Camera Drones
Lecture – 3D data generation

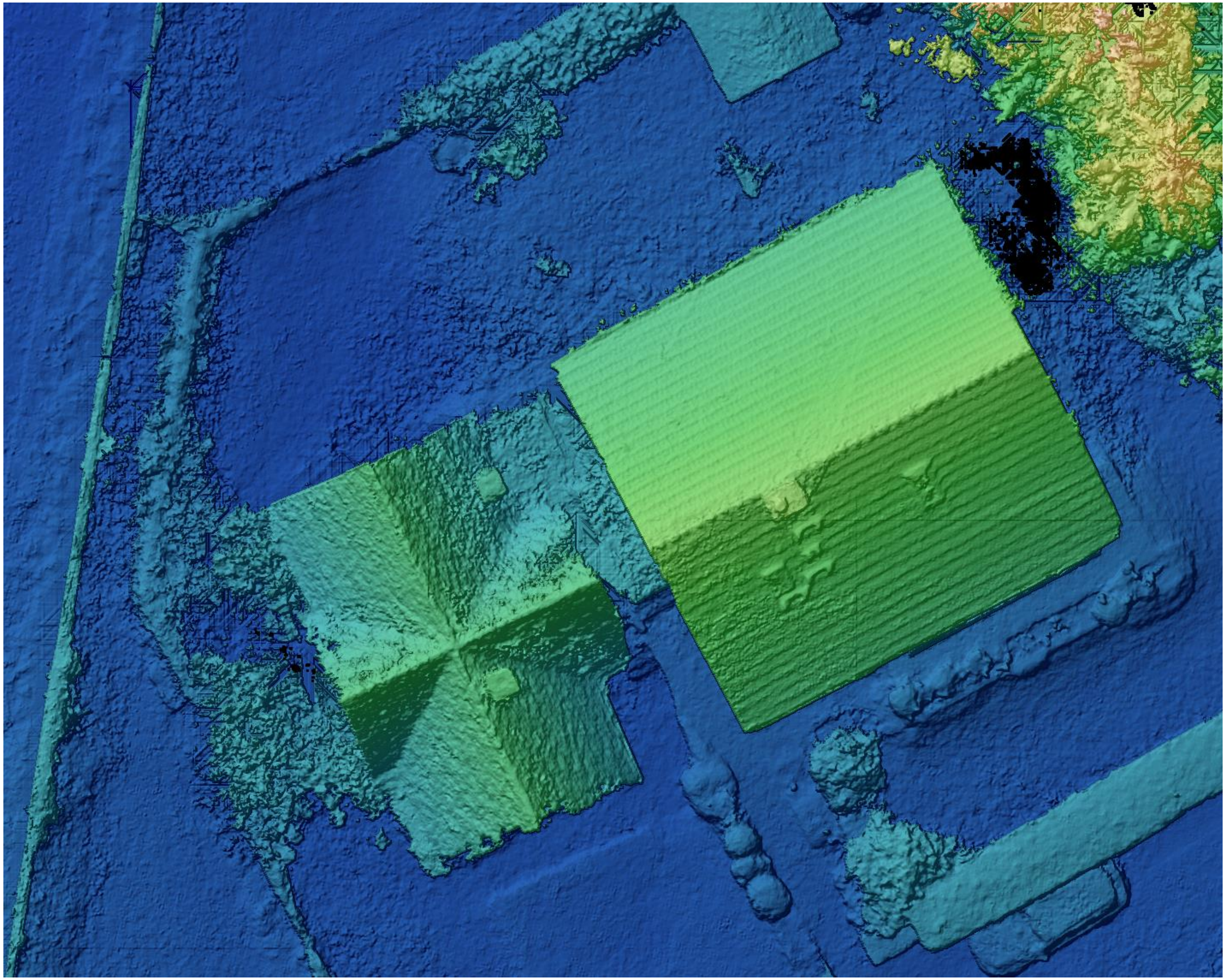
Prof. Friedrich Fraundorfer

WS 2021

Outline

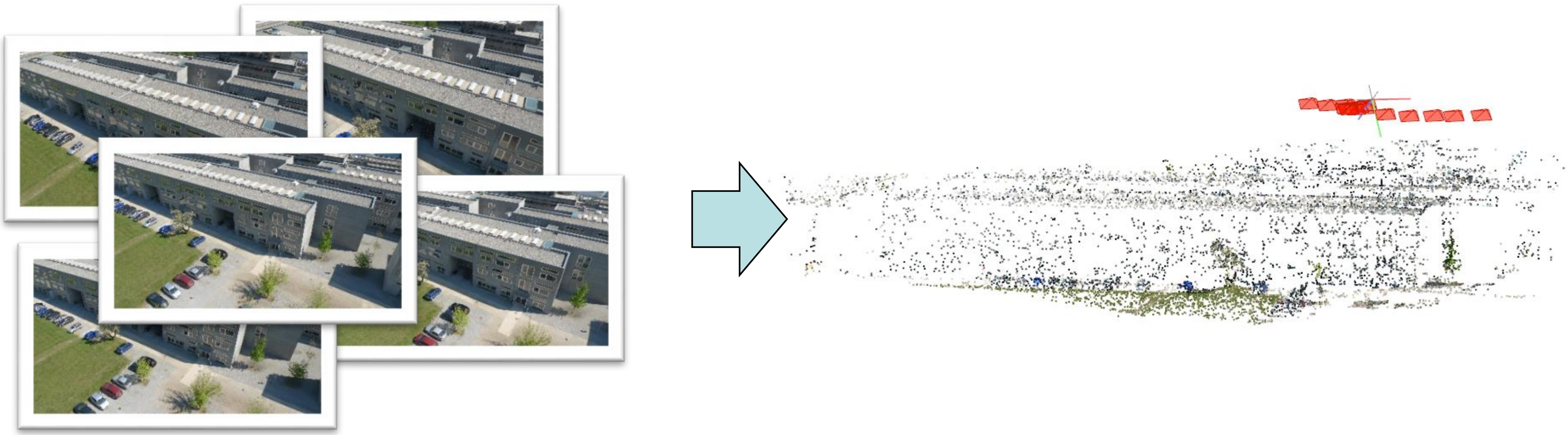
- Structure-from-Motion
  - SfM introduction
  - SfM concept
  - Feature matching
  - Camera pose estimation
  - Bundle adjustment
- Dense matching
- Data products (Orthophoto, DSM)

DSM



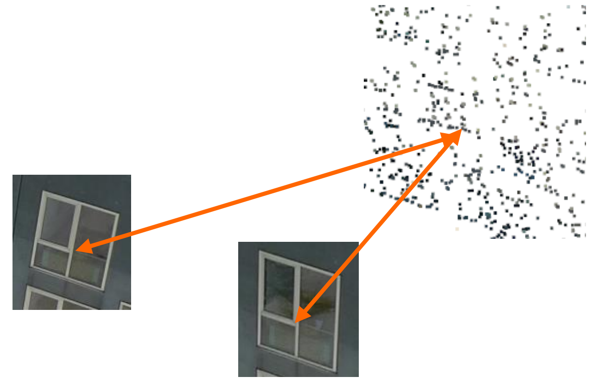
Orthophoto

Structure-from-Motion (SfM) concept

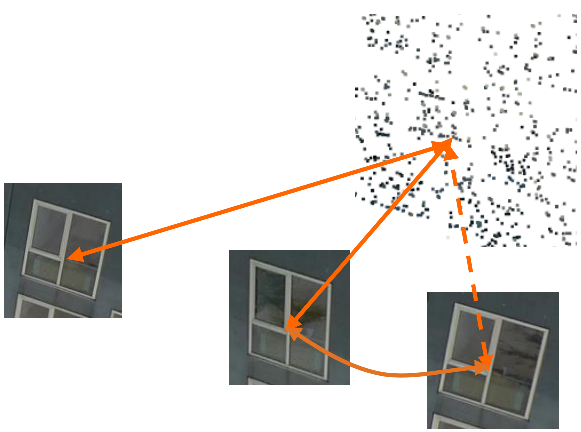


Structure-from-Motion (SfM) concept

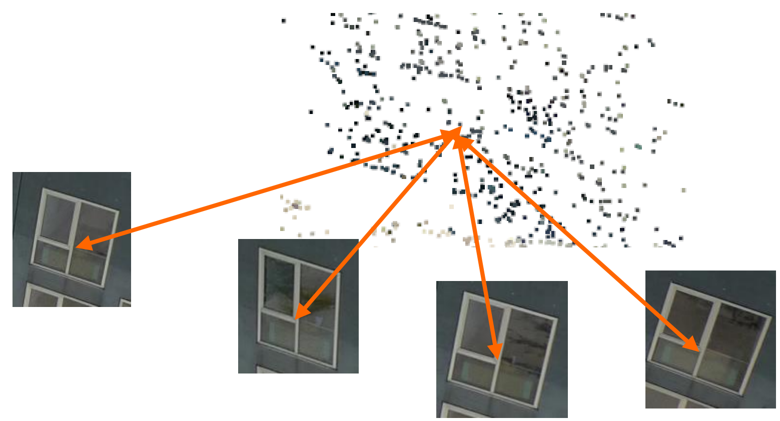
Initialize Motion
(P_1,P_2 compatible with F)



Initialize Structure
(minimize reprojection error)

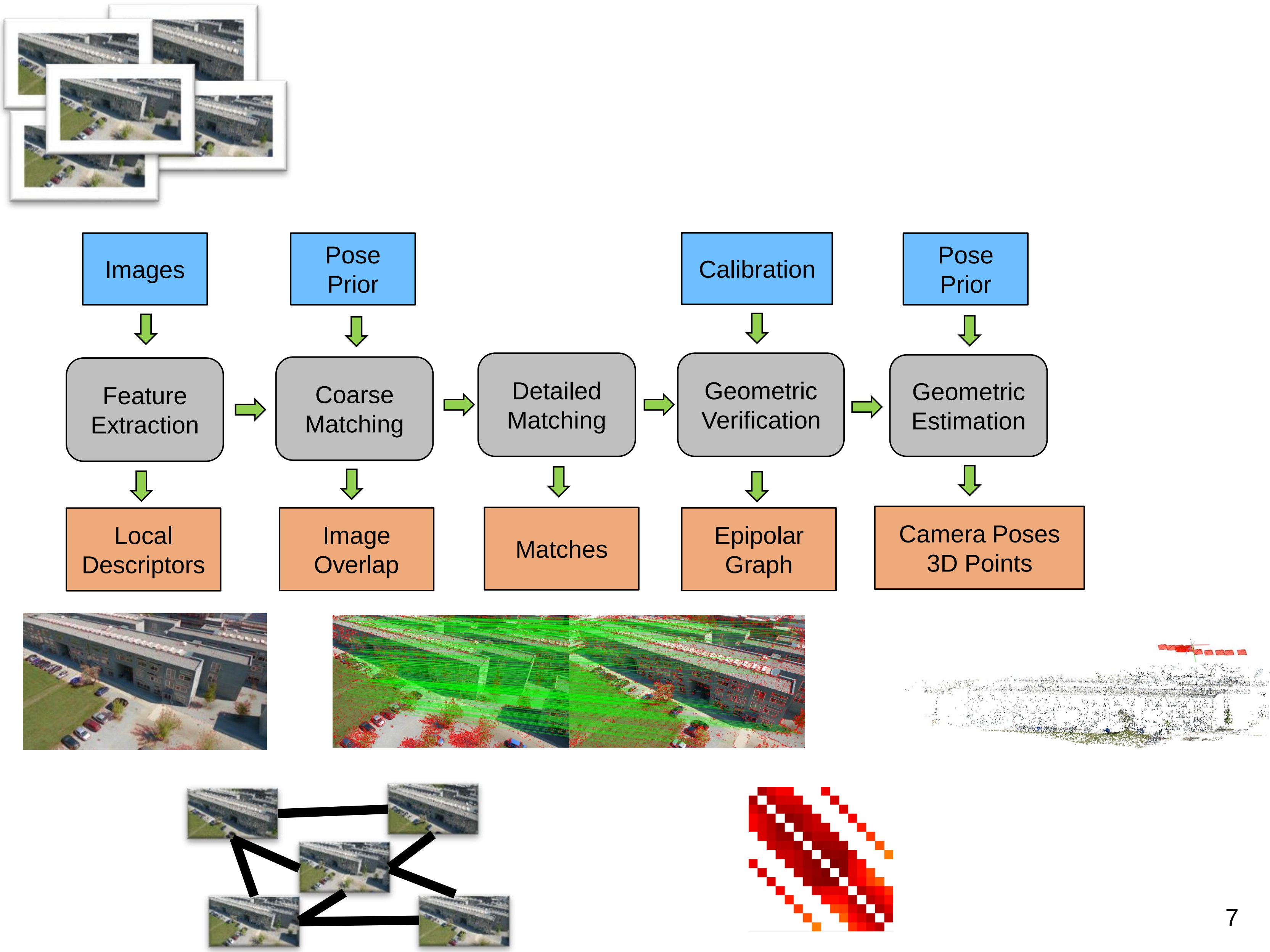


Extend motion
(compute pose through matches
seen in 2 or more previous views)

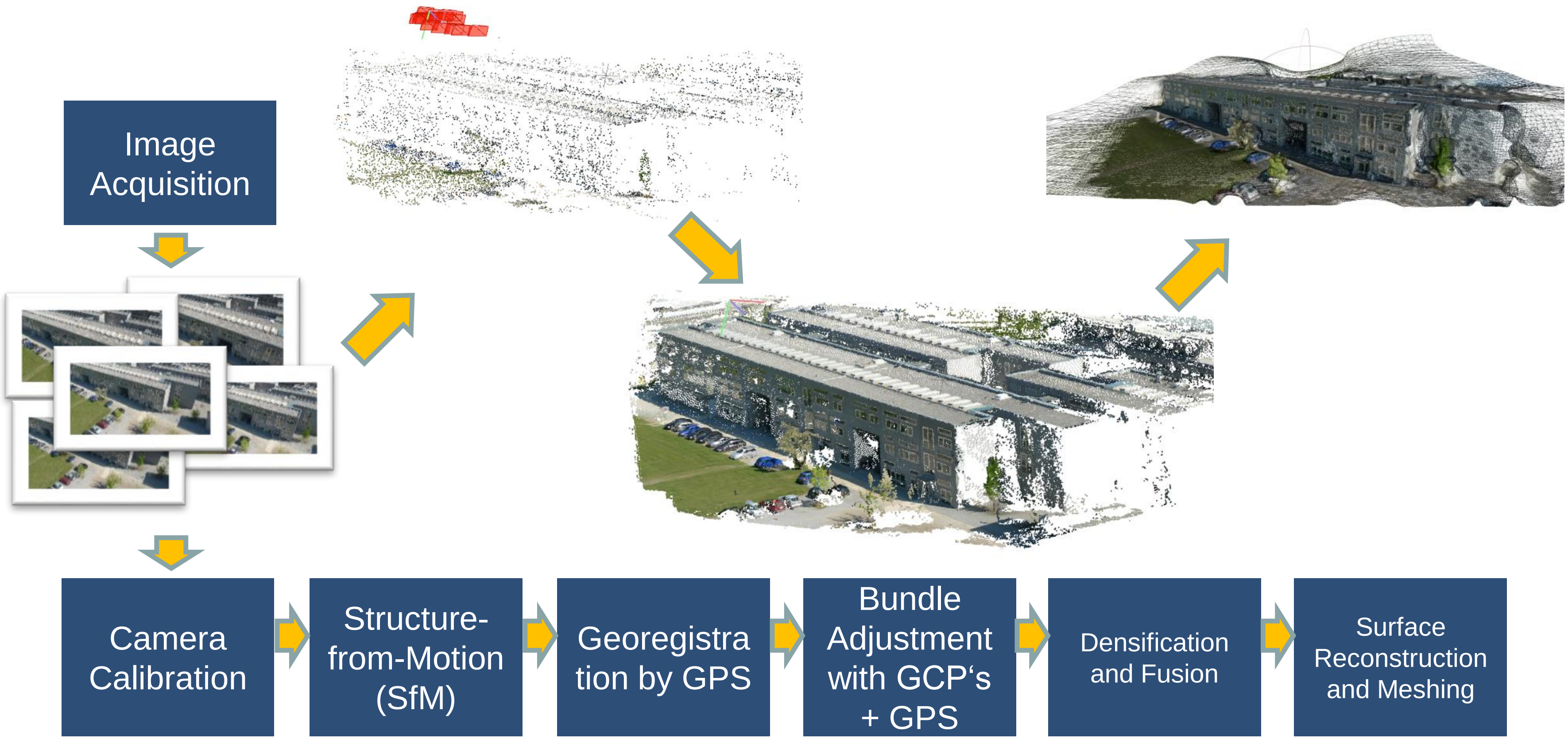


Extend structure
(Initialize new structure,
refine existing structure)

Structure-from-Motion (SfM) core pipeline

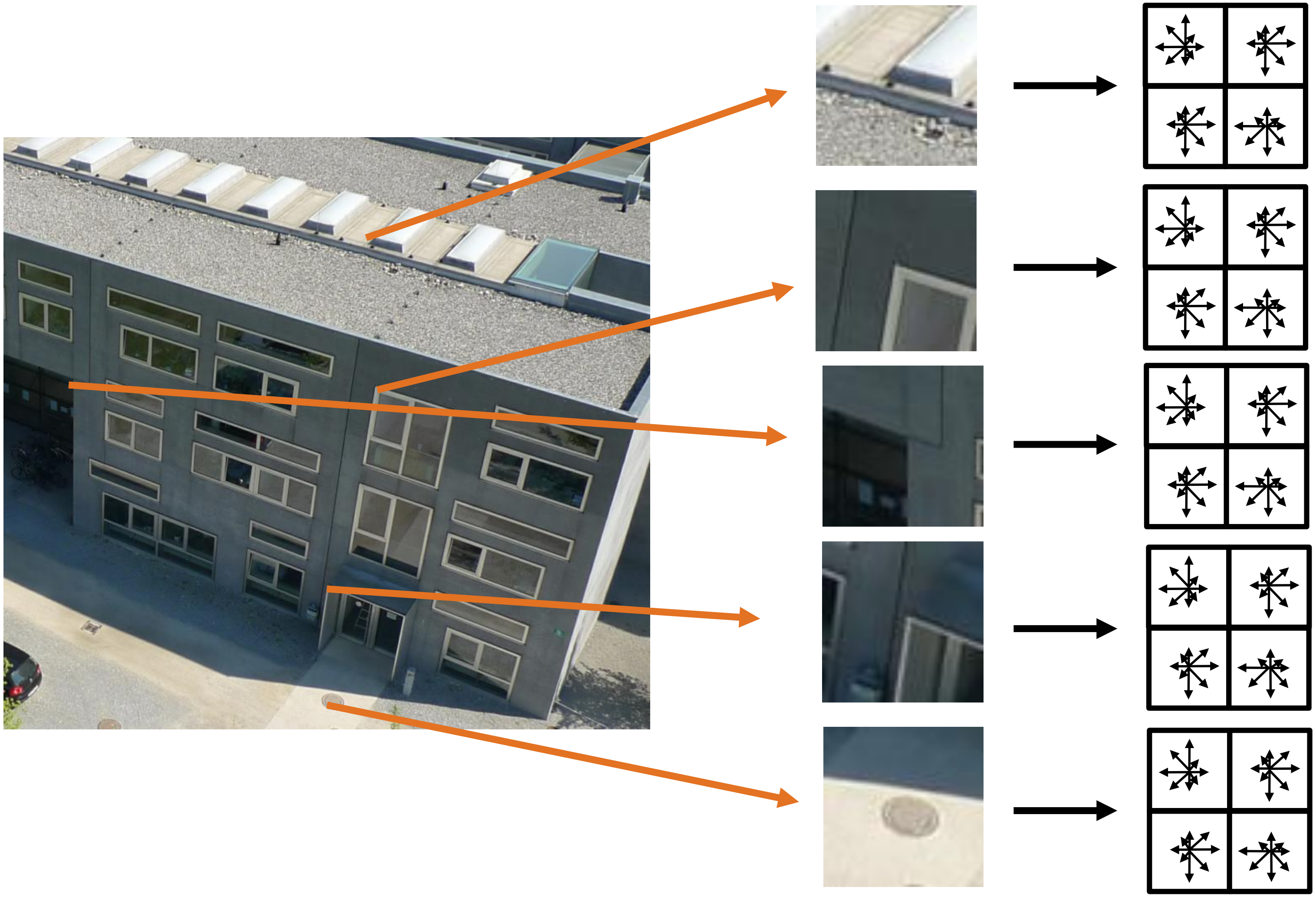


3D data generation overall pipeline



Feature extraction

- SIFT features (best working features for matching right now)
- Each descriptor is a vector of length 128 (gradient histogram)



Coarse matching

- Cluster similar images by similarity using visual words
- Will be used for speeding up exhaustive (nxn) matching

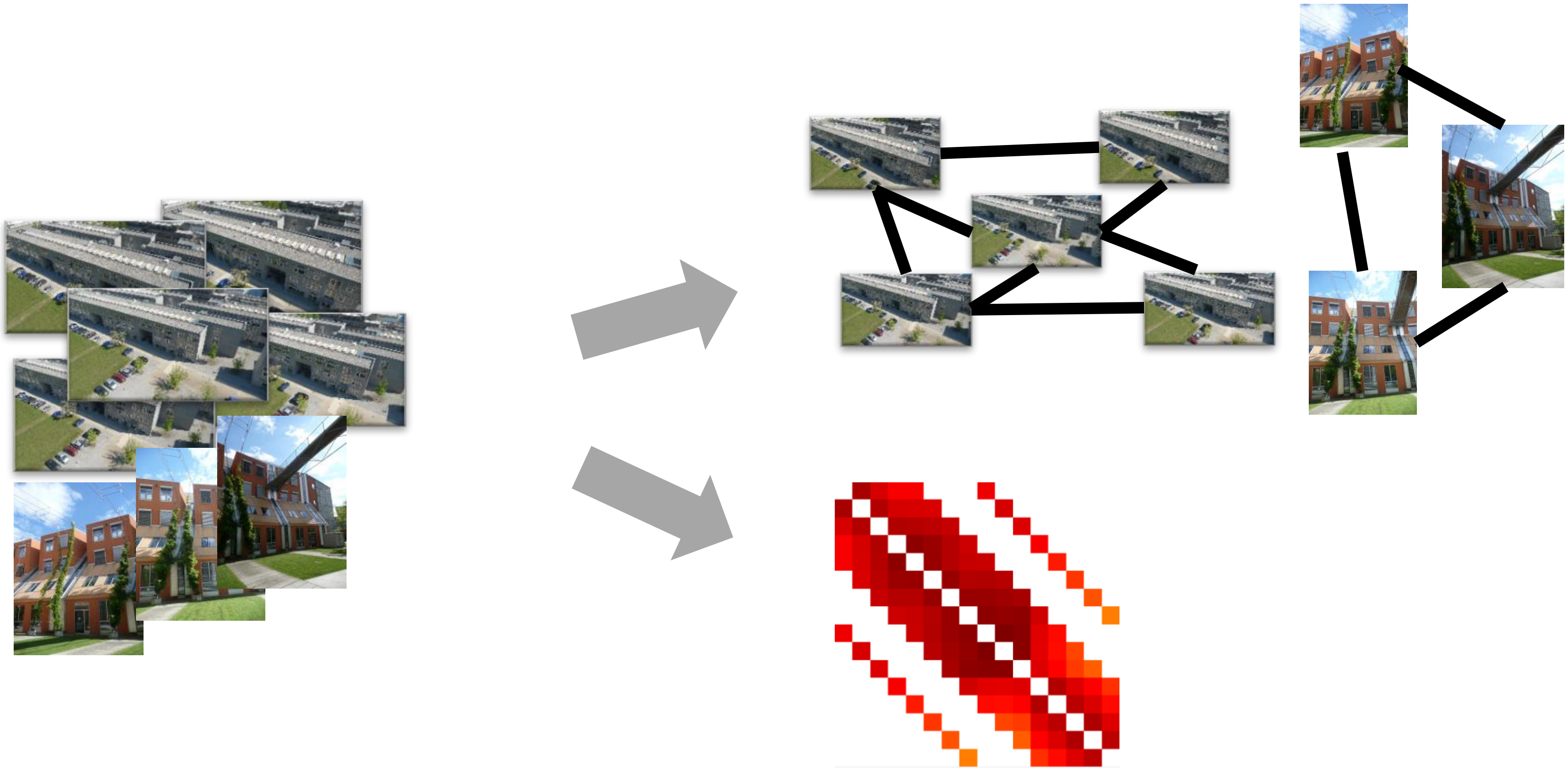
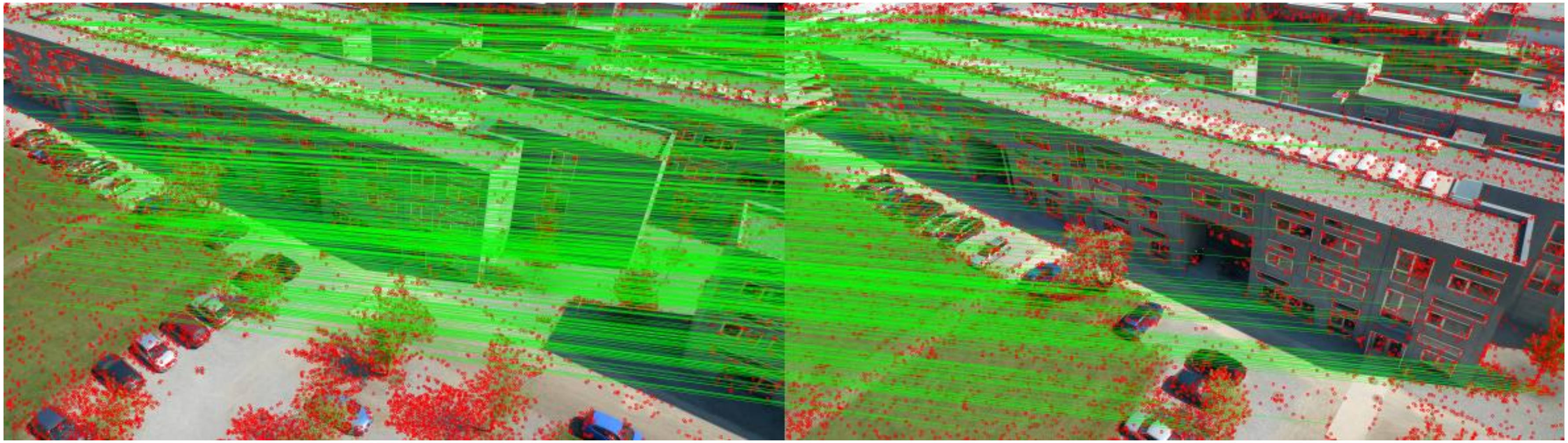
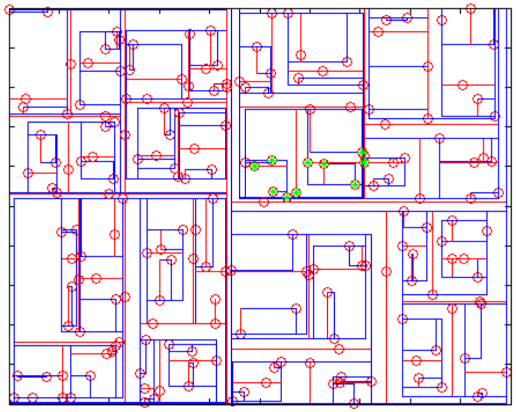


Image similarity

Detailed matching

- Typically using NN-search with a Kd-tree





Epipolar graph

- Defines the sequential order for geometry processing
- Is a plot of the number of geometrically verified feature matches

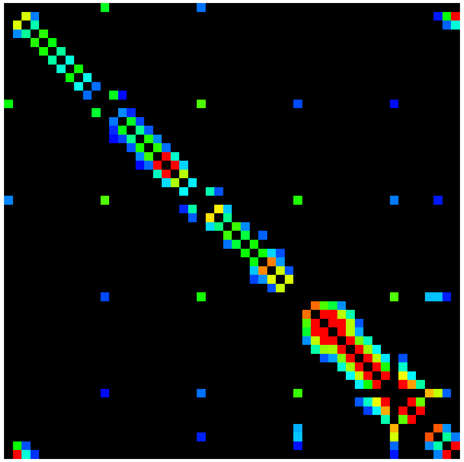
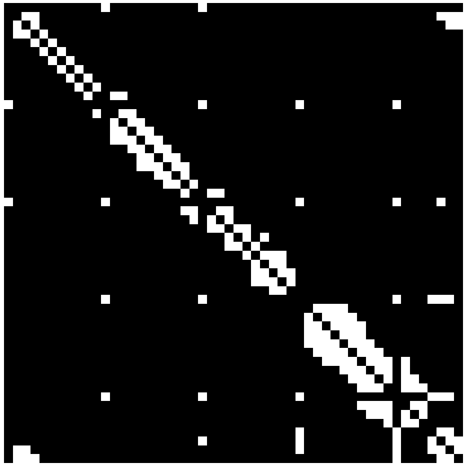


Image similarity



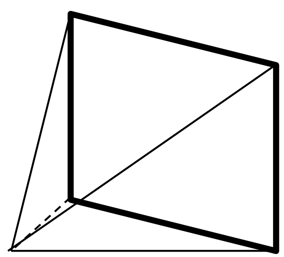
Epipolar graph

Geometry estimation

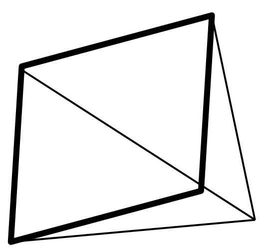
- Following the sequence ordering from the epipolar graph geometry is estimated for all images
- Geometry estimation is an alternating scheme:
  - Estimate camera pose of new images (position, rotation)
  - Triangulate new 3D data points seen in new image
  - Refinement by non-linear optimization (Bundle adjustment)

Geometry estimation steps

- Compute camera poses of the first two images from feature matches



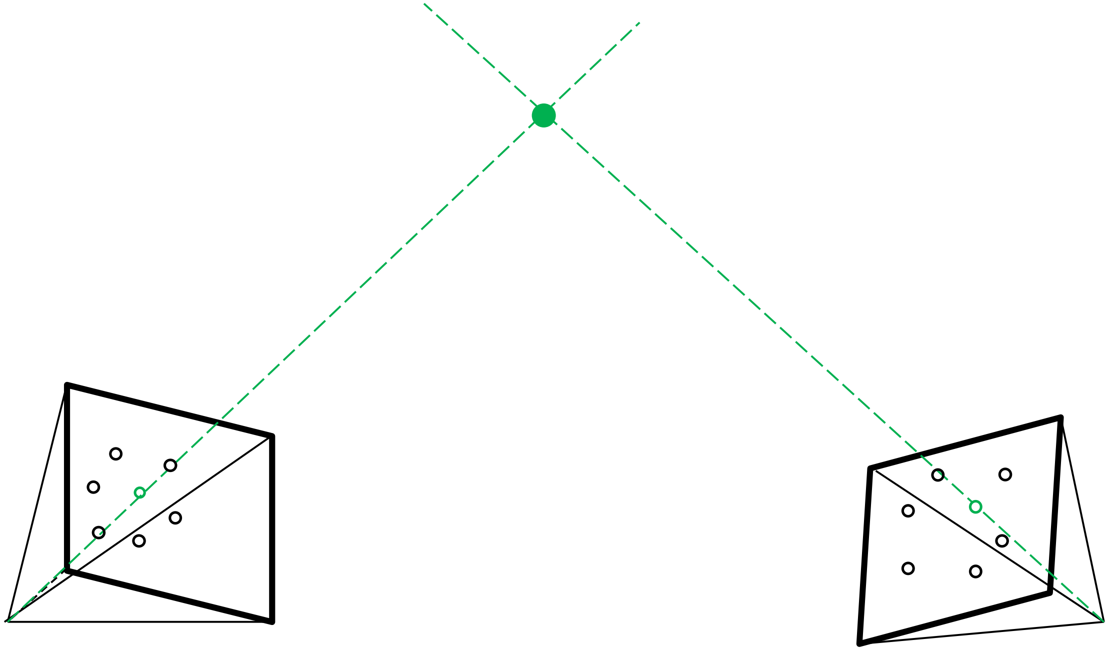
$$P = K[I|0]$$



$$P' = K'[R'|t']$$

Geometry estimation steps

- Computation of first 3D points by triangulation

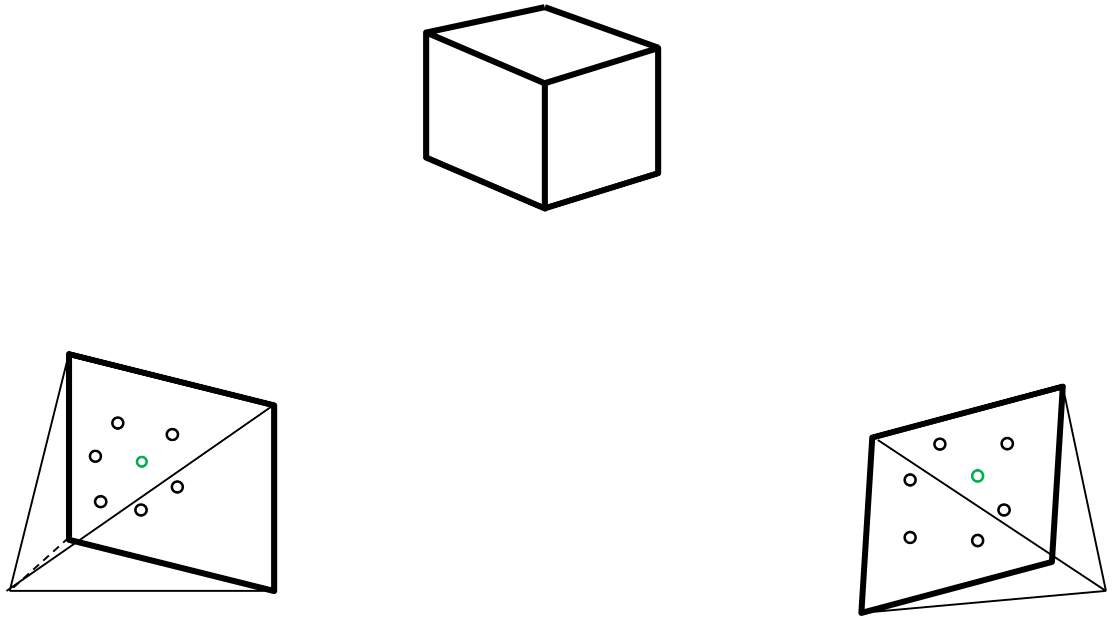


$P = K[I|0]$

$P' = K'[R'|t']$

Geometry estimation steps

- Triangulate all feature matches of the first images

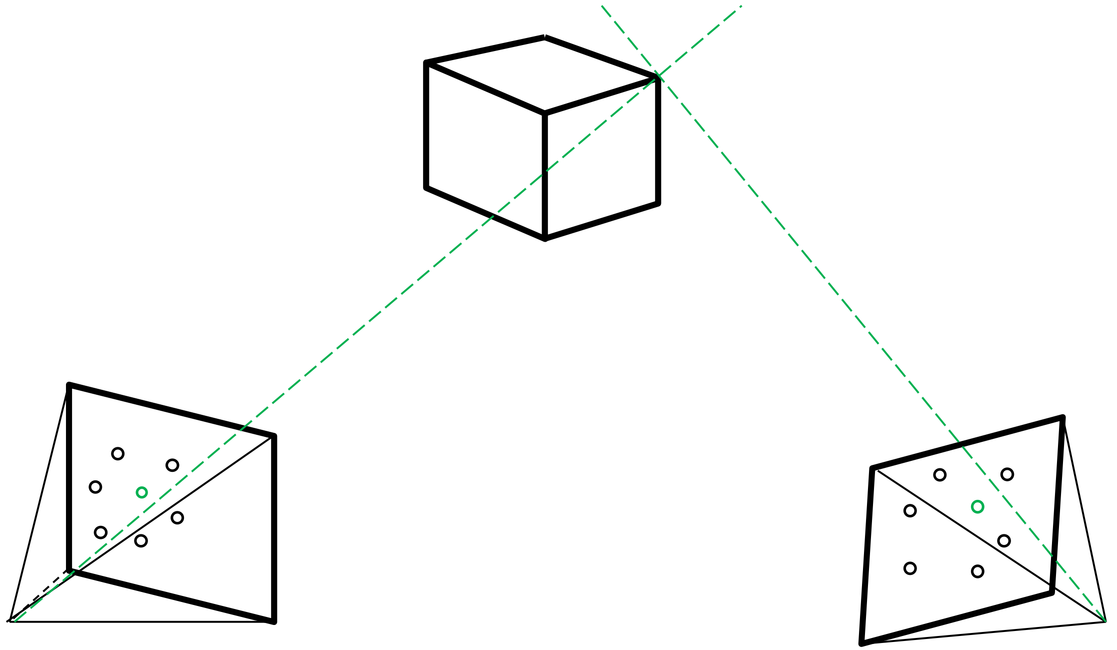


$P = K[I|0]$

$P' = K'[R'|t']$

Geometry estimation steps

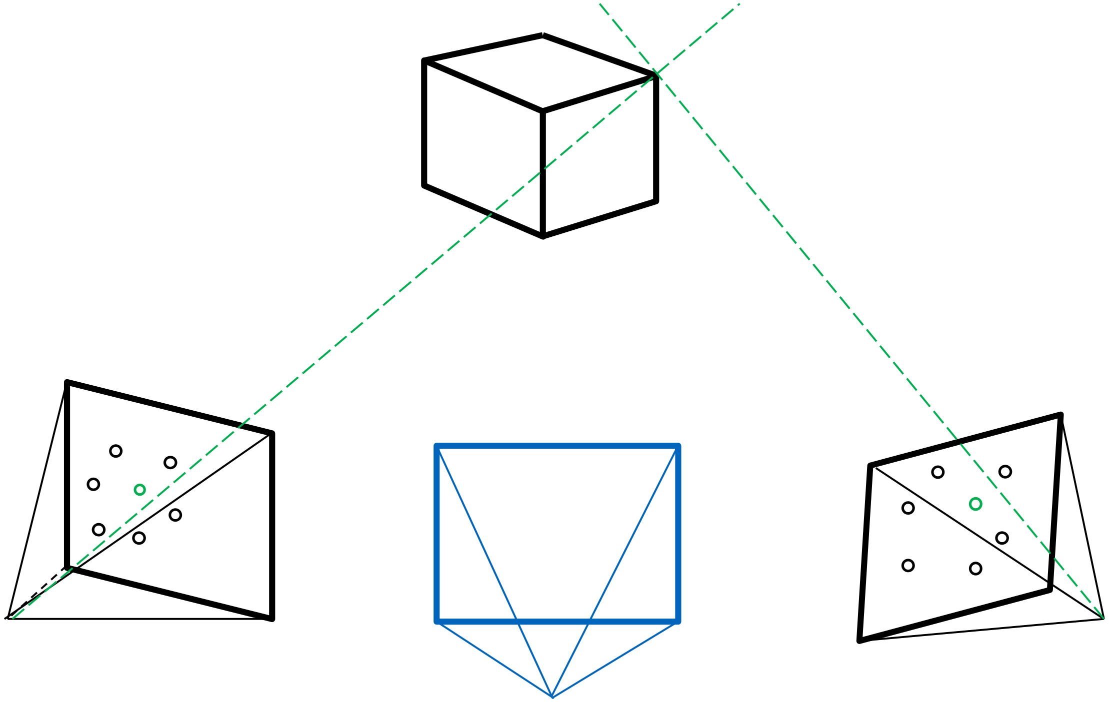
- First refinement of camera poses and 3D points by non-linear estimation of the re-projection error through bundle adjustment



$P = K[I|0]$ $\qquad$ $P' = K'[R'|t']$

Geometry estimation steps

- Start processing the next image



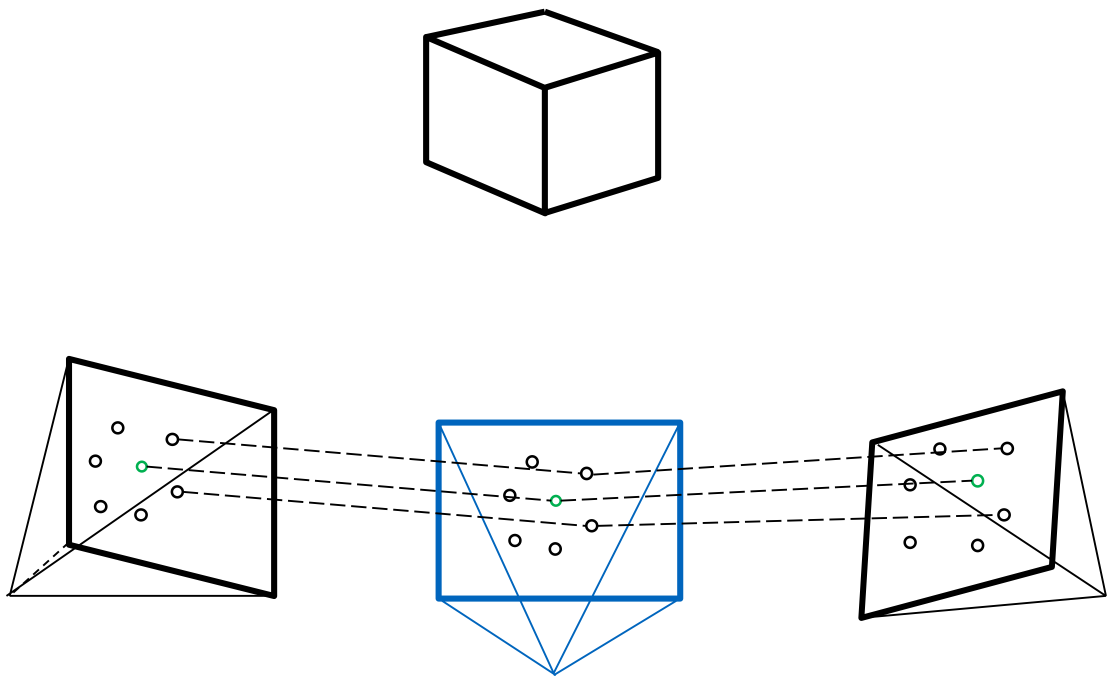
$P = K[I|0]$

$P'' =?$

$P' = K'[R'|t']$

Geometry estimation steps

- First, create feature matches to all the previous, neighboring images



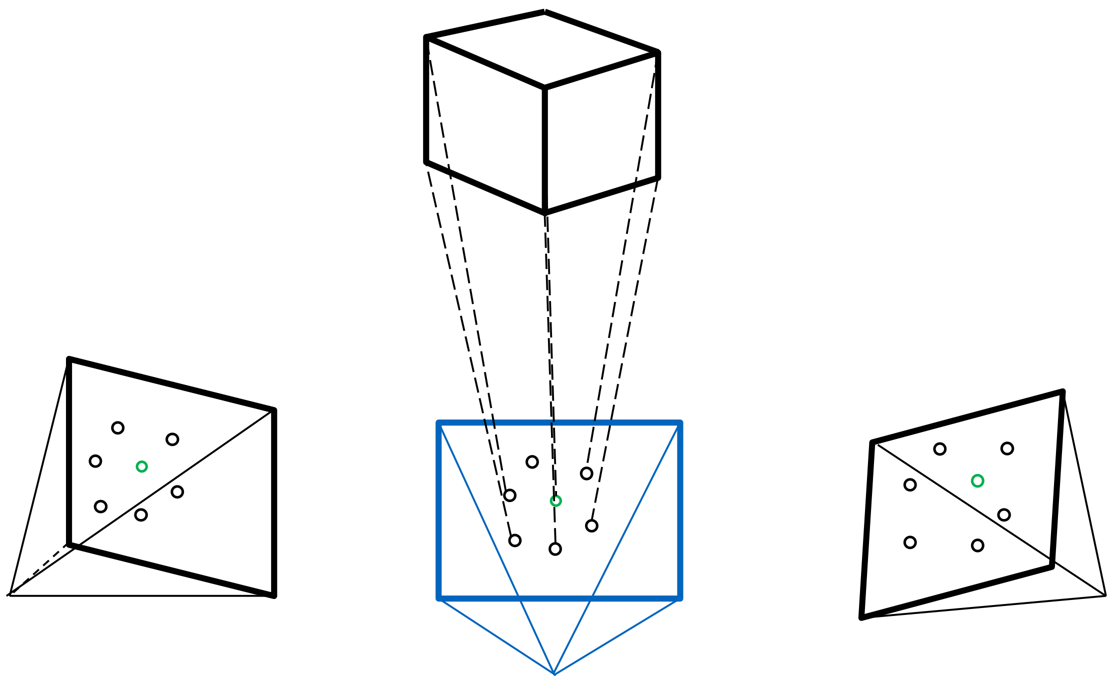
$P = K[I|0]$

$P'' = ?$

$P' = K'[R'|t']$

Geometry estimation steps

- Feature matches give correspondences to already computed 3D points
- From corresponding 2D and 3D points the pose of the new camera can be computed using the PnP-Algorithm



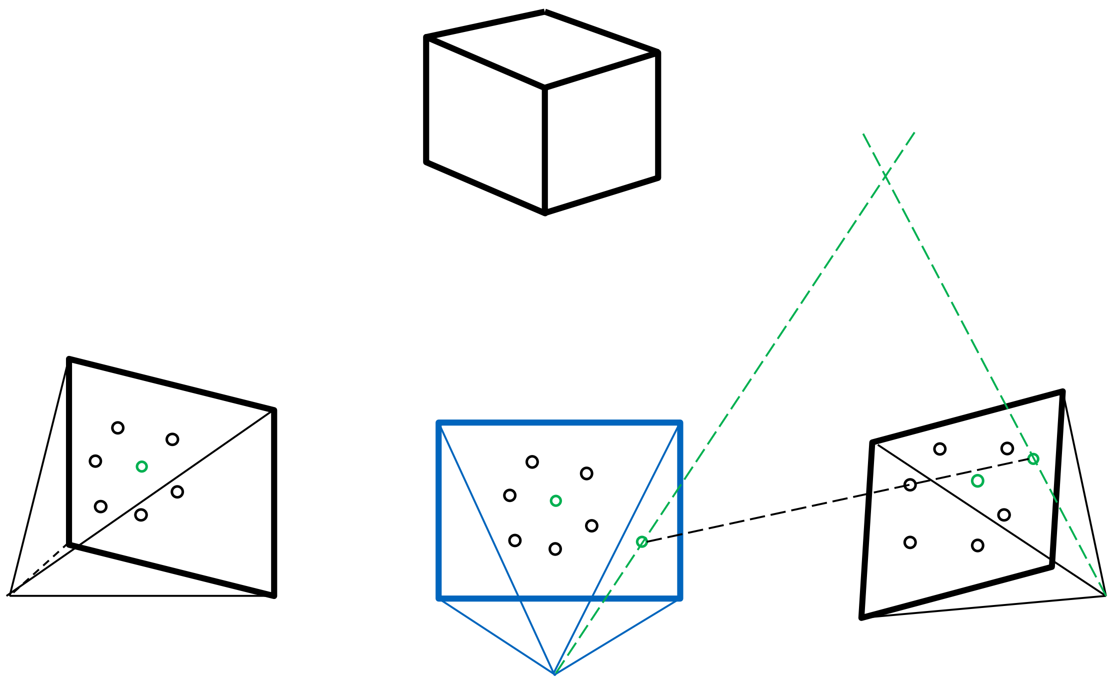
$$P = K[I|0]$$

$$P'' = ?$$

$$P' = K'[R'|t']$$

Geometry estimation steps

- Repeat the process starting again from triangulation of new features



$P = K[I|0]$

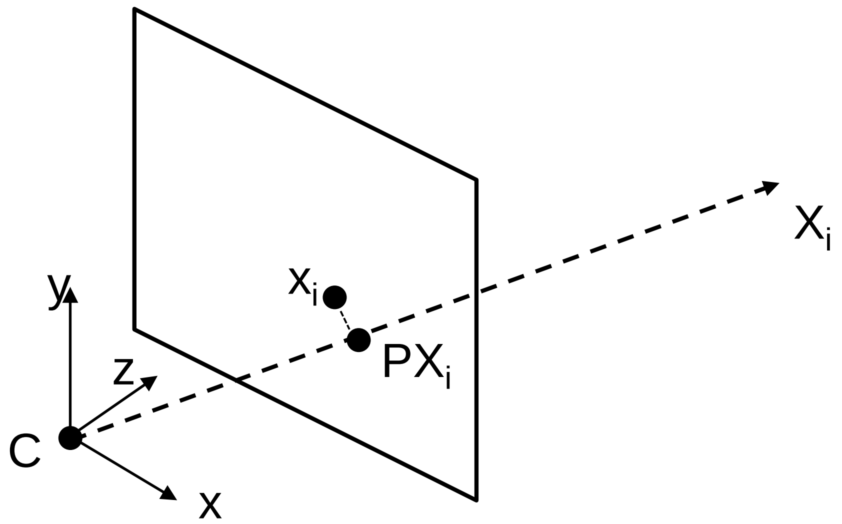
$P' = K'[R'|t']$

$P'' = K''[R''|t'']$

Bundle adjustment

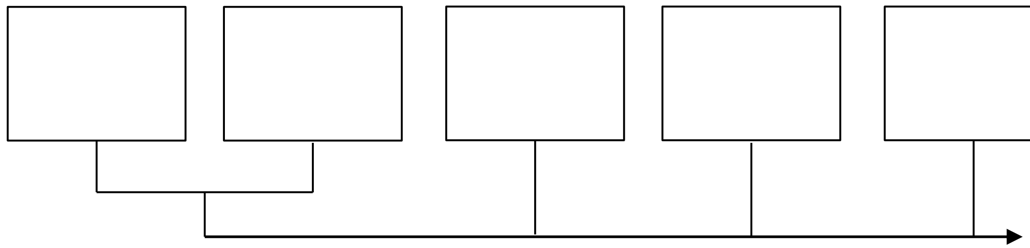
- Levenberg-Marquard optimization of re-projection error
- Parameters are camera poses and all 3D points (millions of parameters to optimize!)

$$\min_{P_j, X_i} \left(\sum_i \sum_j \| x_{i,j} - P_j X_i \| \right)$$

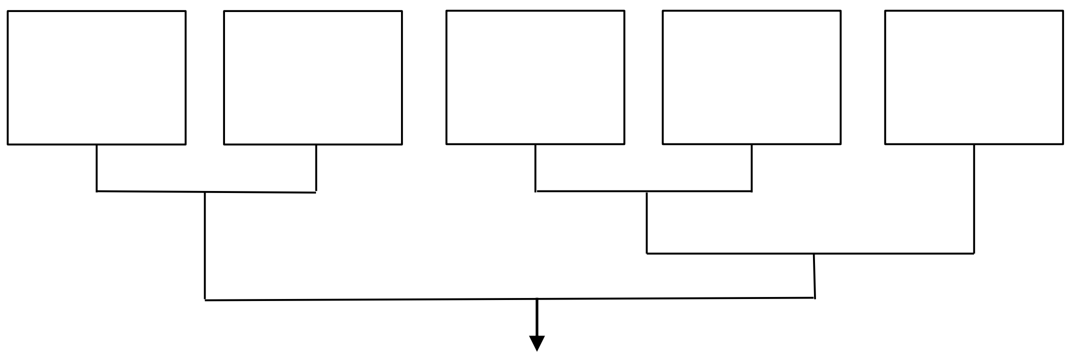


3 paradigms

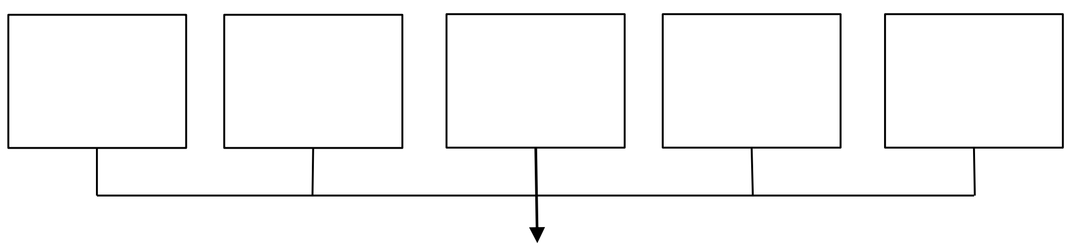
sequential



hierarchical



global

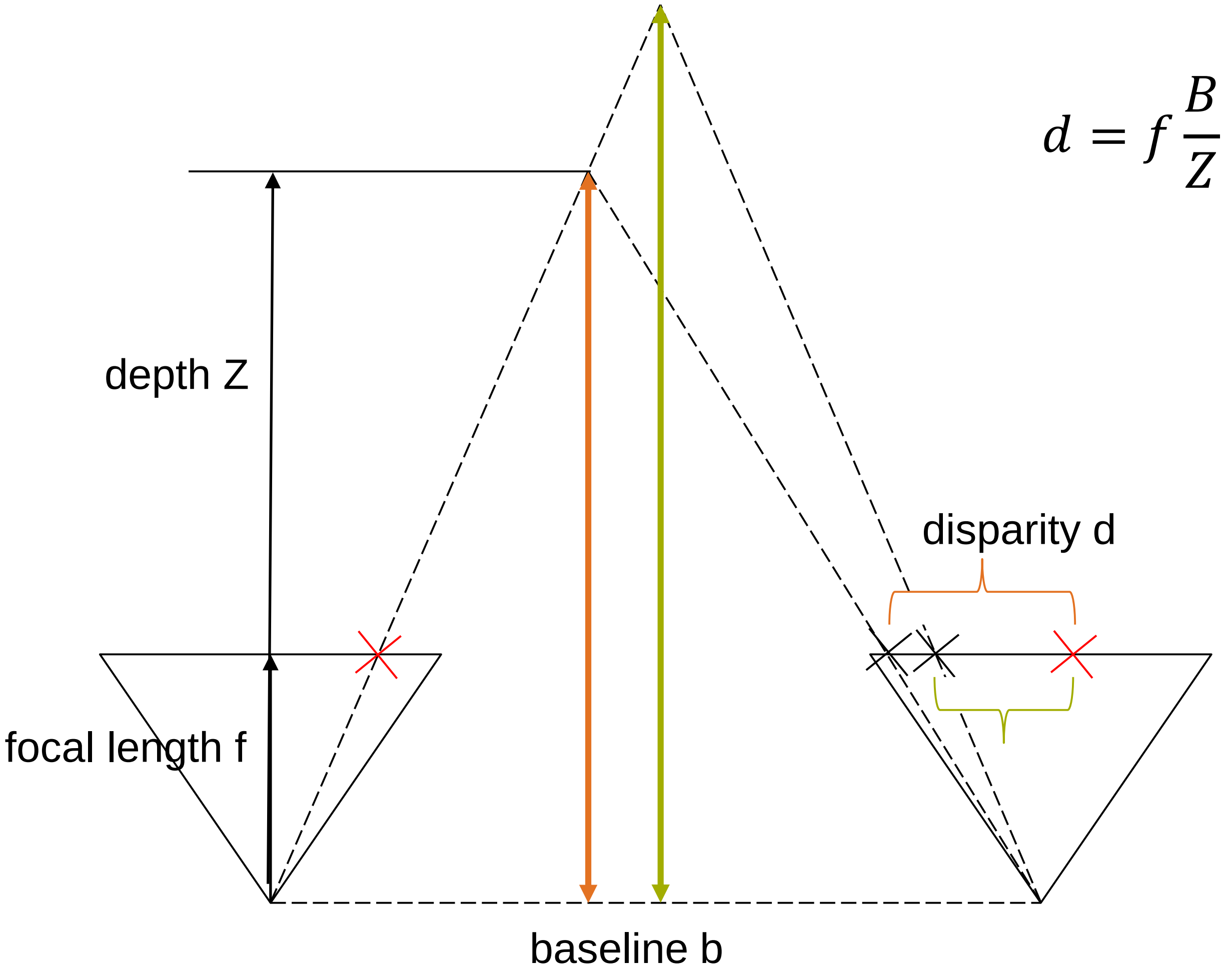


Dense matching

- SfM only gives sparse 3D data
- Only SIFT feature points are triangulated – for most pixel no 3D data is computed
- Dense image matching computes a 3D point for every pixel in the image (1MP image leads to 1 million 3D points)
- Dense matching algorithms need camera poses as prerequisite

Geometric relation

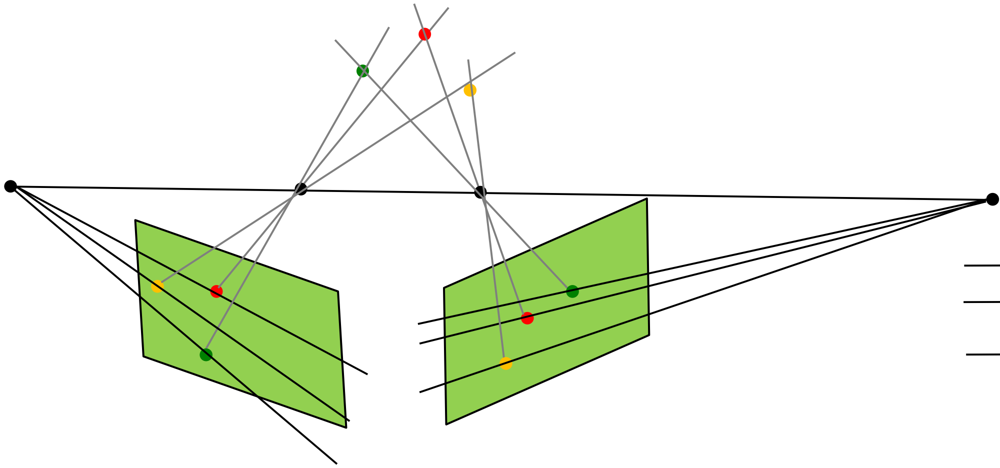
- Stereo normal case
- Depth Z [m] can be computed from disparity d [pixel]



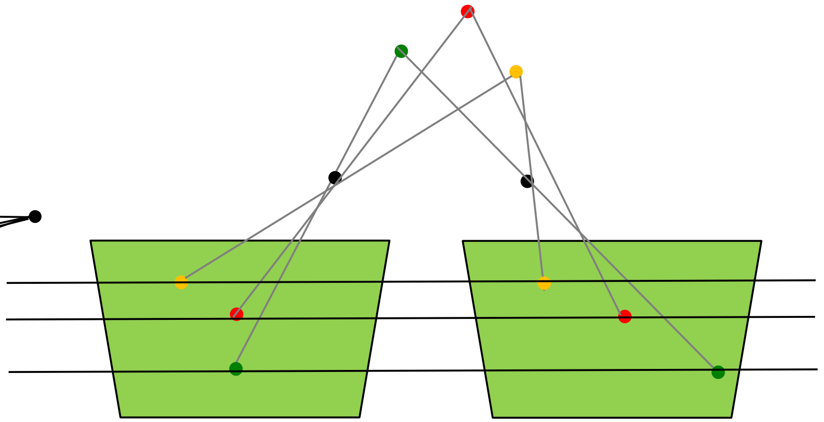
$$d = f\,\frac{B}{Z}$$

Rectification

- Image transformation to simplify the correspondence search
  - Makes all epipolar lines parallel
  - Image x-axis parallel to epipolar line
  - Corresponds to parallel camera configuration



Before rectification

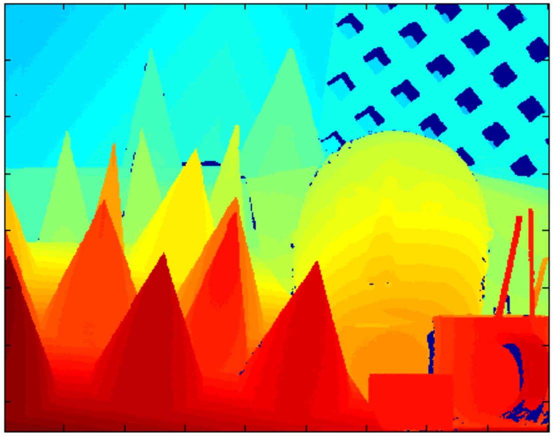


„Stereo normal case“

Dense matching process

+

=



Left View I_l

Right View I_r

Disparity image D

- Estimate disparity (depth) for all pixels in image left.
  - Evaluate correspondence measure for every possible pixel location on the line (e.g. NCC, SAD)

- Disparity *d*: Offset between pixel *p* in the left image and its correspondent pixel *q* in the right image.

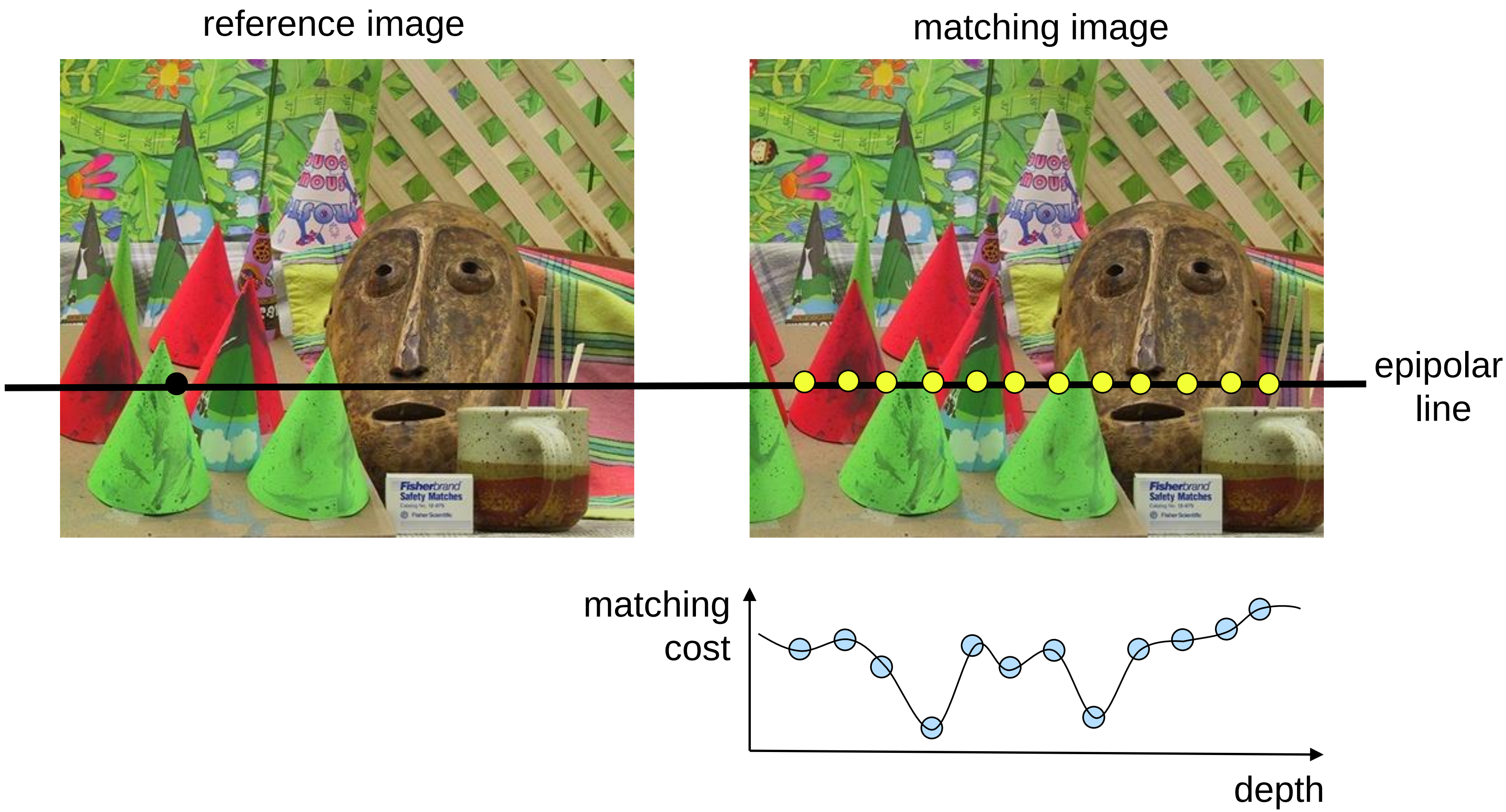
Census Transform

- A popular block matching cost
- Good robustness to image changes (e.g. brightness)
- Matching cost is computed by comparing bit strings using the Hamming distance (**efficient**)
- Bit strings encode if a pixel within a window is greater or lesser than the central pixel (0 .. if center pixel is smaller, 1 .. if center pixel is larger)

89	63	72
67	**55**	64
58	51	49

00000011

Dense matching process



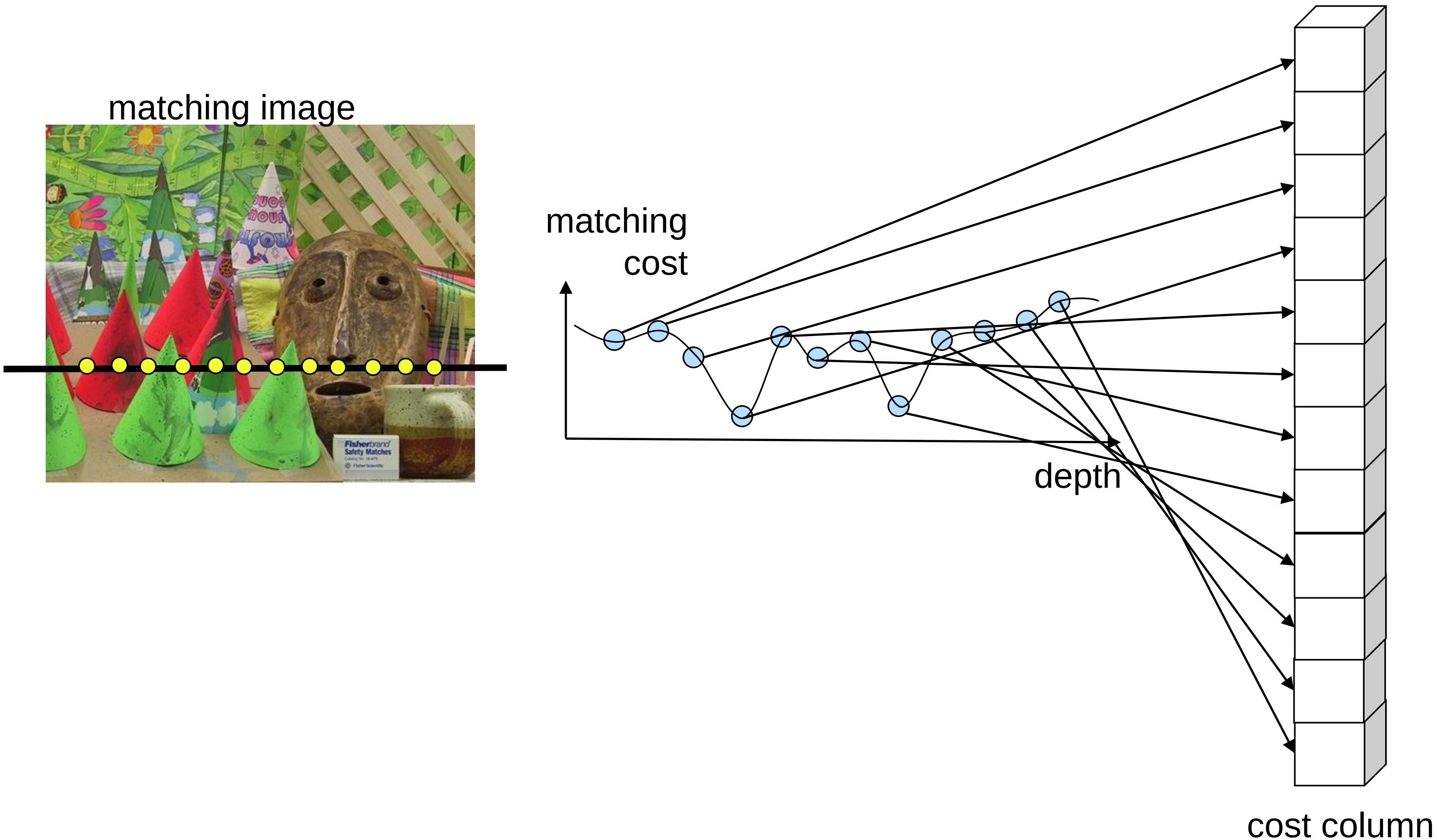
Disparity selection

- Single scanline based
  - Winner takes all (WTA)
    Select the disparity with the lowest cost (i.e. the highest similarity)

  - Scanline optimization (Dynamic programming)
    Select the disparities of the whole scanline such that the total (added up) costs for a scanline is minimal

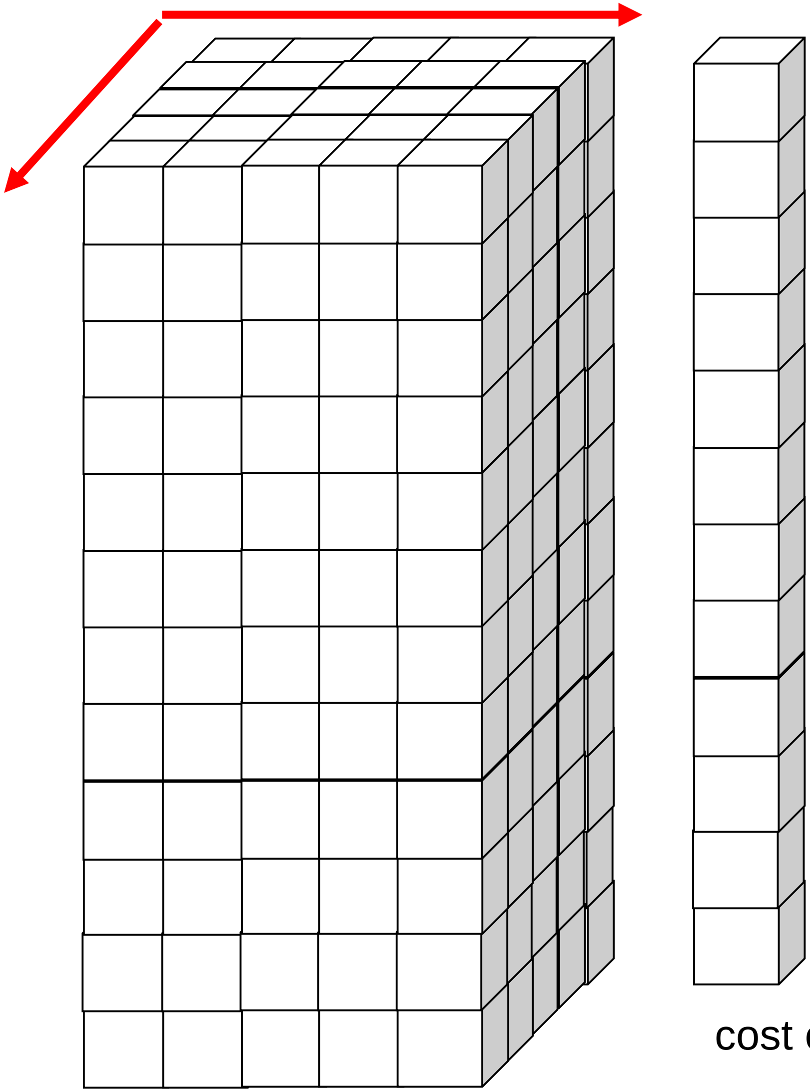
- Global methods (Cost volume optimization)
  - Belief propagation
    Selects the disparities such that the total cost for the whole image is minimal
  - Semi-global Matching
    Approximates the optimization of the whole disparity image

Dense matching process



Dense matching process

matching image

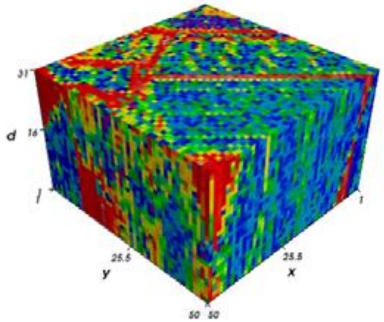


cost volume

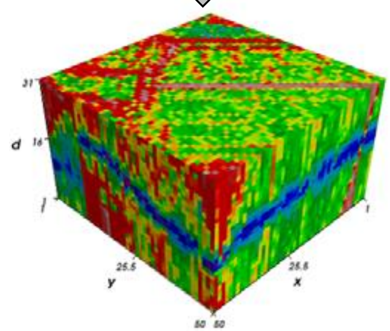
cost column

Semiglobal matching

Dense Cost Computation



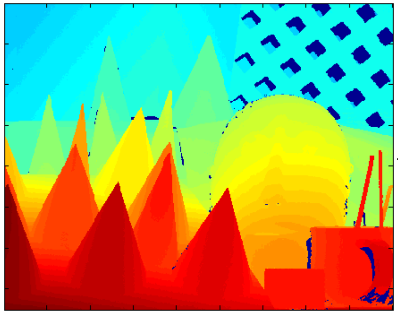
Cost Aggregation



Disparity Selection

Disparity images D(p), D(q)

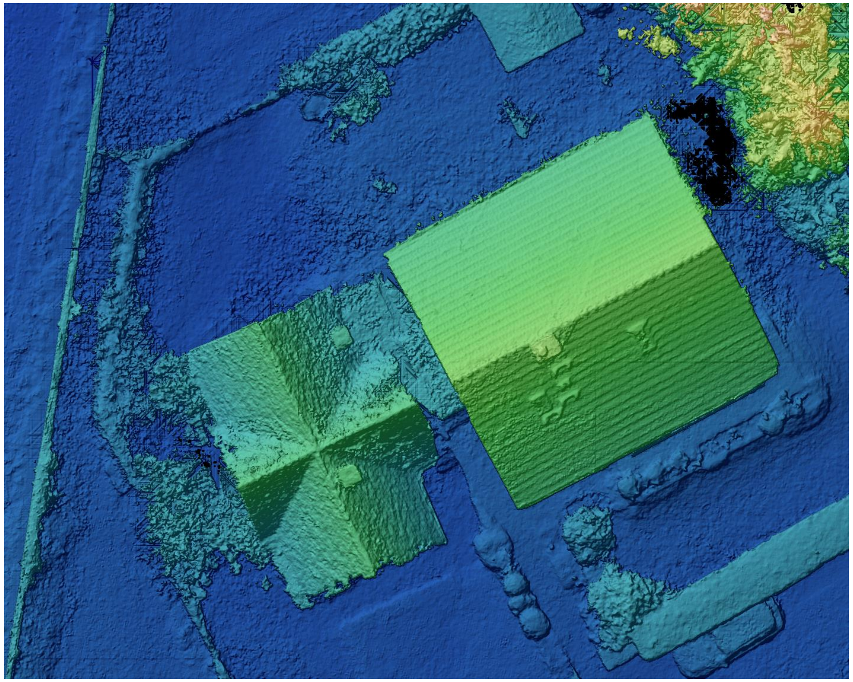
LR Check, Filtering



Digital Surface Model (DSM)

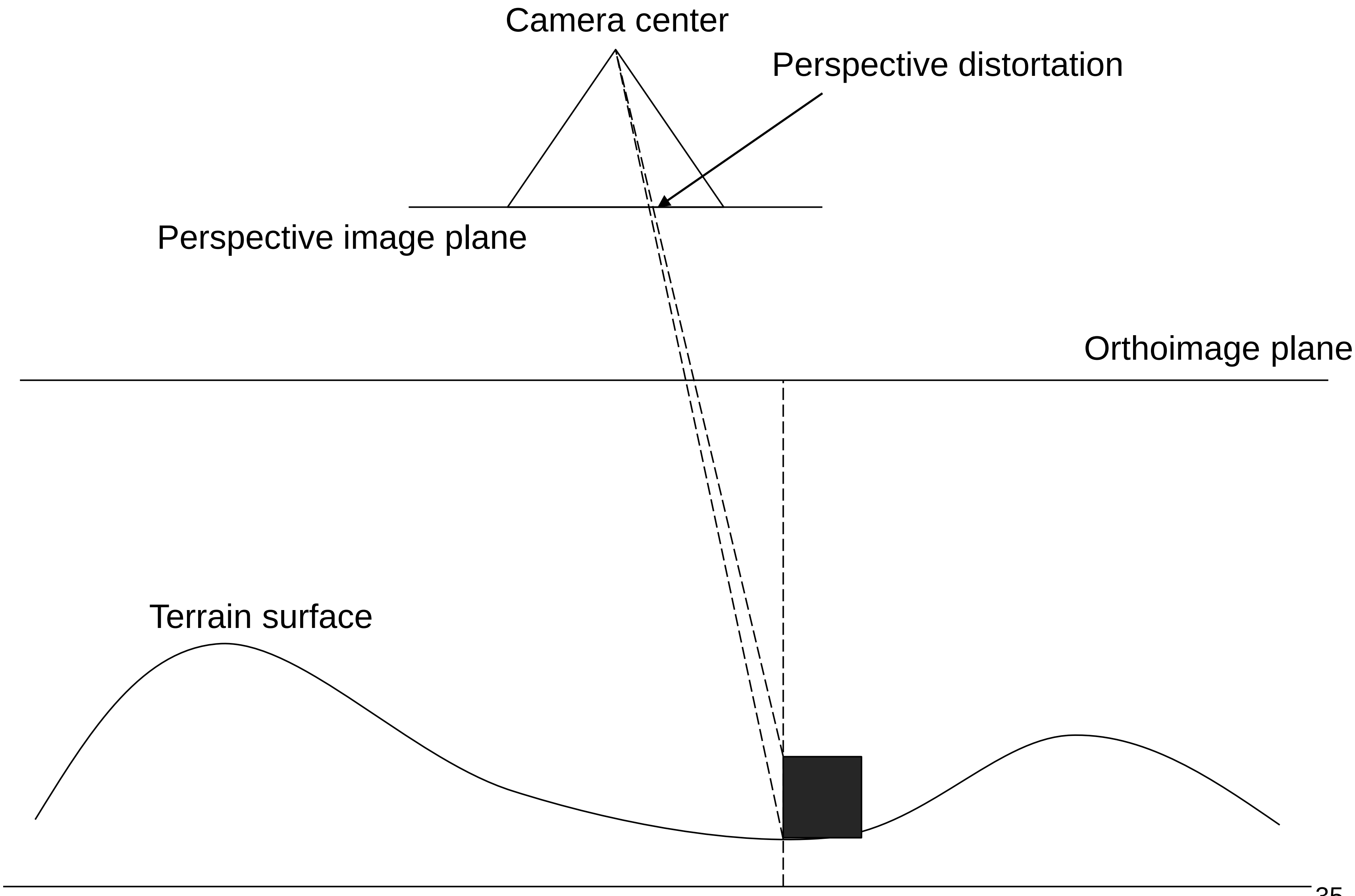
- 1 height value per ground location

original image

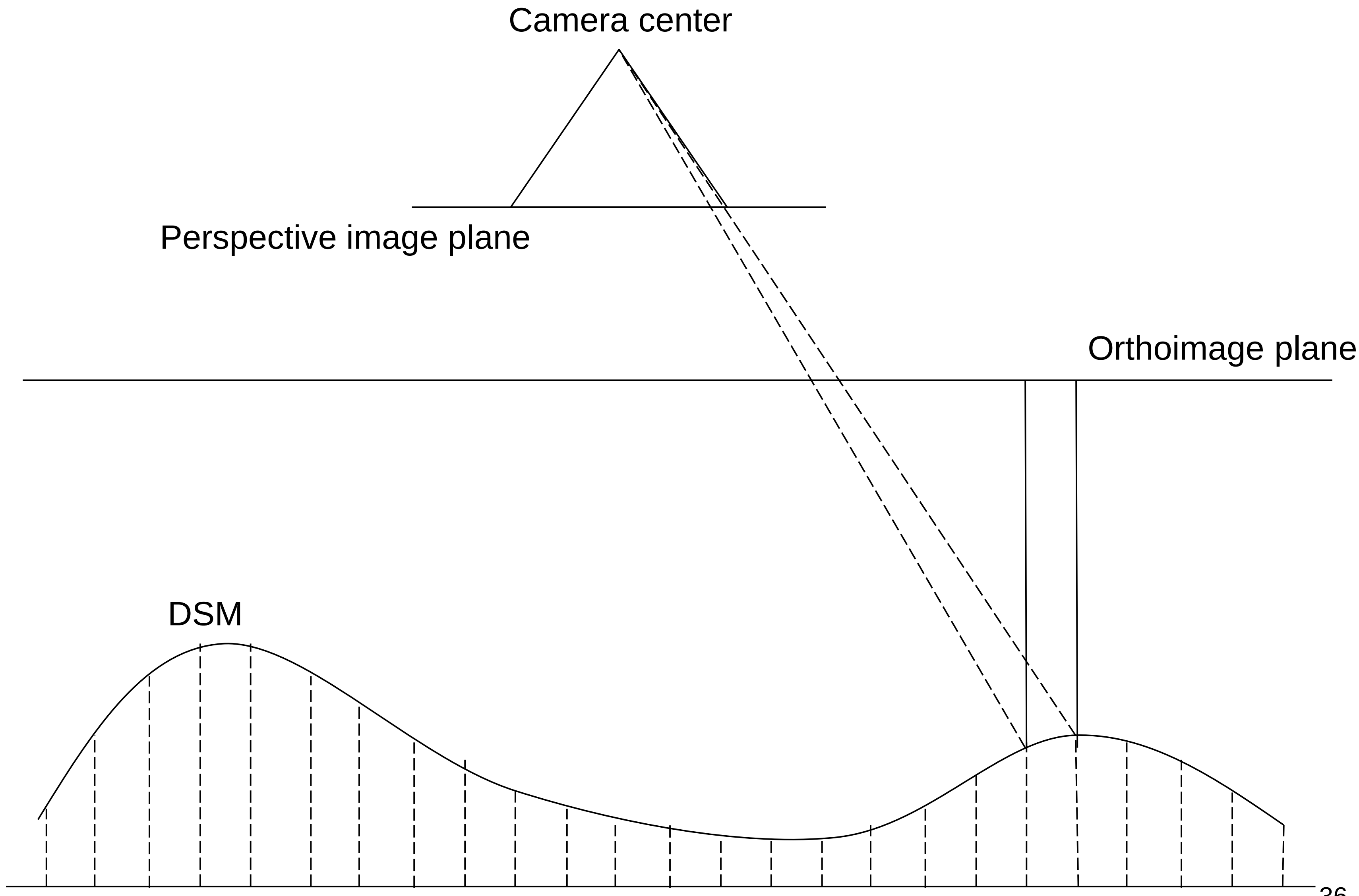


digital surface model
(color represents height)

Orthographic image projection



Orthophoto generation



Orthophoto example

- A true orthophoto has all perspective effects removed

original image

true orthophoto