EEE3032 – Computer Vision and Pattern Recognition

Lab sheet Week 8 -Contours–

Miroslaw Bober (m.bober@surrey.ac.uk)

Task and Learning Objective: In today's lab you will experiment with contours. Specifically you will experiment with different families of curve, and also with active contours (snakes) and the effect that varying the weights on the various energy terms has on the optimization of the contour to fit to image edges.

Resources: The exercises will be performed using Matlab. Download the lab code if you have not already and unzip the code into a folder. Set the Matlab path (File -> Set Path..-> Add with subfolders) to include the code.

Ex1: Hermite curve *(5 minutes)*

Run the program **hermite**. This code generates a piecewise (2 component) cubic spline using two Hermite curves. Experiment by changing the start and end points/tangents of the two cubic curves, and generate alternative piecewise cubic splines that are C1 continuous.

Ensure you understand the role of **G M** and **Q** in the **P=GMQ** framework

Ex2: Bezier curve *(10 minutes)*

Run the program **bezier**. The code prompts you to click 4 points, and draws a single cubic Bézier curve to approximate those points.

Modify the code to also draw the control polygon of the Bézier curve (*hint:* type **help line** in Matlab).

Under what circumstances does the Bézier curve become an *interpolating* rather than *approximating* spline?

*(Recall, an interpolating spline passes through **all** its control points, and approximating spline does not)*

Ex3: Catmull Rom and B-Spline *(10 minutes)*

Run the program **splinedemo** shown in lectures. The code prompts you to click several (more than 4) points to create a piecewise curve modelled using either Catmull-Rom splines or β splines (B-splines).

Experiment with the different behaviours of these curve families by commenting / uncommenting the lines defining the blending matrix. Ensure you understand how the matrix of all control points **Gmaster** is set up – and how during iteration, points are selected from **Gmaster** to populate **G** for the **P=GMQ(s)** framework.

Ex4: Snake on Synthetic Data *(15 minutes)*

In this exercise you will run the program **snakedemo** shown in lectures. The code is dependent on two functions written in C that are called from Matlab. These functions need to be compiled before **snakedemo** will work. The Matlab C compiler is called "mex". At the Matlab prompt within the "snake" subfolder of the lab code type:

```
>> mex snkfit.c
>> mex scancr.c
```

This will create Matlab functions **snkfit** and **scancr** respectively. The function **scancr** will quickly generate a binary mask (i.e. black/white image) of a Catmull Rom spline, given a set of control points and an appropriate image size. This is used in the demo for visualization. The function **snkfit** takes an initial set of control points for a snake, and optimises them to fit an image.

Please note if you haven't run "mex" before you may be asked to choose a compiler. We recommend you use the Microsoft Visual Studio compiler suggested e.g. 2008, on Windows, or for Linux a version of gcc or g++.

When you have compiled the code via **mex** (see above), run the **snakedemo** program.

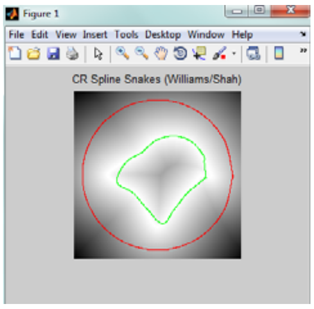
Figure 1
File Edit View Insert Tools Desktop Window Help
CR Spline Snakes (Williams/Shah)

Using a text editor like Windows Notepad, open the **snkfit.c** source code and modify constants ALPHA, BETA and GAMMA. For example try setting each constant in turn to 0.0001.

```
#define ALPHA          (1.0)          /* Alpha weight (1st deriv, internal) */
#define BETA           (1.1)          /* Beta weight (2nd deriv, internal)  */
#define GAMMA          (1.2)          /* Gamma weight (Image data, external)*/
```

Don't forget to re-compile the code using after you have saved your changes to the source code.

You will need to reset Matlab by typing "close all" then "clear all" otherwise you may not be able to successfully recompile (the compiled file is kept "open" by Matlab and so can't be overwritten by mex).

Ensure you are able to explain the effect of these constants in the context of the snake energy equation:

$$E = \int_{s=0}^{s=1} \alpha \frac{dP(s)}{ds} + \beta \frac{dP^2(s)}{ds^2} + \gamma \, \mathrm{I}(P(s))$$

Ex5: Snake on Real Data *(10 minutes)*

Run the program **snakedemo_hand**. This code will attempt to fit a snake to the shape of a hand, extracted from the hand.jpg image bundled with the lab code. Experiment with the weights ALPHA, BETA and GAMMA as in Exercise 4. Try to find a combination of weights that fits the snake to the hand closely (you will find it difficult to fit the snake exactly). Try to use lower weights on BETA and simultaneously, higher weights on ALPHA for best results.

You may also wish to adjust the number of contour control points using the SNAKECP constant in the Matlab code (not the C code). Do not use more than 150 points or the code will take a long time to optimize the snake.