Robot Vision: Matching

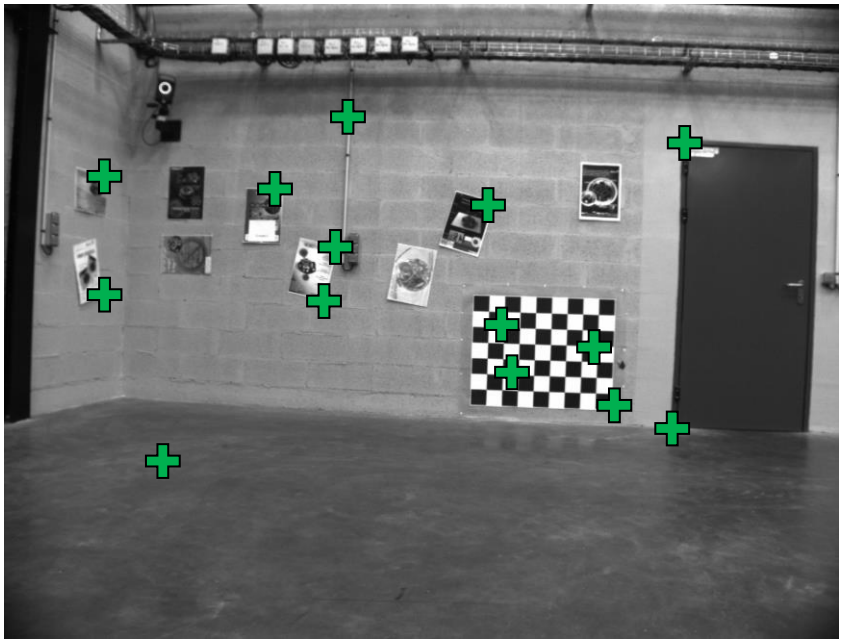
Prof. Friedrich Fraundorfer

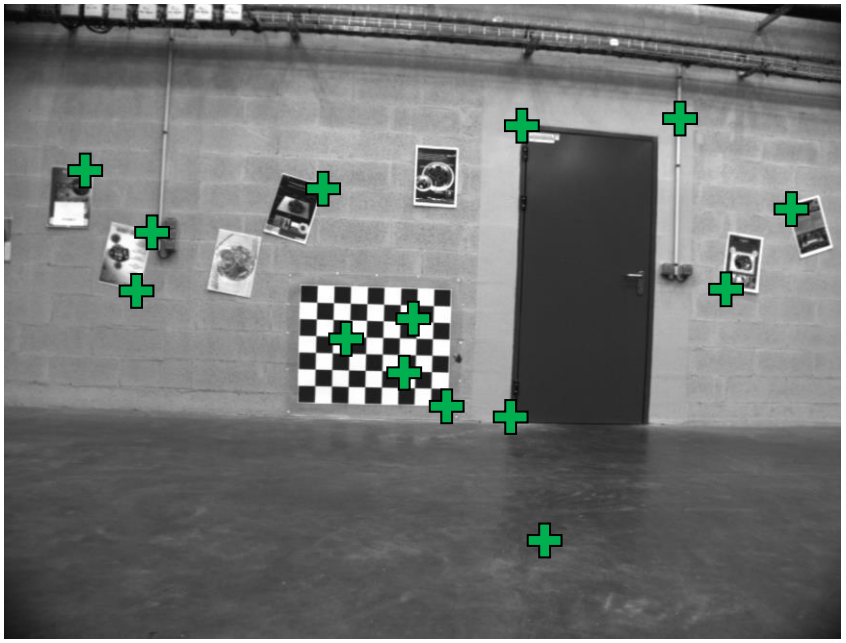
SS 2025

Outline

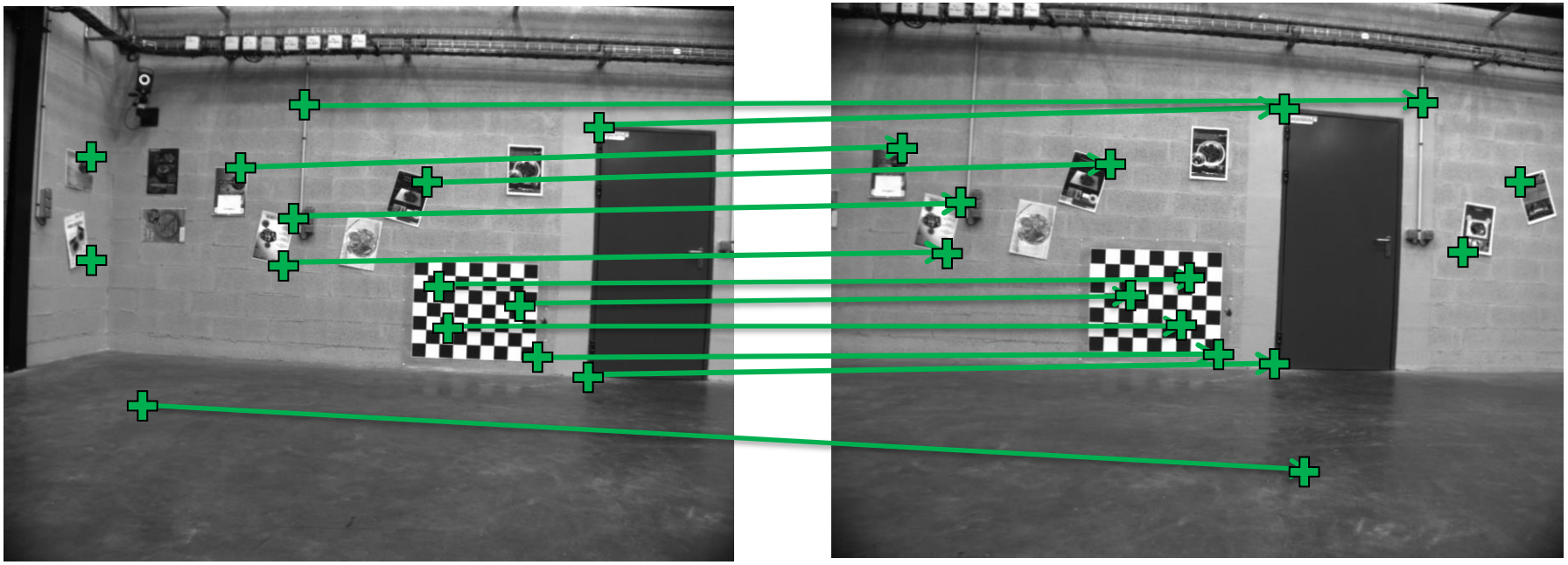
- Feature matching concept
- Descriptors
  - SIFT
  - BRIEF
- Descriptor matching
- Practical issues

Feature matching concept





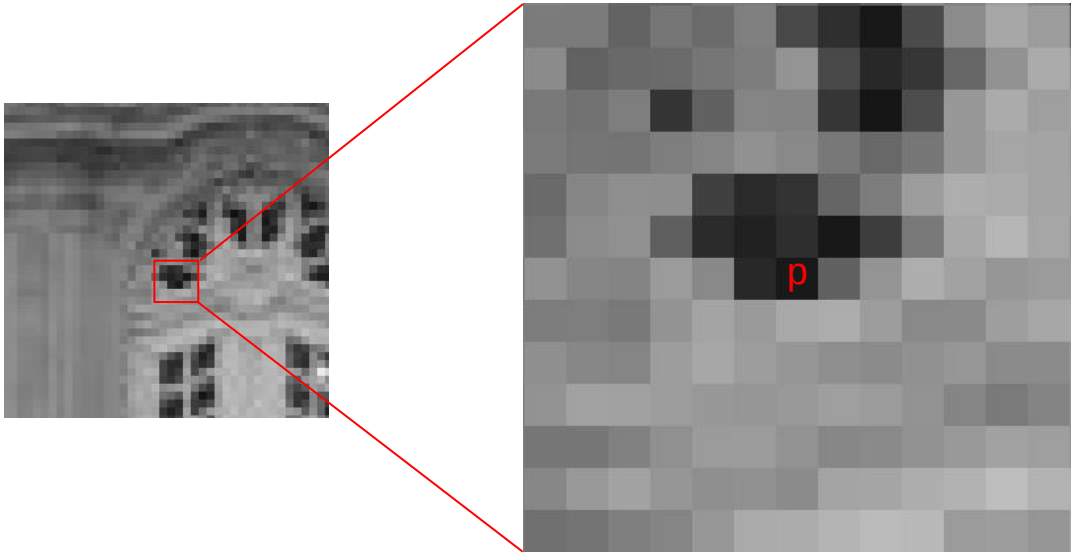
Feature matching concept



Feature matching steps

1. Detect suitable image locations in both images (feature points)
2. Compute a descriptive vector for the image region around each point (feature descriptor)
3. For every feature point in one of the images look for the feature point in the other image that has the most similar descriptor (matching, nearest neighbor search in feature space)

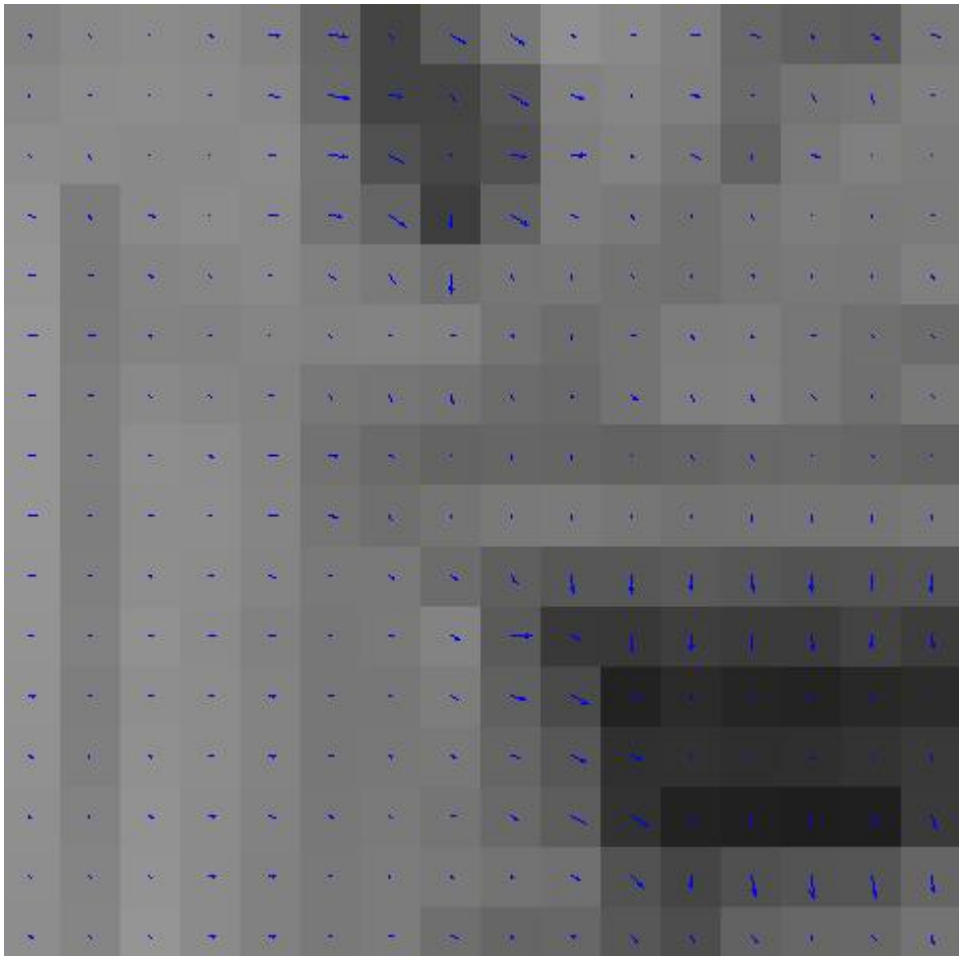
Feature descriptors



- Vector of intensity values within image region
- Intensity value histograms
- Vector of derived attributes (“features”) from image region
  - Gradients, Gradient directions, Filter responses

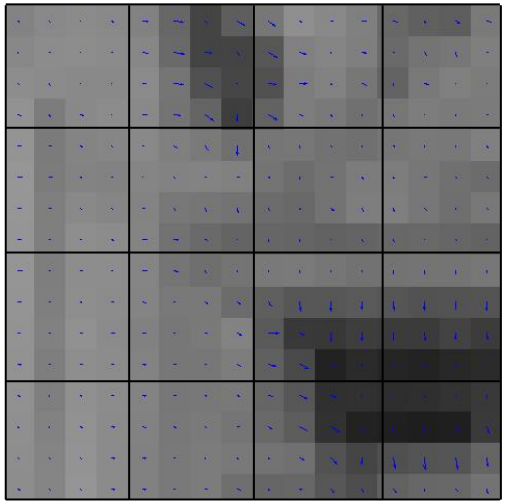
SIFT descriptor

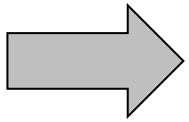
- Computes image gradients and uses a histogram of gradients as descriptor

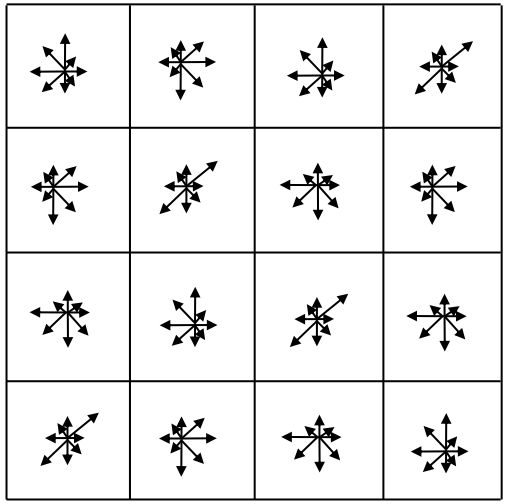


SIFT descriptor

1. Taking a 16 x16 window around interest point location
2. Partitioning into a 4x4 grid of cells.
3. Computation of a histogram of image gradients for each cell (8 bins/orientations each).
4. Normalization of the vector to unit length (L_2-norm = 1)



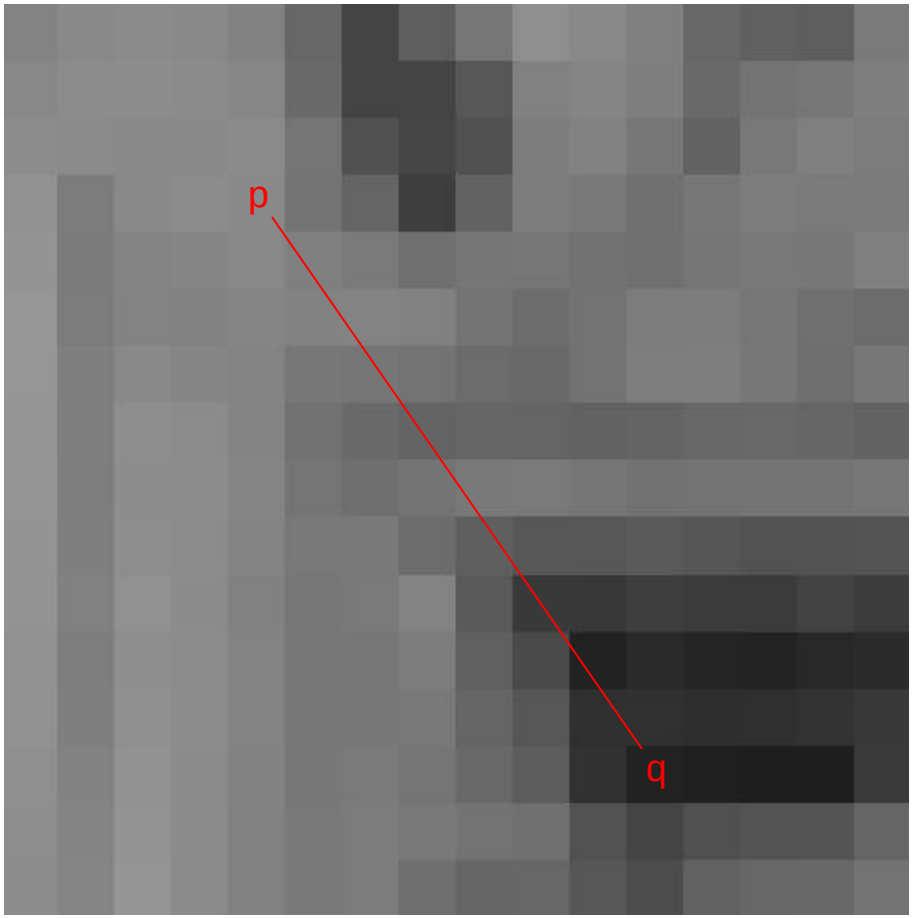




16 histograms with 8 orientations each = 128 features

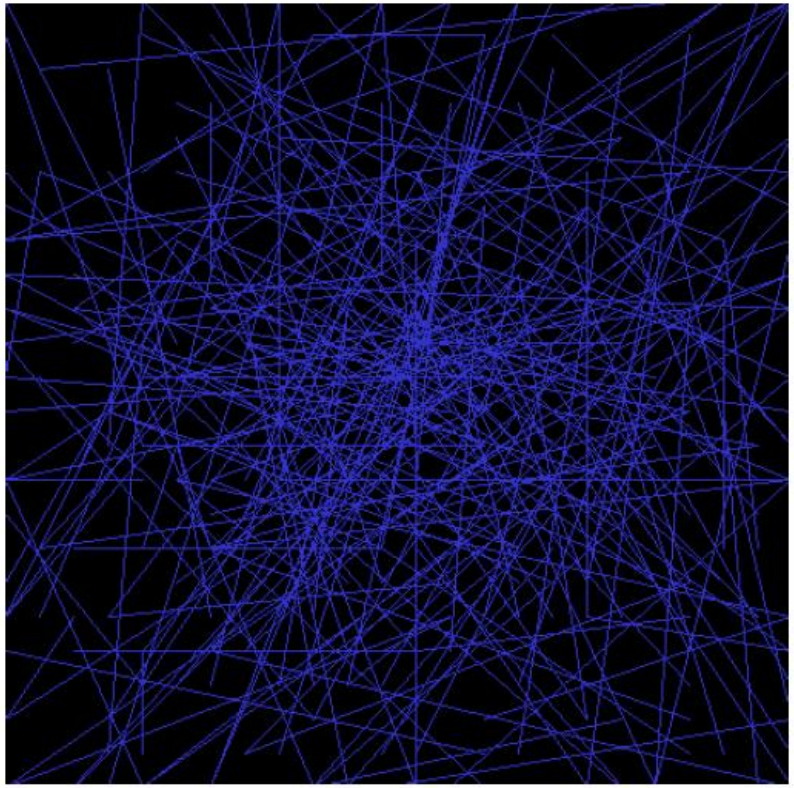
BRIEF descriptor

- Consists of a series of intensity value comparisons
- If I(p) < I(q) then value is 1, otherwise 0.
- Locations p and q are randomly selected and not next to each other



BRIEF descriptor

- The comparison patterns are randomly created but every descriptor is extracted with the same pattern.
- Results in bit strings of 128, 256 or 512 bits lengths.



Descriptor matching

- For every feature point in one of the images look for the feature point in the other image that has the most similar descriptor (matching, nearest neighbor search in feature space)
  - Brute force matching
  - Efficient nearest neighbor search (KD-Tree, ANN)

- Distance measure depends on descriptor
  - SIFT: L_2 distance
  - BRIEF: Hamming distance (number of different bits, number of 1's after XOR operation)

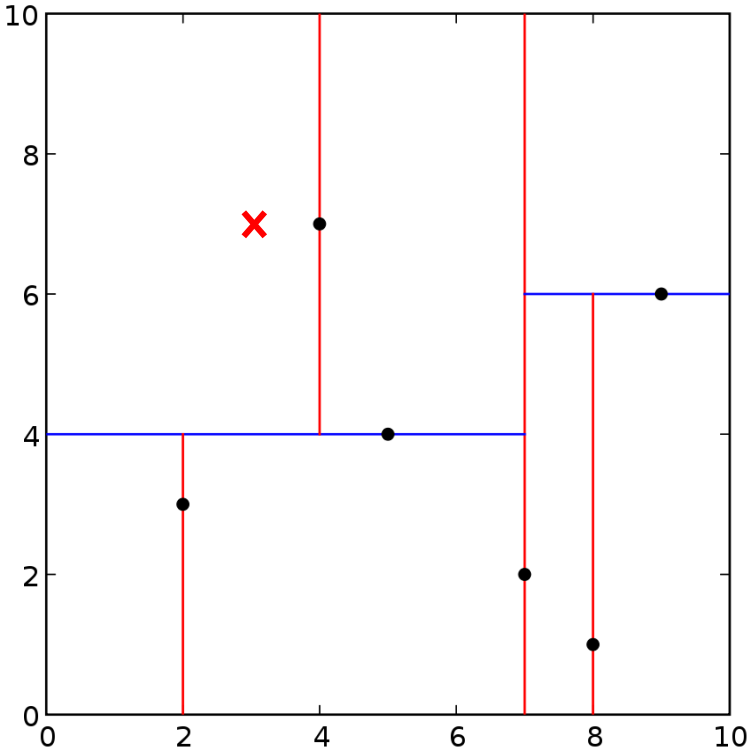
Brute force matching

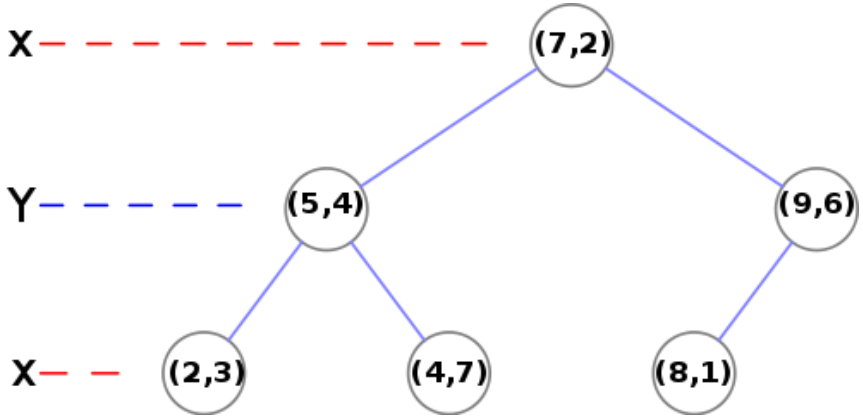
1. Compute all distances between the descriptor vectors ($O(n^2)$)
2. For each descriptor in one image select the one with the smallest distance in the other image.

- Used-up descriptors have to be removed, otherwise one gets one-to-many matches.
- Method is parallelizable and can be run on GPU's

Nearest neighbor search with KD tree

1. Build a KD-Tree from the descriptors of one image O(n logn)
2. For every descriptor of the other image look for the nearest neighbor in the data structure O(n logn)

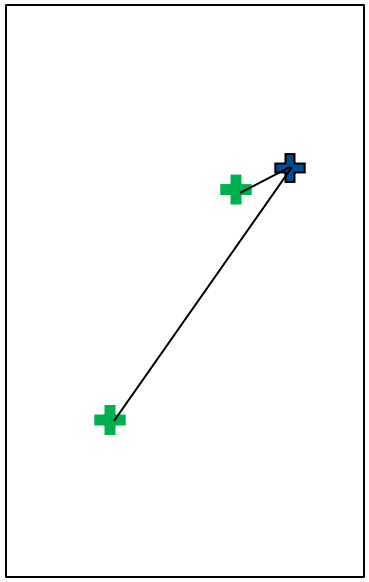




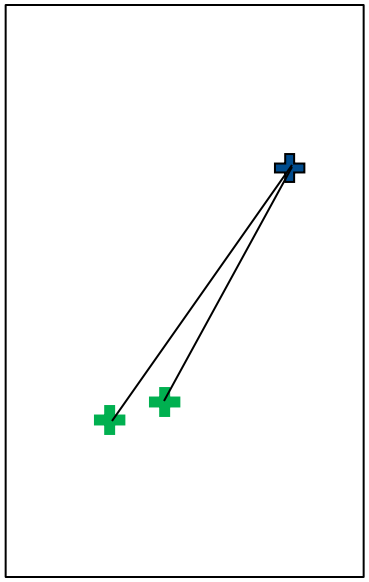
The ratio test

- The nearest neighbor of a feature is not necessarily a correct match
- The distance between two descriptors should be taken into account to classify between a correct or incorrect match
- However, typically a simple threshold on the distance does not work
- Better is to look at the distance ratio of the best and second best match

- Accept a match if $\mathrm{r} = \frac{d(f\ ,f_1)}{d(f\ ,f_2)} < t$
- t=0.8 is a useful threshold



good match
r close to 0



bad match
r close to 1

Geometric constraints

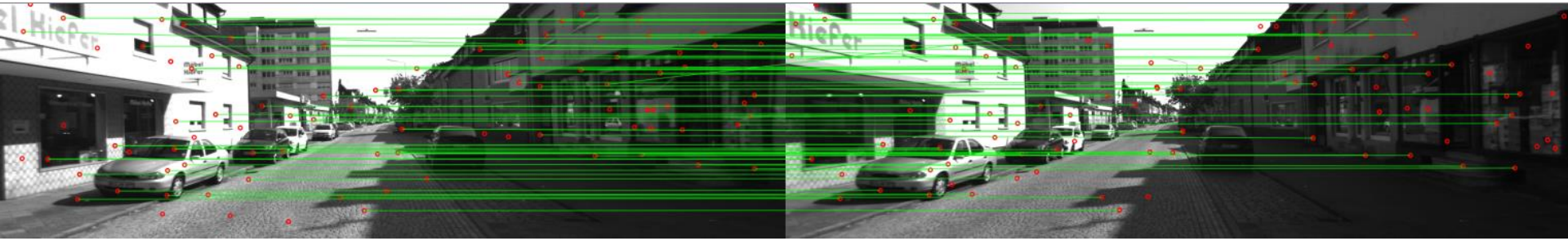
- If knowledge about the location of possible feature matches exists matching can be sped up or made more robust

- E.g. for temporal image sequences (video) the interest points do not change much between frames, feature matching only needs to consider points that have a similar x and y location
- E.g. for stereo systems, feature matches that have the same y coordinate need to be considered only

Geometric verification

- Feature matches between two images need to be geometrically correct (i.e. fulfill the epipolar constraint)
- By computing the fundamental matrix in a robust way from feature matches, incorrect matches can be identified as outliers.

initial matches



geometrically verified matches