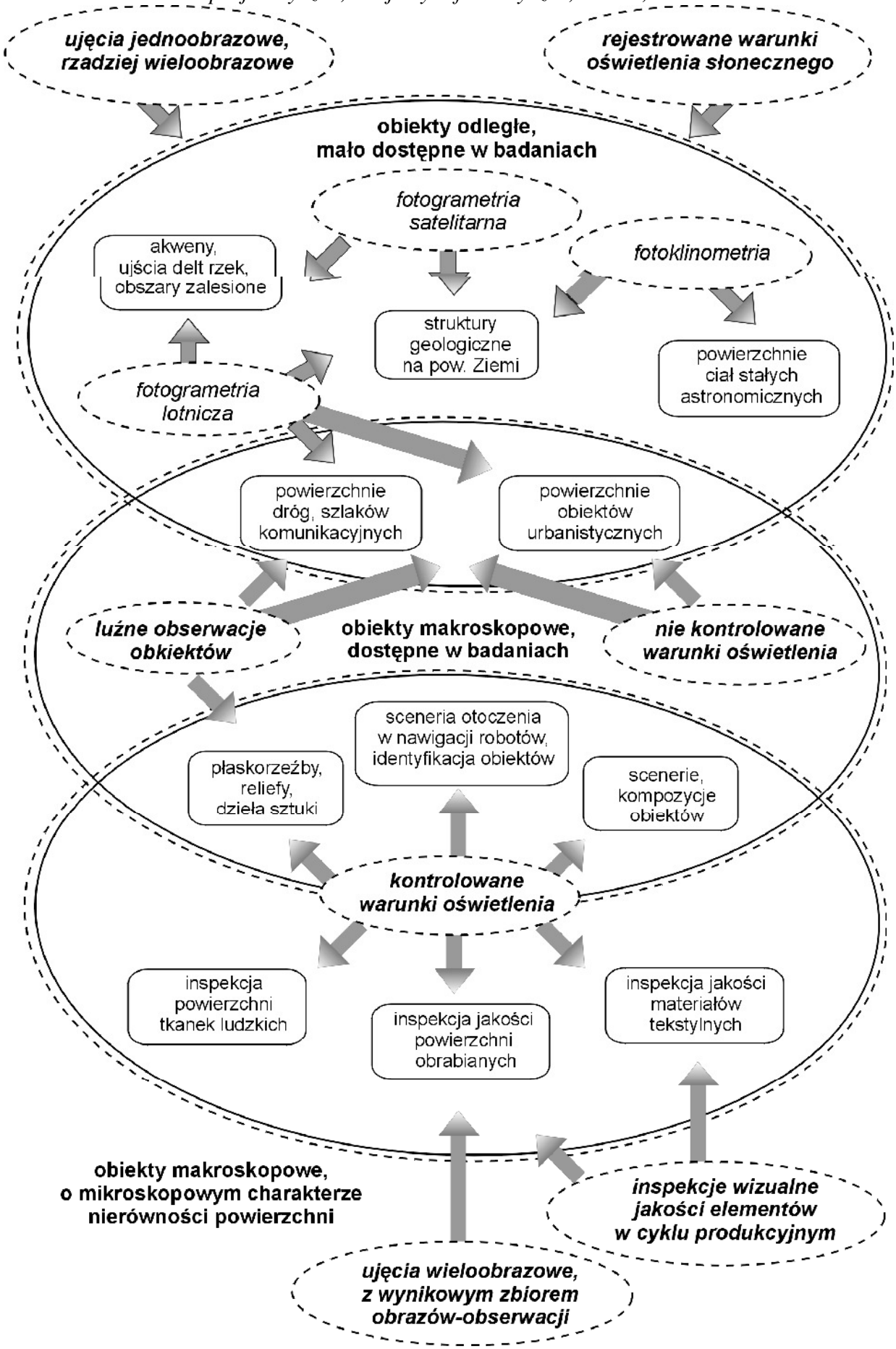
- obiekty mikroskali to obiekty bezpośredniego otoczenia dnia codziennego ludzkiego obserwatora. To znaczy z reguły są to płaskorzeźby, reliefy, elementy architektury budynków i budowli różnych epok, małe przenośne dzieła sztuki, elementy ruchome i przenośne, podręczne dnia codziennego, powierzchni nasion, warzyw i owoców, przetworów i tkanek zwierzęcych produktów spożywczych, elementów gier polowych i zabawowych (na przykład piłek golfowych), posążków, pamiątek, maskotek i zabawek. W tychże wszystkich spotykanych elementach otoczenia dnia codziennego mamy do czynienia zarówno z makro-, jak i mikro-chropowatoscią powierzchni obiektów, które można postrzegać jeszcze nieuzbrojonym okiem.

-obiekty mezoskali w postrzeganiu komputerowym, to obiekty bezpośredniego otoczenia dnia codziennego, w których jednakże mikro-chropowatość i śledzenie zmian mikrotopografii powierzchni zazwyczaj wymaga już stosowania powiększających elementów optycznych (mikroskopów optycznych). Są to między innymi powierzchnie tekstyliów, powierzchni implantów, niektórych powierzchni technicznych, wykonanych na przykład z niektórych gatunków ceramiki technicznej (o znacznej chropowatości) oraz wszelkie materiały służace w obróbce ścieraniem (ścierniwo luźne oraz wiązane na powierzchniach narzędziach ściernych). Ponadto, powierzchnie skał, warstw i tworów osadowych, minerałów w ogólności, rozważanych w przeciwieństwie do tworów geomorfologii, na poziomie mikro-chropowatości powierzchni, która to wymaga mikroskopowych urządzeń optycznych

- obiekty nanoskali, wymagają od ludzkiego obserwatora, by ten dysponował wyłącznie urządzeniami wspomagania obserwacji, urządzeniami optycznymi lub nieoptycznymi obserwacji zarówno chropowatości, jak i mikrochropowatości powierzchni. Są to dla przykładu wygładzane powierzchnie z ceramiki technicznej, powierzchnie lustrzane, lub w ogólności powierzchnie odbić bezpośrednich o niskich wartościach amplitudowych parametrów chropowatości, sięgających dziesiętnych części mikrometra, lub znacznie poniżej mikrometra.

Należy podkreślić, że omawiane poniżej metody komputerowego postrzegania, z uwzględnieniem jedynie modelowania skutków niefalowego, liniowego rozchodzenia się promieni światła w rzutowaniu i odbiciu od powierzchni badanych, w przybliżeniu mogą odnosić się do jedynie pierwszych trzech rozważanych powyżej sytuacji, z pominięciem powierzchni obiektów nanoskali, gdzie natura falowa rozchodzenia się światła przeważa nad naturą korpuskularną (cząstkową fotonów) modelowanych zjawisk rozchodzenia się i odbicia światła.

Poniżej przedstawiono uogólnioną kategoryzację metod postrzegania komputerowego z uwzględnieniem nie tylko zmieniającej się skali obserwowanych obiektów, lecz również różnych warunków oświetlenia powierzchni obiektów badanych, zarówno pod względem liczby, jak i jakości zastosowanych źródeł promieni rzutowanych, jak i w ogólności warunków akwizycji danych obrazów płaskich luminancji (*ang. intensity images*).
Stwierdzono pewną tendencję zmierzającą do zastosowań metod wieloobrazowych, przy jednocześnie dość ściśle kontrolowanych warunkach oświetlenia i akwizycji danych, niż metod jednoobrazowych i niekontrolowanych warunków oświetlenia i akwizycji danych obrazów luminancji, w miarę zmniejszania skali obiektów postrzeganych.
Uwaga: niekontrolowane warunki oświetlenia, niekoniecznie muszą oznaczać brak przesłanek, co do jednoznacznego stwierdzenia parametrów kierunkowych rzutowanego światła, czy właściwości spektralnych i natężeniowych strumienia światła rzutowanego. Niekontrolowane warunki oświetlenia oznaczają najczęściej możliwość ścisłego określenia tychże elementów a priori, bez jednakże możliwości uniwersalnego i wszechwładnego doboru tychże warunków oświetlenia w momencie powziętej decyzji o akwizycji danych obrazu.



Rys. 1 Kategoryzacja metod postrzegania w kontekście skali obiektów obserwacji, warunków i metod odpowiednio akwizycji i przetwarzania danych obrazów luminancji.

Na przykład położenie Słońca nad lokalnym horyzontem powierzchni Ziemi w dowolnym czasie (według czasu lokalnego, uniwersalnego lub gwiezdnego) jest możliwe do określenia z dość dużą dokładnością i jednoznacznością,

co jednakże nie oznacza automatycznie, że warunki tego oświetlenia zawsze muszą być optymalne względem celu obserwacji, jak i właściwości odbicia światła oraz ułożenia względem punktu obserwatora powierzchni obiektu obserwacji nie muszą być optymalne.

Metody generacji powierzchni losowych osadzone w kontekście postrzegania komputerowego mogą, lecz nie muszą koniecznie wymagać posiadania pewnego zasobu wiedzy apriorycznej.

W przypadku powierzchni czysto losowych, o izotropowości (równomierności) wybranych parametrów generowanej powierzchni, bez względu na kierunek rozważany przekroju – profilu 2D, parametry rozkładu normalnego: wierzchołków, wzniesień, czy nachyleń – zboczy nierówności mogą okazać się wystarczające w opisie podstawowych wielkości statystycznych powierzchni. Natomiast w ujęciu dwuwymiarowym dodatkowo taki opis może być wsparty parametrem opisującym gęstość powierzchniową wierzchołków i wzniesień na powierzchni badanej.

W przypadku powierzchni losowych, o anizotropowości (nierównomierności) wybranych parametrów generowanej powierzchni, ze względu na rozważany kierunek – parametru opisu profilu 2D, mogą okazać się niewystarczające, oprócz ponadto zazwyczaj występującym istotnym odstępstwie od rozkładu normalnego wysokości wierzchołków i wzniesień na powierzchni rozważanej.

Oczywiście rozkład inny niż normalny, a w szczególności złożenie kilku rozkładów normalnych o różnym odchyleniu standardowym i średnich może dotyczyć zarówno powierzchni losowych izotropowych, jak i anizotropowych. To ostatnie zagadnienie wiąże się jednakże zarówno z bardziej złożonym procesem generacji, jak i późniejszej identyfikacji powierzchni rozważanej (aktualnie znajdujących poza zasięgiem bieżących rozważań).

Poniżej przytoczono: algorytm i wynik generacji w założeniu losowej powierzchni o izotropowych właściwościach wybranych parametrów – reprezentantów estymowanej topografii. Ponadto tytułem wstępu do metod wieloobrazowych postrzegania komputerowego, w tym zagadnienia rekonstrukcji powierzchni 3D na podstawie zbioru obrazów płaskich luminancji przedstawiono podstawową ścieżkę przetwarzania danych w:

generacji powierzchni – renderingu obrazów 2D – rekonstrukcji powierzchni generowanej na podstawie obrazów 2D.

```
%Artur Bernat, All rights reserved, on 08.II.2010
%przykladowa lista argumentow:
%-----------------------------------------------
%[Zgen]=surface_generator(400,400,0.60,40,1.00,0.25,20,1.00,0.10,10,1.00,0.05,05,1.00);
%[Zgen]=surface_generator(400,400,0.75,40,1.00,0.15,20,1.00,0.07,10,1.00,0.03,05,1.00);
%-----------------------------------------------2 iteracje w generacji trendow T1,T2,T3,T4%
%[Zgen]=surface_generator(400,400,0.60,40,1.00,0.25,20,1.00,0.10,10,1.00,0.05,05,1.00,2);
%[Zgen]=surface_generator(400,400,0.75,40,1.00,0.15,20,1.00,0.07,10,1.00,0.03,05,1.00,2);
%-----------------------------------------------
%-----------------------------------------------5 iteracji w generacji trendow T1,T2,T3,T4%
%[Zgen]=surface_generator(400,400,0.60,40,1.00,0.25,20,1.00,0.10,10,1.00,0.05,05,1.00,5);
%[Zgen]=surface_generator(400,400,0.75,40,1.00,0.15,20,1.00,0.07,10,1.00,0.03,05,1.00,5);
%-----------------------------------------------
function
[Zgen]=surface_generator(size_x,size_y,A1,T1,sigma1,A2,T2,sigma2,A3,T3,sigma3,A4,T4,sigma4,n_iters,interp_method)
;%
%--------------------------------
if nargin < 16
    interp_method = 'spline';
    imethod = 'spline';
end;
%--------------------------------
if nargin < 15
    n_iters = 1;
    interp_method = 'spline';
    imethod = 'spline';
end;
%--------------------------------
if nargin == 16
    switch interp_method
        case {'l','lin'}
            imethod = 'linear';
        case {'n','near'}
            imethod = 'nearest';
```

```
        case {'c','cub'}
            imethod = 'cubic';
        case {'s','spl'}
            imethod = 'spline';
        otherwise
            imethod = interp_method;
    end;
end;
%--------------------------------
[XY,YX] = meshgrid(1:size_x,1:size_y); % podstawowa siatka size_x na size_y elementow %
%--------------------------------
Nx1 = round(size_x/T1);                % w zaokragleniu: liczba punktow generacji dla pierwszej składowej o dł.
okresu T1%
Ny1 = round(size_y/T1);                % w zaokragleniu: liczba punktow generacji - rowniez na kierunku Y%
[XY1,YX1] = meshgrid(linspace(1,size_x,Nx1),linspace(1,size_y,Ny1));     % siatka generacji przebiegu o dl.
okresu T1%
Z1randn = randn_limits_3p3(Ny1,Nx1,sigma1,n_iters); %<=generacja w rozkladzie norm. o zakresie zmiennosci <-3.3
3.3> przy sigma=1%
Z1_interp = interp2(XY1,YX1,Z1randn,XY,YX,imethod);
%--------------------------------
Nx2 = round(size_x/T2);
Ny2 = round(size_y/T2);
[XY2,YX2] = meshgrid(linspace(1,size_x,Nx2),linspace(1,size_y,Ny2));
Z2randn = randn_limits_3p3(Ny2,Nx2,sigma2,n_iters);
Z2_interp = interp2(XY2,YX2,Z2randn,XY,YX,imethod);
%--------------------------------
Nx3 = round(size_x/T3);
Ny3 = round(size_y/T3);
[XY3,YX3] = meshgrid(linspace(1,size_x,Nx3),linspace(1,size_y,Ny3));
Z3randn = randn_limits_3p3(Ny3,Nx3,sigma3,n_iters);
Z3_interp = interp2(XY3,YX3,Z3randn,XY,YX,imethod);
%--------------------------------
Nx4 = round(size_x/T4);
Ny4 = round(size_y/T4);
[XY4,YX4] = meshgrid(linspace(1,size_x,Nx4),linspace(1,size_y,Ny4));
Z4randn = randn_limits_3p3(Ny4,Nx4,sigma4,n_iters);
Z4_interp = interp2(XY4,YX4,Z4randn,XY,YX,imethod);
%--------------------------------
%Zgen=0.6.*Z1_interp + 0.25.*Z2_interp + 0.10.*Z3_interp + 0.05.*Z4_interp;
Zgen=A1.*Z1_interp + A2.*Z2_interp + A3.*Z3_interp + A4.*Z4_interp;
%--------------------------------
gr=linspace(0,1,196);  %196 stopni(kompatybilne z zapisem Movie)
grs=[gr' gr' gr'];
surfl(Zgen);shading interp; colormap(grs);
%/////////////// main Mscript's end//////////////////////////////%
%Artur Bernat, on 08.II.2010
%input/output examplary calling arguments:
%-------------------------------------------------
%[XA,XB]=randn_limits_3p3(100,100,2); %<std:2 100x100 elementow
function [randn_datA,randn_datB]=randn_limits_3p3(size_x,size_y,sigma,n_iters);
%-------------------------------------------------
if nargin <4
    n_iters=5;
end;
randn_datA = 0;
randn_datB = 0;
%////////////////////////////////////////////////////////////////////////%
size_vector = size_x .* size_y;
for i = 1:n_iters,
 %-------------------------------------------------
 randn_dats  = randn(uint16(size_vector*1.1),1);
 randn_dat12 = randn_dats(randn_dats >= -3.3);
 randn_dat13 = randn_dat12(randn_dat12<= +3.3);
 randn_dat14 = randn_dat13(1:size_vector,1);
 randn_dat1  = sigma.*randn_dat14;
 %-------------------------------------------------
 randn_dat1 = reshape(randn_dat1,size_y,size_x);
 %-------------------------------------------------
 randn_datA = randn_datA + randn_dat1;
end;
randn_datA =randn_datA./n_iters;
%////////////////////////////////////////////////////////////////////////%
for i=1:n_iters,
 %-------------------------------------------------
 randn_dat2 = randn(size_vector,1);
 randn_dat2(randn_dat2 <= -3.3) = -3.3;
 randn_dat2(randn_dat2 >= +3.3) = +3.3;
 randn_dat2 = sigma.*randn_dat2;
 %-------------------------------------------------
 randn_dat2 = reshape(randn_dat2,size_y,size_x);
 %-------------------------------------------------
 randn_datB = randn_datB + randn_dat2;
end;
randn_datB = randn_datB./n_iters;
```

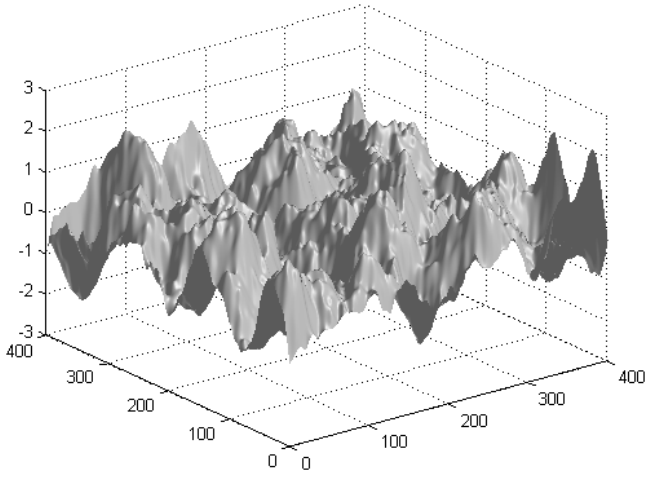
```
%//////////////////////////////////////////////////////////////////////////%
%-------------------------------------------------
%figure,
%if size_vector < 20000
%    size_vector
% subplot(121);hist(randn_datA(:),50);
% subplot(122);hist(randn_datB(:),50);
%else
%    size_vector
% subplot(121);hist(randn_datA(:),150);
% subplot(122);hist(randn_datB(:),150);
%end;
%-------------------------------------------------
%//////////second Mscript's end//////////////////////////////////////%
```

Powyższy skrypt *surface_generator.m* definiuje w przedrostkowym wydruk pomocy kilka przykładowych wywołań z odpowiednimi ciągami listy argumentów wejściowych i wyjściowych:

```
>>help surface_generator
 Artur Bernat, All rights reserved, on 08.II.2010
 przykladowa lista argumentow:
 -----------------------------------------------
 [Zgen]=surface_generator(400,400,0.60,40,1.00,0.25,20,1.00,0.10,10,1.00,0.05,05,1.00);
 [Zgen]=surface_generator(400,400,0.75,40,1.00,0.15,20,1.00,0.07,10,1.00,0.03,05,1.00);
 -----------------------------------------------2 iteracje w generacji trendow T1,T2,T3,T4%
 [Zgen]=surface_generator(400,400,0.60,40,1.00,0.25,20,1.00,0.10,10,1.00,0.05,05,1.00,2);
 [Zgen]=surface_generator(400,400,0.75,40,1.00,0.15,20,1.00,0.07,10,1.00,0.03,05,1.00,2);
 -----------------------------------------------
 -----------------------------------------------5 iteracji w generacji trendow T1,T2,T3,T4%
 [Zgen]=surface_generator(400,400,0.60,40,1.00,0.25,20,1.00,0.10,10,1.00,0.05,05,1.00,5);
 [Zgen]=surface_generator(400,400,0.75,40,1.00,0.15,20,1.00,0.07,10,1.00,0.03,05,1.00,5);
 -----------------------------------------------

    Published output in the Help browser
       showdemo surface_generator
>>
```

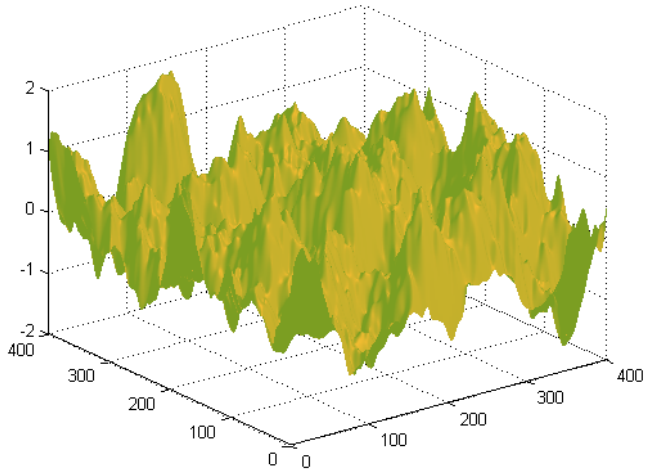
Wówczas, według zamysłu Autora, zwielokrotnione podwójne kliknięcie na jednej z linii przykładowego ciągu argumentów wejściowych/wyjściowych, dot. wywołania funkcji umieszczonej w skrypcie o tej samej nazwie, tj.: *surface_generator.m*, umożliwi na sprawne wywołanie tej funkcji z predefiniowaną listą argumentów, poprzez wciśnięcie klawisza funkcyjnego F9. Jest to operacja równoważna wywołaniu opcji *Evaluate Selection* z menu rozwijanego w rejonie okna poleceń, w reakcji na kliknięcie prawego przycisku myszy na zaznaczonej uprzednio całej linii polecenia.
Konsekwentnie, ta przedrostkowa konwencja, zastosowana w pisaniu komentarzy tekstu auto-pomocy jest stosowana przez Autora (zazwyczaj) wszędzie w poniższych wywołaniach funkcji ze słowem pomocy kontekstowej *help*.
Stąd:

```
[Zgen]=surface_generator(400,400,0.60,40,1.00,0.25,20,1.00,0.10,10,1.00,0.05,05,1.00);
surfl(Zgen);shading interp;colormap([linspace(0,1,196)' linspace(0,1,196)' linspace(0,1,196)']);
```



Lub, z użyciem palety odpowiadającej raczej odcieniom matowym miedzi pokrytej patyną:

```
[Zgen]=surface_generator(400,400,0.60,40,1.00,0.25,20,1.00,0.10,10,1.00,0.05,05,1.00);
surfl(Zgen);shading interp;colormap([linspace(0.2,1.0,196)' linspace(0.55,0.75,196)'
linspace(0.1,0.2,196)']);
```



Powyższe powierzchnie poddano wirtualnemu oświetleniu, z wykorzystaniem 3 źródeł punktowego światła o kącie początkowym azymutu równym 0 stopni oraz kącie elewacji równym dla wszystkich źródeł 60 stopni.
Oto plik kontekstowej auto-pomocy pewnej funkcji-m-skryptu *map_rend_Snx3_spec_16b.m*

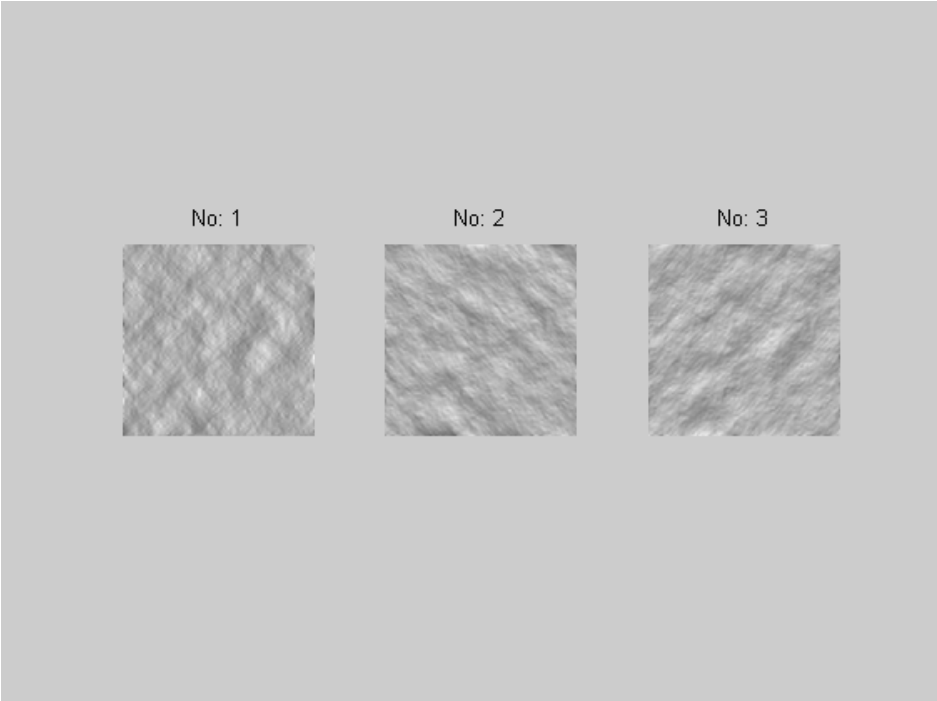
```
>>help map_rend_Snx3_spec_16b
 Artur Bernat
 virtual rendering with Snx3 n directional parameters
 RE-EDITED on 14_Nov_2006
 -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 each of the rendering scripts, have been reedited on 10May2009
 in order to embedde gradient function, instead of diff
 thus, consequently, rendered images are of the same size as depth surface
 -=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
 input data:
 -----------
 map  <= 3D map to be rendered
 ro   <= albedo value within [0;1] (homogenous surface)
 Snx3 <= n directional parameters for n light sources
 ks   <= multiplicative coeff. of specular reflection (Phong model)
 ksn  <= expotential coeff. of specular reflection (Phong model)
 ----------
 output data:
 ----------
 VV    <=3D matrix of kxlxn elemnts., with kxl size of map and n rendered views

>>
```

Powyższy plik auto-pomocy nie zawiera jeszcze, jak większość pozostałych m-skryptów gotowy ciąg poleceń. Przeanalizujmy znaczenie ciągu argumentów wejściowych, wymaganych w renderingu powierzchni. Funkcja realizuje modelowanie odbić matowo-połyskliwych według odpowiednio modelu matowych odbić Lambert'a oraz odbić właściwych dla powierzchni połyskliwych, czy też nawet powierzchni z występowaniem odbić bezpośrednich według modelowania zależnością Phong'a.
Wymagana jest nazwa zmiennej roboczej przechowującej macierz 2D punktów wysokości na przykład.: *Zgen*, względna wartość (w granicach od 0 do 1 włącznie) amplitudowego współczynnika odbić matowych na przykład równa 0.5, macierz 3 na 3 elementy parametrów kierunkowych światła rzutowanego, kolejno i pojedynczo na powierzchnię badaną (realizuje to podwywołanie funkcji-m-skryptu: *angles2Smatrix(0,60,3)*) oraz multiplikatywny i ekspotencjalny współczynnik modelowania odbić wedlug modelu Phonga'a , dla przykładu równe odpowiednio 0.5 i 20.
Oto nagłówek funkcji:

```
function [VV]=map_rend_Snx3_spec_16b(map,ro,Snx3,ks,ksn)
```

oraz przykładowa lista argumentów/podwywołąń-z-argumentami funkcji:

```
[VVV]=map_rend_Snx3_spec_16b(Zgen,0.5,angles2Smatrix(0,60,3),0.5,20);
Azimuth:   0           120           240
Elevation: 60

ks =

    0.5000


ksn =

    20
```

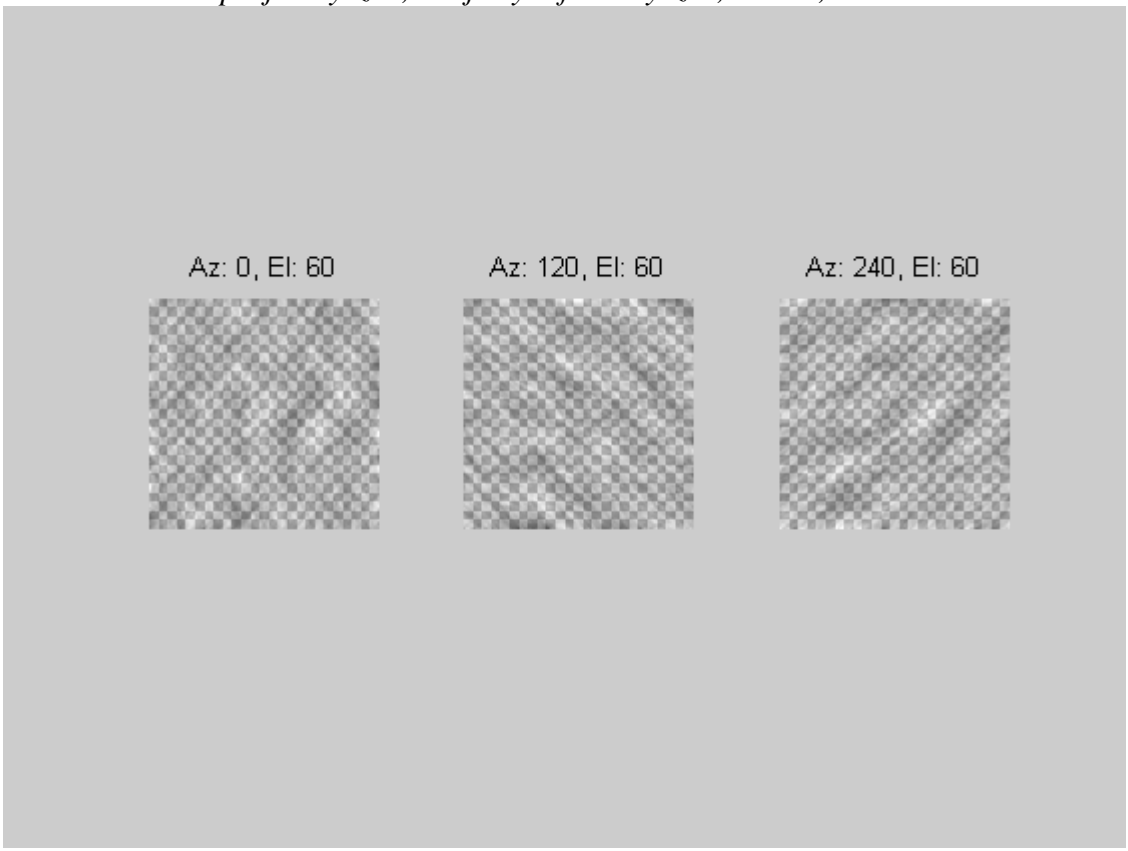


W celach ilustracyjnych korzystniej jest wprowadzić zmienny cyklicznie współczynnik odbicia światła. Sama funkcja wywołania została uzupełniona o wtrącenie trzech spółgłosek *chk* (ang. *checkered* – kratkowany) oraz o jeden dodatkowy argument określający większą wartość (jaśniejszego pola) współczynnika odbić matowych oraz o jeden argument określający częstość zmian tego współczynnika na kierunkach *X* i *Y* obrazów dwuwymiarowych:

```
help map_rend_chk_Snx3_spec_16b
 Artur Bernat 20pm07 All rights reserved
 RE-EDITED ON 14Nov2006
 renderer for three views with: diffuse(matte) and specular reflections
 input:
 ------
 map   <= given 3D map  (kxl points of data)
 ro    <= 1st reflectance coefficient
 ro2   <= 2nd reflectane coefficient
 Snx3  <= directional parameters for light sources (n vectors of [Sx Sy Sz] components)
 nums  <= numbers of squares in checkered pattern imposed on 3D map
 ks    <= multiplicative coefficient of specular reflections (Phong model)
 ksn   <= expotential coefficient of specular reflections (Phong model)
 output:
 -------
 VVV  <= 3D matrix of k x l x n elemnts., kxl-input/output data size, n-number of rendered views
 -------
 see also:
 --------
 help modified_PS_16bits_SVD_nnpxlsNew                  <= determining vector gradient field content's%
 syntax:
 -------
 function [VVV]=map_rend_chk_Snx3_spec_16b(map,ro,ro2,Snx3,nums,ks,ksn)
```

Oto wydruk wywołania dla funkcji tworzącej szachownicę zmiennosci cyklicznej współczynnika odbić matowych:

```
[VVV]=map_rend_chk_Snx3_spec_16b(Zgen,0.4,0.6,angles2Smatrix(0,60,3),25,0.5,20);
Azimuth:   0           120           240
Elevation: 60

ks =

    0.5000


ksn =

    20


ks =

    0.5000


ksn =

    20

  Name        Size              Bytes  Class     Attributes

  vkkA      400x400            320000  uint16
  vkkB      400x400            320000  uint16

  Name        Size              Bytes  Class     Attributes

  vkkA      400x400            320000  uint16
  vkkB      400x400            320000  uint16

  Name        Size              Bytes  Class     Attributes

  vkkA      400x400            320000  uint16
  vkkB      400x400            320000  uint16


Az =

         0  120.0000  240.0000


El =

   60.0000   60.0000   60.0000

  Name       Size            Bytes  Class     Attributes

  Aztab      1x3                24  double
  Eltab      1x3                24  double


Snx3 =

    0.5000         0    0.8660
   -0.2500    0.4330    0.8660
   -0.2500   -0.4330    0.8660


size_n =

     3
```



Powyższe wywołanie funkcji-m-skryptu ujawnia pewny brak gospodarności w tworzeniu kodu do interpretacji w wywołaniu, bowiem funkcja wywoływana w skrypcie: *map_rend_chk_Snx3_spec_16b.m* korzysta z dwóch pod-wywołań funkcji *map_rend_Snx3_spec_16b.m* dla współczynników odbić matowych równych odpowiednio 0.4 oraz 0.6, a następnie dokonuje selekcji podobszarów obrazów 2D dla jasnych i ciemnych pól szachownicy.

```
%Artur Bernat 20pm07 All rights reserved
%RE-EDITED ON 14Nov2006
%renderer for three views with: diffuse(matte) and specular reflections
%input:
%------
%map   <= given 3D map  (kxl points of data)
%ro    <= 1st reflectance coefficient
%ro2   <= 2nd reflectane coefficient
%Snx3  <= directional parameters for light sources (n vectors of [Sx Sy Sz] components)
%nums  <= numbers of squares in checkered pattern imposed on 3D map
%ks    <= multiplicative coefficient of specular reflections (Phong model)
%ksn   <= expotential coefficient of specular reflections (Phong model)
%output:
%-------
%VVV  <= 3D matrix of k x l x n elemnts., kxl-input/output data size, n-number of
rendered views
%-------
%see also:
%--------
%help modified_PS_16bits_SVD_nnpxlsNew                  <= determining vector gradient
field content's%
%syntax:
%-------
%function [VVV]=map_rend_chk_Snx3_spec_16b(map,ro,ro2,Snx3,nums,ks,ksn)
function [VVV]=map_rend_chk_Snx3_spec_16b(map,ro,ro2,Snx3,nums,ks,ksn)
%ro2=ro/2;
size_n=size(Snx3,1);

size_X=size(map,2);
size_Y=size(map,1);
```

```
check_X=size_X/nums;
check_Y=size_Y/nums;
%-------------------------------------
%[vg_1A,vg_2A,vg_3A]=map_rend(map,ro,S3x3);
%-------------------------------------
%[vg_1B,vg_2B,vg_3B]=map_rend(map,ro2,S3x3);
%-------------------------------------
[VVA]=map_rend_Snx3_spec_16b(map,ro,Snx3,ks,ksn);
%-------------------------------------
[VVB]=map_rend_Snx3_spec_16b(map,ro2,Snx3,ks,ksn);
%-------------------------------------
%figure,
for kk=1:size_n,   %GLOBAL for LOOP of size_n CHEQUERED PATTERNS

    vkkA=VVA(:,:,kk);
    vkkB=VVB(:,:,kk);
 whos vkkA vkkB
 %----------------------
    for i=0:2:nums-1,
     for j=0:2:nums-1,
       vkkA(1+i*check_Y:(i+1)*check_Y,1+j*check_X:(j+1)*check_X)=...
       vkkB(1+i*check_Y:(i+1)*check_Y,1+j*check_X:(j+1)*check_X);
     end;
 end;

 %----------------------
    for i=1:2:nums-1  %-2,
     for j=1:2:nums-1 %-2,
       vkkA(1+i*check_Y:(i+1)*check_Y,1+j*check_X:(j+1)*check_X)=...
       vkkB(1+i*check_Y:(i+1)*check_Y,1+j*check_X:(j+1)*check_X);
     end;
 end;
 %----------------------
  VVV(:,:,kk)=vkkA(:,:);
end; % end OF  for kk=1:... LOOP
%----------------------
[Aztab,Eltab]=Smatrix2angles(Snx3);
whos Aztab Eltab
Snx3
size_n
figure,
for ll=1:size_n,

    switch size_n
        case 6
         subplot(2,size_n/2,ll);imshow(uint16(VVV(:,:,ll)));
         hold on;title(['Az: ' num2str(round(Aztab(ll))) ', El: '
num2str(round(Eltab(ll)))]);hold off;
        case 8
         subplot(2,size_n/2,ll);imshow(uint16(VVV(:,:,ll)));
         hold on;title(['Az: ' num2str(round(Aztab(ll))) ', El: '
num2str(round(Eltab(ll)))]);hold off;
        otherwise
         subplot(1,size_n,ll);imshow(uint16(VVV(:,:,ll)));
         hold on;title(['Az: ' num2str(round(Aztab(ll))) ', El: '
num2str(round(Eltab(ll)))]);hold off;
    end;

end;
%----------------------
```

Published with MATLAB® 7.6

, gdzie sama funkcja wirtulnego oświetlenia przy jednolitym współczynniku odbić matowych:

```
%Artur Bernat
%virtual rendering with Snx3 n directional parameters
%RE-EDITED on 14_Nov_2006
%-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
%each of the rendering scripts, have been reedited on 10May2009
%in order to embedde gradient function, instead of diff
%thus, consequently, rendered images are of the same size as depth surface
%-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
%input data:
%-----------
%map  <= 3D map to be rendered
%ro   <= albedo value within [0;1] (homogenous surface)
%Snx3 <= n directional parameters for n light sources
%ks   <= multiplicative coeff. of specular reflection (Phong model)
%ksn  <= expotential coeff. of specular reflection (Phong model)
%----------
%output data:
%----------
%VV    <=3D matrix of kxlxn elemnts., with kxl size of map and n rendered views
function [VV]=map_rend_Snx3_spec_16b(map,ro,Snx3,ks,ksn)

V=[0 0 1];  % viewer direction

size_n=size(Snx3,1);
%----------------
roxy=double((2.^16)*ro); %conversion of ro to roxy of absolute values within range
[0,2.^16]
roxyspec=double((2.^16)*ks);
%----------------
size_X=size(map,2);
size_Y=size(map,1);
[mapxdiff,mapydiff]=gradient(map);
mapzdiff=ones(size_Y,size_X);
ks
ksn
%--------------------------
for kk=1:size_n, %MAIN for LOOP for virtual rendering of n images, based on map data
 Sn=Snx3(kk,:);
 for i=1:size_Y,
    for j=1:size_X,
        N=[mapxdiff(i,j);mapydiff(i,j);mapzdiff(i,j)];
        N=N/norm(N);
        %vout(i,j)=double(roxy).*Sn*N;
         vout(i,j)=double(roxy).*Sn*N+double(roxyspec)*power(((Sn+V)/norm(Sn+V))*N,ksn);
    end;
 end;
 VV(:,:,kk)=uint16(vout);
end;
%--------------------------

imshowm(VV);
```

jest prostą konsekwencją określenia pierwszej pochodnej na kierunku *X* i *Y* płaszczyzny odniesienia *XY* oraz obliczeń iloczynu skalarnego odpowiednio kierunku *L* światła rzutowanego z wektorem normalnym do powierzchni *N*. Nieco bardziej złożona sytuacja dotyczy modelowania Phonga. Wyobraźmy sobie pewien wektor wypadkowy, pełniący dwusieczną kąta zawartego pomiędzy kierunkiem *L*, a kierunkiem obserwacji *V*. Pożądaną sytuacją jest, by w miejscu odbicia bezpośredniego promienia światła wektor normalny do powierzchni *N* dokładnie odpowiadał orientacji dwusiecznej tego kąta, jeśli chcemy by efekt odbić bezpośrednich był zmaksymalizowany, w strumieniu światła odbitego na obrazie 2D, dla danej lokalizacji (*x*,*y*).

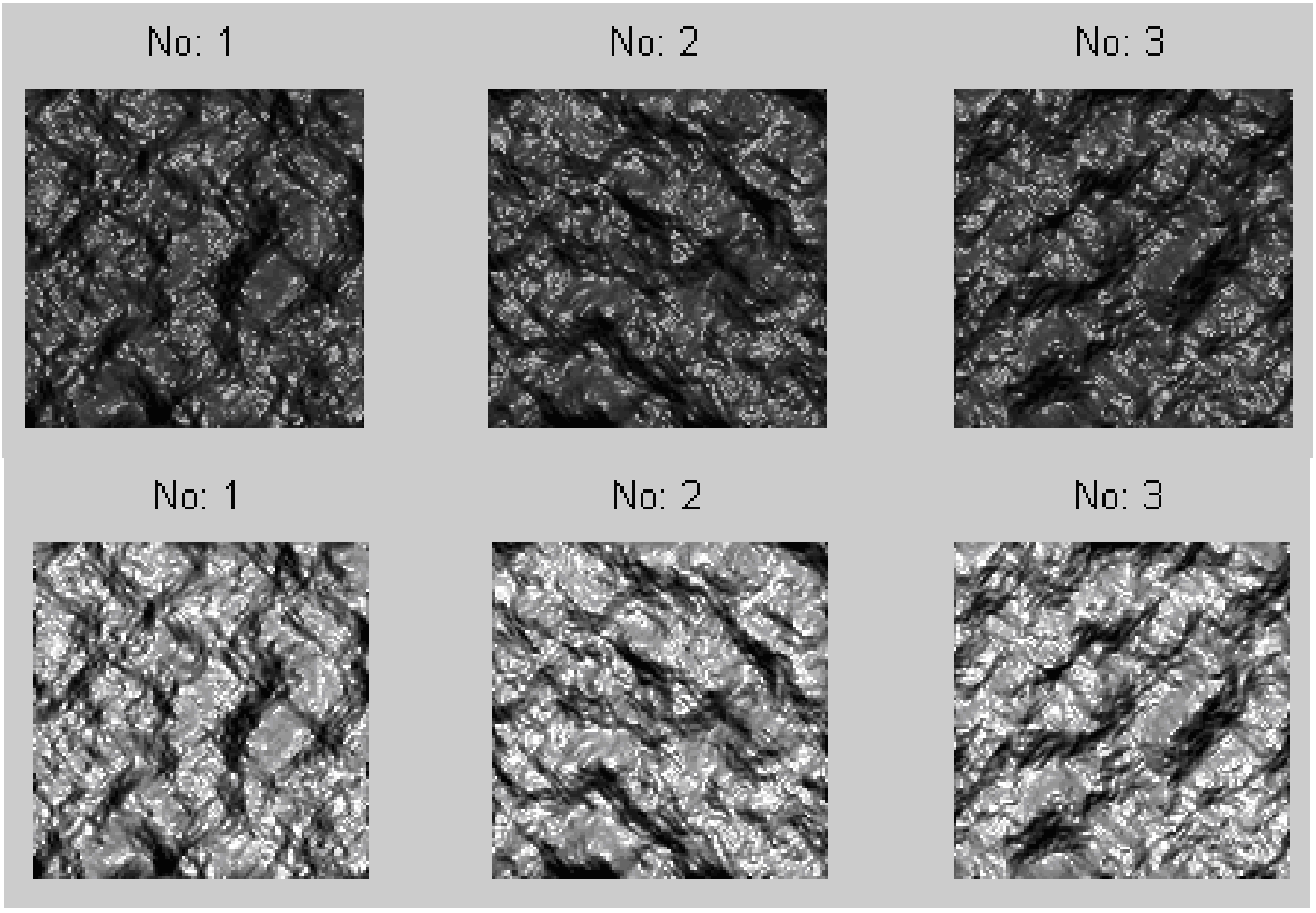
Bardziej intrygujące być może są własnego wytworu efektowne algorytmy uwzględniające infinityzemalne zawiłości mini-długości łuków na zakrzywieniach analizowanej w wymiarze trzecim topografii powierzchni w renderingu. A wszystko to w celu wywołania bardziej zbliżonego do rzeczywistości efektu elastycznego naciągania szachownicy na renderowaną powierzchnię:

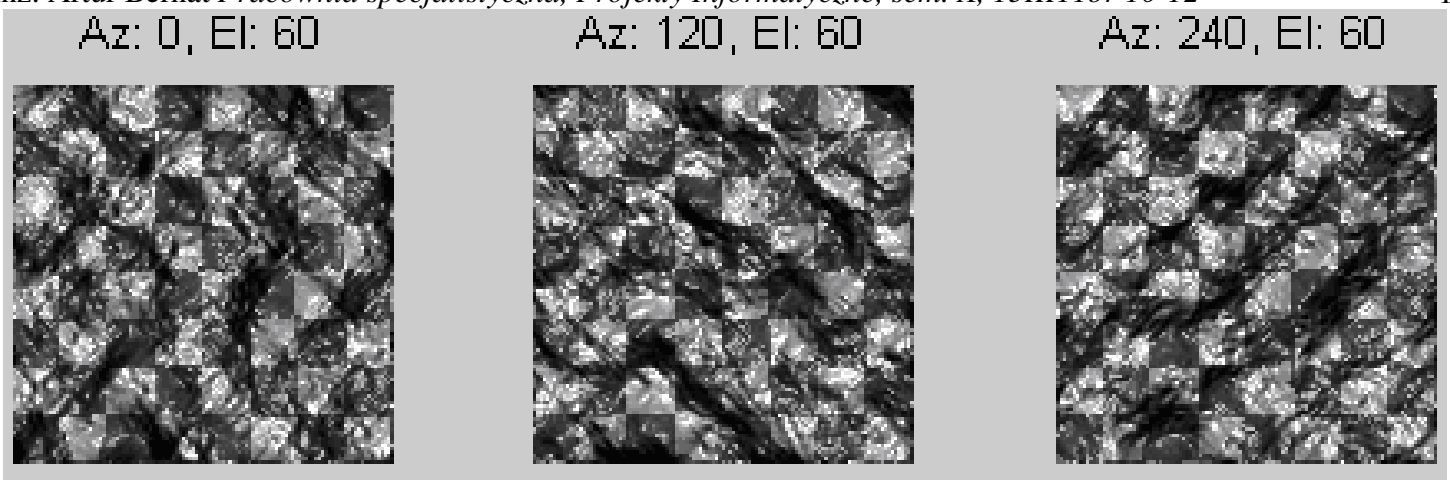
```
help map_rend_check3D_Snx3_spec_16b.m
 Artur Bernat 20pm07 All rights reserved
 RE-EDITED ON 14Nov2006
 updated on16 June 2009, 17pm26
 renderer for three views with: diffuse(matte) and specular reflections
 input:
 ------
 map   <= given 3D map  (kxl points of data)
 ro    <= 1st reflectance coefficient
 ro2   <= 2nd reflectane coefficient
 Snx3  <= directional parameters for light sources (n vectors of [Sx Sy Sz] components)
 nums  <= numbers of squares in checkered pattern imposed on 3D map
 ks    <= multiplicative coefficient of specular reflections (Phong model)
 ksn   <= expotential coefficient of specular reflections (Phong model)
 output:
 -------
 VVV  <= 3D matrix of k x l x n elemnts., kxl-input/output data size, n-number of rendered views
 -------
 see also:
 --------
 help modified_PS_16bits_SVD_nnpxlsNew                  <= determining vector gradient field content's%
 syntax:
 -------
 function [VVV]=map_rend_check3D_Snx3_spec_16b(map,ro,ro2,Snx3,nums,ks,ksn)
```

Sprawdźmy, wywołanie funkcji-m-skryptu *map_rend_check3D_Snx3_spec_16b.m* zasadniczo niczym istotnym w liście argumentów nie różni się i nie musi się wcale różnić od wywołania poprzedniego, chociaż… Wprowadzono inną względną jasność pól jasnych i ciemnych. Są to odpowiednio wartości względne: 0.3 oraz 0.7 ‘radykalizując’ nieco proces wcześniej delikatnego nakładania wzoru szachownicy na powierzchni renderowanej. Wprowadzono 20-krotne powiększenie rozpiętości punktów wysokości na powierzchni, celem ukazania, jak szachownica ‘naciągana’ jest na różnice w wysokościach punktów na powierzchni renderowanej. A samą częstość zmienności cyklicznej pól jasnych na ciemne i na odwrót zmniejszono z 25 do tylko 8 na całej długości i szerokości renderowanego wycinka powierzchni:

```
[VVV]=map_rend_check3D_Snx3_spec_16b(20*Zgen,0.3,0.7,angles2Smatrix(0,60,3),08,0.5,20);
Azimuth:   0           120           240
Elevation: 60
  Name         Size                 Bytes  Class     Attributes

  XX        1600x1600            20480000  double
  YY        1600x1600            20480000  double

  Name         Size                 Bytes  Class     Attributes

  Sx         400x400              1280000  double
  Sy         400x400              1280000  double
  XX        1600x1600            20480000  double
  YY        1600x1600            20480000  double


ks =

    0.5000


ksn =

    20


ks =

    0.5000


ksn =

    20

  Name             Size                 Bytes  Class     Attributes

  Sx             400x400              1280000  double
  Sy             400x400              1280000  double
  XX            1600x1600            20480000  double
  YY            1600x1600            20480000  double
  albedoVVV     1600x1600            20480000  double
```

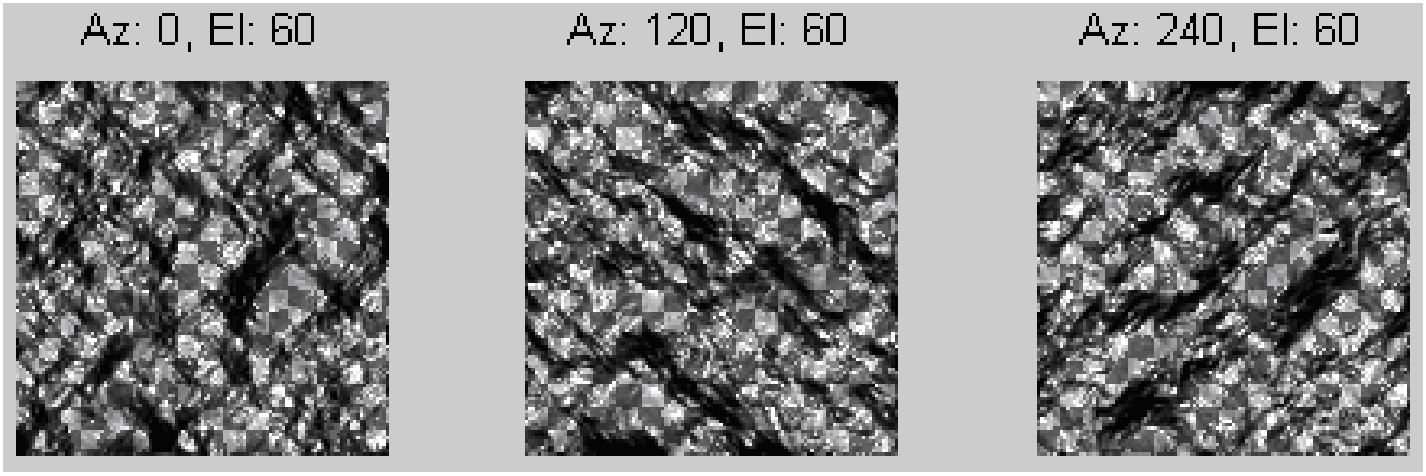
```
  Name        Size              Bytes  Class     Attributes

  vkkA      400x400            320000  uint16
  vkkB      400x400            320000  uint16

  Name        Size              Bytes  Class     Attributes

  vkkA      400x400            320000  uint16
  vkkB      400x400            320000  uint16

  Name        Size              Bytes  Class     Attributes

  vkkA      400x400            320000  uint16
  vkkB      400x400            320000  uint16


Az =

         0  120.0000  240.0000


El =

   60.0000   60.0000   60.0000

  Name       Size            Bytes  Class     Attributes

  Aztab      1x3                24  double
  Eltab      1x3                24  double


Snx3 =

    0.5000         0    0.8660
   -0.2500    0.4330    0.8660
   -0.2500   -0.4330    0.8660


size_n =

     3
```





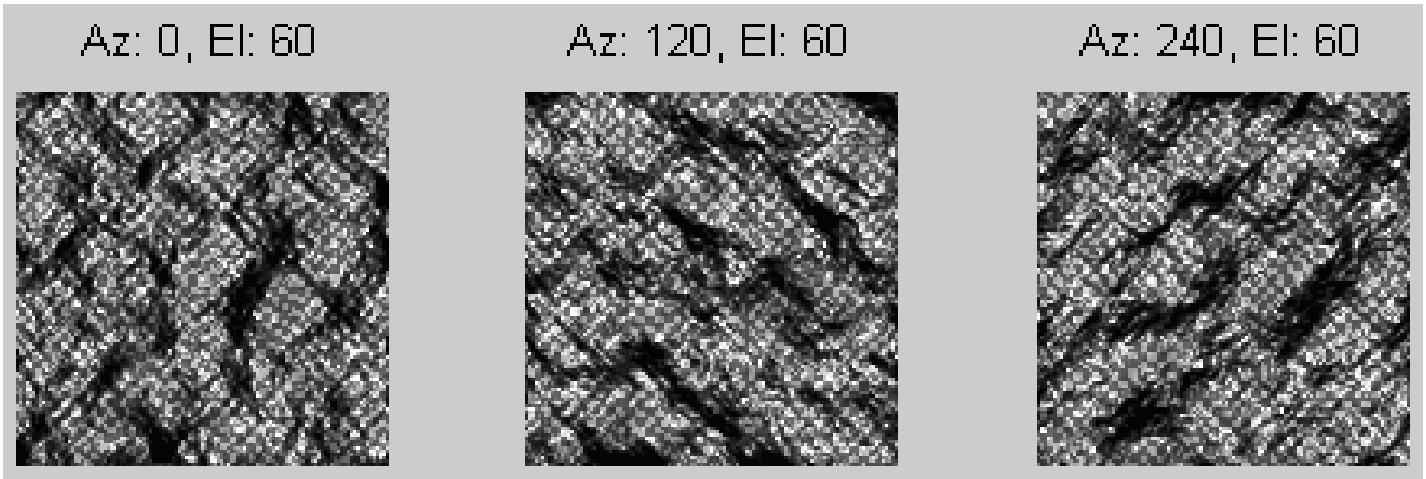
Inny przykład wywołania funkcji z uwzględnieniem infinityzemalnych łuków po zmiennej topografii powierzchni uwzględnia nieco jaśniejsze: ‘jasne’ i ‘ciemne’ pole szachownicy, 25-krotne zwiększenie rozpiętości wysokości na powierzchni renderowanej oraz 16-krotną częstotliwość zamiany ‘jasnego’ pola w ‘ciemne’ i na odwrót:

```
[VVV]=map_rend_check3D_Snx3_spec_16b(25*Zgen,0.4,0.8,angles2Smatrix(0,60,3),16,0.5,20);
```



Albo też: nieco drobniejsze bo 40-krotne zamiany pól jasnych w ciemne(tutaj podano tylko wyniki końcowe renderingu w przykładach przedostatnim i ostatnim):

```
[VVV]=map_rend_check3D_Snx3_spec_16b(25*Zgen,0.4,0.8,angles2Smatrix(0,60,3),40,0.5,20);
```



```
%Artur Bernat 20pm07 All rights reserved
%RE-EDITED ON 14Nov2006
%updated on16 June 2009, 17pm26
%renderer for three views with: diffuse(matte) and specular reflections
%input:
%------
%map   <= given 3D map  (kxl points of data)
%ro    <= 1st reflectance coefficient
%ro2   <= 2nd reflectane coefficient
%Snx3  <= directional parameters for light sources (n vectors of [Sx Sy Sz] components)
%nums  <= numbers of squares in checkered pattern imposed on 3D map
%ks    <= multiplicative coefficient of specular reflections (Phong model)
%ksn   <= expotential coefficient of specular reflections (Phong model)
%output:
%-------
%VVV  <= 3D matrix of k x l x n elemnts., kxl-input/output data size, n-number of rendered views
%-------
%see also:
%--------
%help modified_PS_16bits_SVD_nnpxlsNew                  <= determining vector gradient field content's%
%syntax:
%-------
%function [VVV]=map_rend_check3D_Snx3_spec_16b(map,ro,ro2,Snx3,nums,ks,ksn)
function [VVV,albedoVVV3D]=map_rend_check3D_Snx3_spec_16b(map,ro,ro2,Snx3,nums,ks,ksn,albedo_mapping_step)
%-------------------------------------
if nargin<8
    albedo_mapping_step=4;
end;
%ro2=ro/2;
MaxGray16b=2.^16-1;
%-------------------------------------
size_n=size(Snx3,1);
size_X=size(map,2);
size_Y=size(map,1);
%-------------------------------------
check_X=size_X/nums;
check_Y=size_Y/nums;
%-------------------------------------
[p,q]=gradient(map./3);                  % slopes p, q on X, Y directions
dsx=(p.^2+1).^(1/2);                     % infinitizemal arc lengths, in horizontal lines%
dsy=(q.^2+1).^(1/2);                     % infinitizemal arc lengths, in vertical lines%
%-------------------------------------
[XX,YY]=meshgrid(1:4.*size_X,1:4.*size_Y); %developped albedo 4-times larger than depth map %
%-------------------------------------
whos XX YY albedoVVV Sx Sy
sub_areasX=size_X./albedo_mapping_step;
sub_areasY=size_Y./albedo_mapping_step;
%-------------------------------------
%Sx=round(cumsum(dsx,2));                        % current arc length at (x,y) currently considered location
from border%
%Sy=round(cumsum(dsy,1));                        % current arc length at (x,y) currently cosidered location,
taken from border%
 for m=1:albedo_mapping_step,
  for n=1:albedo_mapping_step,
%-------------------------------------
   Sx(1+(m-1).*sub_areasY:m.*sub_areasY,1+(n-1).*sub_areasX:n.*sub_areasX)=...
   cumsum(dsx(1+(m-1).*sub_areasY:m.*sub_areasY,1+(n-1).*sub_areasX:n.*sub_areasX),2);
%-------------------------------------
   Sy(1+(m-1).*sub_areasY:m.*sub_areasY,1+(n-1).*sub_areasX:n.*sub_areasX)=...
   cumsum(dsx(1+(m-1).*sub_areasY:m.*sub_areasY,1+(n-1).*sub_areasX:n.*sub_areasX),1);
%-------------------------------------
  end;
 end;

%-------------------------------------
whos XX YY albedoVVV Sx Sy
sub_areasX=size_X./albedo_mapping_step;
sub_areasY=size_Y./albedo_mapping_step;
%-------------------------------------
%figure,imshow(uint16(albedoVVV));title('Albedo 2D, checkered chessboard pattern');hold off;
%-------------------------------------
 for m=1:albedo_mapping_step,
  for n=1:albedo_mapping_step,
%   albedoVVV3D(1+(m-1).*sub_areasY:m.*sub_areasY,1+(n-1).*sub_areasX:n.*sub_areasX)=...
%   interp2(XX,YY,albedoVVV,Sx(1+(m-1).*sub_areasY:m.*sub_areasY,1+(n-1).*sub_areasX:n.*sub_areasX),...
 %                          Sy(1+(m-1).*sub_areasY:m.*sub_areasY,1+(n-1).*sub_areasX:n.*sub_areasX));
  end;
 end;
%-------------------------------------
%figure,imshow(uint16(albedoVVV3D));title('Albedo 2D, morphed 3D');hold off;
%-------------------------------------
%Sx=padarray(Sx,[1 1],'replicate','post');
%Sy=padarray(Sy,[1 1],'replicate','post');
%-------------------------------------
```

```
[VVA]=map_rend_Snx3_spec_16b(map,ro,Snx3,ks,ksn);
%-------------------------------------
[VVB]=map_rend_Snx3_spec_16b(map,ro2,Snx3,ks,ksn);
%-------------------------------------
albedoVVV=ro.*MaxGray16b.*ones(size_X,size_Y);
%-------------------------------------
    for i=0:2:4.*nums-1,
     for j=0:2:4.*nums-1,
       albedoVVV(1+i*check_Y:(i+1)*check_Y,1+j*check_X:(j+1)*check_X)= ro2.*MaxGray16b;
     end;
    end;
 %-------------------------------------
    for i=1:2:4.*nums-1,  %-2,
     for j=1:2:4.*nums-1, %-2,
       albedoVVV(1+i*check_Y:(i+1)*check_Y,1+j*check_X:(j+1)*check_X)=ro2.*MaxGray16b;
     end;
 end;
 %-------------------------------------
 whos XX YY albedoVVV Sx Sy
 sub_areasX=size_X./albedo_mapping_step;
 sub_areasY=size_Y./albedo_mapping_step;
 %-------------------------------------
 %
 %-------------------------------------
 for m=1:albedo_mapping_step,
  for n=1:albedo_mapping_step,
   albedoVVV3D(1+(m-1).*sub_areasY:m.*sub_areasY,1+(n-1).*sub_areasX:n.*sub_areasX)=...
   interp2(XX,YY,albedoVVV,Sx(1+(m-1).*sub_areasY:m.*sub_areasY,1+(n-1).*sub_areasX:n.*sub_areasX),...
                           Sy(1+(m-1).*sub_areasY:m.*sub_areasY,1+(n-1).*sub_areasX:n.*sub_areasX));
  end;
 end;
 %-------------------------------------
for kk=1:size_n,   %GLOBAL for LOOP of size_n CHEQUERED PATTERNS
 vkkA=VVA(:,:,kk);
 vkkB=VVB(:,:,kk);
 whos vkkA vkkB
%-------------------------------------
  for j=1:size_X,
    for i=1:size_Y,
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        if (albedoVVV3D(i,j)==ro.*MaxGray16b)
            vkkA(i,j)=vkkA(i,j);
        elseif (albedoVVV3D(i,j)==ro2.*MaxGray16b)
            vkkA(i,j)=vkkB(i,j);
        end;
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    end;
  end;
%-------------------------------------
%-------------------------------------
 VVV(:,:,kk)=vkkA(:,:);
end; % end OF  for kk=1:... LOOP
%-----------------------
albedoVVV3D=uint16(albedoVVV3D);
%-----------------------
[Aztab,Eltab]=Smatrix2angles(Snx3);
whos Aztab Eltab
Snx3
size_n
figure,
%-----------------------------------------
for ll=1:size_n,
    switch size_n
        case 6
         subplot(2,size_n/2,ll);imshow(uint16(VVV(:,:,ll)));
         hold on;title(['Az: ' num2str(round(Aztab(ll))) ', El: ' num2str(round(Eltab(ll)))]);hold off;
        case 8
         subplot(2,size_n/2,ll);imshow(uint16(VVV(:,:,ll)));
         hold on;title(['Az: ' num2str(round(Aztab(ll))) ', El: ' num2str(round(Eltab(ll)))]);hold off;
        otherwise
         subplot(1,size_n,ll);imshow(uint16(VVV(:,:,ll)));
         hold on;title(['Az: ' num2str(round(Aztab(ll))) ', El: ' num2str(round(Eltab(ll)))]);hold off;
    end;

end;
%-----------------------------------------
```

Na podstawie kodu źródłowego do infityzelmalnego ‘naciągania’ szachownicy względem obliczanej długości łuku topografii powierzchni, najpierw obliczane są estymaty długości łuku na kierunkach *X* i *Y*, w wyrażeniach typu pierwiastek z (1+(dz/dx)^2) oraz pierwiastek z (1+(dz/dy)^2, a następnie są obliczane ‘elastycznie naciągane’ sumy kumulacyjne interpolacji szachownicy względem tych zmian po dz topografii powierzchni…