Computer Vision 1 - AI Assignment4
Image Alignment and Stitching

March 2, 2016

All the files should be zipped and sent to **computervision1.uva(at)gmail.com** before **09-03-2016** , **23.59** (Amsterdam Time).

1 Image alignment

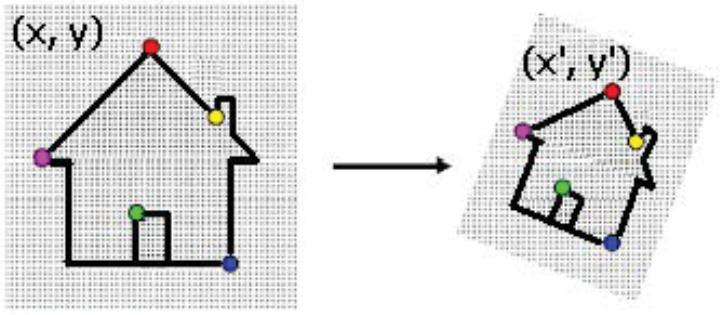
In this practice you will write a function that takes two images as input and computes the affine transformation between them. You will work with supplied boat images. The overall scheme is as follows:

1. Detect interest points in each image.
2. Characterize the local appearance of the regions around interest points.
3. Get the set of supposed matches between region descriptors in each image.
4. Perform RANSAC to discover the best transformation between images.

The first three steps can be performed using David Lowes SIFT. You can use the functions `[frames,desc] = sift(im)` and `[matches] = siftmatch(desc1, desc2)` from Andrea Vedaldi sift implementation (to download check http:// www.vlfeat.org/ vedaldi/assets/sift/versions/, you need the latest version 0.9.19).

The final stage, running RANSAC, should be performed as follows:

- Repeat N times:
- Pick P matches at random from the total set of matches T.
  How many matches do we need to solve an affine transformation which can be formulated as shown in Figure 1?
- Construct a matrix A and vector b using the P pairs of points and find transformation parameters $(m1, m2, m3, m4, t1, t2)$ by solving the equation $Ax = b$. Hint: Equation $Ax = b$ can be solved using pseduo-inverse: $x = (A^T A)^{-1} A^T b$, or `x = pinv(A)*b` in Matlab.



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} m_1 & m_2 \\ m_3 & m_4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \tag{1}$$

It can be rewritten as

$$\begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \end{bmatrix} \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{2}$$

or

$$Ax = b,\ A = \begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \end{bmatrix},\ x = \begin{bmatrix} m_1 \\ m_2 \\ m_3 \\ m_4 \\ t_1 \\ t_2 \end{bmatrix},\ b = \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{3}$$

Figure 1: Affine transformation.

- Using the transformation parameters, transform the locations of all T points in image 1. If the transformation is correct, they should lie close to their counterparts in image 2. Plot the two images side by side with a line connecting the original T points in image 1 and transformed T points over image 2.
- Count the number of inliers, inliers being defined as the number of transformed points from image 1 that lie within a radius of 10 pixels of their pair in image 2.
- If this count exceeds the best total so far, save the transformation parameters and the set of inliers.

- End repeat.
- Finally, transform image 1 using this final set of transformation parameters. If you display this image you should find that the pose of the object in the scene should correspond to its pose in image 2. To transform the image, implement your own function based on nearest-neighbor interpolation. Then use the build-in Matlab functions `imtransform` and `maketform` and compare your results.

How many iterations in average are needed to find good transformation parameters?

2 Image Stitching

In this assignment you will write a function that takes two images as input and stitch them together. You will work with supplied images *left.jpg* and *right.jpg* The overall scheme can be summarized as follows:

1. As in previous task you should first find the best transformation between input images.
2. Then you should estimate the size of the stitched image. Hint: calculate the transformed coordinate of corners of the *right.jpg*.
3. Finally, combine the *left.jpg* with the transformed *right.jpg* in one image.

3 Deliverables

1. **[6 pts]** For section 1:
   - [2 *pts*] A demo function should take image pair "boat/img1.pgm" and "boat/img2.pgm" as input, and return the keypoint matchings between the two images. (Hint: you can use vlfeat function vl_ubcmatch to do the matching.)
   - [1 *pts*] Take a random subset (with set size set to 50) of all matching points, and plot on the image. (Note: connect matching pairs with lines.)
   - [3 *pts*] A demo function which perform RANSAC algorithm should return the best transformation found. For visualization, show the transformations from img1 to img2 and from img2 to img1.
2. **[4 pts]** For section 2:
   - [2 *pts*] A demo function which runs the whole routine with all other functions you have implemented.
   - [2 *pts*] Visualizations of the combined image from *left.jpg* to transformed *right.jpg* should be submitted.