Úvod do počítačovej grafiky
bez geometrie: Image-based Rendering

Andrej FERKO
Comenius University Bratislava
22. marca 2018, FMFI UK

Motivation

- Time, costs...
- human visual system
- field of view
- of around 135x200 degrees,
- but a typical camera
- only 35 x 50 degrees...

- Plenoptic modeling... 1995

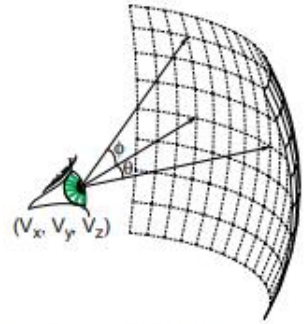


FIGURE 1. The plenoptic function describes all of the image information visible from a particular viewing position.

In the case of a dynamic scene, we can additionally choose the time, t, at which we wish to evaluate the function. This results in the following form for the plenoptic function:

$$p = P(\theta, \phi, \lambda, V_x, V_y, V_z, t) \quad (1)$$

Plenoptic modeling

- **Plenoptic modeling**... Bishop & McMillan 1995
- "Image-based rendering is a powerful new approach for generating real-time photorealistic computer graphics... convincing animations without an explicit geometric representation."

- Tools: Dersch, Hugin, PTGui...

- AutoStitch - Brown-Lowe 2003
- http://matthewalunbrown.com/autostitch/autostitch.html

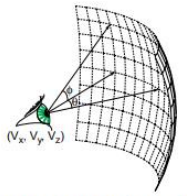


FIGURE 1. The plenoptic function describes all of the image information visible from a particular viewing position.

In the case of a dynamic scene, we can additionally choose the time, t, at which we wish to evaluate the function. This results in the following form for the plenoptic function:

$$p = P(\theta, \phi, \lambda, V_x, V_y, V_z, t) \quad (1)$$

- **7D Plenoptic Function >> 2D panorama**

Plenoptic function [BM95]

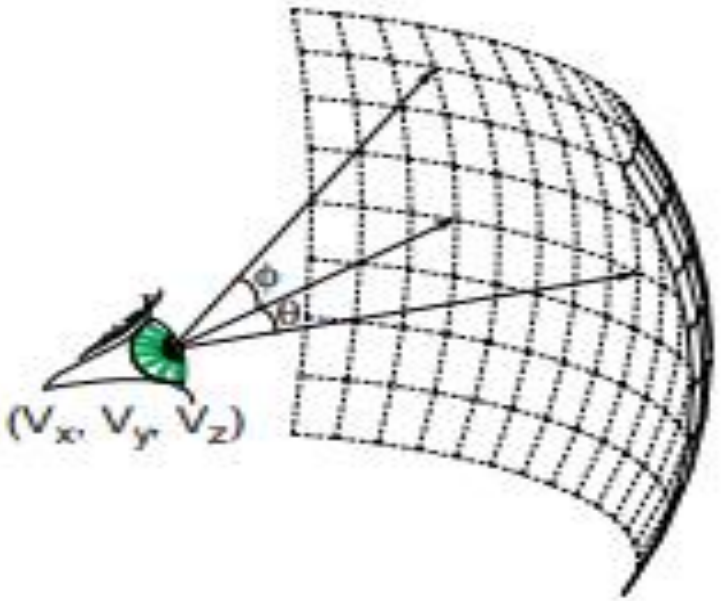


FIGURE 1. The plenoptic function describes all of the image information visible from a particular viewing position.

In the case of a dynamic scene, we can additionally choose the time, t, at which we wish to evaluate the function. This results in the following form for the plenoptic function:

$$p = P(\theta, \phi, \lambda, V_x, V_y, V_z, t) \qquad (1)$$

Sea of Images 2002

Daniel G. Aliaga, Thomas Funkhouser, Dimah Yanovsky, Ingrid Carlbom

Photosynth 2006-10

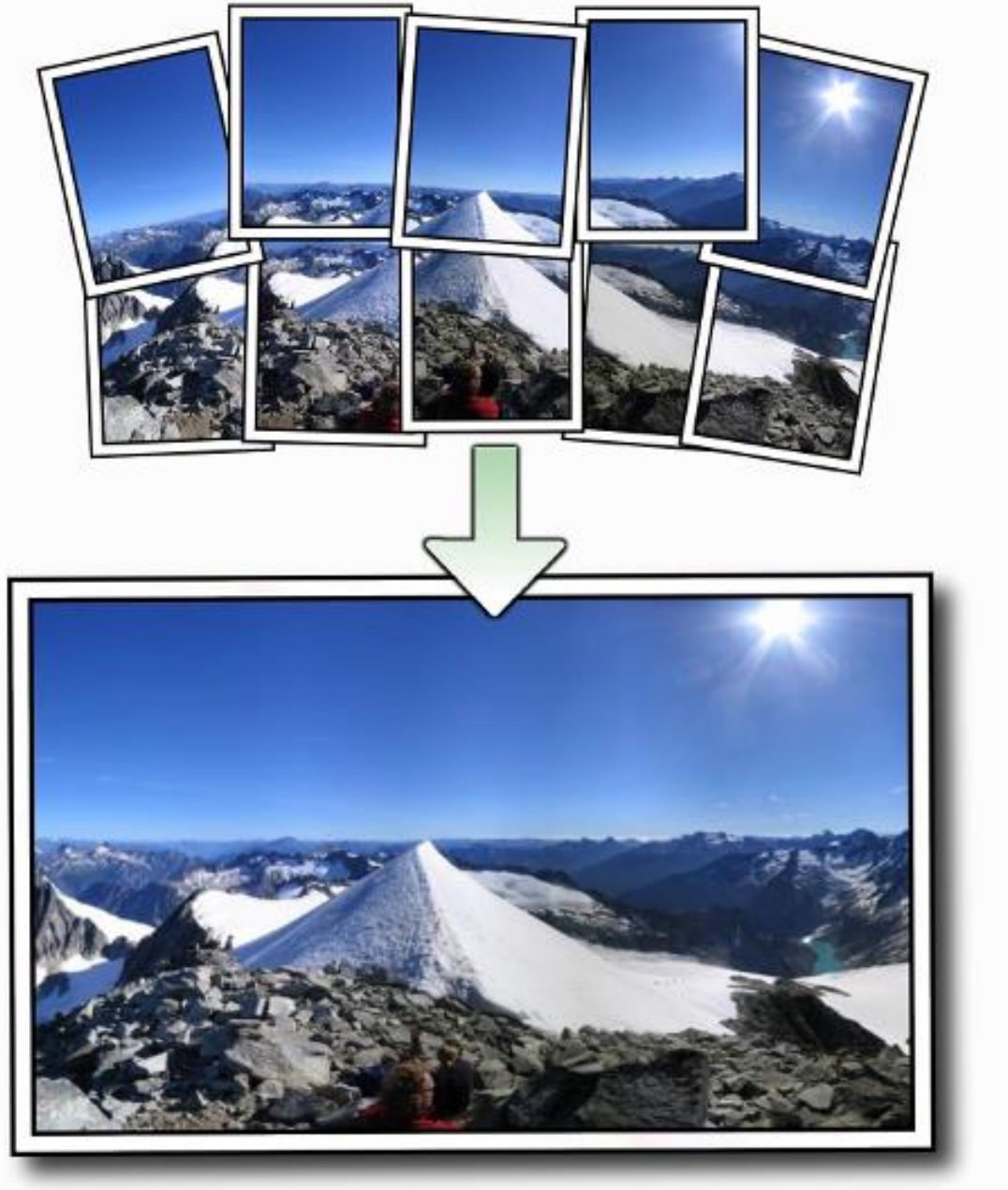
Microsoft Photosynth

Photosynth technology showing Spider Meadows Central Washington

Developer(s)	Microsoft
Initial release	August 20, 2008; 9 years ago
Last release	2.110.317.1042 / March 18, 20 7 years ago
Development status	Discontinued[1]
Type	3D modeling, panorama stitching
Website	photosynth.net

Autostitch

- [BL03]



Process

- **[BL03]**

25 of 57 images aligned

All 57 images aligned

Final Result

- **http://matthewalunbrown.com/autostitch/autostitch.html**

Principles of geometric analysis and synthesis of a mathematic model

- princíp kontinuity
  - (nepreryvnost: spojitost, koherencia)
- princíp zhody
  - (sootvetstvije: dodržiavanie, consistency, conformity)
- princíp kompatibility
  - (sovmestimost: zlucitelnost)

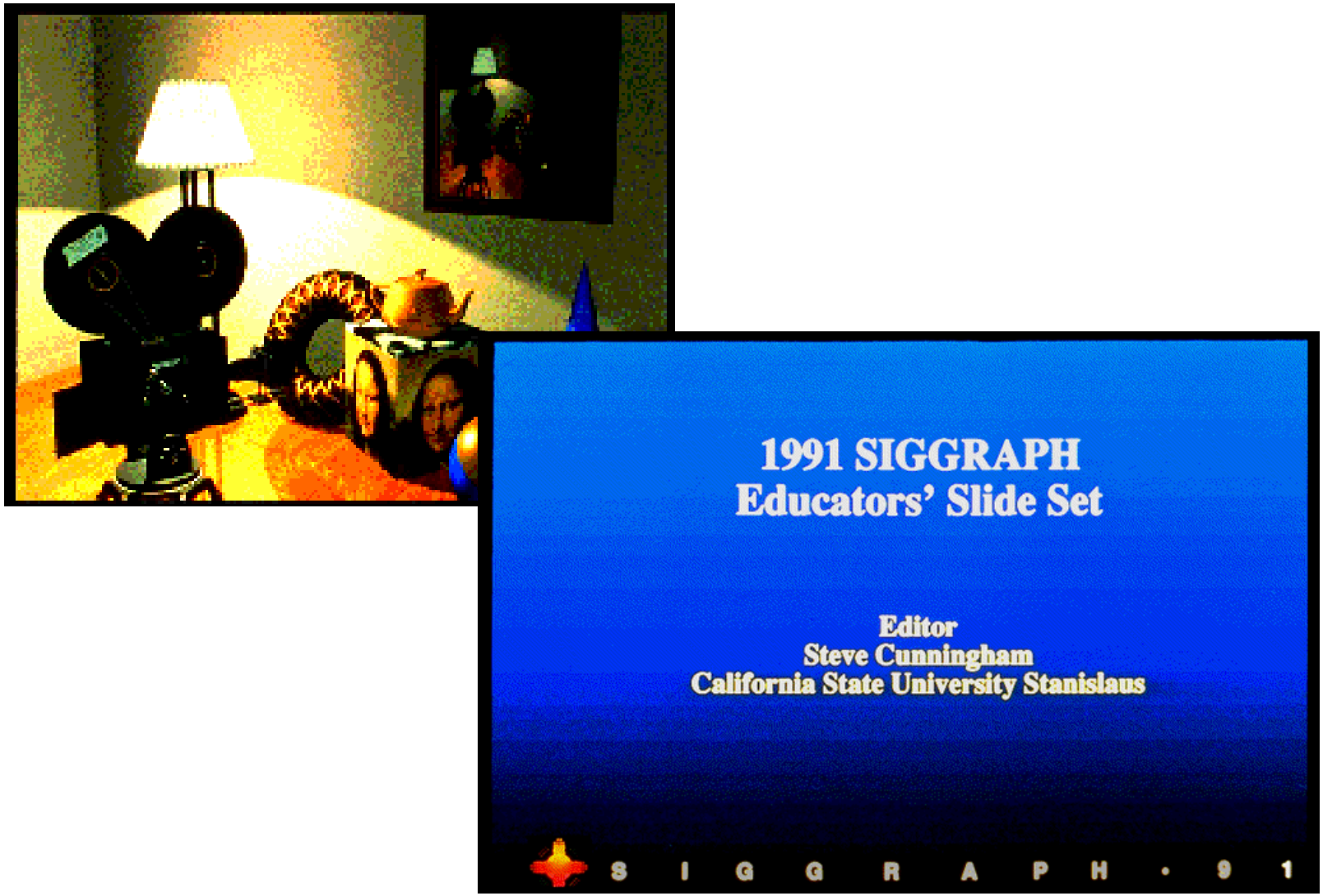
- Baganyan, GA. 1985. Mašinnaja grafika v upravľenii. Jerevan: Ajastan.

History before computers

- Panorama of 'Old Edinburgh' by Robert Barker
- Barker's patent for painting panoramas expired in 1801, which meant the 360-degree images could be produced by rival artists

his Panorama of 'Old Edinburgh' by Robert Barker is still in existance and resides at the The Edinburgh Virtual Environment Centre, University of Edinburgh. The Panorama was as wide as 300 feet and as high as 50. It is known as "Edinburgh From The Crown Of St. Giles". Image Copyright © City Arts Centre.

SIGGRAPH Slide Show

- 

ShutterBug Credits

Produced by Tom Williams and H. B. Siegel, with the assistance of M. W. Mantle

All images rendered with PhotoRealistic RenderMan 3.2

Copyright Pixar, 1990

Produced for Computer Graphics, Principles and Practice, Second Edition, by Foley, van Dam, Feiner, and Hughes

Copyright Addison-Wesley, 1990

S I G G R A P H • 9

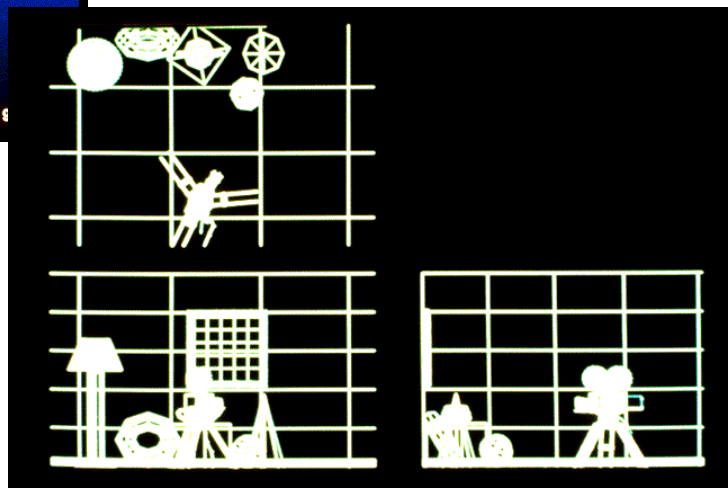
The Shutterbug Rendering Progression

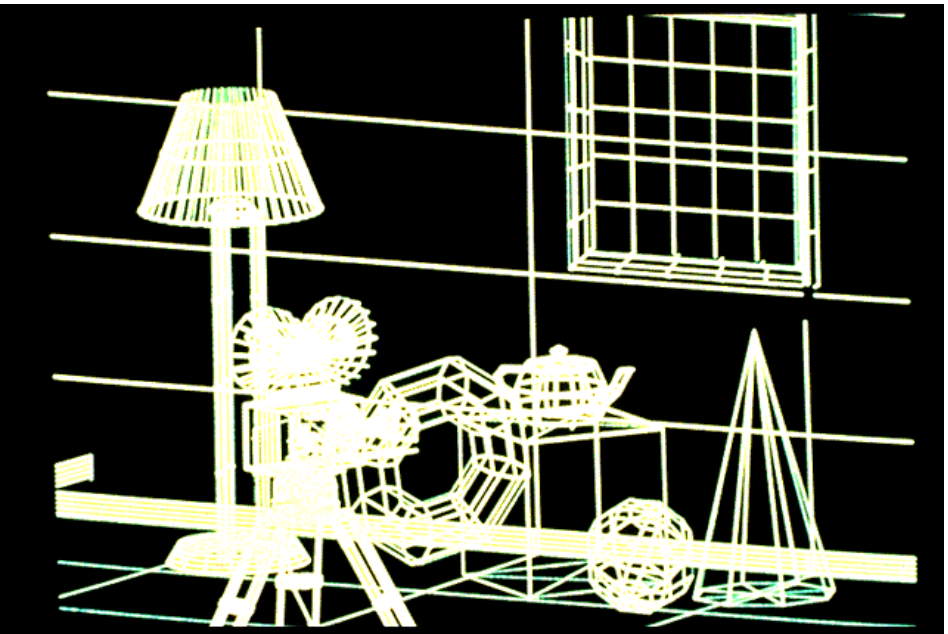
This sequence illustrates the progressive refinement of rendering algorithms.

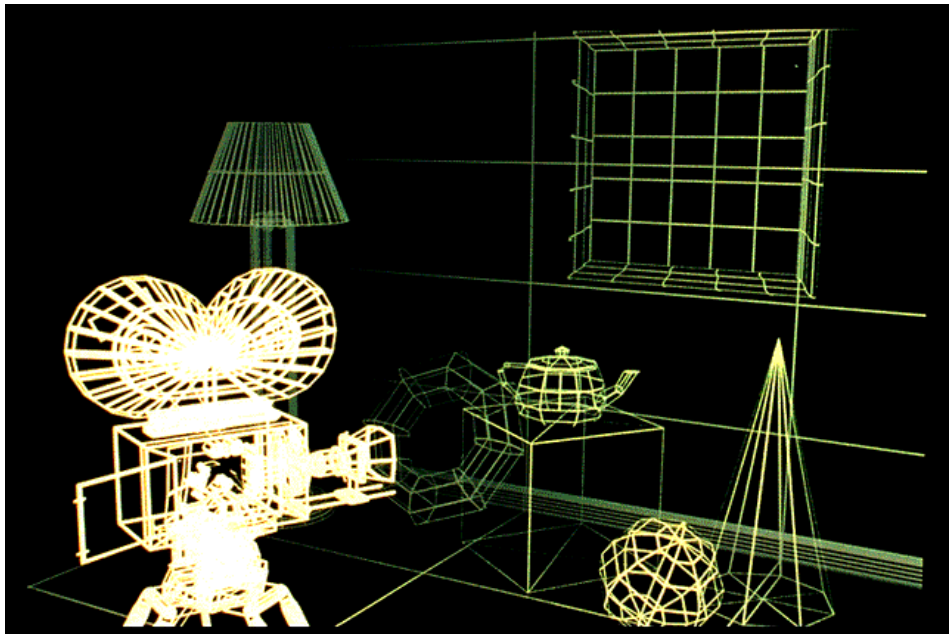
The images range from wire frames to photo-realistic renditions including reflections and shadows.

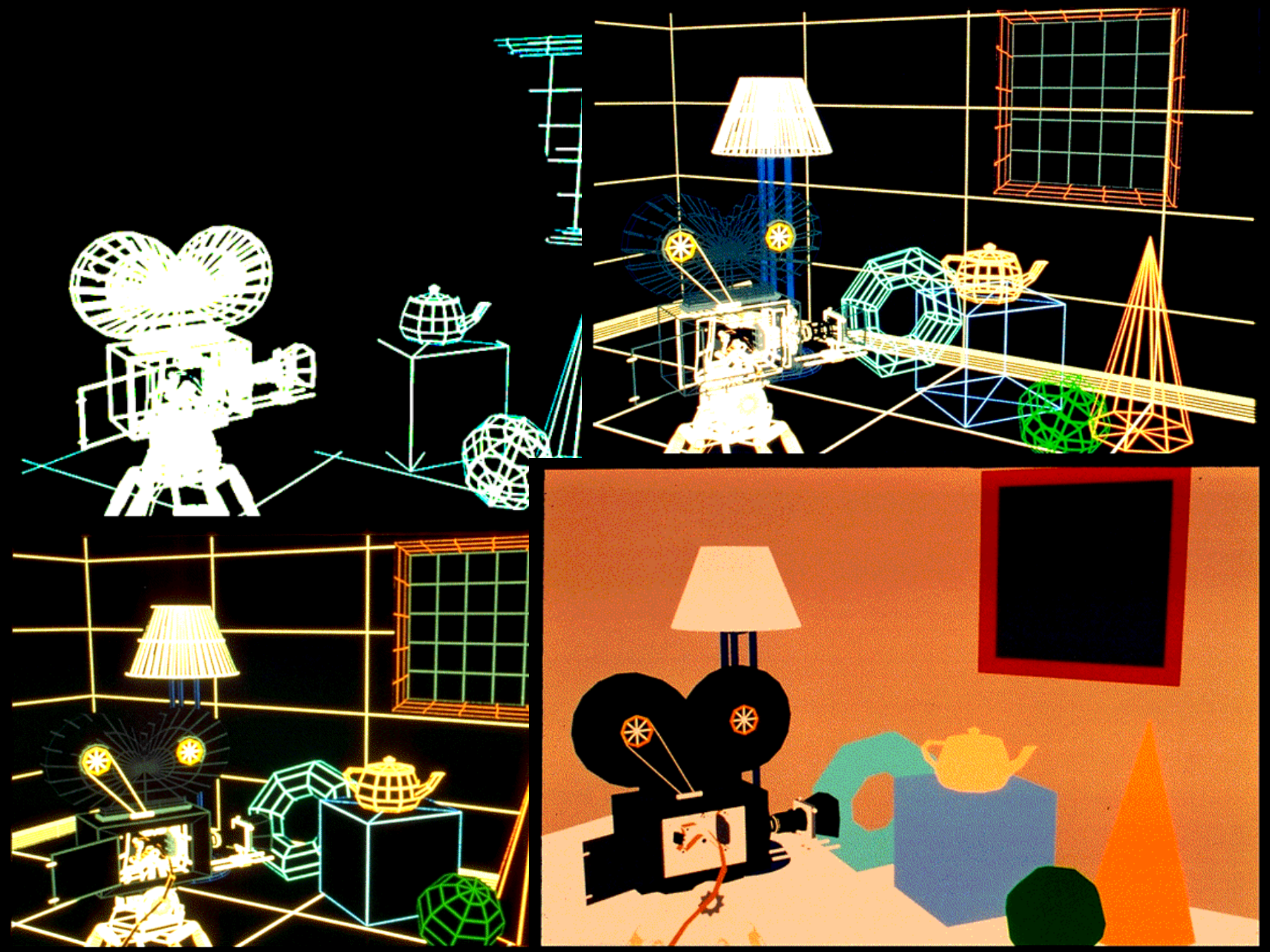
The rendering algorithm affects the quality and information conveyed by the image, independent of the underlying three-dimensional model.

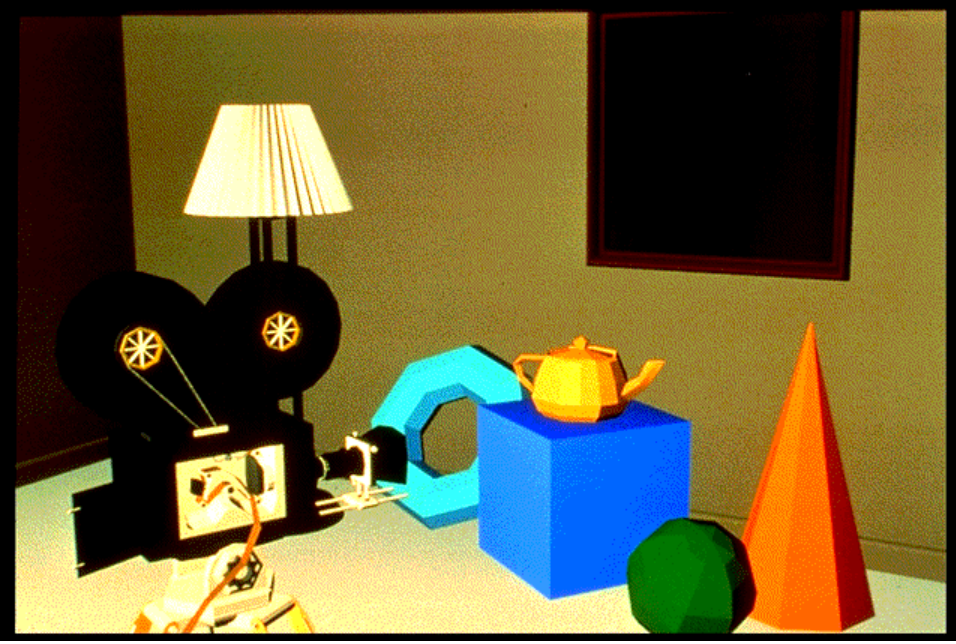
S I G G R A P H • 9 1

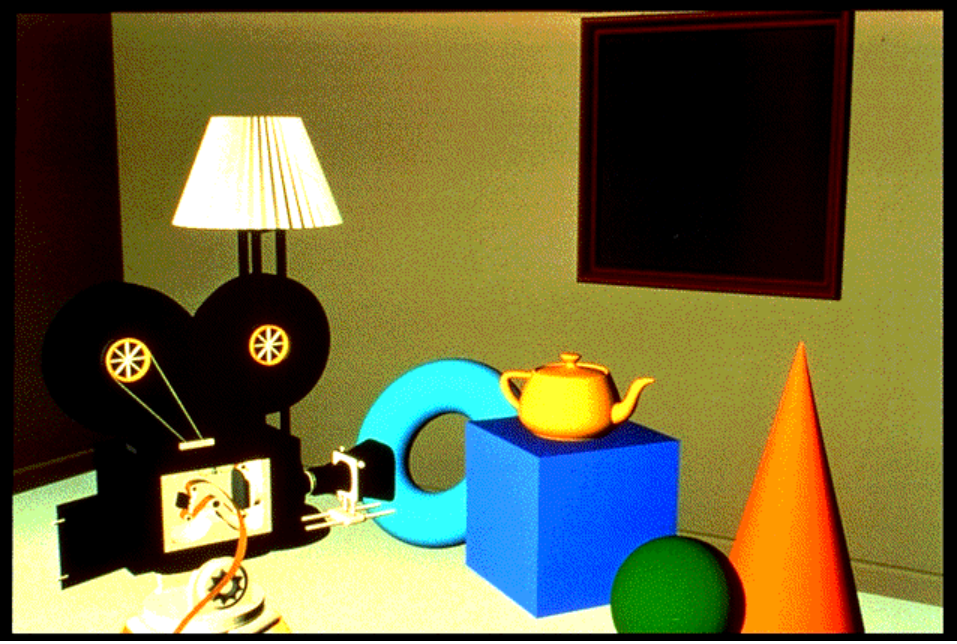


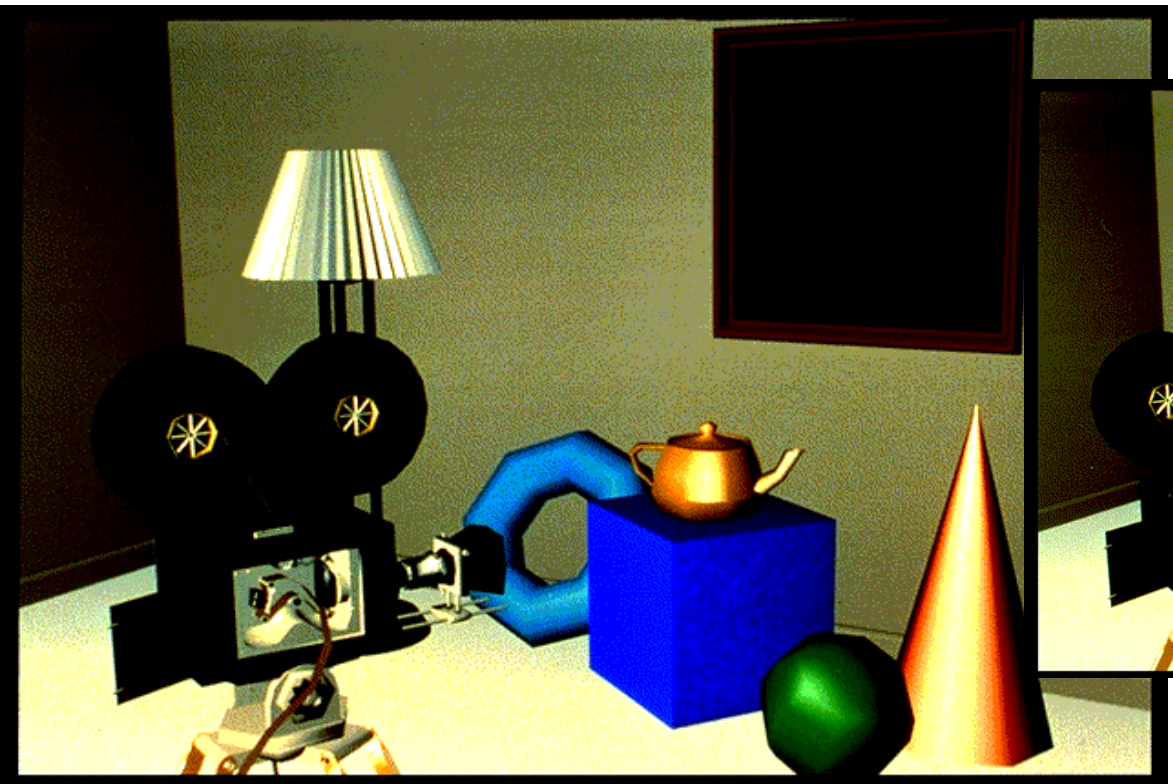


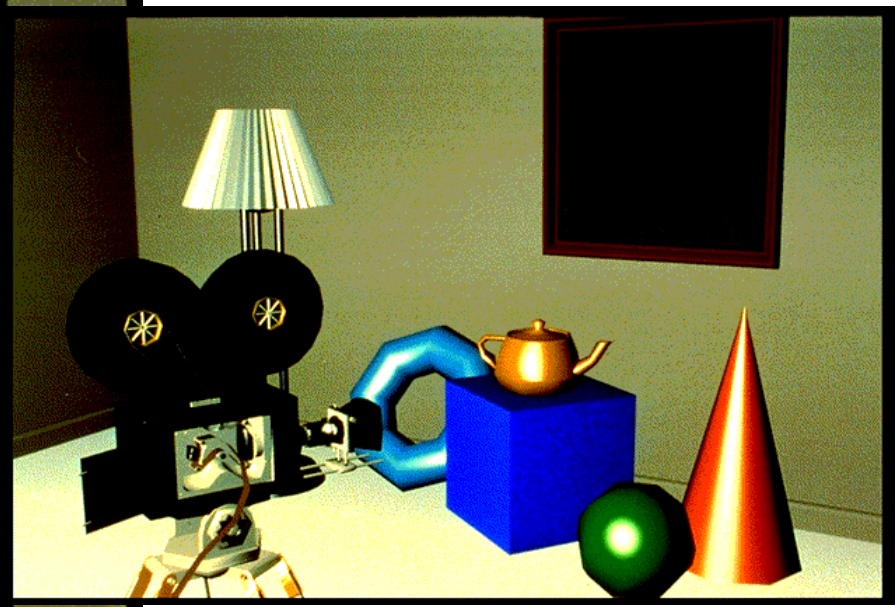


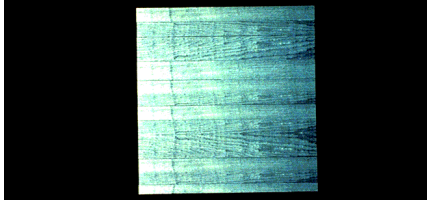


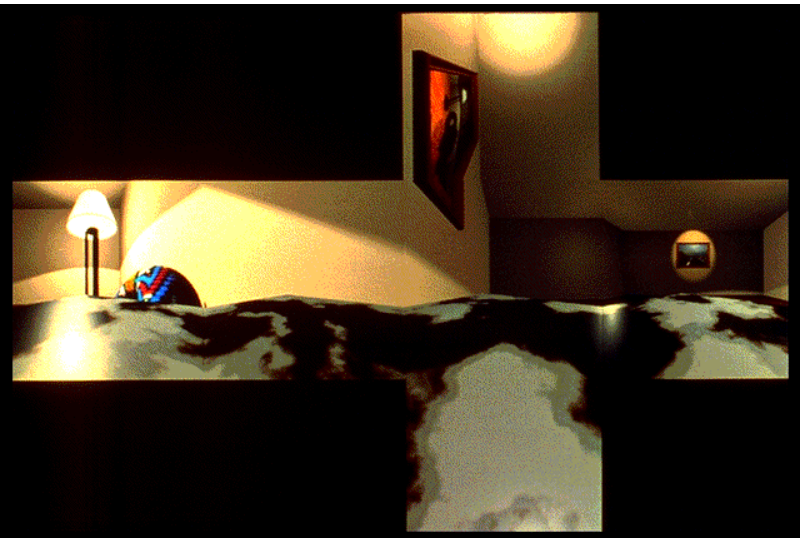


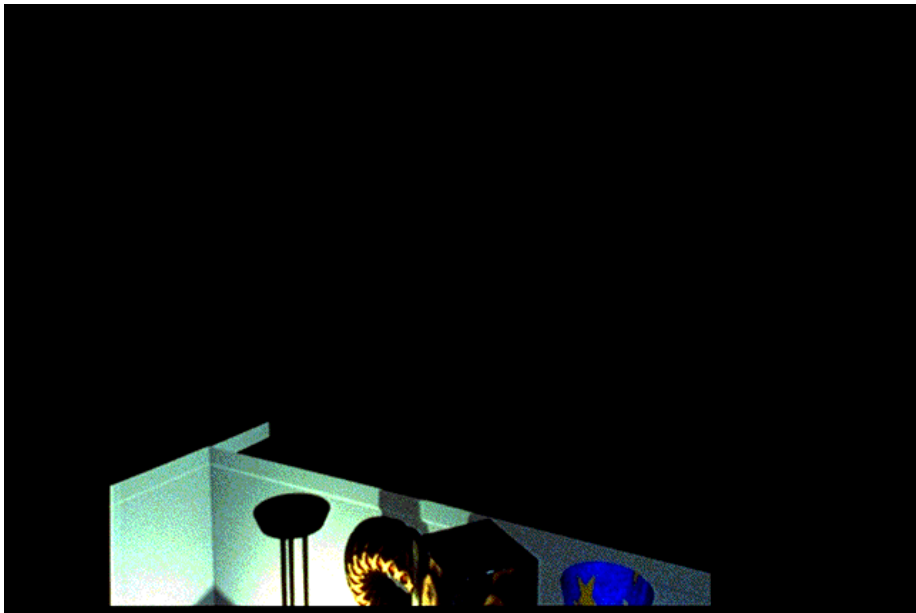






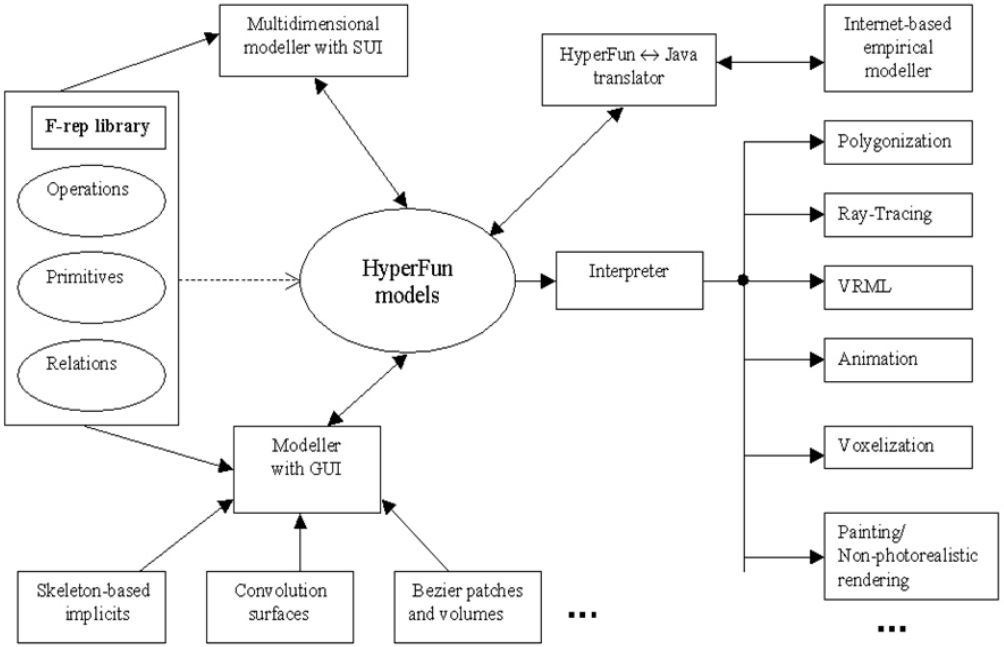






Model >> Picture, context

System Architecture



FRep Library

Primitives

HfSphere, HfEllipsoid, HfCylinder, HfEllipticCylinder, HfEllipticCone, HfTorus, HfSuperellipsoid, HfBlock, HfBlobby, HfMetaball, HfSoft, hFBezierVolume, HfCubicSpline, HfNoiseG

Convolution objects:

HfConvPoint, HfConvLine, HfConvArc, HfConvTriangle, HfConvCurve, HfConvMesh

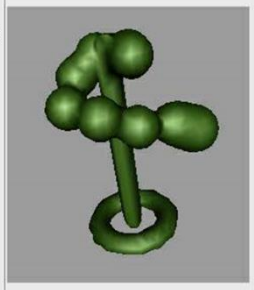
Operations

HfBlendingUnion, HfBlendingIntersection, HfScale, HfShift, HfRotate, HfTwist, HfStretch, HfTapering, HfSpaceMapCubic

Specialist operations for hypervolume texturing:

HfColor, HfColorUnion, HfGradient, HfWave, HfMapNoise, HfColorPattern, HfColorWall, ...

```
--This HyperFun program consists of one object:
--union of superellipsoid, torus and soft object

my_model(x[3], a[1])
{
array x0[9], y0[9], z0[9], d[9], center[3];
x1=x[1];
x2=x[2];
x3=x[3];

-- superellipsoid by formula
superEll = 1-(x1/0.8)^4-(x2/10)^4-(x3/0.8)^4;

-- torus by library function
center = [0, -9, 0];
torus = hfTorusY(x,center,3.5,1);

-- soft object
x0 = [2.,1.4, -1.4, -3, -3, 0, 2.5, 5., 6.5];
y0 = [8, 8, 8, 6.5, 5, 4.5, 3, 2, 1];
z0 = [0, -1.4,-1.4, 0, 3, 4, 2.5, 0, -1];
d = [2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.5, 2.7, 3];
sum = 0.;
i = 1;
while (i<10) loop
        xt = x[1] - x0[i];
        yt = x[2] - y0[i];
        zt = x[3] - z0[i];
        r = sqrt(xt*xt+yt*yt+zt*zt);
        if (r <= d[i]) then
                r2 = r*r; r4 = r2*r2; r6 = r4*r2;
                d2 = d[i]^2; d4 = d2*d2; d6 = d4*d2;
                sum = sum + (1 - 22*r2/(9*d2) +
                17*r4/(9*d4) - 4*r6/(9*d6));
        endif;
        i = i+1;
endloop;
soft = sum - 0.2;

-- final model as set-theoretic union
my_model = superEll | torus | soft;
}
```



HyperFun Polygonizer

- Polygonization Pasko et al. [1988]
- Command line interface
- VRML export
- MAM/VRS + Tcl/Tk
- Multi-Platform

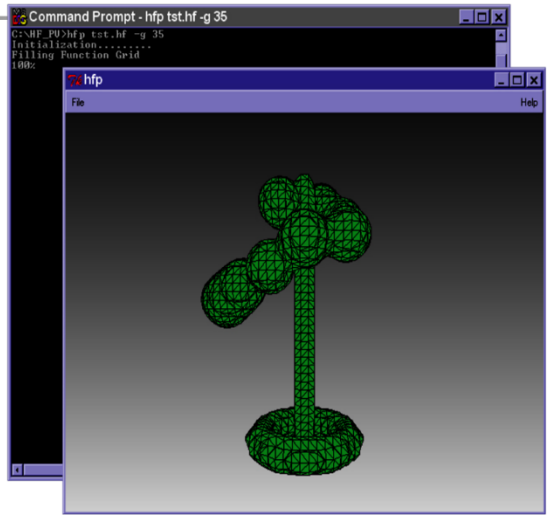


Figure 2.1

Your first VRML world

02fig01.wrl

Click on the image to view the VRML scene.

```
#VRML V2.0 utf8
# The VRML 2.0 Sourcebook
# Copyright (c) 1997
# Andrea L. Ames, David R. Nadeau, and John L. Moreland
# A brown hut
Group {
    children [
    # Draw the hut walls
        Shape {
            appearance DEF Brown Appearance {
                material Material {
                    diffuseColor 0.6 0.4 0.0
                }
            }
            geometry Cylinder {
                height 2.0
                radius 2.0
            }
        },
    # Draw the hut roof
        Transform {
            translation 0.0 2.0 0.0
            children Shape {
                appearance USE Brown
                geometry Cone {
                    height 2.0
                    bottomRadius 2.5
                }
            }
        }
    ]
}
```

- **https://www.wiley.com/legacy/compbooks/vrml2sbk/ch02/02fig01.htm**

Virtual Heart of Central Europe, Culture 2000

- Awarded by EuroPrix Quality Seal

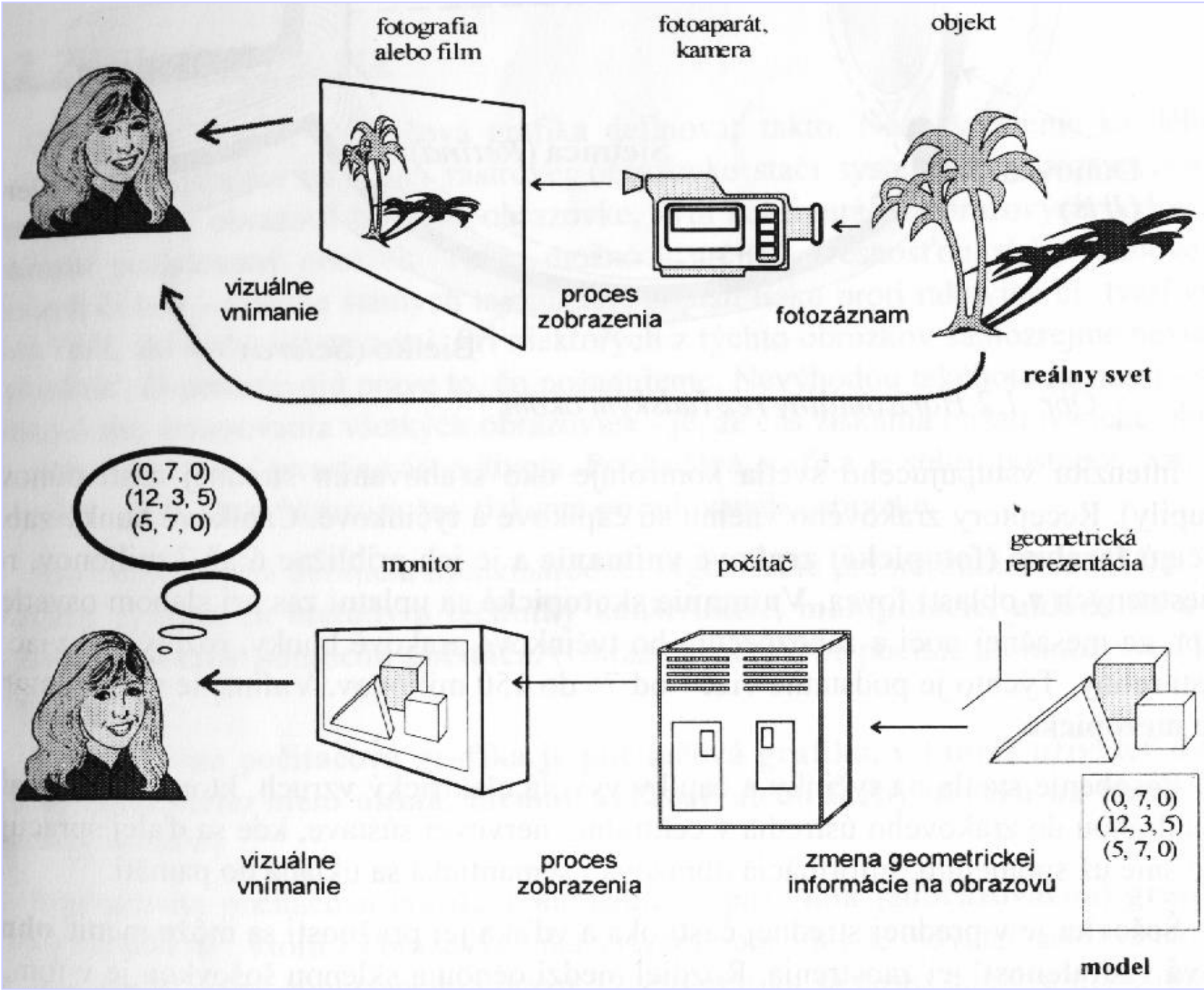
www.VHCE.info

- 330 kEUR, 150 kEUR from EC, Slovak Prix
- follow-up 2005-2006 (SK, SI, PL, CZ), submitted, 256 kEUR, rejected

Compare Reality - Synthesis

Photograph

Rendering using the deterministic method


fotografia
alebo film
fotoaparát,
kamera
objekt
vizuálne
vnímanie
proces
zobrazenia
fotozáznam
reálny svet
(0, 7, 0)
(12, 3, 5)
(5, 7, 0)
monitor
počítač
geometrická
reprezentácia
vizuálne
vnímanie
proces
zobrazenia
zmena geometrickej
informácie na obrazovú
(0, 7, 0)
(12, 3, 5)
(5, 7, 0)
model

Compare Workflow (PTGui)

Photograph

Rendering using the deterministic method

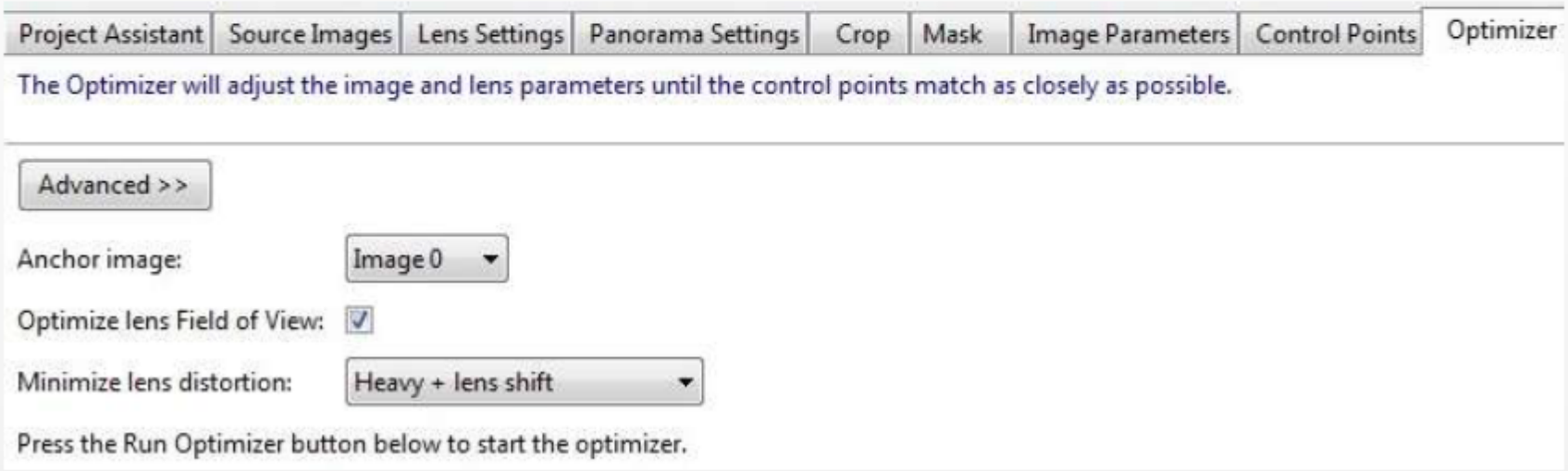
Stitching Workflow using PtGui

- Load the 5 aerial pictures into PtGui
- Assuming RAW images have been processed with LR

	Image	File	Width	Height
0		R:\Pano\Kopter\Olympiapark\P1050983.JPG	4000	3000
1		R:\Pano\Kopter\Olympiapark\P1050985.JPG	4000	3000
2		R:\Pano\Kopter\Olympiapark\P1050987.JPG	4000	3000
3		R:\Pano\Kopter\Olympiapark\P1050989.JPG	4000	3000
4		R:\Pano\Kopter\Olympiapark\P1050992.JPG	4000	3000

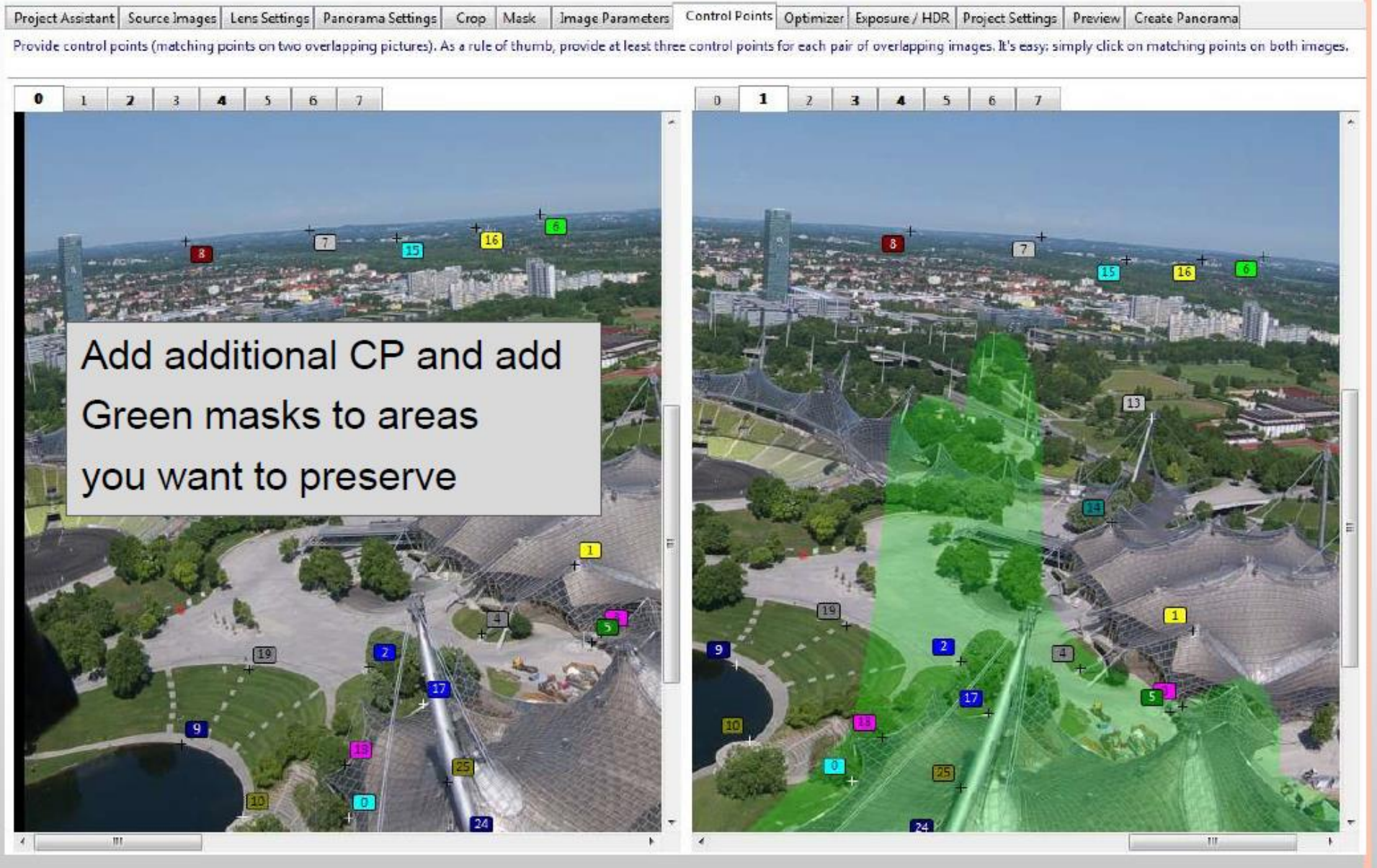
Stitching Workflow using PTGui

- Set Optimizer to Heavy + lens shift
- Align Images



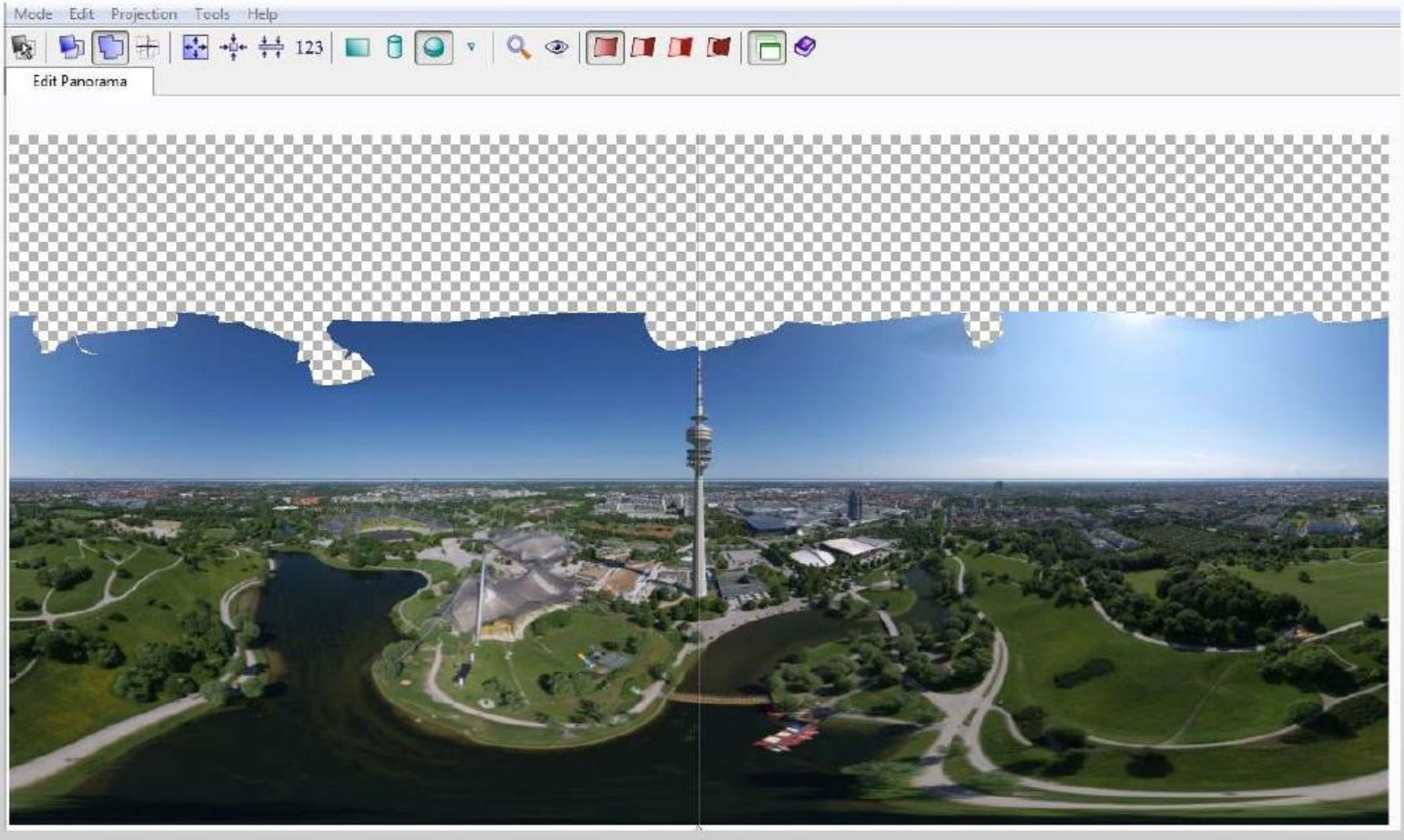
STITCHING WORKFLOW USING PTGUI

STITCHING WORKFLOW USING PTGUI

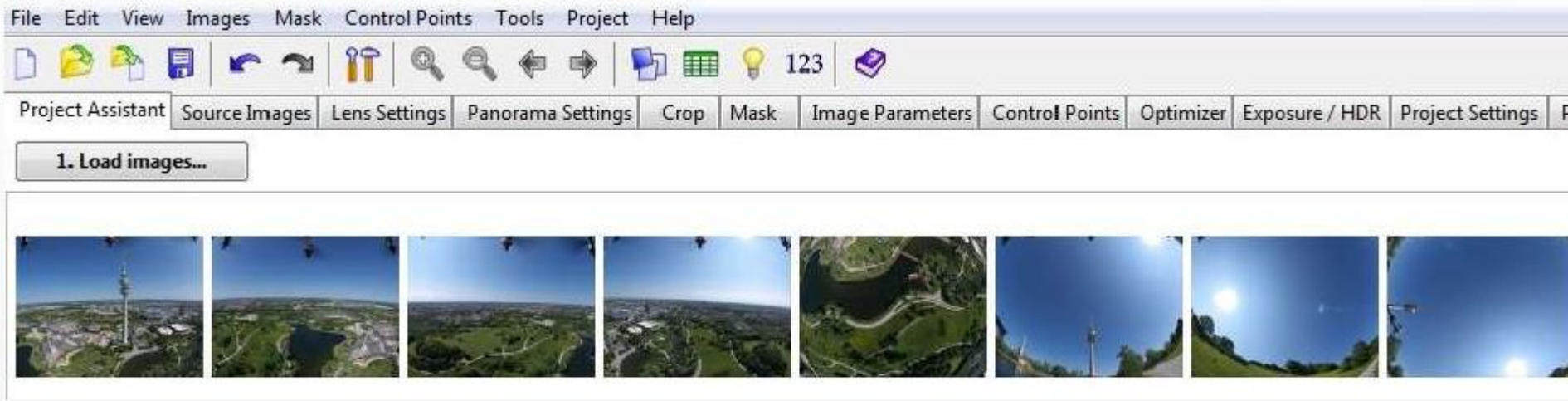


STITCHING WORKFLOW USING PTGUI

Align and optimize the spherical panorama



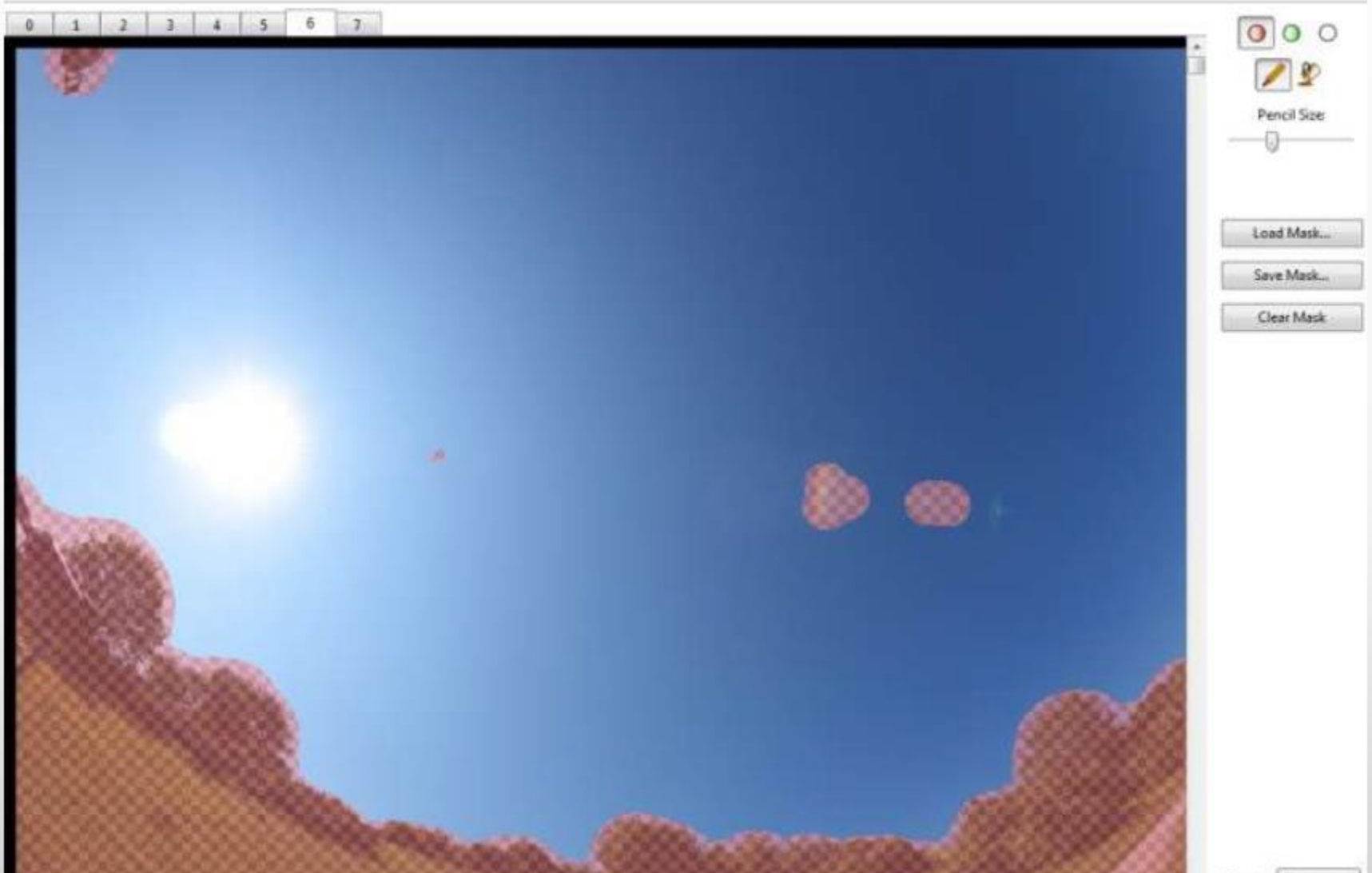
STITCHING WORKFLOW USING PTGUI



- Add 3 Zenith pictures
  - Advantage of having enough overlap to remove lens flares and ground objects
- Move the 3 pictures manually using the panorama editor into the ‘right’ position

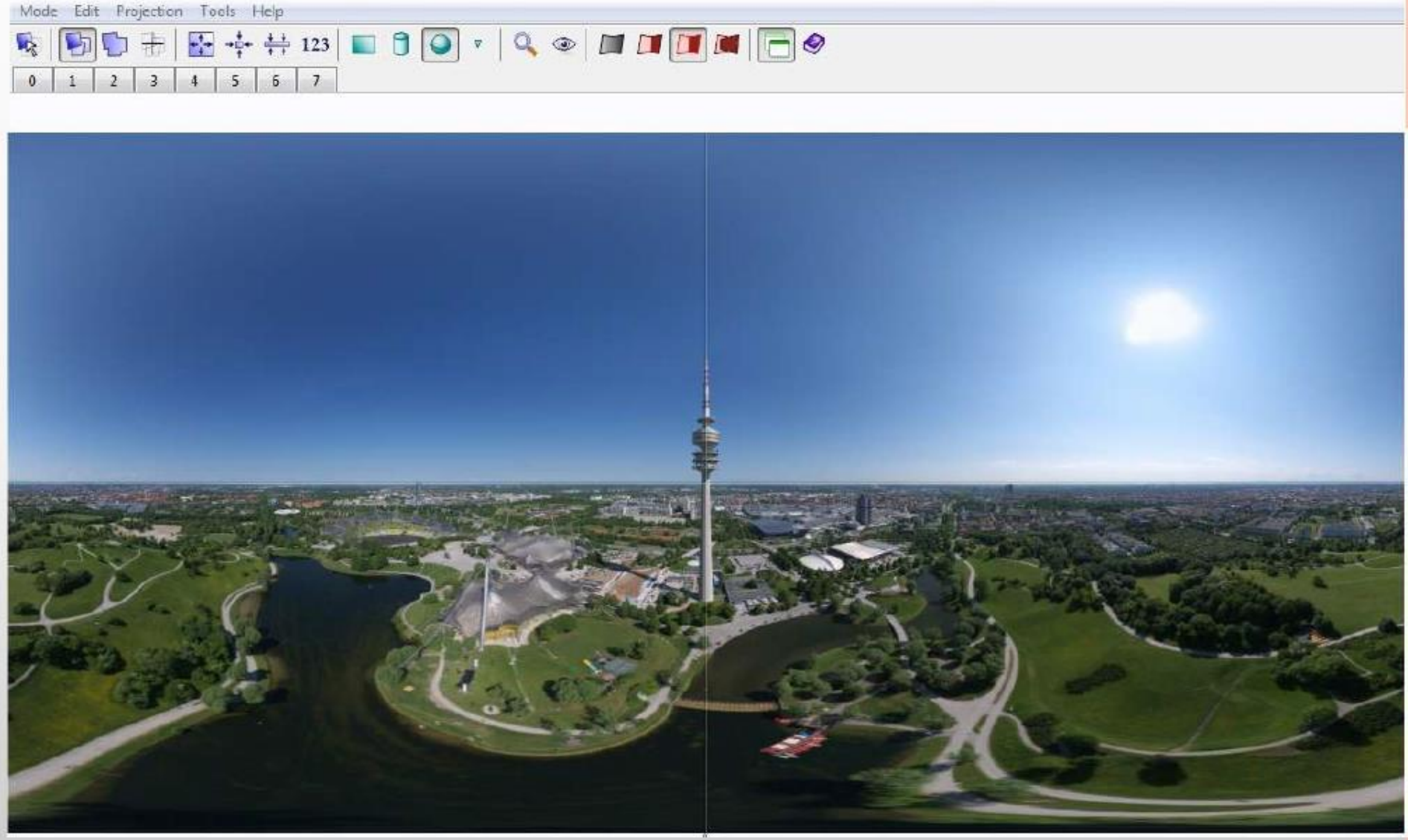
STITCHING WORKFLOW USING PTGUI

- Mask ground objects and lens flares



STITCHING WORKFLOW USING PTGUI

- Check the panorama in Editor



Compare Workflow (PTGui)

Photograph

Rendering using the deterministic method

Image-based rendering
motivates Image Processing...CV

Spracovanie obrazu je proces aplikovania akejkoľvek operácie na obraz alebo obrazové dáta pre daný účel. Napr. analýza scény, kompresia obrazu, konštrukcia 2D alebo 3D modelov objektov, ai.

vstup je daný ako	výstup je daný ako		
	popis	**obrázok**	**zvuk**
popis	symbolická manipulácia	počítačová grafika	hlasový výstup
obrázok	rozpoznávanie obrazov	spracovanie obrazu	
zvuk	rozpoznávanie zvuku		spracovanie zvuku

Obr. 1.4 Oblasti spracovania informácie

Počítačovej grafike príbuzné úlohy rieši aj **rozpoznávanie obrazcov** (*pattern recognition*). Vzťahy týchto a ďalších oblastí spracovania informácie objasňuje obr. 1.4, [SKAL93]. Prázdne políčka označujú zatiaľ nepomenované oblasti.

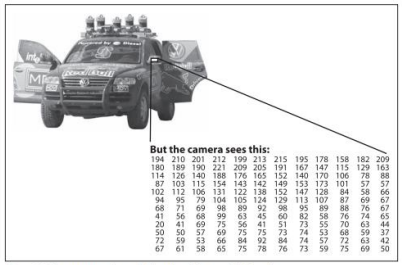


Figure 1-1. To a comput

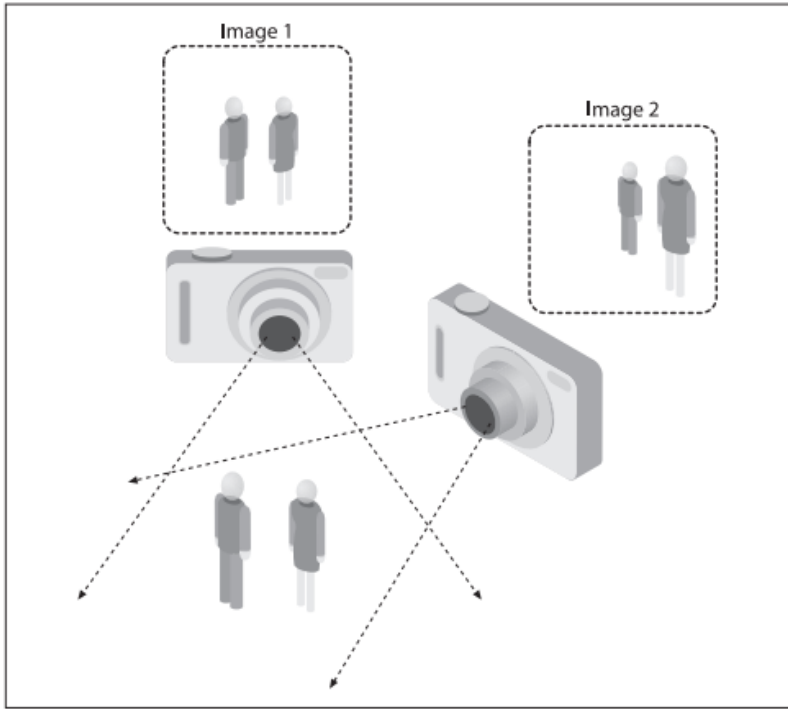
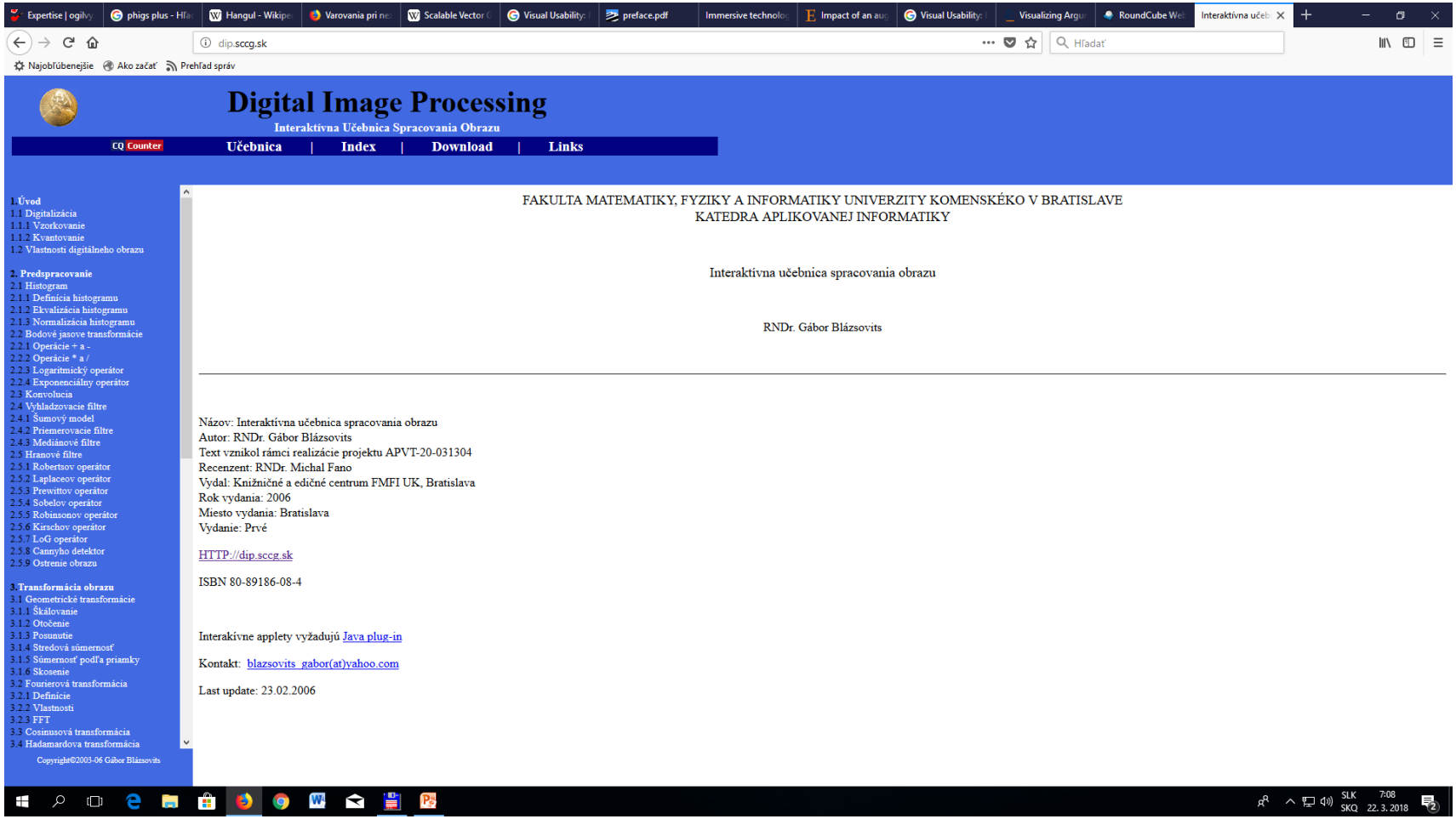


Figure 1-2. The ill-posed nature of vision: the 2D appearance of objects can change radically with viewpoint

Learning OpenCV: Computer vision with the OpenCV … - Bradski Citované 5742-krát

SK: [Ru] & dip.sccg.sk



Digital Image Processing

Interaktívna Učebnica Spracovania Obrazu

CQ Counter | Učebnica | Index | Download | Lin

dip.sccg.sk

FAKUL

Názov: Interaktívna učebnica spracovania obrazu
Autor: RNDr. Gábor Blázsovits
Text vznikol rámci realizácie projektu APVT-20-031304
Recenzent: RNDr. Michal Fano
Vydal: Knižničné a edičné centrum FMFI UK, Bratislava
Rok vydania: 2006
Miesto vydania: Bratislava
Vydanie: Prvé

HTTP://dip.sccg.sk

ISBN 80-89186-08-4

Interakívne applety vyžadujú Java plug-in

Kontakt: blazsovits_gabor(at)yahoo.com

Last update: 23.02.2006

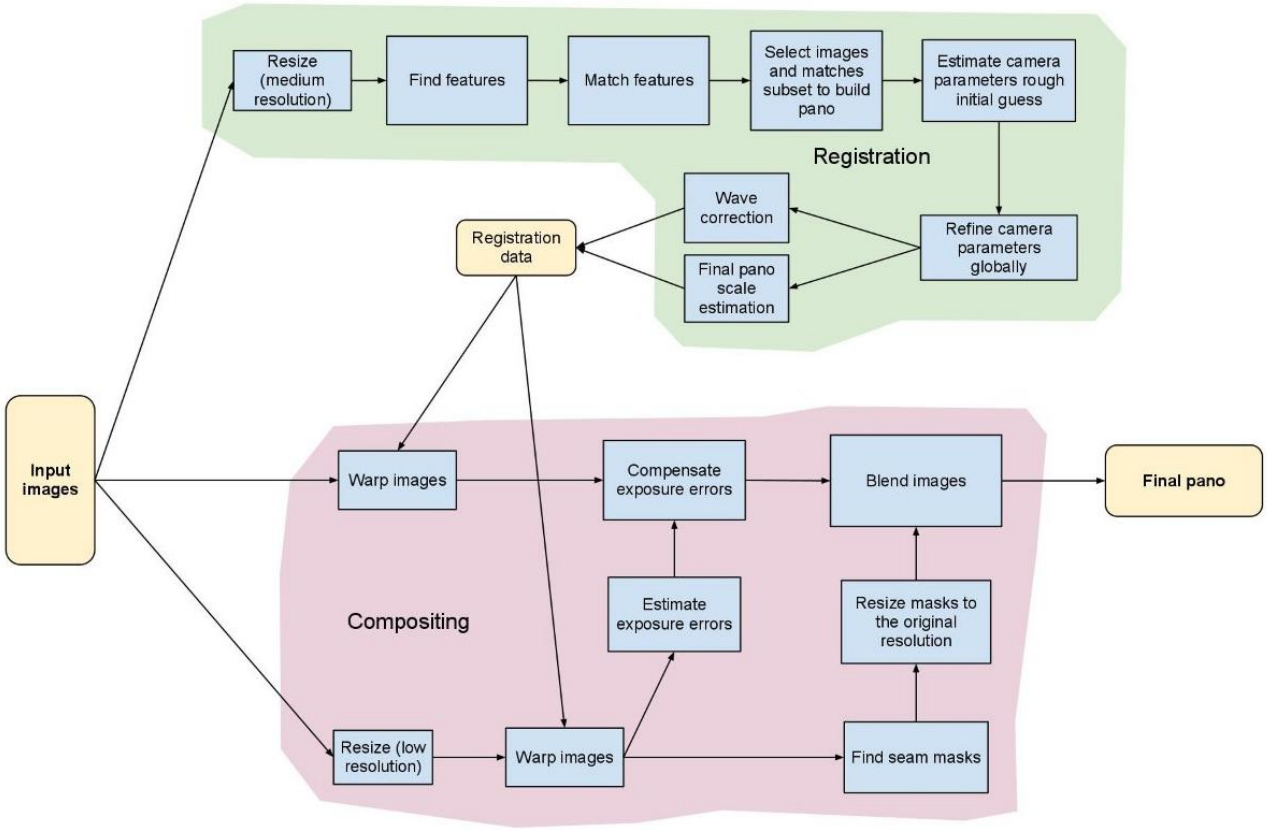
Copyright©2003-06 Gábor Blázsovits

From digital image (Ruzicky) to IBR (OpenCV pipeline)

Obr. 6.1 Obrazová funkcia získaná vzorkovaním

http://www.sccg.sk/~ferko/PGASO2012-bookmarks.pdf

Image-based rendering
(e.g. OpenCV pipeline)



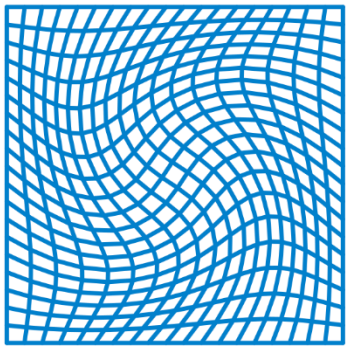
Intro Reading: e.g. https://en.wikipedia.org/wiki/Image_stitching

Image registration is the process of transforming different sets of data into one coordinate system.

Image Registration

Image registration, transforming different sets of data into one coordinate system.

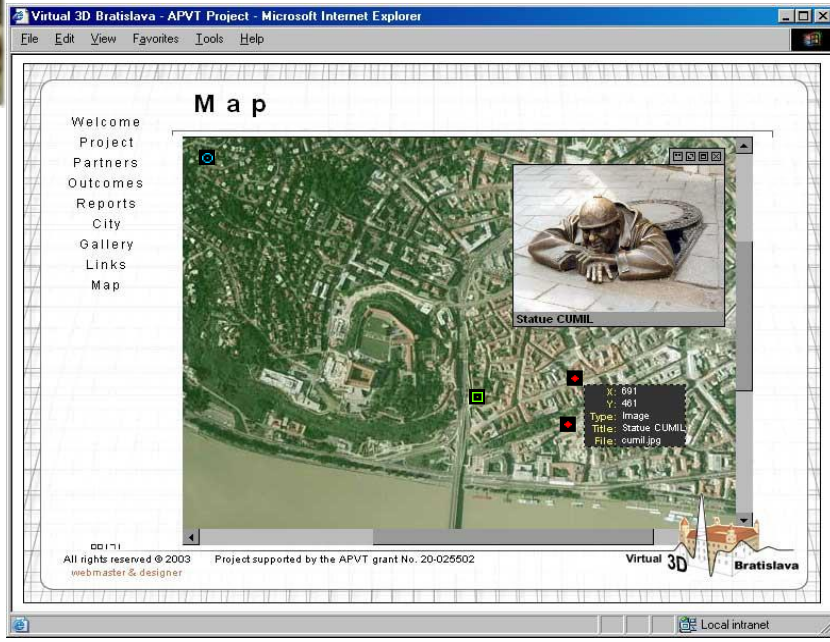
e.g. **Diffeomorphism,** an isomorphism of smooth manifolds, an invertible function that maps one differentiable manifold to another such that both the function and its inverse are smooth



Why?
to see things previously impossible to see,
e.g. the distant Alps

https://en.wikipedia.org/wiki/Image_registration

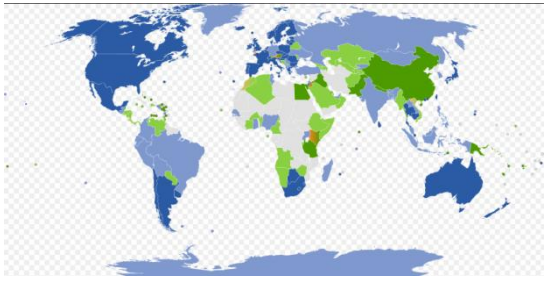
History 2, Bratislava



Veduta, malovana rovinna panorama a dvojpohladova vizualizacia, VrBa.

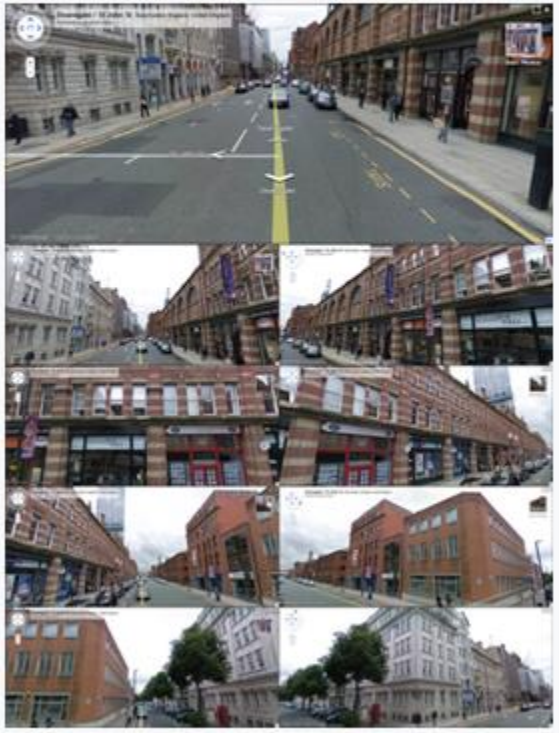
History 3… Street View 2007

The facades of buildings were texture-mapped onto 3D models. The same 3D model was used to translate 2D screen coordinates into a database of buildings in order to provide hyperlinks to additional data.



Aspen Movie Map, MBR >> IBR, 20 peta 2012

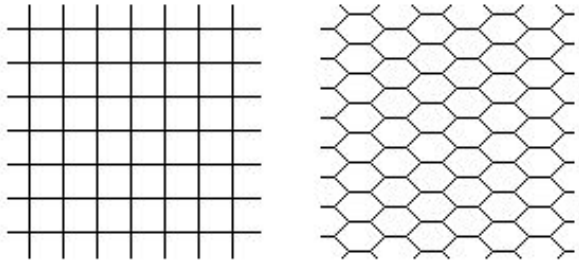
Google Street View



A road junction in Manchester, England, showing nine different angles

Initial release May 25, 2007; 10 years ago

Digital Image



Štvorcová a hexagonálna vzorkovacia mriežka.

1.2 Vlastnosti digitálneho obrazu

V tejto učebnici slovo **obraz**, alebo šedotónový obraz bude vyjadrovať dvojrozmernú jasovú funkciu f(x,y). Definičným oborom obrazovej funkcie bude rovinná oblasť R:

$$R = \{(x,y), 0 \le x \le x_n, 0 \le y \le y_m\} \quad (1)$$

kde x,y sú celé čísla, x_n, y_m sú maximálne súradnice. Obor hodnôt je celočiselná množina jasových hodnôt.

V digitálnom obraze môžeme zaviesť **vzdialenosť** medzi dvoma bodmi. Nech (i,j) (k,l) sú dva obrazové elementy, potom vzdialenosť môžeme definovať nasledujúcimi spôs

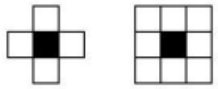
$$D_E = \sqrt{(i-k)^2 + (j-l)^2} \quad (2)$$

$$D_4 = |i-k| + |j-l| \quad (3)$$

$$D_8 = \max\{|i-k|, |j-l|\} \quad (4)$$

Kde D_E je Euklidovská vzdialenosť, ktorá ale nie je vhodná pre diskrétny obraz, lebo nemusí vrátiť celé číslo. Vzdialenosti D_4, D_8 určujú najmenší počet jednotkových krok mriežke. V prípade D_4 posun je povolený len vo zvislom alebo vo vodorovnom smere. V prípade D_8 sú povolené aj diagonálne pohyby.

Ďalším dôležitým pojmom je **susednosť**. Rozlišujeme **4-susednosť** a **8-susednosť**. 4-susedia daného obrazového elementu sú body s jednotkovou vzdialenosťou v vzdialenosťou v D_8. Tiež sa hovorí **4-okolie**, alebo **8-okolie**.

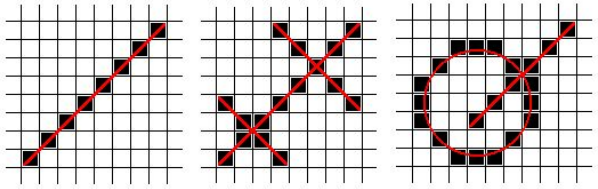


Digital Image Properties

Oblasť je súvislá množina obrazových elementov, pre ktorú platí, že medzi každými dvoma bodmi existuje cesta patriaca celá do tejto množiny. Predpokladajme, že Ri sú oblasti obrazu. Nech R je oblasť ktorá vznikne zjednotením všetkých oblastí Ri. Potom R^C je množinovým doplnkom oblasti R, nazývame ho pozadím.

Objekty sú oblasti, ktoré obvykle odpovedajú entitám zobrazovaného sveta. V jednoduchom praktickom prípade, keď má bod jas väčší ako určitý prah, priradíme ho k objektu.

Súvislosť a susednosť definovaná na diskrétnej štvorcovej mriežke nás privedie k určitým paradoxom. Predstavme si úsečku s 45 stupňovým sklonom v digitálnom obraze. Ak uvažujeme 4-susednosť, potom táto úsečka je v každom svojom bode nesúvislá. Ďalším paradoxom je, že dve pretínajúce sa úsečky v digitálnom obraze sa len dotýkajú.



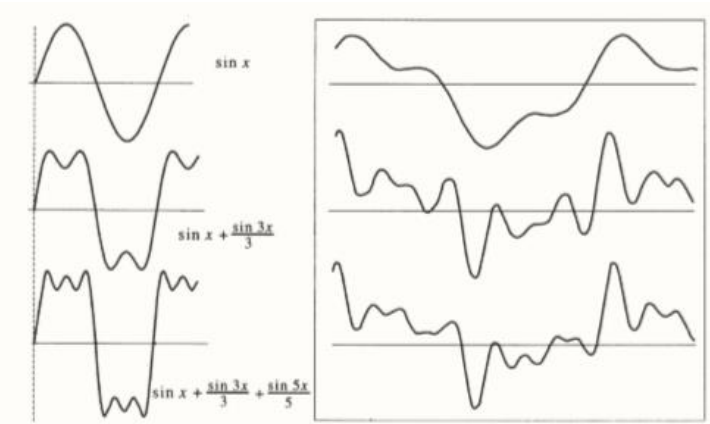
a) v prípade 4-susednosti, úsečka je v každom svojom bode nesúvislá.
b) v pravo hore sa úsečky pretínajú, kým ľavo dole sa len dotýkajú, t.j. nemajú spoločný bod.
c) z euklidovskej geometrie platí, že uzavretá krivka delí priestor na dve časti. V digitálnom obraze to ale nemusí byť pravda. Na obrázku vidíme kruh, t.j. uzavretú krivku, a úsečku ktorá ju nepretína, ale spája body z vnútra s bodmi z vonkajška.

Jedným riešením pre tieto paradoxy je použiť 8-susednosť pre objekty a 4-susednosť pre pozadie. Je to ale nepraktické riešenie. Ďalšou možnosťou je použiť hexagonálnu mriežku, v ktorej paradoxy nevznikajú. Narazíme ale na realizačný problém, pretože väčšina grafických zariadení podporuje štvorcový raster.

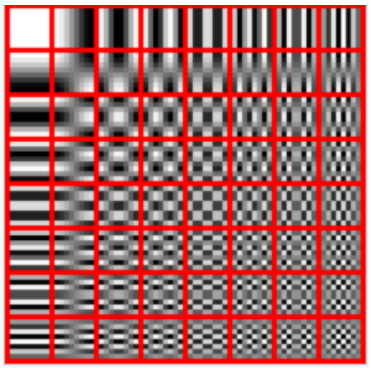
Pod pojmom **hranica oblasti** rozumieme množinu všetkých bodov, ktoré majú aspoň jedného suseda, ktorý nepatrí do oblasti. V digitálnom obraze rozlišujeme **vonkajšiu** a **vnútornú** hranicu. Pre vnútornú hranicu potom platí predošlá definícia. Vonkajšia hranica je hranicou pozadia.

DIP - Digital Image Processing, Interaktívna učebnica spracovania obrazu
Copyright©2003-06 Gábor Blázsovits, Katedra aplikovanej informatiky FMFI UK Bratislava

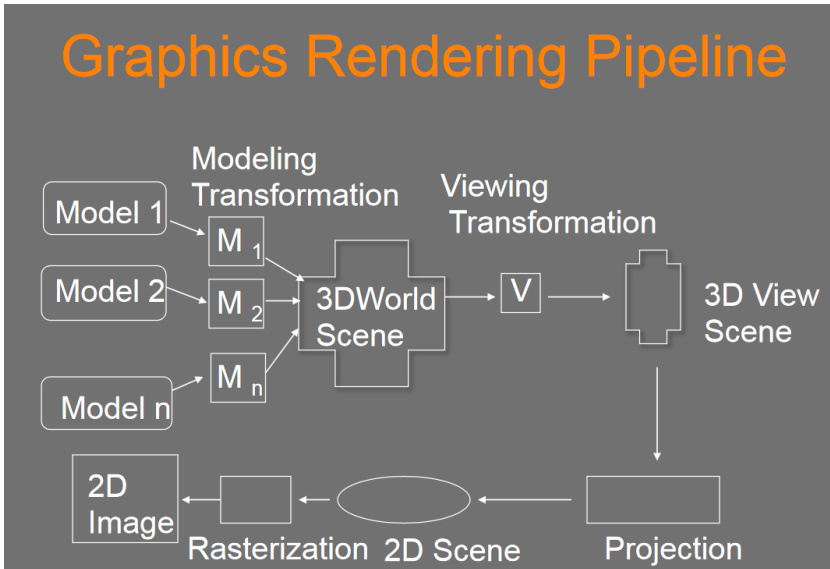
[Ru], Wi, [Kalra]



Obr. 6.2 Signál vyjadrený frekvenciou harmonických funkcií



The DCT transforms an 8×8 block of input values to a linear combination of these 64 patterns. The patterns are referred to as the two-dimensional DCT *basis functions*, and the output values are referred to as *transform coefficients*. The horizontal index is u and the vertical index is v.



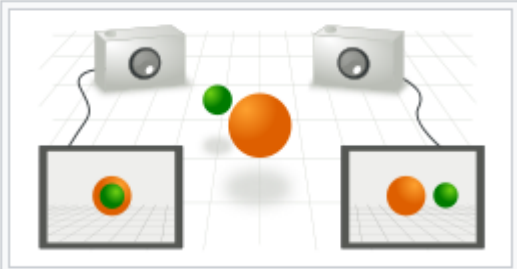
Digital Image >> Model

- Model>Picture: SetPixel
- Model>Picture: GKS 6, **polyline, polymarker, fill area, text, cell array, NUB (GDP)**
- Model>Picture: SVG 14, **path, basic shapes, text… or feature sets**

- Image>Model: **e.g. Harris Corner Detector, interaction**
- Image>Model: **e.g. Harris Edge Detector, snake… CV (OCR etc.)**

- **Image Registration**
- **Mathematic Morphology**

- **Image Correspondence**
- **Image Segmentation**



Typical use case for epipolar geometry
Two cameras take a picture of the same scene from different points of view. The epipolar geometry then describes the relation between the two resulting views.

Source image.

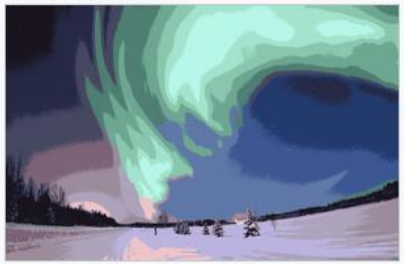


Image after running *k*-means with *k* = *16*. Note that a common technique to improve performance for large images is to downsample the image, compute the clusters, and then reassign the values to the larger image if necessary.

Digital Image >> Model

- Model>Picture: SetPixel
- Model>Picture: GKS 6, **polyline, polymarker, fill area, text, cell array, NUB (GDP)**
- Model>Picture: SVG 14, **path, basic shapes, text… or feature sets**

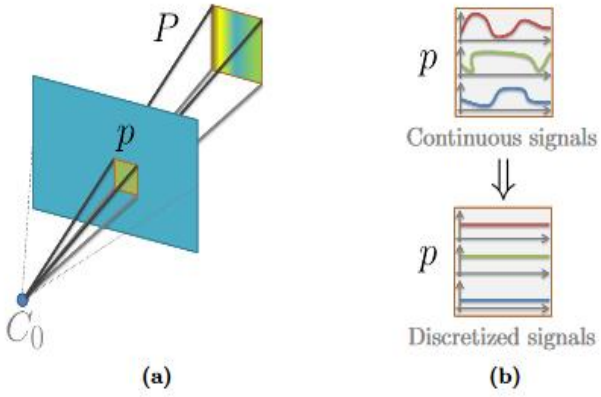


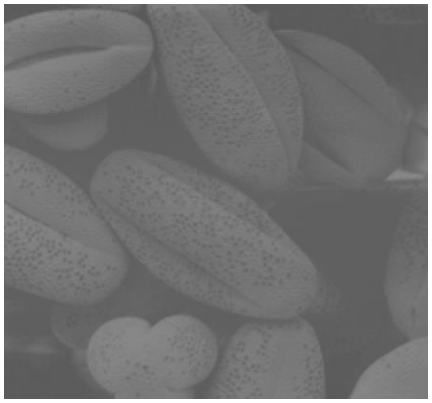
Figure 2.1: Image Formation. A 2D pixel **p** is formed by projecting a 3D frustrum onto the camera sensor (a). Within this pixel, continuous RGB signals are discretized to an average RGB value, losing spatial resolution (b).

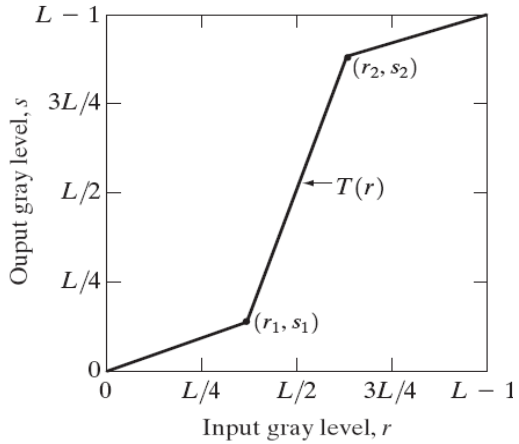
- Image>Model: **e.g. Harris Corner Detector, interaction**
- Image>Model: **e.g. Harris Edge Detector, snake… CV (OCR etc.)**

- **Image Registration**
- **Mathematic Morphology**

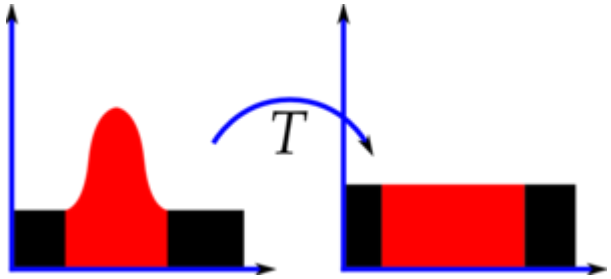
Digital Image Processing (Low Level)

- **The negative of an image with gray levels in the range [0,255], s = 255 – r**
- **Contrast stretching by increasing the dynamic range [Benesova]**



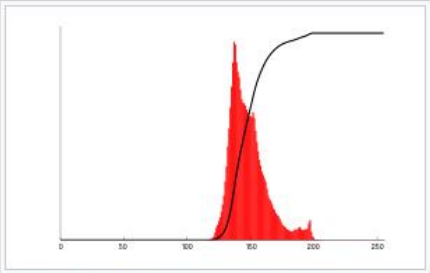


- **The histogram of a digital image with gray levels in the range [0, L-1] is a discrete function h (r_k) = n_k, where r_k is the *k*-th gray level and n_k is the number of pixels in the image having gray level r_k [Benesova] >> EQUALIZATION**
- **Image averaging**
- **Image subtraction**
- **Smoothing/sharpening**



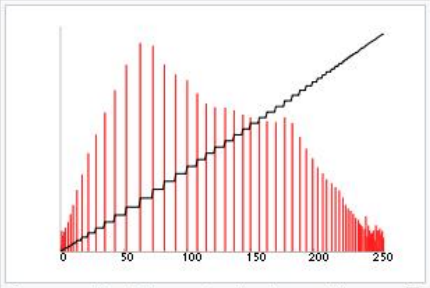
Histogram Equalization

Before Histogram Equalization



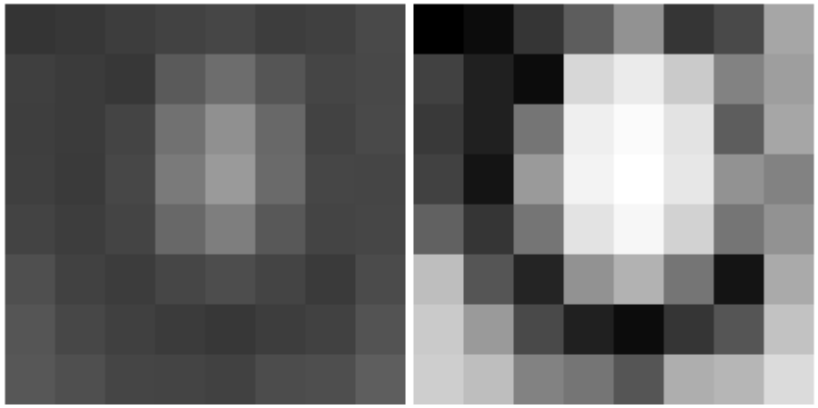
Corresponding histogram (red) and cumulative histogram (black)

After Histogram Equalization



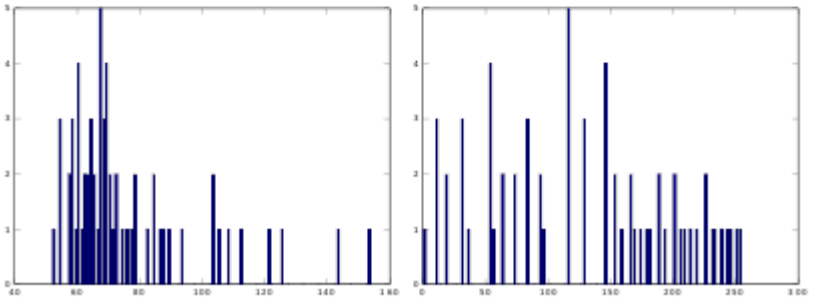
Corresponding histogram (red) and cumulative histogram (black)

https://en.wikipedia.org/wiki/Histogram_equalization



Original

Equalized



Histogram of Original image

Histogram of Equalized image

Digital Image Transformations

- **Translation, rotation, scale, symmetry, skew >> dip.sccg.sk**

- **Pixel approximation: nearest neighbour (1 pixel), bilinear interpolation (4 pixels), bicubic interpolation (9 pixels)**

V praxi sa táto rovnica nahradzuje **bilineárnou transformáciou**, alebo **afinnou transformáciou**. Bilineárna transformácia má tvar:

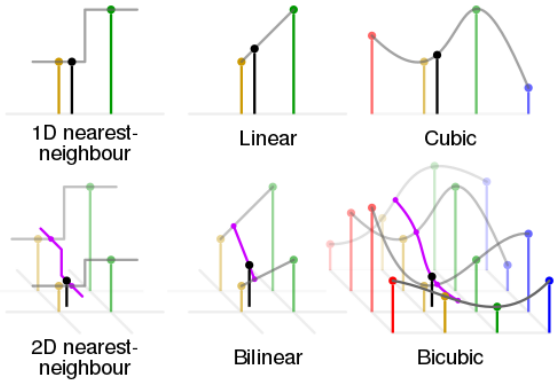
$$x_1 = a_0 + a_1 x + a_2 y + a_3 xy$$
$$y_1 = b_0 + b_1 x + b_2 y + b_3 xy \qquad (3)$$

Na jeho určenie potrebujeme štyri dvojice vstupných a výstupných bodov.
Na určenie afinnej transformácie stačia tri dvojice bodov, a má tvar:

$$x_1 = a_0 + a_1 x + a_2 y$$
$$y_1 = b_0 + b_1 x + b_2 y \qquad (4)$$

Pomocou **homogénnych súradníc** môžeme afinné transformácie vyjadriť v maticovom tvare

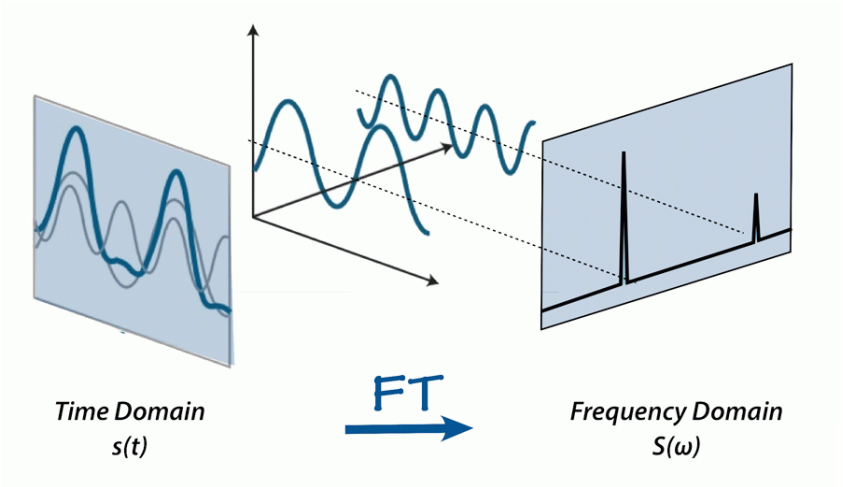
$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad (5)$$

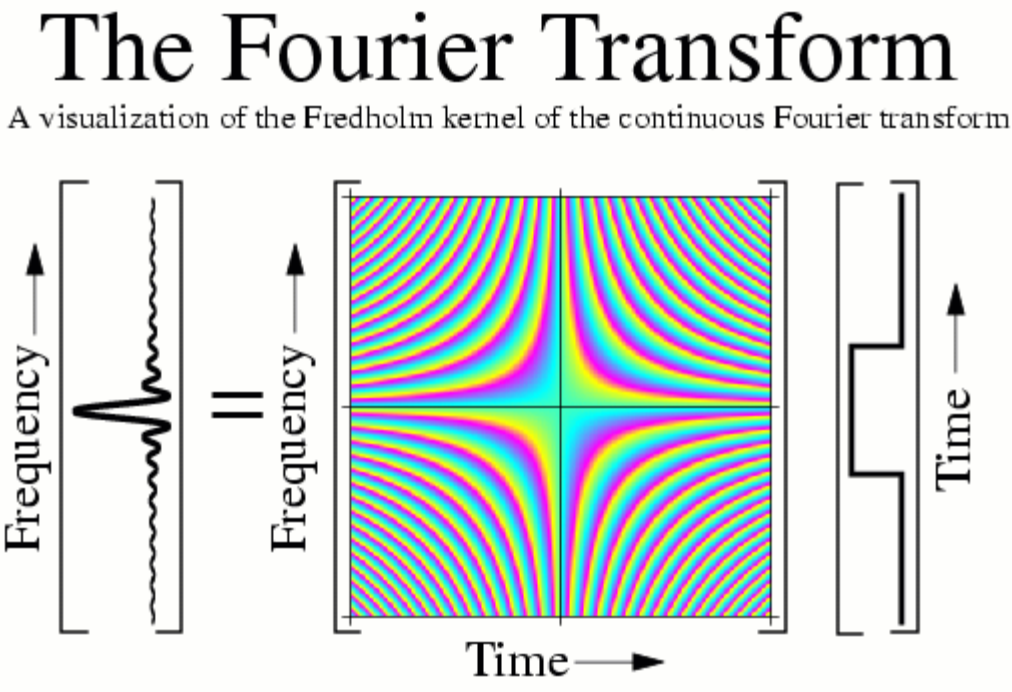


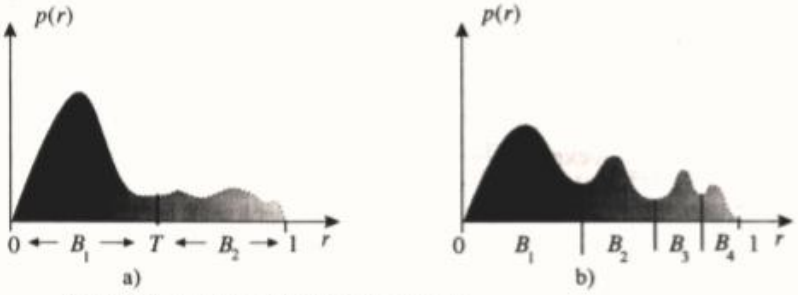
https://en.wikipedia.org/wiki/Bicubic_interpolation#/media/File:Comparison_of_1D_and_2D_interpolation.svg

- **Fourier transform >> dip.sccg.sk, Ruzicky, Sikudova**

Fourier Transform Visually







Obr. 8.2 Histogram globálneho prahovania

Z toho dôvodu určíme hranice v horizontálnom aj vertikálnom smere dvoma prechodmi obrazom $f(x, y)$. Vo všeobecnosti môžu byť určené intervaly hodnôt $B1$, $B2$, ..., Bn.

Algoritmus vyhľadania hranice

1. Pre každý riadok v obraze $f(x, y)$ (t.j. $y = 0, 1, ..., N-1$), vytvoríme príslušný riadok v pomocnom obraze $g_1(x, y)$ použitím nasledovného vzťahu pre $x = 1, 2, ..., N-1$:

$g_1(x, y) = L_E$, ak $f(x, y)$ a $f(x-1, y)$ sú v rôznych intervaloch, (1)
L_B, inak,

kde L_E a L_B sú špecifikované úrovne **hranice** (*edge*) a **pozadia** (*background*).

2. Pre každý stĺpec v obraze $f(x, y)$ (t.j. $x = 0, 1, ..., N-1$), vytvoríme príslušný stĺpec v pomocnom obraze $g_2(x, y)$ použitím nasledovného vzťahu pre $y = 1, 2, ..., N-1$:

$g_2(x, y) = L_E$, ak $f(x, y)$ a $f(x, y-1)$ sú v rôznych intervaloch, (2)
L_B, inak.

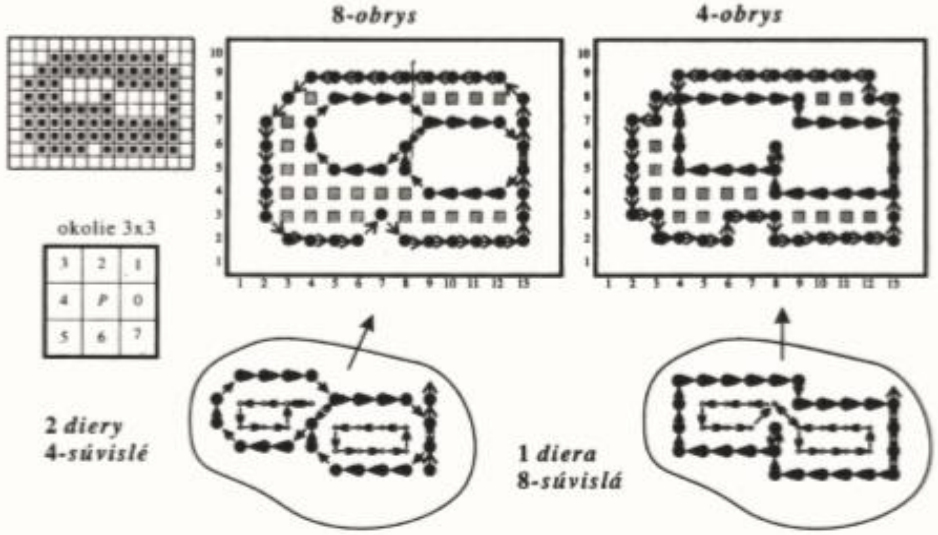
3. Požadovaný obraz, pozostávajúci z hranice objektov odlíšených od pozadia je získaný použitím nasledovného vzťahu pre $x, y = 1, 2, ..., N-1$:

$g(x, y) = L_E$, ak $g_1(x, y)$ alebo $g_2(x, y)$ je rovné L_E,
L_B, inak.

Predpokladáme, že oblasť je zadaná vo vnútri a nie na okraji obdĺžnika obrazovej mapy $h[i, j]$, pretože pre tieto body funkcia *neighb_h* nie je definovaná jednoznačne.

Algoritmus postupného prehľadávania hraničných bodov

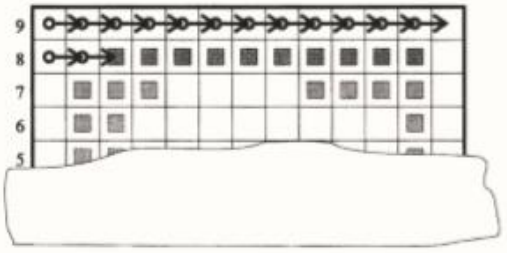
```
Procedure Bound_8;                          { farby sú: 0 - doplnok, 1 - množina }

begin
   for i:= 1 to Xmax do
      for j:= 1 to Ymax do
      begin
        if h[i, j] = 1 then                          { bod z množiny }
        begin
          if neighb_h (i, j, 0 ) = 0 then h[i, j]:= 2 else     { bod z doplnku množiny }
          if neighb_h (i, j, 2 ) = 0 then h[i, j]:= 2 else
          if neighb_h (i, j, 4 ) = 0 then h[i, j]:= 2 else
          if neighb_h (i, j, 6 ) = 0 then h[i, j]:= 2
        end
      end
end;
```



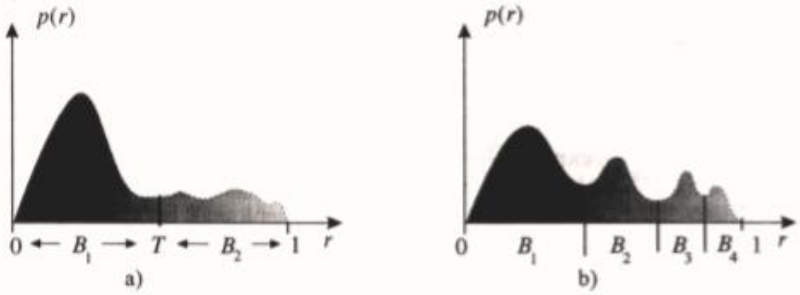
Obr. 8.5 Rozdiel medzi 8-obrysom a 4-obrysom tej istej oblasti

8.3.2 Jednoduchý algoritmus obrysu

Predpokladajme ako v predchádzajúcej časti, že oblasť máme zadanú v matici h[]. Určíme prvý hraničný bod, ktorý bude prvým bodom postupnosti obrysu. Tento bod vyhľadáme napríklad tak, že postupne prehľadávame body v riadkoch, až kým nenájdeme prvý bod s hodnotou 1 (pozri obr. 8.6). Túto procedúru vyhľadania prvého bodu označme *first*. Aktuálny bod obrysu označme *P*.



Obr. 8.6 Vyhľadanie prvého bodu v algoritme určenia obrysu



Obr. 8.2 Histogram globálneho prahovania

Z toho dôvodu určíme hranice v horizontálnom aj vertikálnom smere dvoma prechodmi obrazom $f(x, y)$. Vo všeobecnosti môžu byť určené intervaly hodnôt $B1$, $B2$, ..., Bn.

Algoritmus vyhľadania hranice

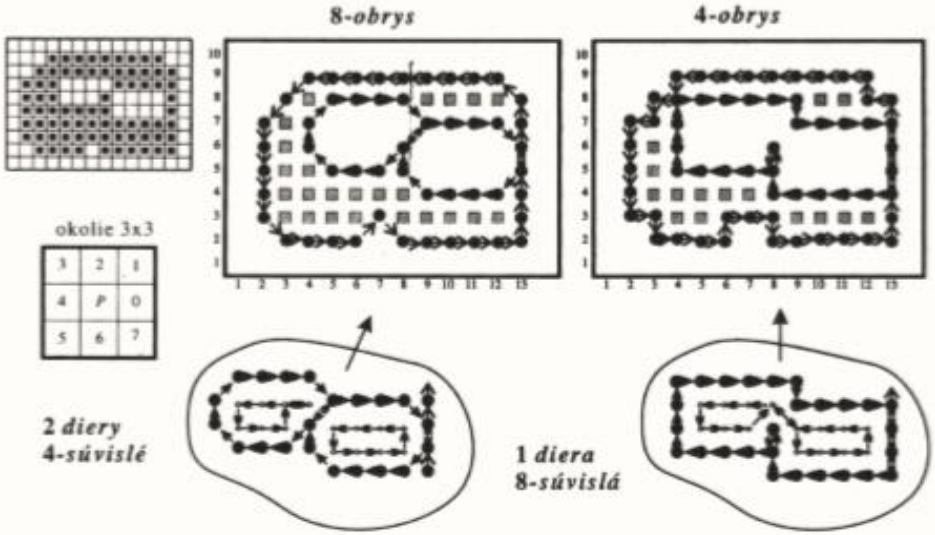
1. Pre každý riadok v obraze $f(x, y)$ (t.j. $y = 0, 1, ..., N-1$), vytvoríme príslušný riadok v pomocnom obraze $g_1(x, y)$ použitím nasledovného vzťahu pre $x = 1, 2, ..., N-1$:
   $g_1(x, y) = L_E$, ak $f(x, y)$ a $f(x-1, y)$ sú v rôznych intervaloch, (1)
   L_B, inak,
   kde L_E a L_B sú špecifikované úrovne **hranice** (*edge*) a **pozadia** (*background*).
2. Pre každý stĺpec v obraze $f(x, y)$ (t.j. $x = 0, 1, ..., N-1$), vytvoríme príslušný stĺpec v pomocnom obraze $g_2(x, y)$ použitím nasledovného vzťahu pre $y = 1, 2, ..., N-1$:
   $g_2(x, y) = L_E$, ak $f(x, y)$ a $f(x, y-1)$ sú v rôznych intervaloch, (2)
   L_B, inak.
3. Požadovaný obraz, pozostávajúci z hranice objektov odlíšených od pozadia je získaný použitím nasledovného vzťahu pre $x, y = 1, 2, ..., N-1$:

Predpokladáme, že oblasť je zadaná vo vnútri a nie na okraji obdĺžnika obrazovej mapy $h[i, j]$, pretože pre tieto body funkcia *neighb_h* nie je definovaná jednoznačne.

Algoritmus postupného prehľadávania hraničných bodov

```
Procedure Bound_8;                          { farby sú: 0 - doplnok, 1 - množina }

begin
   for i:= 1 to Xmax do
      for j:= 1 to Ymax do
      begin
        if h[i, j] = 1 then                          { bod z množiny }
        begin
          if neighb_h (i, j, 0 ) = 0 then h[i, j]:= 2 else     { bod z doplnku množiny }
          if neighb_h (i, j, 2 ) = 0 then h[i, j]:= 2 else
          if neighb_h (i, j, 4 ) = 0 then h[i, j]:= 2 else
          if neighb_h (i, j, 6 ) = 0 then h[i, j]:= 2
        end
      end
end;
```

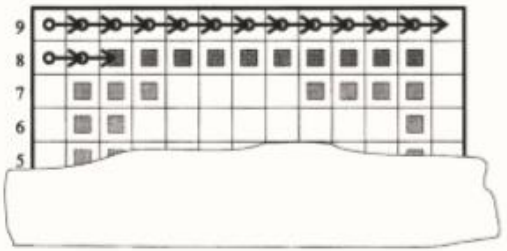
8.3.2 Jednoduchý algoritmus obrysu



Obr. 8.5 Rozdiel medzi 8-obrysom a 4-obrysom tej istej oblasti

8.3.2 Jednoduchý algoritmus obrysu

Predpokladajme ako v predchádzajúcej časti, že oblasť máme zadanú v matici h[]. Určíme prvý hraničný bod, ktorý bude prvým bodom postupnosti obrysu. Tento bod vyhľadáme napríklad tak, že postupne prehľadávame body v riadkoch, až kým nenájdeme prvý bod s hodnotou 1 (pozri obr. 8.6). Túto procedúru vyhľadania prvého bodu označme *first*. Aktuálny bod obrysu označme *P*.



Obr. 8.6 Vyhľadanie prvého bodu v algoritme určenia obrysu

Úvod do počítačovej grafiky
bez geometrie: Image-based Rendering

Andrej FERKO
Comenius University Bratislava
21. marca 2018, FMFI UK