Moving Least Squares

Image reconstruction

Your goal is to fill the missing parts of (or simply smooth) an input image using the moving least squares technique. To that end, you will use C++, Qt (for the interface) and Eigen (for linear algebra). If Eigen is not installed, simply download it (http ://eigen.tuxfamily.org) and uncompress it where you want. The provided sources can be compiled using the following command lines :

```
qmake && make
```

You may have to edit imageMLS.pro to specify the path to eigen headers. Run the program with

```
./imageMLS [filename] [noise value - in 0,1] [sigma - in 1,inf]
```

By default, the program simply loads an image, adds some noise (create holes) and replaces the missing part by a red color. Check the different files. You will mainly have to modify "mlsRec.cpp". The "apply" function contains examples of how to manipulate images and how to use matrices and vectors with Eigen. You will have to modify this function to obtain your desired result.

Plane fitting

The function to minimize is the following :

$$E(a,b,c) = \sum_{i=1}^{n} w_i(||\mathbf{x} - \mathbf{x_i}||)(ax_i + by_i + c - z_i)^2$$

The algorithm thus consists in iterating over all pixels of the input image and apply a weighted least square on each of them, using a gaussian weight.

$$\begin{pmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{pmatrix} \Longleftrightarrow Ax = b$$

By computing the partial derivatives of E, and taking weights into account, we get $(A^TWA)x = A^TWb$ and then $x = (A^TWA)^{-1}A^TWb$. W is a diagonal matrix containing the weights for each sample :

$$w(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp -\frac{x^2}{2\sigma^2}$$

Once the coeficients found, the polynomial has to be evaluated at the corresponding point to obtain the result.

The iteration will be done in the "apply" function and the local minimization will be done in "estimateColorPlane" function. Note that, in case of RGB images, the weighted least square will have to be applied 3 times per pixel (one per channel R, G and B).

Try with different values of sigma and noise ratio.

Quadratic surface fitting

Edit the “estimateColorQuadric” function to apply the same technique and estimate a quadric instead of a plane :

$$E(a,b,c,d,e,f) = \sum_{i=1}^{n} w_i(||\mathbf{x} - \mathbf{x_i}||)(ax_i^2 + by_i^2 + cx_iy_i + dx_i + ey_i + f - z_i)^2$$

Note : if your plane fitting is correct, this should take no more than 2 min.

Try with different values of sigma and noise ratio. Compare with the plane.

Estimation of σ

In case we want to fill some missing data, it is difficult to define a good value for σ because it might provide a good neighborhood in some pixels and not enough data at other positions. Complete the function “estimateSigma” to better approximate the neighborhood in which the weight will be applied. Use it according to the pixel positions in the image.

Adjust your estimation. Try interpolating (keeping original valid pixels) vs approximating (least square on all pixels without distinction). Also try different kind of missing data (see main.cpp) and with different input images/height fields.