Fundamentals of Computer Graphics

Shape visualization I
Piecewise-linear approximation

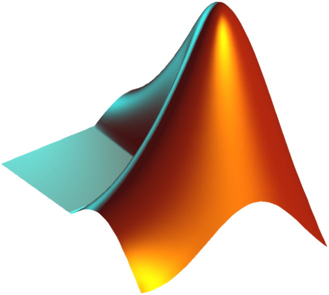
Emanuele Rodolà
rodola@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

1st semester a.y. 2018/2019 · October 04, 2018

Exercises

- **Warning:** .off files might have 0-based or 1-based indices depending on the tool used to create them
- **Demo:** Mesh from edge lengths



Visualizing shapes

Throughout this course we will primarily deal with manifold triangle meshes (possibly with boundary) and point clouds

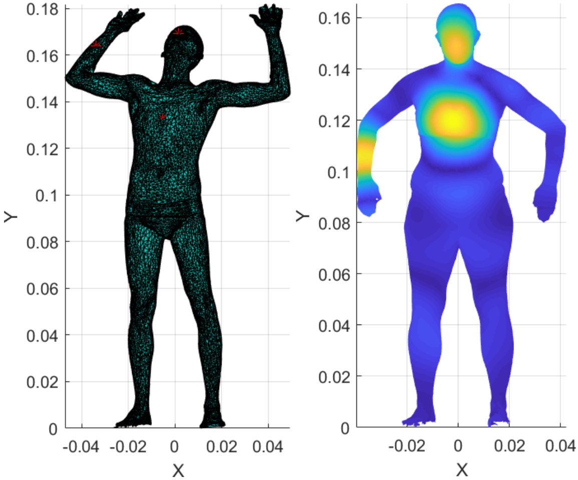
Visualizing shapes

Throughout this course we will primarily deal with manifold triangle meshes (possibly with boundary) and point clouds

How can we visualize our shapes (as well as functions, points, etc.) in the "best" possible way?

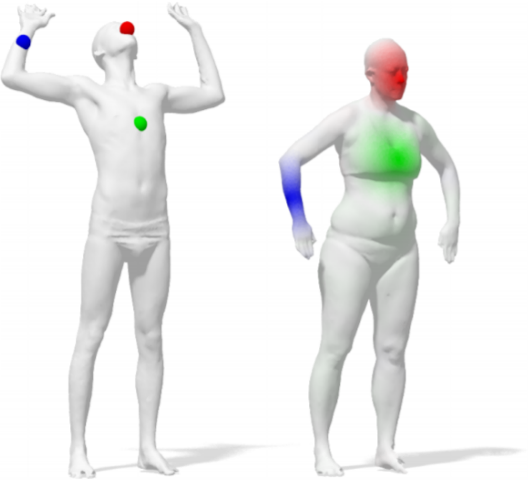
Visualizing shapes

Throughout this course we will primarily deal with manifold triangle meshes (possibly with boundary) and point clouds



Visualizing shapes

Throughout this course we will primarily deal with manifold triangle meshes (possibly with boundary) and point clouds



Visualizing shapes

Throughout this course we will primarily deal with manifold triangle meshes (possibly with boundary) and point clouds

Good visualization brings several benefits:

- Makes the key messages clear
- Puts the emphasis on the relevant aspects
- Sidesteps technical clutter and makes for a pleasant reading

Visualizing shapes

Throughout this course we will primarily deal with manifold triangle meshes (possibly with boundary) and point clouds

Good visualization brings several benefits:

- Makes the key messages clear
- Puts the emphasis on the relevant aspects
- Sidesteps technical clutter and makes for a pleasant reading

On the other hand, bad visualization:

- Might convey the wrong message, or hide it in unnecessary details
- Might convey no message at all
- Does not enrich the text

Visualizing shapes

Throughout this course we will primarily deal with manifold triangle meshes (possibly with boundary) and point clouds

Good visualization brings several benefits:

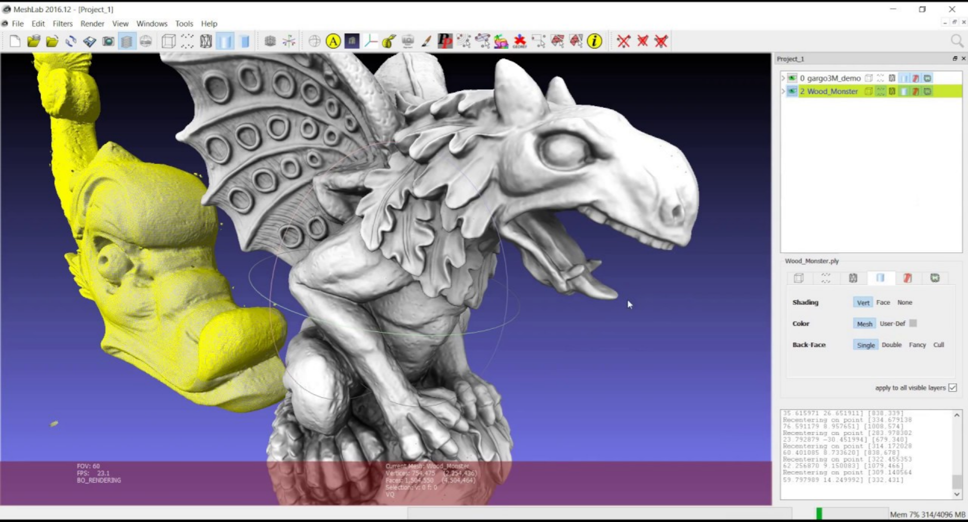
- Makes the key messages clear
- Puts the emphasis on the relevant aspects
- Sidesteps technical clutter and makes for a pleasant reading

On the other hand, bad visualization:

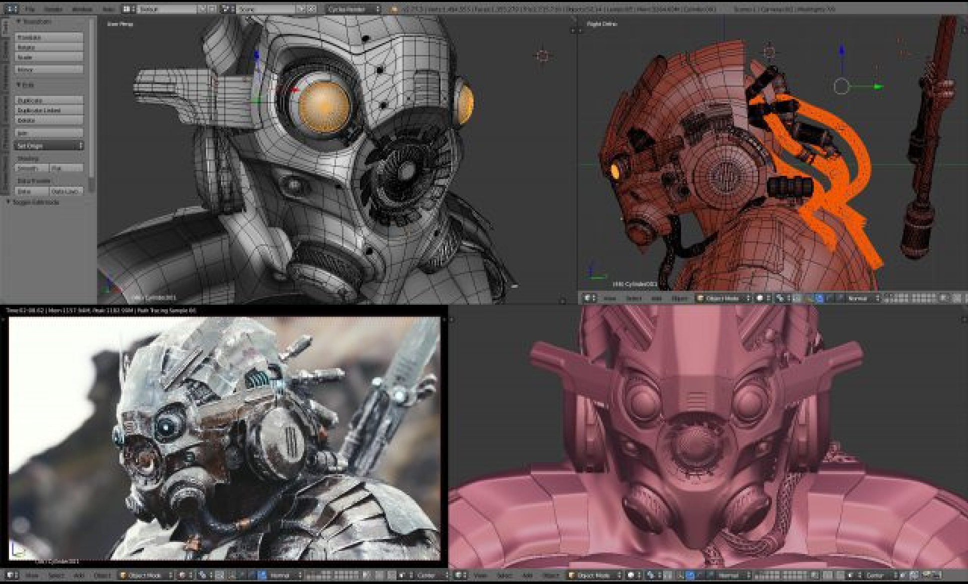
- Might convey the wrong message, or hide it in unnecessary details
- Might convey no message at all
- Does not enrich the text

Scientific visualization is a research area by itself!

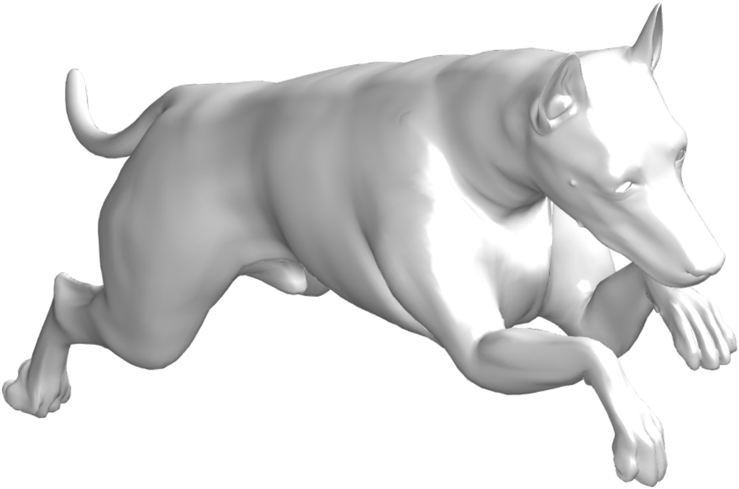
MeshLab (www.meshlab.net)



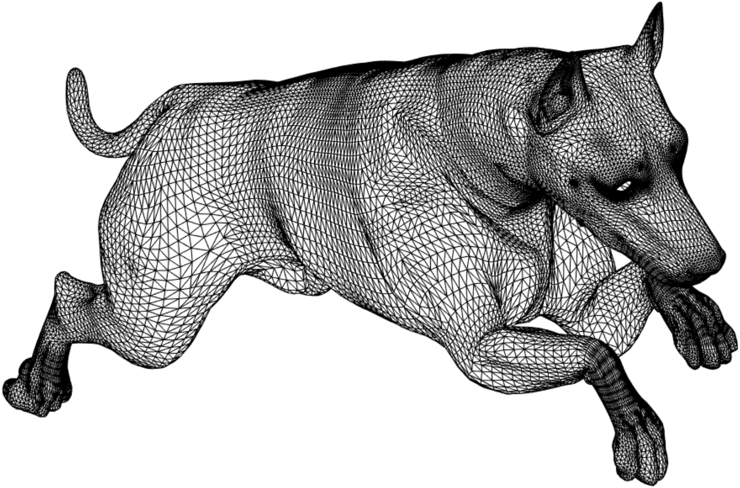
Blender (www.blender.org)



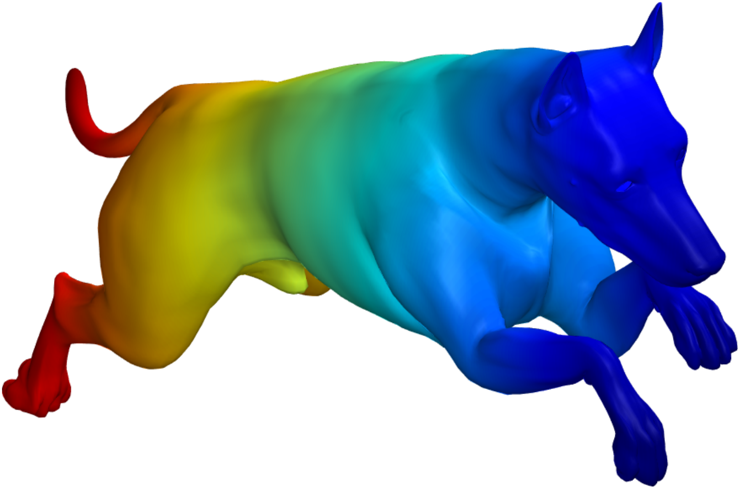
Plotting scalar functions



Plotting scalar functions

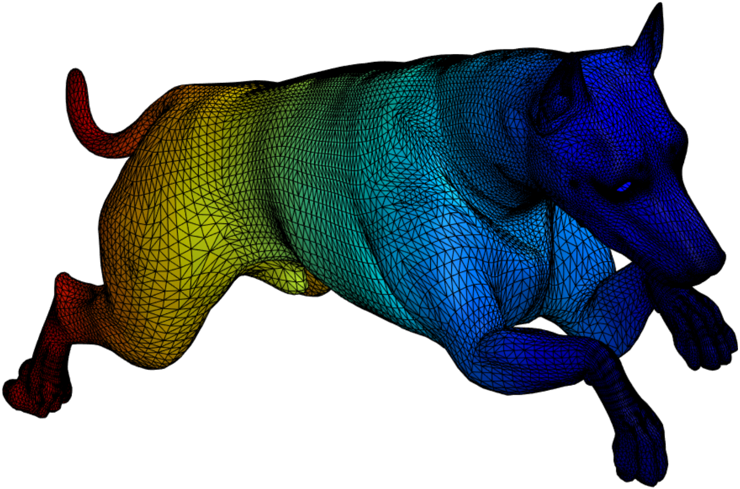


Plotting scalar functions



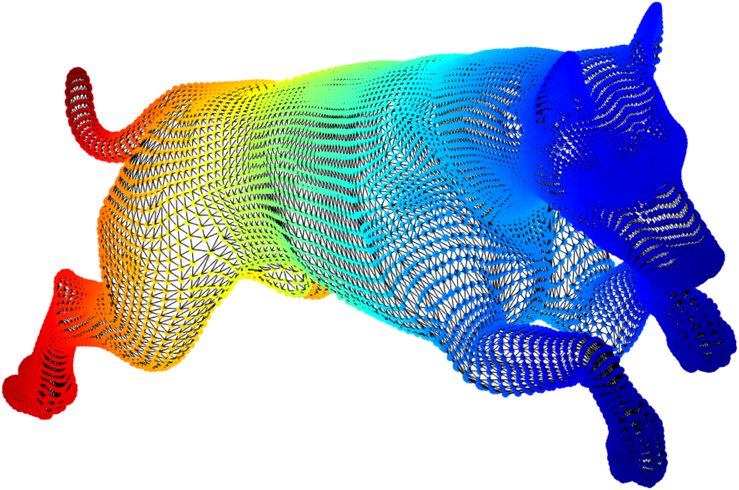
In Matlab: `trisurf(TRIV, X, Y, Z, f); shading interp;`
where $\mathbf{f}$ is a $n \times 1$ vector with a number per vertex

Plotting scalar functions



Matlab is actually coloring the triangles, not the vertices!

Plotting scalar functions

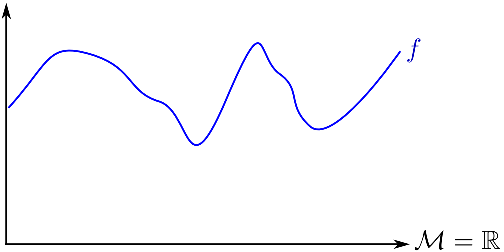


So what is happening inside the triangles?

Piecewise-linear approximation

Let us take one step back.

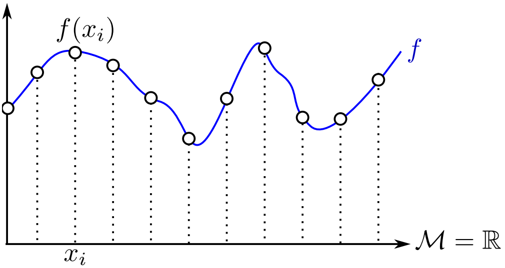
Consider a scalar function $f : \mathcal{M} \to \mathbb{R}$, which we want to represent on some discrete domain



Piecewise-linear approximation

Let us take one step back.

Consider a scalar function $f : \mathcal{M} \to \mathbb{R}$, which we want to represent on some discrete domain (here, a uniform partition of $\mathcal{M} = \mathbb{R}$)



Piecewise-linear approximation

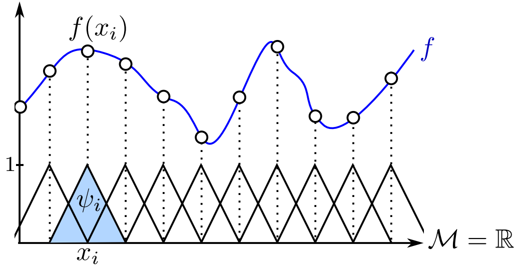
Let us take one step back.

Consider a scalar function $f : \mathcal{M} \to \mathbb{R}$, which we want to represent on some discrete domain (here, a uniform partition of $\mathcal{M} = \mathbb{R}$)

To do so, we approximate f by some other function $\tilde{f}$ using linear combinations of basis functions:

$$f \approx \tilde{f}$$
$$\tilde{f} = \sum_i f(x_i)\psi_i$$

Here, ψ_i are "hat" basis functions and $f(x_i)$ are approx. coefficients



Piecewise-linear approximation

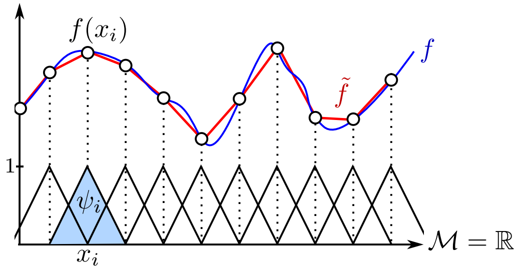
Let us take one step back.

Consider a scalar function $f : \mathcal{M} \to \mathbb{R}$, which we want to represent on some discrete domain (here, a uniform partition of $\mathcal{M} = \mathbb{R}$)

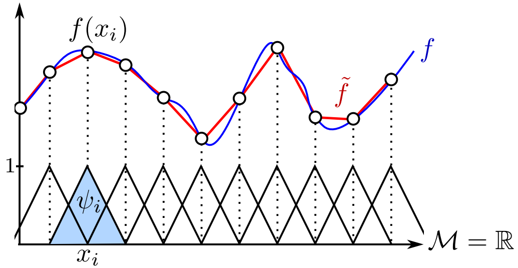
To do so, we approximate f by some other function $\tilde{f}$ using linear combinations of basis functions:

$$f \approx \tilde{f}$$
$$\tilde{f} = \sum_i f(x_i)\psi_i$$

Here, ψ_i are "hat" basis functions and $f(x_i)$ are approx. coefficients

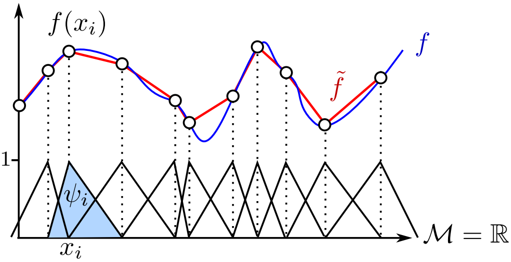


Piecewise-linear approximation



The vector $\mathbf{f} \in \mathbb{R}^n$ contains the approximation coefficients $\mathbf{f}_i = f(x_i)$ wrt the hat basis.

Piecewise-linear approximation

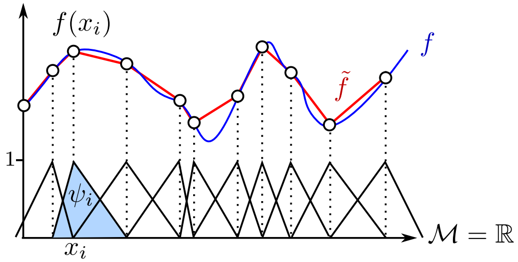


The vector $\mathbf{f} \in \mathbb{R}^n$ contains the approximation coefficients $\mathbf{f}_i = f(x_i)$ wrt the hat basis.

Note that:

- The domain could also be non-uniformly sampled while keeping a piecewise-linear approximation

Piecewise-linear approximation



The vector $\mathbf{f} \in \mathbb{R}^n$ contains the approximation coefficients $\mathbf{f}_i = f(x_i)$ wrt the hat basis.

Note that:

- The domain could also be non-uniformly sampled while keeping a piecewise-linear approximation
- Piecewise-linear is not the only option: other basis functions can be chosen (quadratic, cubic, ...)

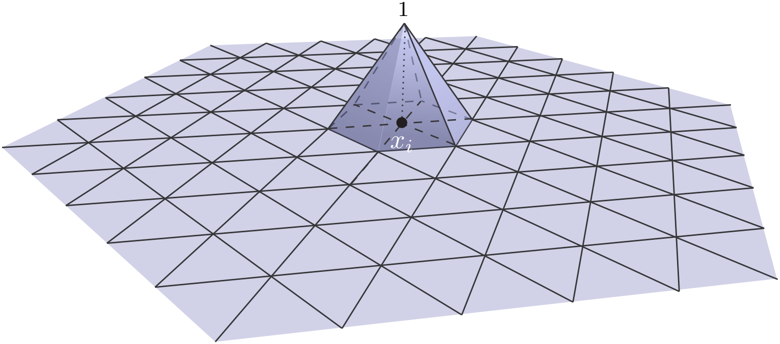
Piecewise-linear approximation on meshes

Triangle meshes discretize an underlying surface domain $\mathcal{M}$

Piecewise-linear approximation on meshes

Triangle meshes discretize an underlying surface domain $\mathcal{M}$

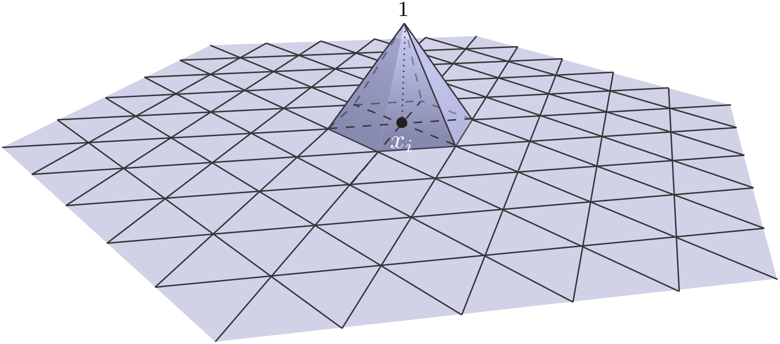
Hat basis functions can be defined just as before:



Piecewise-linear approximation on meshes

Triangle meshes discretize an underlying surface domain $\mathcal{M}$

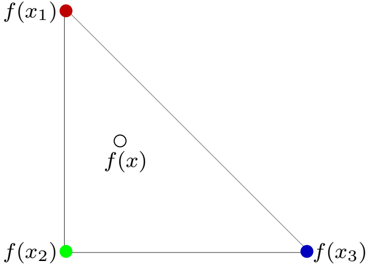
Hat basis functions can be defined just as before:



With this choice, we look at piecewise-linear approximations of functions on our triangle meshes (this will be a key assumption in future lectures!)

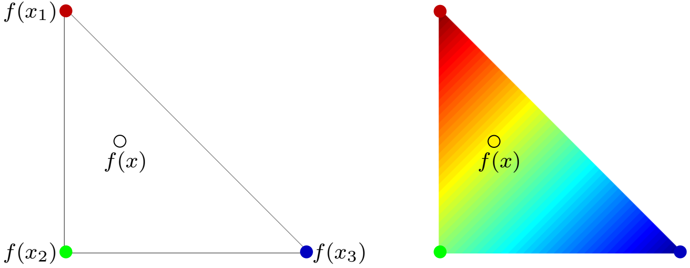
Piecewise-linear approximation on meshes

Piecewise-linear clearly means linear behavior within each triangle:



Piecewise-linear approximation on meshes

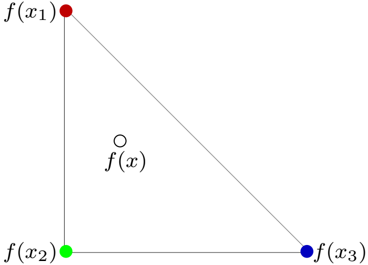
Piecewise-linear clearly means linear behavior within each triangle:



Given function values $f(x_1), f(x_2), f(x_3)$ at the 3 vertices, the function values $f(x)$ inside the triangle are obtained by bilinear interpolation

Piecewise-linear approximation on meshes

Piecewise-linear clearly means linear behavior within each triangle:



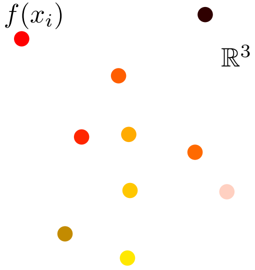
Given function values $f(x_1), f(x_2), f(x_3)$ at the 3 vertices, the function values $f(x)$ inside the triangle are obtained by bilinear interpolation

With Matlab's shading interp command, vector values are therefore interpreted as approximation coefficients wrt the hat basis

Functions on point clouds

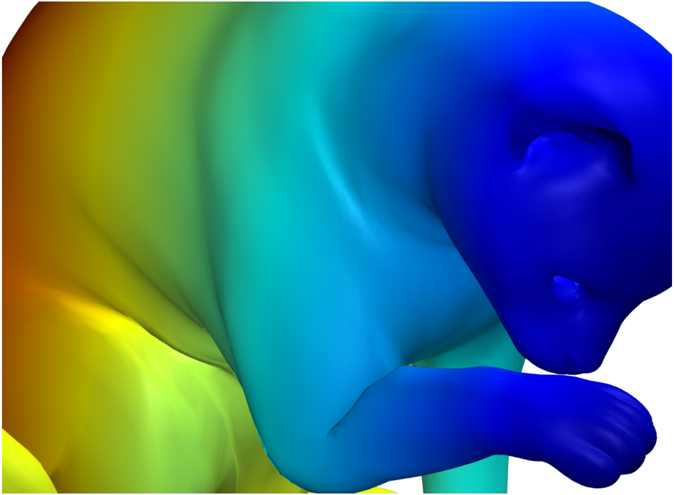
On point clouds P, things are much easier:

The standard way is to interpret $\mathbf{f}_i = f(x_i)$ precisely as the value of the function at $x_i \in P$



Shading in Matlab: interp

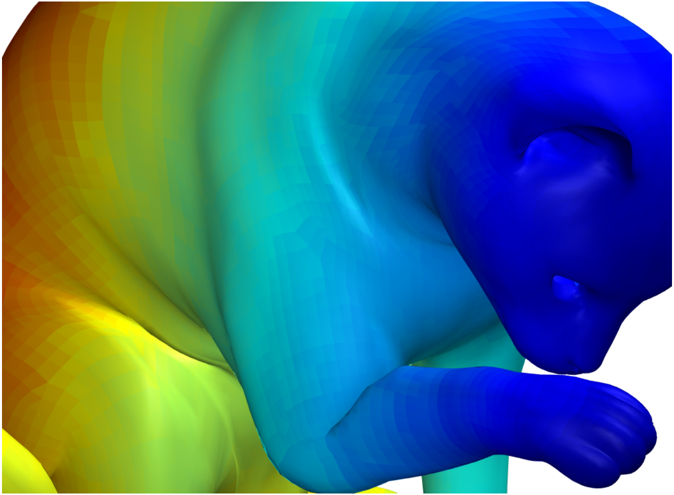
The choice of shading depends on what we want to visualize.



shading interp is good for scalar functions

Shading in Matlab: flat

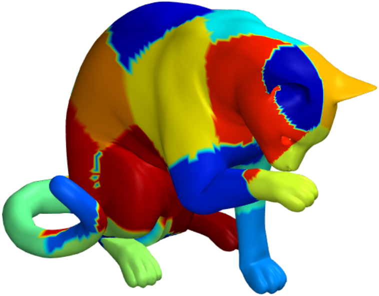
The choice of shading depends on what we want to visualize.



shading flat does not apply any interpolation

Shading in Matlab: flat

The choice of shading depends on what we want to visualize.



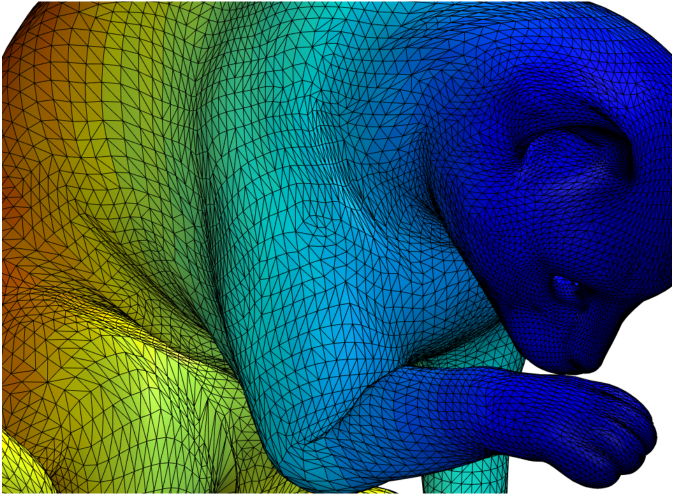
shading interp

shading flat

shading flat does not apply any interpolation (good for segmentations)

Shading in Matlab: faceted

The choice of shading depends on what we want to visualize.



shading faceted is flat with visible mesh edges

Lighting and materials

Shape appearance is affected by lights and how the surface reacts to them

no light

light

Lighting and materials

Shape appearance is affected by lights and how the surface reacts to them

no light

light

lighting gouraud

- lighting gouraud interpolates linearly across the triangles

Lighting and materials

Shape appearance is affected by lights and how the surface reacts to them

no light · light x2 · lighting gouraud

- lighting gouraud interpolates linearly across the triangles
- lights add up (two lights are brighter than one)

Lighting and materials

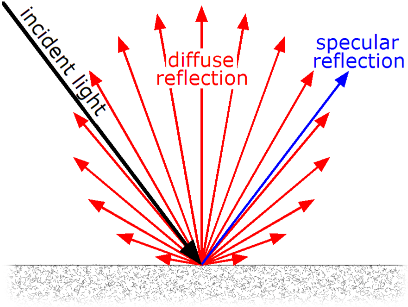
Several surface properties interact with lights and affect appearance

- Ambient light intensity: a nondirectional light that illuminates the scene

Lighting and materials

Several surface properties interact with lights and affect appearance

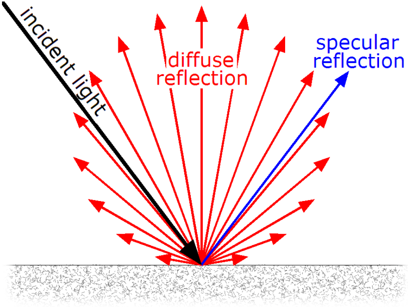
- Ambient light intensity: a nondirectional light that illuminates the scene
- Diffuse reflection: the nonspecular reflectance (think of non-shiny wood or chalk)



Lighting and materials

Several surface properties interact with lights and affect appearance

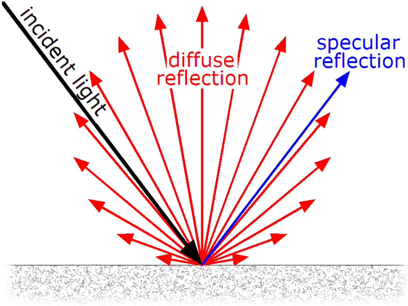
- Ambient light intensity: a nondirectional light that illuminates the scene
- Diffuse reflection: the nonspecular reflectance (think of non-shiny wood or chalk)
- Specular reflection: the bright spots on the surface



Lighting and materials

Several surface properties interact with lights and affect appearance

- Ambient light intensity: a nondirectional light that illuminates the scene
- Diffuse reflection: the nonspecular reflectance (think of non-shiny wood or chalk)
- Specular reflection: the bright spots on the surface



A specific set of these property values make up a material

Ambient light

Ambient light mainly affects shadows:

In Matlab: Change trisurf's 'AmbientStrength' property from 0 to 1

Diffuse reflection

Diffuse reflection mainly affects color brilliance:

In Matlab: Change trisurf's 'DiffuseStrength' property from 0 to 1

Specular reflection

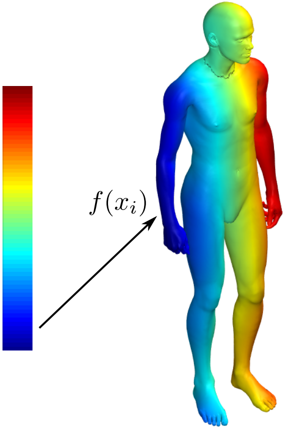
Specular reflection directly affects specularities:

In Matlab: Change trisurf's ‘SpecularStrength’ property from 0 to 1

Colormaps

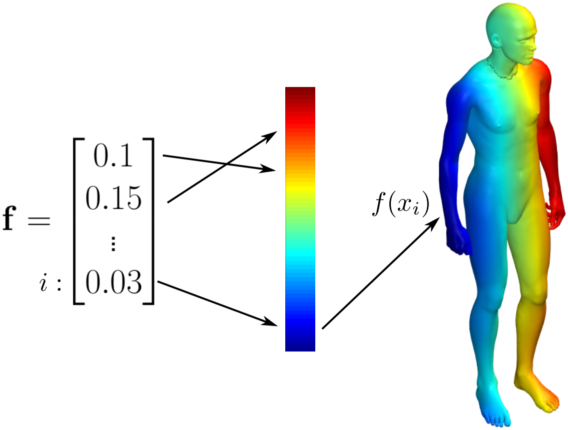
Colormaps are used to determine which color corresponds to which value:

$$\mathbf{f} = \begin{array}{r} \\ \\ \\ i: \end{array}\begin{bmatrix} 0.1 \\ 0.15 \\ \vdots \\ 0.03 \end{bmatrix}$$



Colormaps

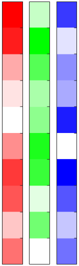
Colormaps are used to determine which color corresponds to which value:



Scale does not matter: $\alpha\mathbf{f}$ will look the same as $\mathbf{f}$ for any $\alpha \neq 0$

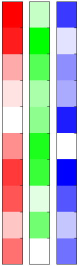
Colormaps as range quantization

Colormaps are usually represented as $n \times 3$ matrices with values in $[0, 1]$:



Colormaps as range quantization

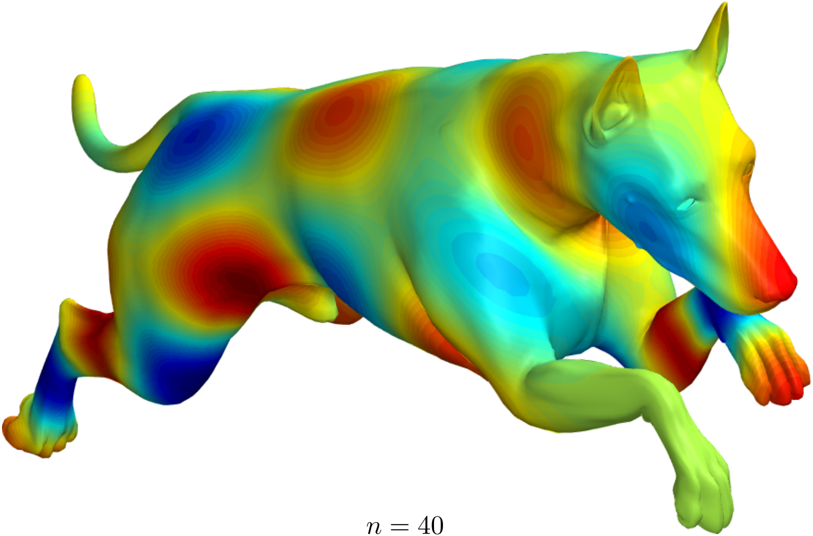
Colormaps are usually represented as $n \times 3$ matrices with values in $[0, 1]$:



The number n of colors in a colormap doesn't have anything to do with the size of $\mathbf{f}$

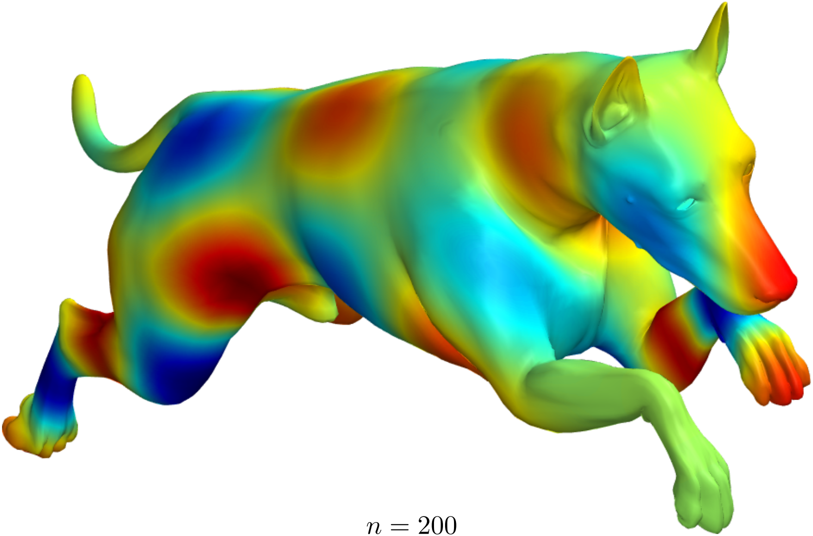
Rather, the colormap represents a quantization of the range of $f : \mathcal{M} \to \mathbb{R}$ into a discrete number (n) of "color bins"

Example: Quantization



$n = 40$

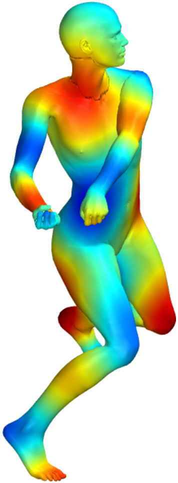
Example: Quantization



$n = 200$

Standard colormaps

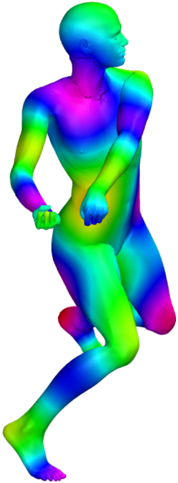
- jet



Standard colormaps

- jet

- hsv



Standard colormaps

- jet

- hsv

- spring

Standard colormaps

- jet

- hsv

- spring

- hot

Standard colormaps

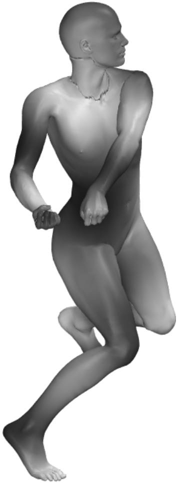
- jet

- hsv

- spring

- hot

- gray



Standard colormaps

- jet

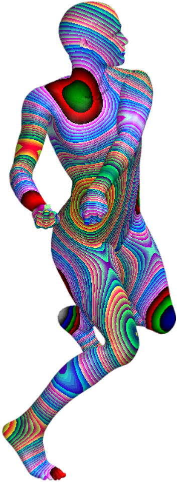
- hsv

- spring

- hot

- gray

- colorcube



Standard colormaps

- jet

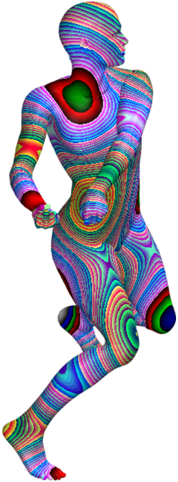
- hsv

- spring

- hot

- gray

- colorcube



Custom colormaps may also be defined as needed

- By modifying an existing colormap (e.g. reverse color order)
- By creating a new one from scratch

Example: whitered

Create a colormap that linearly grows from white to red in n steps:

$n = 10$

Example: whitered

Create a colormap that linearly grows from white to red in n steps:

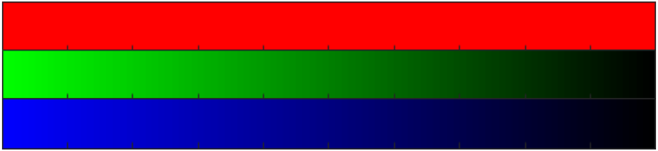
$n = 100$

Example: whitered

Create a colormap that linearly grows from white to red in n steps:

$n = 100$

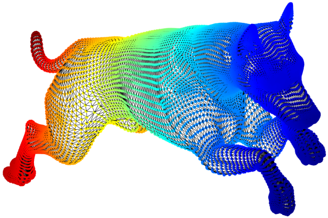
Simply fix the red channel to constant value 1, and linearly decrease the blue and green channels from 1 to 0 in n discrete steps:



Example: Colored point cloud over a mesh

How did I produce this figure (see slide #16)?

A white mesh with visible edges, and a colored point cloud on top

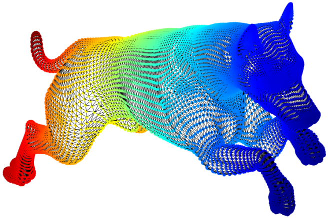


Example: Colored point cloud over a mesh

How did I produce this figure (see slide #16)?

A white mesh with visible edges, and a colored point cloud on top

- It involves two colormaps: white (mesh) and jet (point cloud)

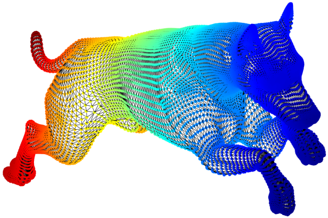


Example: Colored point cloud over a mesh

How did I produce this figure (see slide #16)?

A white mesh with visible edges, and a colored point cloud on top

- It involves two colormaps: white (mesh) and jet (point cloud)
- f is the y coordinates of the mesh

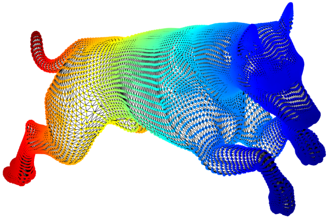


Example: Colored point cloud over a mesh

How did I produce this figure (see slide #16)?

A white mesh with visible edges, and a colored point cloud on top

- It involves two colormaps: white (mesh) and jet (point cloud)
- f is the y coordinates of the mesh
- In order to index the jet colormap, we must bring the range of f to within $\{1, \ldots, 256\}$

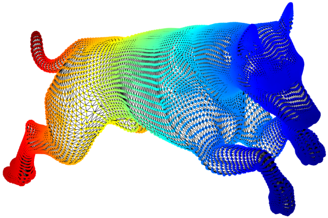


Example: Colored point cloud over a mesh

How did I produce this figure (see slide #16)?

A white mesh with visible edges, and a colored point cloud on top

- It involves two colormaps: white (mesh) and jet (point cloud)
- f is the y coordinates of the mesh
- In order to index the jet colormap, we must bring the range of f to within $\{1, \ldots, 256\}$
- This is done by the transformation:



$$\hat{f}(x) = \frac{f(x) - \min(f)}{\max(f) - \min(f)}$$

$$\hat{f}(x) = \text{round}(1 + 255f(x))$$

Exercise: Rendering of farthest point sampling

For the shape cat_partial .off (download from course website):

- Compute a Euclidean farthest point sampling of 50 points
- Render the mesh with flat white color in Matlab (or your favorite environment), with the farthest point samples rendered as blue points on top, and the boundary of the mesh visualized in red

Use materials and lights as you like.

Send your renderings in .png format to rodola@di.uniroma1.it, using [FundCG] as the email subject.

Exercise: Rendering of 1d Euclidean embedding

For the human shape `tr_reg_000.off`, compute its minimum distortion Euclidean embedding into $\mathbb{R}^1$ using the gradient descent algorithm with a quadratic stress.

Interpret the resulting embedding as a scalar function $f : \mathcal{M} \to \mathbb{R}$.

- Create a new colormap 'bluewhitered' growing linearly from blue to white to red
- Render f as a colored mesh in Matlab (or your favorite environment) in the bluewhitered colormap

Use materials and lights as you like.

Send your renderings in .png format to rodola@di.uniroma1.it, using [FundCG] as the email subject.