Computer Vision
Exercise 2: Interest Points and Image Alignment

Raquel Urtasun

Due date: Friday February 8th at noon

1 Programming exercises

This week we will look into local features and image alignment for the programming exercise

Image Warping: Implement the forward and inverse warping functions defined as

```
image_warped = forward_image_warping(image,transformation);
image_warped = inverse_image_warping(image,transformation);
```

Which warping function is better? which one suffers from holes? Employ the filters you developed last week if necessary. Apply the inverse of the transformation to the resulting image. Do you get the same as the original image? why? If not, what would you suggest to do? when can you retrieve the same?

Interest point detection: Implement a Harris interest point detector. Make use of the code you did last week to compute the gradients

```
points = harris(image,criteria,sigma,threshold);
```

Note that here *sigma* represents the variance of the Gaussian to compute derivatives, and *type* represents the type of criteria (i.e., functions on the singular values) use to define whether it is an interest point. Run it in your favorite image. Play with different criteria. Do you see any notable difference?

Feature description: Implement a descriptor as the normalized (mean 0, variance 1) grayscale patch. Make use of the SIFT descriptor from VLFeat. `http://www.vlfeat.org/overview/sift.html` Take your favorite image and perform several transformations (scale, rotation, translation, affine, etc). Test the robustness of these two descriptors to this transformations. What do you observe?

Matching: Implement a criteria that looks only at the best match. Implement a second criteria that looks at the best 2 matches. Compute precision-recall curves for both (given the transformed images you did before). When does this matter? Show an image where this matters and point to why and where is the difference.

Image Alignment: Add Gaussian noise to your warped image. Perform least squares to compute the transformation and compute the error. Which metric would you use? why? Plot this metric as a function of the Gaussian noise variance. What do you observe? Add outliers to the set of good matches. Is least squares robust? recompute the plots.

Robust Matching and Alignment: Implement RANSAC. Test it in the two cases (i.e., Gaussian and outlier noise). Is it robust? to what?

2 Critic reading exercise

Read the paper by David Lowe on SIFT [1]

3 Problem Sets

General Formulation (slide 20, lecture 6). Show how to obtain the Jacobian for translation, euclidean, similarity and affine transformations.

Uncertainty weighting: Show what is the least-squares solution if we have a different known weight for each match. Provide different criteria you think might be useful to compute this weight. What are the potential problems?

Iterative Least Squares: Show the derivation from the M-estimator to the iterative reweighted least squares. Derive each update and mentioned which of those updates can be done in close form. Is this procedure guaranteed to converge? why? and to the global optimum?

Ransac: Suppose that we have a problem, where 40% of our points are outliers. Compute the number of rounds necessary to have a 99% probability of success for each type of transformation. Plot the number of rounds as a function of the probability of success for each type. What do you observe?

References

[1] D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999.