G53VIS Tutorials 4 – Matching functions

Xin Chen & Andy French, 2018-19

1 Matching functions

This section aims to give an overview of the matching functions used for the lab work and coursework. Assume that we are given one template T_e and one target image T_a. Both T_e and T_a have size $N_1 \times N_2$ pixels. Firstly, we convert these images into $N_1N_2 \times 1$ vectors. In Matlab this is done as follows: x1 = T_e (:) and x2 = T_a (:). Then we can match x_1 and x_2 in the following ways:

- Euclidean Distance (ED): Calculate first the difference vector e = x_1- x_2. Then ED = sqrt (sum(e.^2)). The smaller the distance the more similar x_1 and x_2 are.

- Correlation: C = x_1^T x_2. The higher the correlation the more similar x_1 and x_2 are.

- Normalised Correlation: First normalise both x_1 and x_2 as follows. For example, for x_1:
  Step 1: Calculate the mean value m_1 =mean(x_1) and then x_1= x_1- m_1
  Step 2: Calculate the norm n_1 = sqrt(sum(x_1.^2)) and then x_1= x_1/ n_1 . Once we do steps 1-2 for both x_1 and x_2, then use correlation C = x_1^T x_2. The higher the normalised correlation the more similar x_1 and x_2 are. Normalised correlation cannot be larger than 1.

2 Suggestions for Coursework

This document summarizes suggestions for the Coursework as discussed during the lecture. We assume that rectified images are given. Hence, in this case, the problem of Stereo reduces to finding pixel disparities between the corresponding scanlines of the two images. The disparity d is defined as the amount of horizontal shift (vertical shift is zero) between a point p in the left scanline and its corresponding point p' in the right image. Hence, given p, to find d we just need to find p'. To this end, so far we have used the Euclidean Distance (ED) or the Normalised Correlation (NC) between pixel intensities as the matching function (see also the tracking task for the lab work). Extensions to this include:

- Gradient-based features: rather than using pixel intensities, you can extract the horizontal and vertical gradients g_x and g_y for each pixel location and use these as features to do the matching. Let us denote by W the N×N window centred at pixel p. We can apply *imgradient* Matlab function on W to extract the N×N gradient images G_x and G_y. Denote f1 = Gx(:) and f2 = Gx(:) the $N^2 \times 1$ vectors obtained by vectorizing G_x and G_y. Define x = [f1 ; f2] the $2N^2 \times 1$ feature vector to describe window W. Now use x to calculate ED or NC.

- HoG-based features: rather than using pixel intensities, you can also use SIFT, HOG (or any other feature descriptors) as a basis to perform correlation. For example use *extractHOGFeatures* Matlab function to extract HOG features from W and use the resulting $N_f \times 1$ vector, where N_f is the HOG feature length, to calculate ED or NC.

- Dense-SIFT features: the previous approach extracts SIFT features from the N×N window centred at pixel p. What you can also do is to extract a SIFT descriptor for each pixel within W, concatenate all resulting feature descriptors in a single vector (the length of this will be $N^2N_f \times 1$), and use this to calculate ED or NC.

- Multi-scale approach: All approaches so far have assumed that the window W has a fixed size of N×N. Firstly, you can experiment with different window sizes. Secondly, you can define new multi-scale descriptors by simply concatenating the features obtained using each window size.

- Effect of noise: So far we have assumed that the images are noise-free. You can also test the performance of difference approaches when image noise is added. To add noise to an image you can use Matlab function *imnoise*.

- Use scanline methods based on Dynamic Programming.

- Use 2D optimization methods based on graph-based optimisation.

- Any other extension based on research that you have done on the Internet.