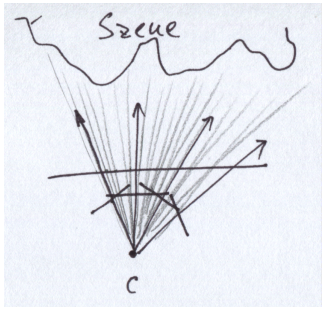
Computer Vision

Exercise 3 – Panorama Stitching

06-07/12/2016

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 …

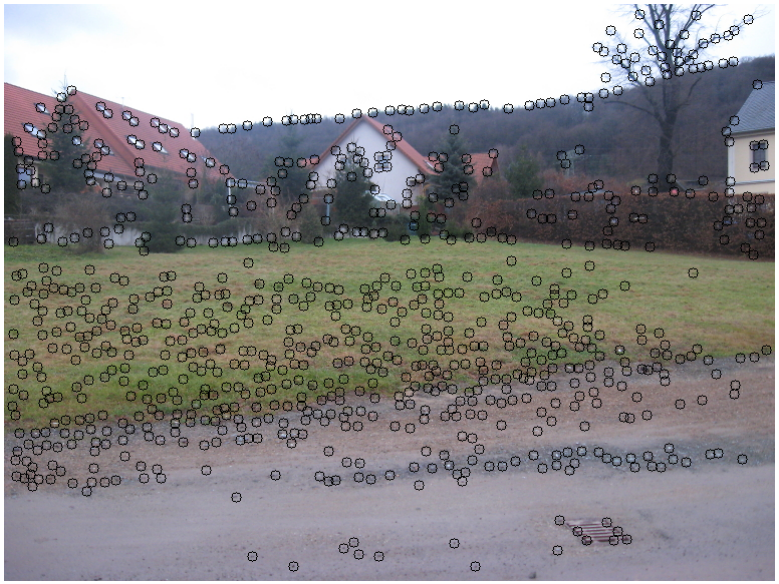
3. Matching: **brute force + cross-validation**, marriage problem, graph matching …

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

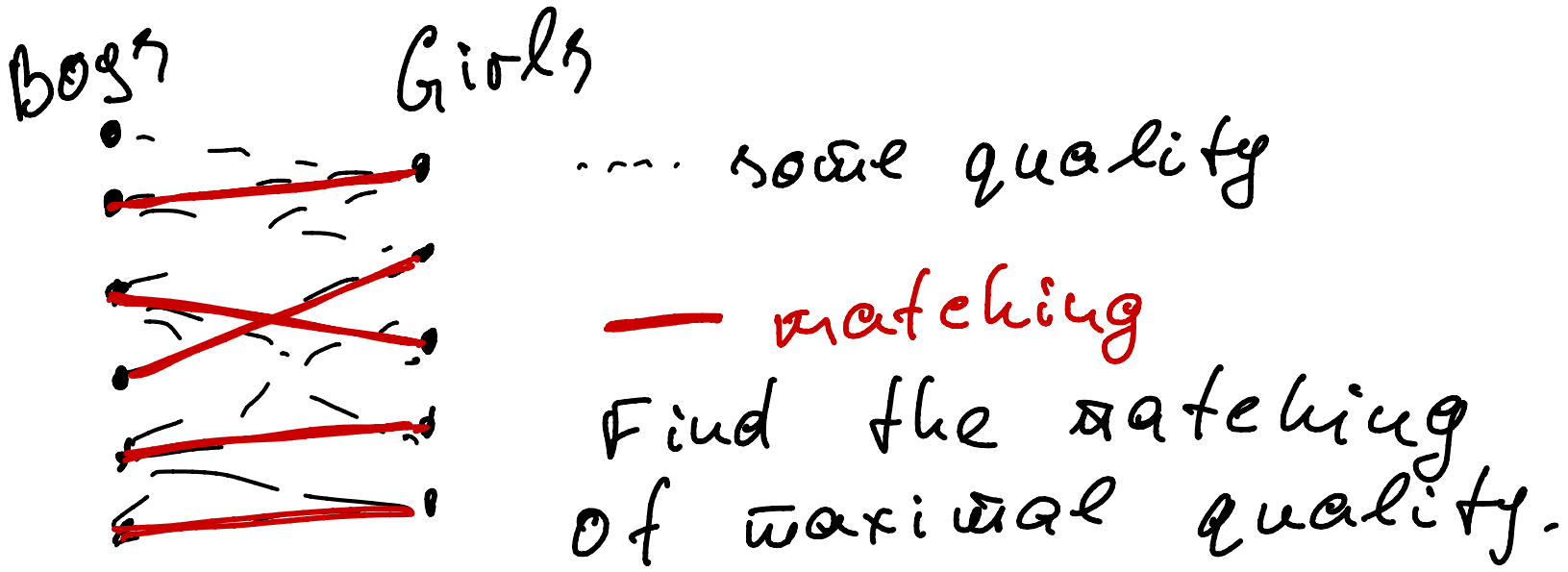
Here:

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 …



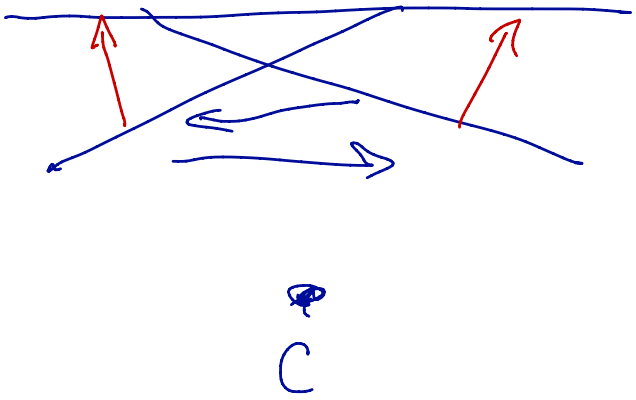
Min Cost Max Flow Problem.

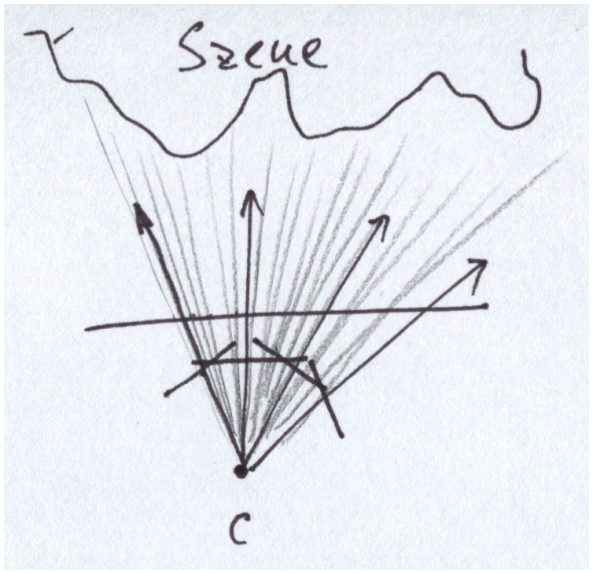
Marriage Problem.

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

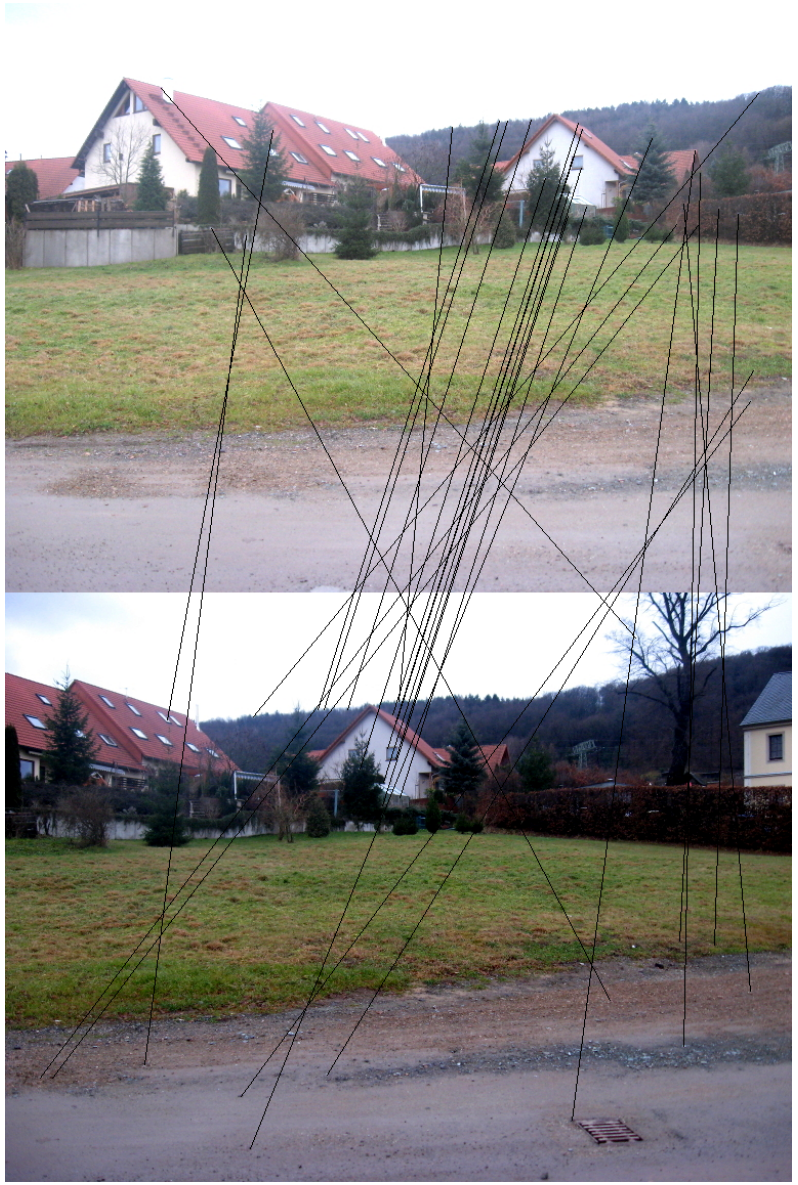
Problem – different colors

Original:

Interest points – Ok.

Problem – different colors

Matches –
obviously too few



Problem – different colors

Panorama – the homographies are wrong

Assignments

Get started:

All the stuff (these slides, the code, images) is available under http://cvlab-dresden.de/courses/computer-vision-1/, play with it – change the parameters, introduce image distortions (color, geometry), identify problems …

General:

You are allowed (and even encouraged) to use the code, you can replace its parts by OpenCV functions or by your own implementations

However:

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

Assignments

Implement one of:

1. Replace Harris detector by another one, e.g. SIFT, MSER (?)
2. Use other features instead of squared distance between the patches, e.g. some color/geometric normalization, correlation coefficient, SIFT descriptors …
3. Replace the matching procedure by another one, e.g. Marriage problem (max 3 points)
4. Implement “incremental” estimation (see next slides)
5. Implement resolution pyramid (see next slides)

Deadline – **11.01.2017** per e-mail an Dmytro.Shlezinger@...

Defenses on request (from my side, if something does not work, from your side, if you are not happy with points).

Incremental estimation

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

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

2. 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

How to make use of this?

1. Apply the “standard pipeline” -> estimate the initial H_0
2. Use H_0 to correct coloring, reduce search space etc.
3. Perform the “standard pipeline” again using these corrections, improve H_t -> H_{t+1}
4. Iterate 2-3 until convergence

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