Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

MATHEMATICAL FOUNDATIONS OF COMPUTER GRAPHICS AND VISION

EXERCISE 1 - MLS FOR CURVES, MESHES AND IMAGES

Handout date: 21.03.2016
Submission deadline: 10.04.2016, 23:59
Demo date: 11.04.2016, at exercise session

General Rules

Plagiarism note. Copying code (either from other students or from external sources) is strictly prohibited! We will be using automatic anti-plagiarism tools, and any violation of this rule will lead to expulsion from the class.

Late submissions will not be accepted, except in case of serious illness or emergency. In that case please notify the assistants and provide a relevant medical certificate.

Software. All exercises of this course use the MATLAB programming language. The MATLAB distribution is available from IDES for ETH students. See the exercise session slides for hints or specific functions that could be useful for your implementation.

What to hand in. Email a .zip file of your solution to rroveri@inf.ethz.ch . The file must be called "MATHFOUND16-*-firstname-familyname.zip" (replace * with the assignment number) and the subject of the email must be "MATHFOUND16". The .zip file MUST contain a single folder called "MATHFOUND16-*-firstname-familyname" with the following data inside:

- A folder named "code" containing your code
- A README file (in pdf format) containing a description of what you've implemented and instructions for running it, as well as explanations/comments on your results.
- Screenshots of all your results with associated descriptions in the README file.

Grading. This homework is 11.7% of your final grade. Your submission will be graded according to the quality of the images produced by your program, the conformance of your program to the expected behaviour of the assignment, and your understanding of the underlying techniques used in the assignment. The submitted code must produce exactly the same images included in your submission.

To ensure fairness of your grade, you will be asked to briefly present your work to the teaching assistants. Each student will have 3-4 minutes to demo their submission and explain in some detail what has been implemented, report potential problems and how they tried to go about solving them, and point the assistants to the code locations where the various key points of the assignments have been implemented. See above for the scheduled demo date for this particular assignment.

Goal of this exercise

In this exercise you will apply what you learned about the Moving Least Squares (MLS) method to curves, meshes and images. You will read and implement parts of recent research papers that used MLS.

1. EXERCISE PART 1: Curve and Surface Reconstruction Using MLS

As described in the lectures, moving least squares (MLS) can be used to approximate a function given samples from that function at sample points. In this exercise, we will utilize this method to reconstruct curves (1.1 and 1.2) and smoothe meshes (1.3). We provide a few example datasets.

1.1. **Curves derivation.** MLS based surfaces can be computed by considering different degrees for the local approximations and constraints, as described in the lectures. The first part of the exercise is to derive on paper one of the simplest definitions for MLS based surfaces. For this task, we assume that the locally fit polynomial consists of a single constant term, and we use the local functional approximation, described in slide 32 of the lecture "Representation, Sampling, and Reconstruction". Please derive the closed-form expression for the resulting implicit function $f(\mathbf{x})$.

Hint: it is a weigthed least squares problem for the scalar c_0, and hence reduces to a simple normalized average.

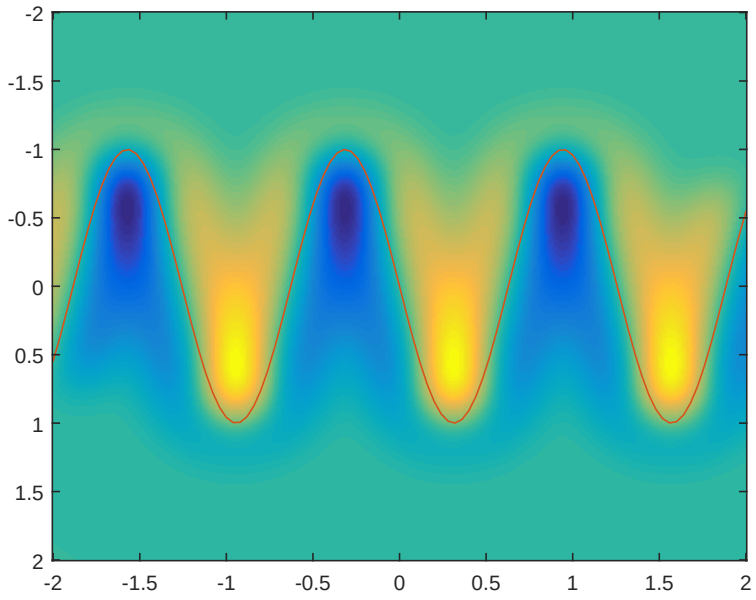

Figure 1. Possible result for the first dataset of exercise 1, task 2.

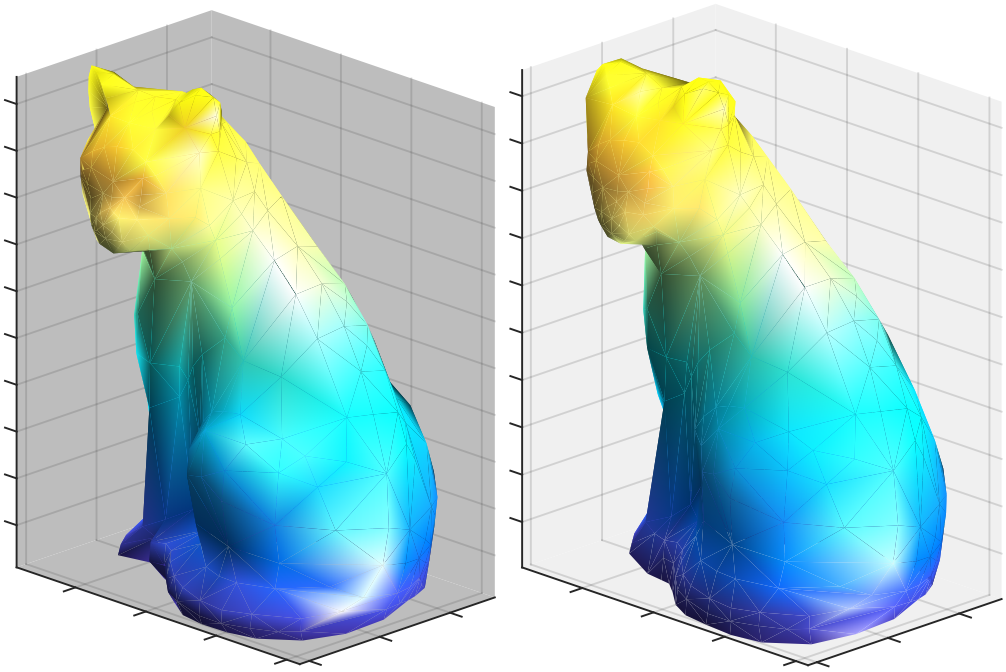

FIGURE 2. A possible result for task 3 of exercise 1. Original mesh on the left, smoothed version on the right.

1.2. **Curves plotting.** Evaluate the derived $f(\mathbf{x})$ for the given example point sets with 2D points and normals, on a regular grid. For the definition of ϕ, you may use a Gaussian of a chosen σ such that $\phi(r) = e^{-r^2/\sigma^2}$. Initialize a matrix representing the grid. Then, for each grid center, evaluate the function f and store the value at the entry of the matrix corresponding to that grid. Finally, plot this matrix as a heatmap or similar, overlain with the sample points in the same image (as shown in Fig. 1.

Generate a few images with different σ values to illustrate how it affects the approximation. You can see the reconstructed curve by observing the values of f close to zero.

A few hints about the implementation:

- You can use the "imagesc" function of Matlab to visualize the computed matrix.
- If the plotted $f(\mathbf{x})$ looks too smooth, try decreasing σ. Similarly, if you get discontinous $f(\mathbf{x})$, try increasing σ.

1.3. **Smoothing meshes.** Load the vertices and normals of the provided meshes. These will serve as the sample points and normals. Then, project each vertex coordinate $\mathbf{v}$ of the mesh by $\mathbf{v}' = \mathbf{v} - \nabla f(\mathbf{x})$, where $f(\mathbf{x})$ is the same as in the first task. This will give you new smoothed coordinates $\mathbf{v}'$ of the mesh vertices. Apply this smoothing a few times, obtaining a result similar to Fig. 2. Write the final vertex positions into a .off file. Visualize the mesh using the application "Meshlab" (`http://meshlab.sourceforge.net/`).

A few hints about the implementation:

- Matlab code to load and write .off meshes is provided, as well as two simple meshes you can use to test your algorithm.
- The expression for $\nabla f(\mathbf{x})$ can be easily derived on paper. If you wish, you may confirm your expression by comparing it to the expression in Section 4.3 of the paper "Feature Preserving Point Set Surfaces based on Non-Linear Kernel Regression".
- The function $f(\mathbf{x})$ is always computed by using the initial positions and normals of the vertices, even if you do multiple iterations of projections.
- As in the previous exercise, try changing σ to see its smoothing effect.

1.4. **Speeding up mesh smoothing (optional bonus exercise).** Incorporate a spatial data structure such as a grid, octree, or kd-tree, to speed up the computations and thus handle bigger datasets. We recommend the Approximate Nearest Neighbor Library (`http://www.cs.umd.edu/~mount/ANN/`), which also has a Matlab wrapper (`http://www.mathworks.com/matlabcentral/fx_files/15521/1/content/ann_mwrapper/Readme.htm`). But you may also use a simple grid.

Hint: For a given query point $\mathbf{x}$, you need to consider only the points 3σ away from this point in the sum for computing $f(\mathbf{x})$.

1.5. **Advanced methods (optional bonus exercise).** Extend the basic definition by incorporating the sharp feature preserving and robust definition of the paper "Feature preserving point set surfaces based on non-linear kernel regression". You may apply this definition for curve reconstruction or mesh smoothing as in the previous exercises. Hint: The paper utilizes an expression very similar to the one you derived in the first task, and provides pseudo-code!

Required output of this section:

- Derivation of the implicit function for task 1.1 in your .pdf report.
- Code that reads in provided 2D data points and generates a heatmap visualizing the implicit function overlain with the input data for task 1.2.
- Derivation of the gradient expression in your .pdf report, screenshots of the results and smoothed meshes in .off format for task 1.3.

2. EXERCISE PART 2: Image Deformation Using Moving Least Squares

The second task of this assignment is to write a matlab program to deform images based on moving least squares as presented in the paper "Image deformation using moving least squares" by Schaefer et al. [06] (provided with the exercise). In this paper, the authors propose a method to smoothly deform images using different kinds of transformations and allow the user to manipulate sets of points and segments to specify the deformations. For this exercise you will only focus on image deformations using sets of points.

Write a matlab program which takes as input an image and performs the deformation after the user selects some input and output control points. The algorithm should follow the description from the paper. The three kinds of transformation, i.e. affine, similarity and rigid transformations should

be supported by your program. Please note that we dont ask for an interactive application so that you dont need to worry about making the code fast (precomputation of some terms, approximation of the image by a grid), but you can do it if you wish.

Some pointers to help with your implementation:

- Read section 2 of the paper to understand the types of transformations you need to apply.
- Implement backwards warping to avoid artifacts.
- You can use the "meshgrid" and "griddata" functions to implement the warping process.
- See 3 for an example of what your results should look like.

Required output of this section:

- Code that deforms images according to user input using all three methods described above (affine, similarity and rigid).
- Representative image results of the gingerbread man (provided with the exercise) and at least one more image of your choice.

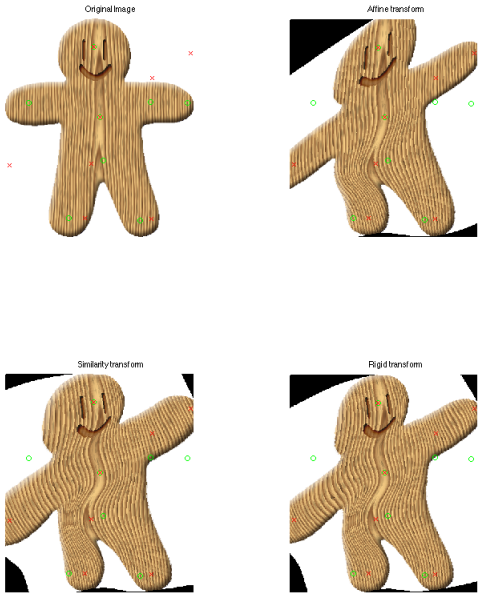

Figure 3. A possible result for exercise part 2.