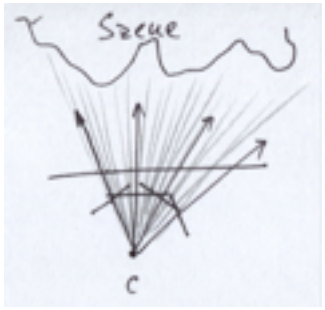
Computer Vision

Exercise 3 – Panorama Stitching

01/12/2015

The task



Pipeline

1. Interest point detector: **Harris**, Scale-space analysis, Edge detection, MSER-s …

2. Image features: Patches (**squared distance**, correlation coefficients), SIFT, SURF, moments, geometric features, affine invariance, projective invariance …

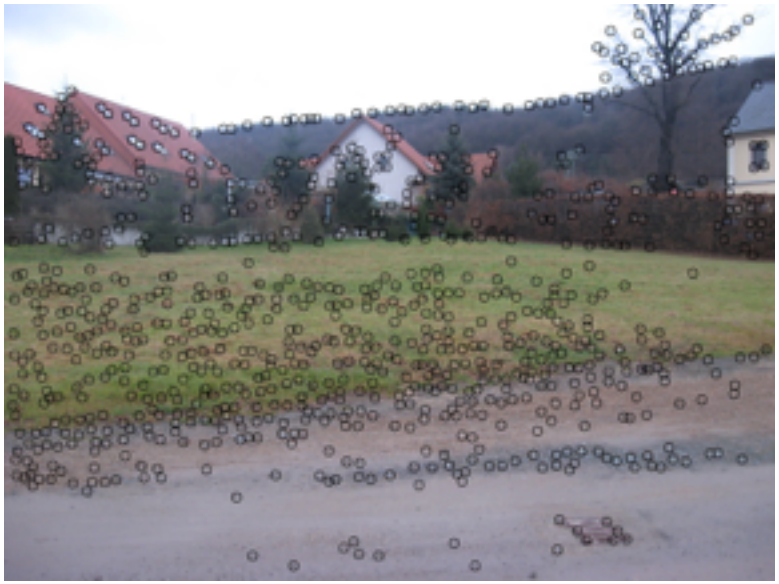
3. Matching: **brute force + cross-validation**, marriage problem …

4. Homography estimation: **RANSAC (OpenCV)**, other RANSAC-variants, robust optimization …

5. Rendering

Harris detector

1. Computation of coefficients
2. Smoothing – **integral image**, separable Gaussian kernel
3. Computation of the “Cornerness”.
4. Non-maxima suppression (by fast max-filter)
5. Some post-processing



Matching

Here:

squared difference of image patches, brute force comparison, cross-validation

Other possibilities:

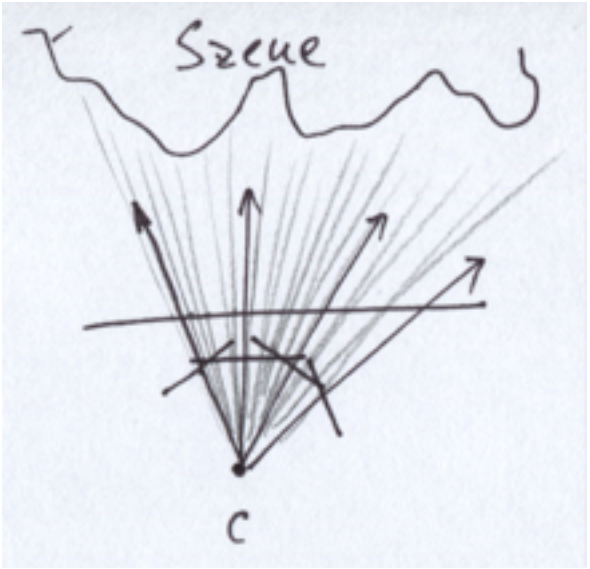
Features – correlation coefficients, SIFT, explicit color correction (iterative)

Matching – graph matching, marriage problem …

Homography estimation

Here:

1. RANSAC (OpenCV implementation)
2. Estimation of **two** homographies (needed for rendering)



Rendering

Standard (OpenCV) – left to right and vice versa

Symmetric:

Problems – homogenous areas

Interest points are found on textured regions only

Problems – small overlap

Not enough matches – very coarse alignment

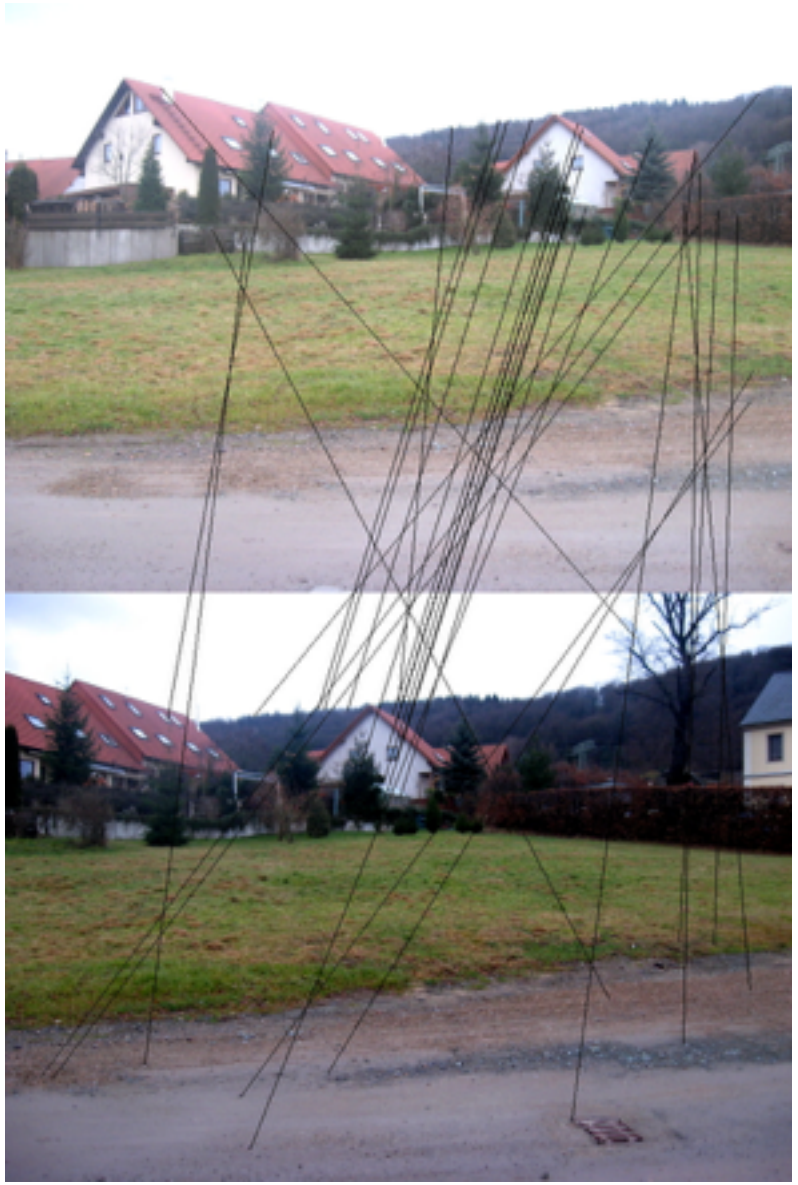
Problem – different colors

Original:

Interest points – Ok.

Problem – different colors

Matches –
obviously too few



Problem – different colors

Panorama – the homographies are wrong

Assignments

- You have the code and the images, play with it – change the parameters, introduce image distortions, identify problems …
- You are allowed to use the code, you can replace its parts (or even completely) by OpenCV functions or by your own implementations
- There should be at least one step (function) in the whole pipeline implemented by yourself !!! Note: own solutions for the 2nd exercise (e.g. Harris detector, filtering etc.) will be not considered as solutions for this one – something new should be done for this exercise as well
- Address problems considered above e.g. colors, homogenous regions etc. – think what steps could be improved (and how) in order to reach better performance
- Address the pipeline steps – think how to do them generally better

Assignments

All the stuff (these slides + the code) is available under

http://cvlab-dresden.de/courses/computer-vision-1/

Deadline – 04.01.2016 (?) per e-mail an Dmytro.Shlezinger@...

Add-ons (extra points):

- Stereo – estimate the fundamental matrix instead of the homography
- Many images instead of only two (in a row)
- Many images aligned in a grid
- Many not aligned images
- …

Some suggestions, hints, other possible works …

Let some “approximate” homography be known. What it can be useful for?

– You can adjust image coloring in order to achieve better dissimilarity measure for patches (since you know the “true” matches)

– You can (drastically) reduce the search space for estimation of the matches from interest points, i.e. do not compare all to all. If so, you can use much more interest points without to essentially increase the time complexity

Some suggestions, hints, other possible works …

How to make use of this ?

Apply an iterative procedure:

1. "Standard" pipeline (i.e. with the original images and less interest points) → initial homography estimation H^0
2. Use H^0 to correct colorings, reduce search space …
3. Perform "standard" pipeline (taking into account the current homography H^t) to improve $H^t \rightarrow H^{t+1}$
4. Iterate 2 – 3 until convergence

Some suggestions, hints, other possible works …

Another way – use resolution pyramid

1. Scale down the images
2. Estimate the homography for downscaled images
3. Use this homography to estimate the one for the originals
4. Use this estimation to adjust colorings (speed up the matching etc.) for the original images

Some other possible works:

– Think about an iterative approach using the “symmetric” rendering

– Do stereo-matching or multi-view 3D-reconstruction

– …