Imperial College London

SMART ROBOTICS LAB

01/03/2021

Lecture 7:
Vision-Based Simultaneous Localisation and Mapping

70003 Advanced Robotics

Dr Stefan Leutenegger

Teaching Assistants: Dr Masha Popovic,
Sotiris Papatheodorou, Binbin Xu, and Nils Funk

Linear Least Squares – Polynomial Fit

Given:

- N data points $\tilde{z}_i$ at t_i.
- Assumption that the data can be explained with a model $z = (a_2 t^2 + a_1 t + a_0)$.

Find:

$\mathbf{x}^* = [a_0^*, a_1^*, a_2^*]^T$ minimising the sum of square differences

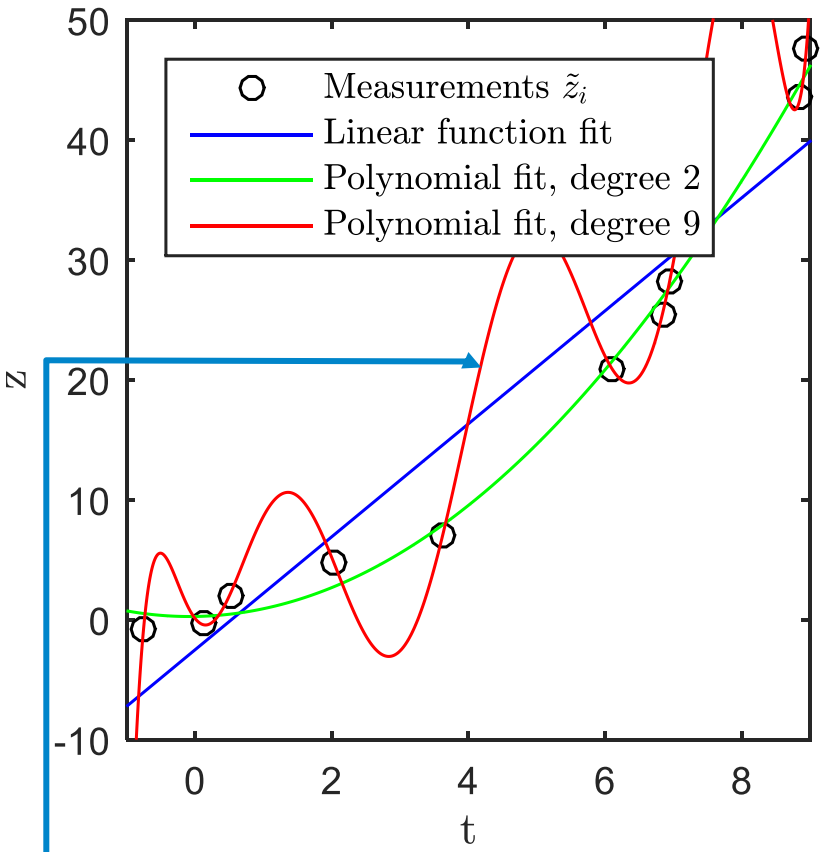
$$c(\mathbf{x}) = \frac{1}{2} \sum_{i=1}^{N} \| \tilde{z}_i - (a_2 t_i^2 + a_1 t_i + a_0) \|_2^2.$$

Solution:

Analogous to linear function fit, but with

$$\mathbf{H} = \begin{bmatrix} 1 & t_1 & t_1^2 \\ \vdots & \vdots & \vdots \\ 1 & t_N & t_N^2 \end{bmatrix}.$$

Note: This is a nonlinear function to fit, but still a **linear** regression problem!



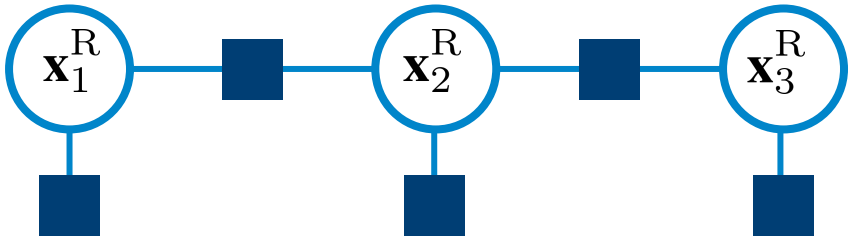
Overfitting: the problem of explaining too little (noisy) data with too many parameters…

Imperial College London

SMART ROBOTICS LAB

Factor Graph – Differential Drive Robot Example

- All states of the trajectory are being estimated



Measurements: error terms

Measurements are of the form $p(\mathbf{z}_i|\mathbf{x}_j)$.

This will allow us to formulate measurement errors $\mathbf{e}_i = \tilde{\mathbf{z}}_i - \mathbf{f}(\mathbf{x}_j)$.

The Probabilistic View on Least Squares

Now assume:

- Gaussian measurement likelihood $p(\mathbf{z}_i|\mathbf{x}) = \mathcal{N}(\mathbf{h}_i(\mathbf{x}), \mathbf{R}_i)$.
- Optionally: Gaussian prior $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}_0, \mathbf{P})$.

Find: $\mathbf{x}^* = \operatorname{argmax} p(\mathbf{x}|\mathbf{z}) = \operatorname{argmin}(-\log(p(\mathbf{x}|\mathbf{z})))$

ML Estimator: minimise the weighted sum of least squares

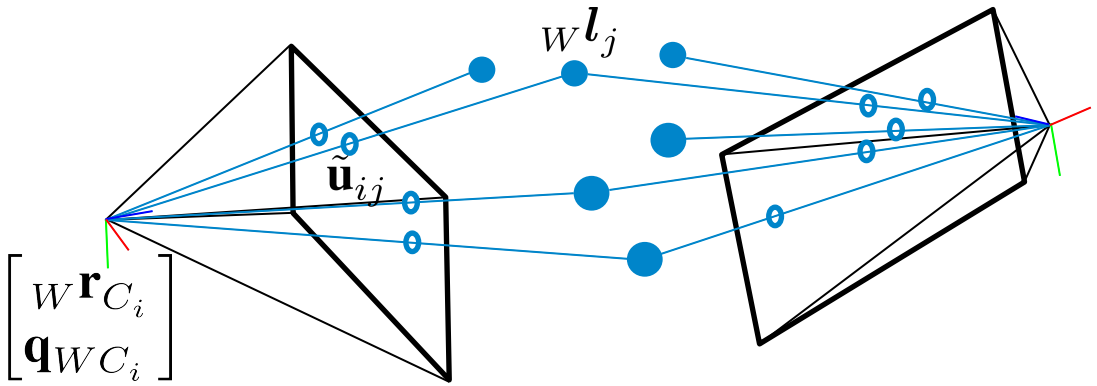
$$\begin{aligned}\mathbf{x}^* &= \operatorname{argmin}(-\log(p(\mathbf{x}|\mathbf{z}))) = \operatorname{argmin}(-\log(p(\mathbf{z}|\mathbf{x}))) \\ &= \operatorname{argmin}\left(-\log\left(\prod_{i=1}^{N} p(\mathbf{z}_i|\mathbf{x})\right)\right) = \operatorname{argmin}\left(-\log\left(\prod_{i=1}^{N} \exp\left(-\frac{1}{2}(\tilde{\mathbf{z}}_i - \mathbf{h}_i(\mathbf{x}))^{\mathrm{T}} \mathbf{R}_i^{-1} (\tilde{\mathbf{z}}_i - \mathbf{h}_i(\mathbf{x}))\right)\right)\right) \\ &= \operatorname{argmin} \frac{1}{2} \sum_{i=1}^{N} (\tilde{\mathbf{z}}_i - \mathbf{h}_i(\mathbf{x}))^{\mathrm{T}} \mathbf{R}_i^{-1} (\tilde{\mathbf{z}}_i - \mathbf{h}_i(\mathbf{x})).\end{aligned}$$

MAP Estimator: minimise the sum of least squares plus prior cost

$$\begin{aligned}\mathbf{x}^* &= \operatorname{argmin}(-\log(p(\mathbf{x}|\mathbf{z}))) = \operatorname{argmin}(-\log(p(\mathbf{z}|\mathbf{x})p(\mathbf{x}))) \\ &= \operatorname{argmin}\left(\frac{1}{2}(\mathbf{x}_0 - \mathbf{x})^{\mathrm{T}} \mathbf{P}^{-1} (\mathbf{x}_0 - \mathbf{x}) + \frac{1}{2} \sum_{i=1}^{N} (\tilde{\mathbf{z}}_i - \mathbf{h}_i(\mathbf{x}))^{\mathrm{T}} \mathbf{R}_i^{-1} (\tilde{\mathbf{z}}_i - \mathbf{h}_i(\mathbf{x}))\right).\end{aligned}$$

Bundle Adjustment: Visual SLAM as Optimisation Problem

Given: 2D detections $\tilde{\mathbf{u}}_{ij}$ of corresponding unknown 3D landmarks ${}_W\boldsymbol{l}_j$ in images $i = 1 \dots N$.



Find: 3D landmark positions ${}_W\boldsymbol{l}_j$ and 6D camera poses $[{}_W\mathbf{r}_{C_i}, \mathbf{q}_{WC_i}]^T$ where the images were taken.

Solution: minimise the reprojection error

$$\mathbf{e}_{ij}({}_W\mathbf{r}_{C_i}, \mathbf{q}_{WC_i}, {}_W\boldsymbol{l}_j) = \tilde{\mathbf{u}} - \mathbf{u}(\boldsymbol{T}_{C_iW}\ {}_W\boldsymbol{l}_j),$$

over all images

$$\min \sum_{i=1}^{N} \sum_{j \in \mathcal{J}(i)} \|\mathbf{e}_{ij}\|^2,$$

$\mathcal{J}(i)$: Set of landmark indices as visible in camera i.

applying Levenberg-Marquardt.

Jacobians needed for the Bundle Adjustment Problem

We will need **Jacobians** of $\mathbf{e}_{ij}({}_W\mathbf{r}_{C_i}, \mathbf{q}_{WC_i}, {}_W\mathbf{l}_j) = \tilde{\mathbf{u}} - \mathbf{u}(\boldsymbol{T}_{C_iW}\, {}_W\boldsymbol{l}_j)$.
Let's rewrite first: $\mathbf{e}_{ij} = \tilde{\mathbf{u}} - \mathbf{u}(\boldsymbol{T}_{WC_i}^{-1}\, {}_W\boldsymbol{l}_j) = \tilde{\mathbf{u}} - \mathbf{u}(\mathbf{C}_{WC_i}^T\, {}_W\mathbf{l}_j - \mathbf{C}_{WC_i}^T\, {}_W\mathbf{r}_{C_i})$.

We will employ the chain rule. So, we first need $\frac{\partial}{\partial {}_{C_i}\mathbf{l}}\mathbf{u}({}_{C_i}\mathbf{l}) = \frac{\partial}{\partial {}_{C_i}\mathbf{l}}\mathbf{k}(\mathbf{d}(\mathbf{p}({}_{C_i}\mathbf{l})))$.

$$\mathbf{U}({}_{C_i}\bar{\mathbf{l}}_j) := \frac{\partial}{\partial {}_{C_i}\mathbf{l}_j}\mathbf{u}({}_{C_i}\mathbf{l}_j) = \begin{bmatrix} f_u & 0 \\ 0 & f_v \end{bmatrix} \mathbf{D} \begin{bmatrix} \frac{1}{\bar{l}_3} & 0 & -\frac{\bar{l}_1}{\bar{l}_3^2} \\ 0 & \frac{1}{\bar{l}_3} & -\frac{\bar{l}_2}{\bar{l}_3^2} \end{bmatrix}$$

Jacobian of distortion

Now:

$$\frac{\partial {}_C\mathbf{l}_j}{\partial {}_W\mathbf{r}_{C_i}} = -\overline{\mathbf{C}}_{WC_i}^T, \quad \frac{\partial {}_C\mathbf{l}_j}{\partial \delta\boldsymbol{\alpha}_i} = \overline{\mathbf{C}}_{WC_i}^T [{}_W\bar{\mathbf{l}}_j - {}_W\bar{\mathbf{r}}_{C_i}]^\times, \quad \frac{\partial {}_C\mathbf{l}_j}{\partial {}_W\mathbf{l}_j} = \overline{\mathbf{C}}_{WC_i}^T.$$

Now we have everything to build the Gauss-Newton System of equations:

$$\underbrace{\sum_{ij} \mathbf{E}_{ij}(\bar{\mathbf{x}})^T \mathbf{R}_{ij}^{-1} \mathbf{E}_{ij}(\bar{\mathbf{x}})}_{=:\mathbf{A}} \Delta\mathbf{x} = \underbrace{-\sum_{ij} \mathbf{E}_{ij}(\bar{\mathbf{x}})^T \mathbf{R}_{ij}^{-1} \mathbf{e}_{ij}(\bar{\mathbf{x}})}_{=:\mathbf{b}},$$

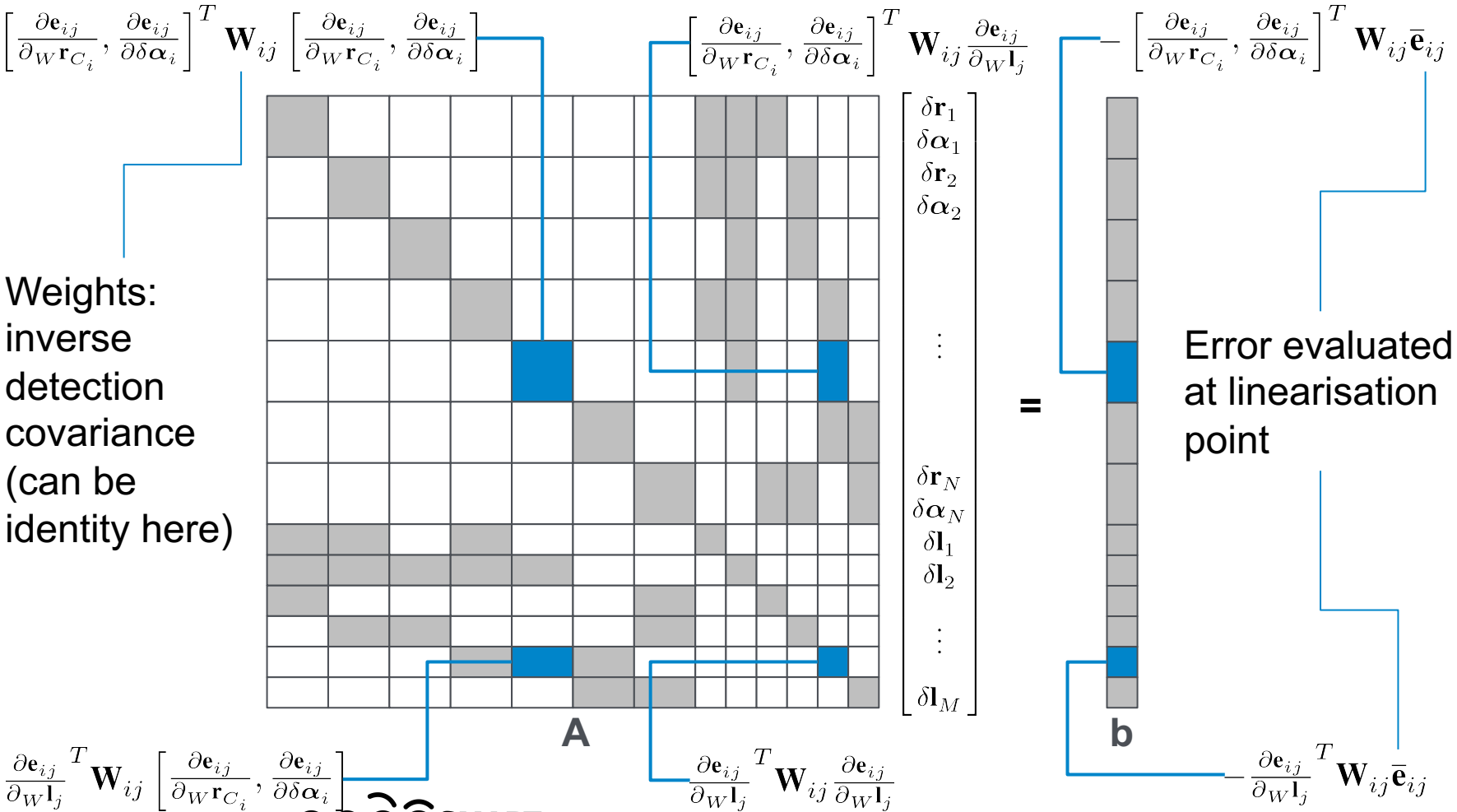
$\mathbf{E}_{ij}(\bar{\mathbf{x}}) = \frac{\partial}{\partial \mathbf{x}}\, \mathbf{e}_{ij}(\mathbf{x})\big|_{\bar{\mathbf{x}}}$.
Jacobian of reproj. error w.r.t. all the variables (lots of zeros)

Imperial College London

SMART ROBOTICS LAB

Structure of the Bundle Adjustment Problem

The **Gauss-Newton system of equations** smartly (additive per reproj. error):



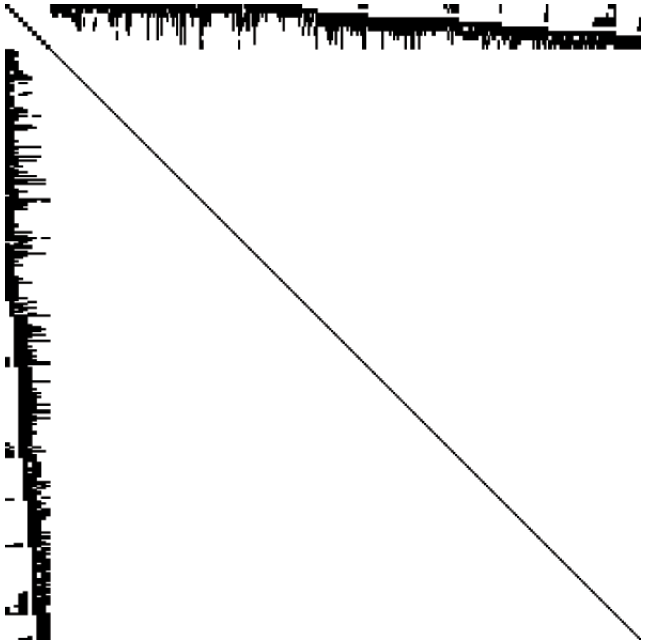
Imperial College London

SMART ROBOTICS LAB

Some Notes on Bundle Adjustment

- Typical problems contain many more points than poses
- Due to the sparsity of the linear equation system, it needs to be solved in a with some tricks, e.g.
  - Use Schur-Complement elimination of landmarks
  - Use sparse Cholesky composition
- In practise, **use an off-the-shelf solver** like
  - ceres: http://ceres-solver.org/
  - g2o: https://github.com/RainerKuemmerle/g2o

  This means, you will only have to implement the error function and Jacobians!



Example matrix A of a modestly sized problem. https://en.wikipedia.org/wiki/Bundle_adjustment

Batch Optimisation vs. Recursive Estimation (Filtering)

Batch Optimisation

- Estimate all states ever at once.
- No need for a temporal model, but can employ one.
- Enables re-linearization, leading to highest possible accuracy.
- However: when estimating a time-varying state, this means inserting variables for each time-step (i.e. the problem will grow with time)!

Recursive Estimation

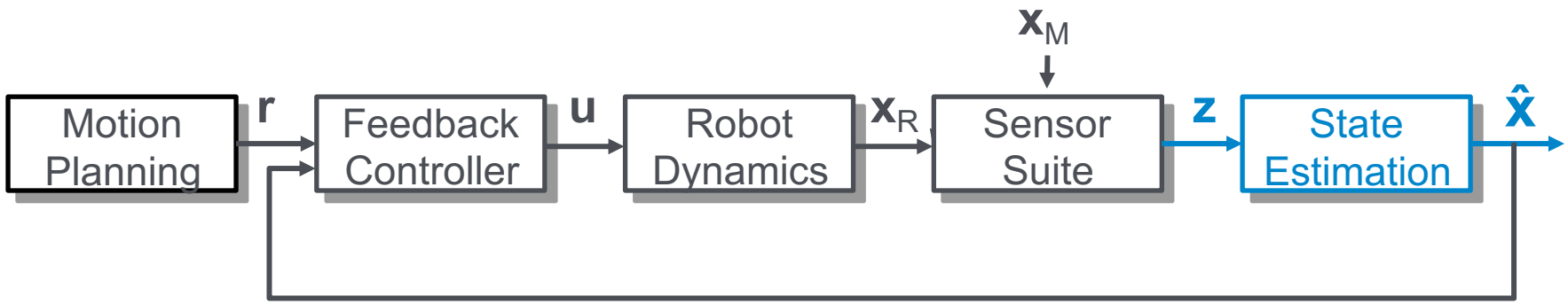
- Estimate only the current state and maintain some distribution.
- Needs a temporal model for prediction, which is alternated with measurement updates.
- Constant state vector size, constant complexity.
- Re-linearization for old state is impossible (reduced accuracy).
- Delayed measurements cannot be included.

Schedule: Lectures

What	Date	Topic
Lecture 1	18/01/21	Introduction, Problem Formulation, and Examples
Lecture 2	25/01/21	Representations and Sensors
Lecture 3	01/02/21	Kinematics and Temporal Models
Lecture 4	08/02/21	The Extended Kalman Filter
Lecture 5	15/02/21	Feedback Control
Lecture 6	22/02/21	Nonlinear Least Squares
Lecture 7	**01/03/21**	**Vision-based Simultaneous Localisation and Mapping**
Lecture 8	08/03/21	Revision, Q&A

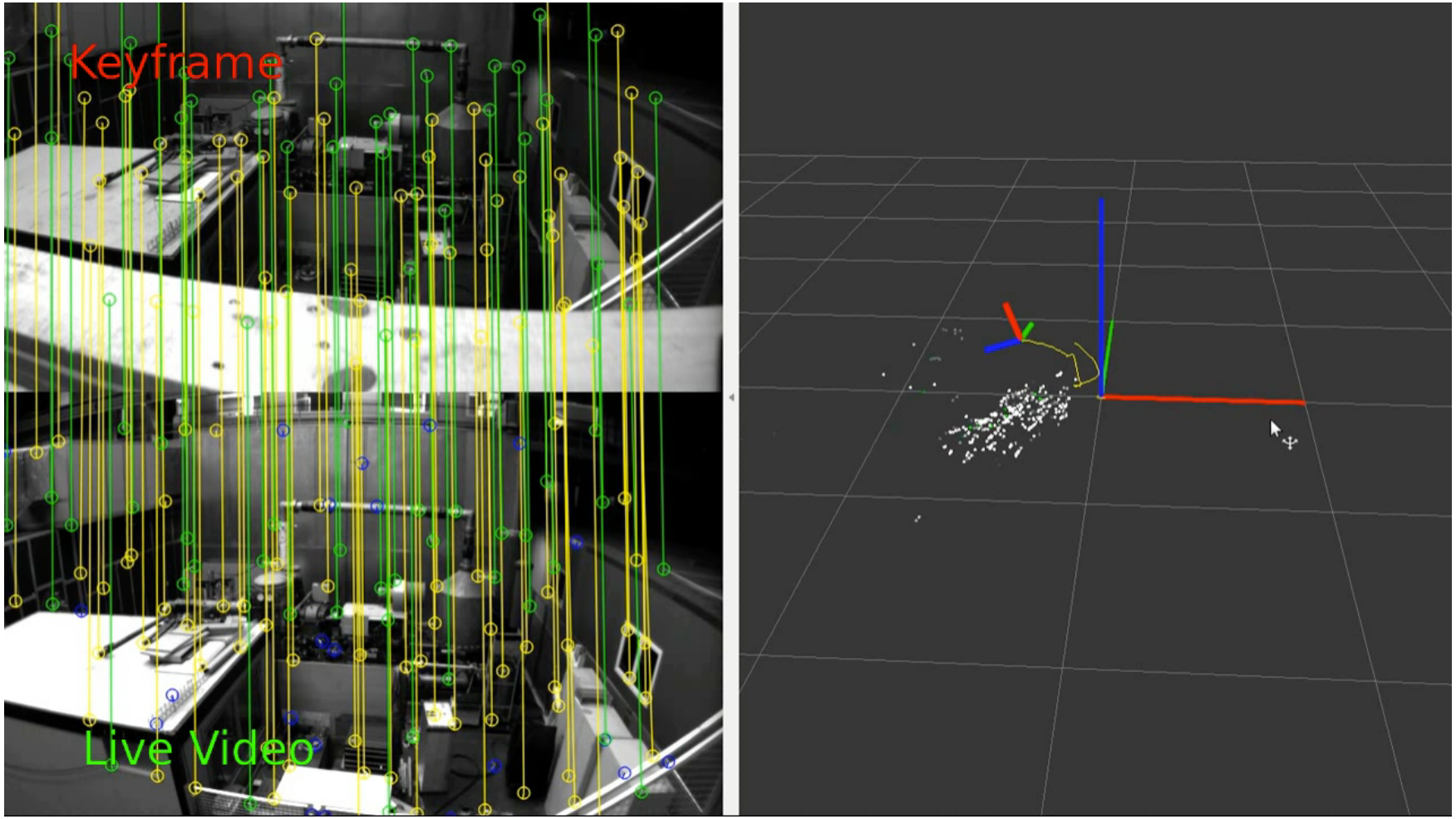
Today

1. Keypoint Detection, Description and Matching
2. Bootstrapping
3. Loop-Closure
4. Dense SLAM



OKVIS: Open Keyframe-based Visual-Inertial SLAM

Joint work with P. Furgale, M. Bosse, S. Lynen, M. Chli, V. Rabaud, K. Konolige, and R. Siegwart



https://www.youtube.com/watch?v=TbKEPA2_-m4

Keypoints: What Do We Want to Achieve?

Challenge 1:

Detector repeatability

Challenge 2:

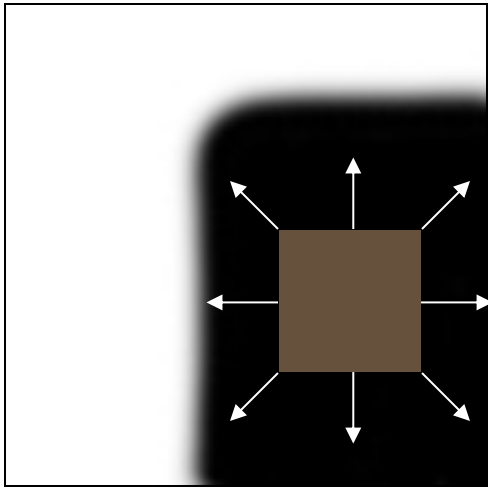
Descriptor quality

More Challenges:

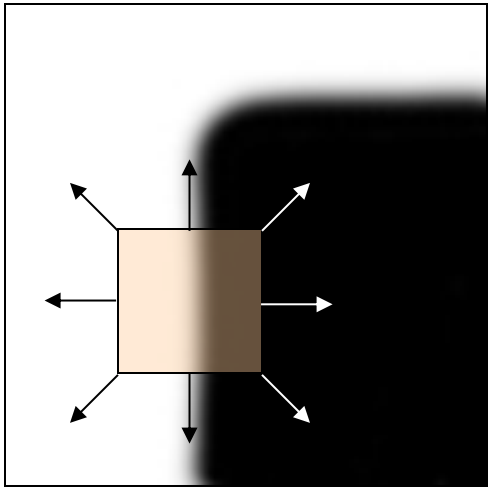
- Speed: Detection and description / Matching, database retrieval
- Invariance with respect to rotation and/or scale?
- Tolerance to brightness / contrast changes
- Distribution in the image

Identifying Corners

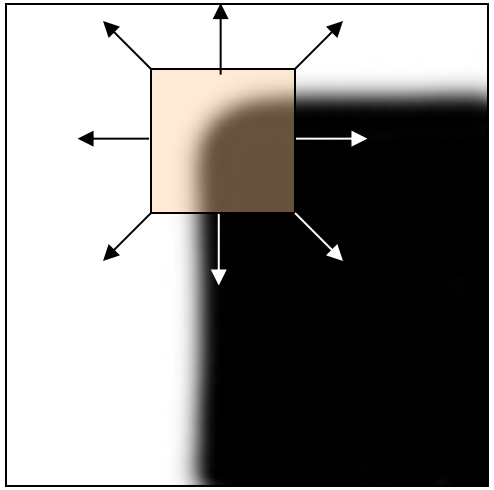
Shifting a window in **any direction** should give a **large change** in intensity in at least 2 directions



"**flat**" region:
no intensity change



"**edge**":
no change along the edge direction



"**corner**":
significant change in at least 2 directions

How Do We Implement This?

$$SSD(\Delta x, \Delta y) \approx \sum_{x,y \in P} \left(I_x(x,y)\Delta x + I_y(x,y)\Delta y \right)^2 = \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

Intensity gradient along y-direction

Intensity gradient along x-direction

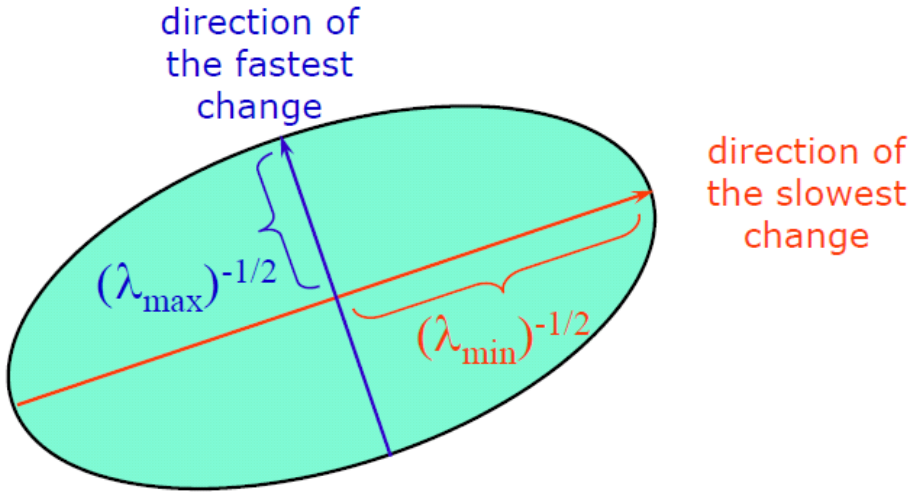
M is the "Second Moment Matrix"

$$M = \sum_{x,y \in P} \begin{bmatrix} I_x^{\,2} & I_x I_y \\ I_x I_y & I_y^{\,2} \end{bmatrix}$$

Since M is symmetric, we can rewrite M:

$$M = R^T \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} R$$

where λ_1 and λ_2 are the Eigenvalues of M.



Corner Response Function [Harris and Stephens, AVC88]

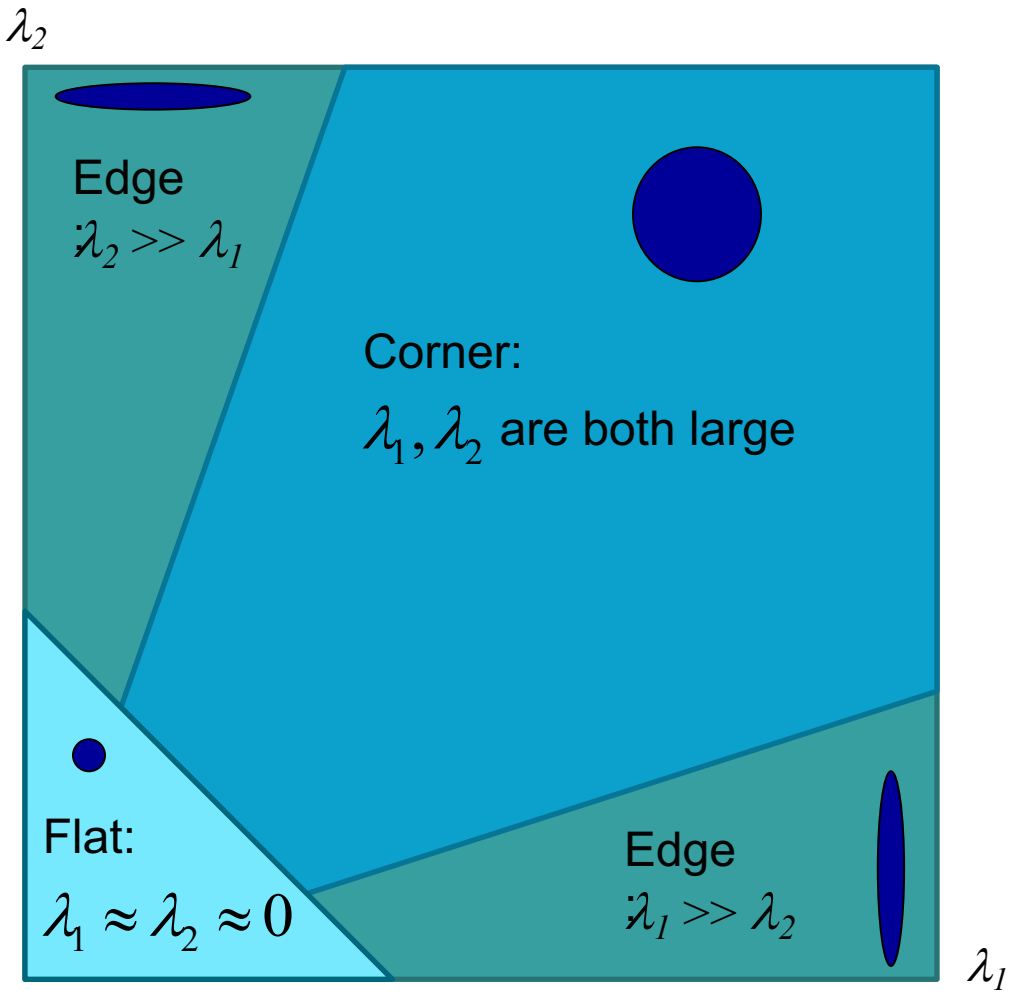
«Cornerness Function»

$$R = \det(M) - \kappa \,\mathrm{trace}(M)^2$$

$$= \lambda_1\lambda_2 - \kappa(\lambda_1 + \lambda_2)^2$$

Where κ is a constant between 0.04 and 0.15.

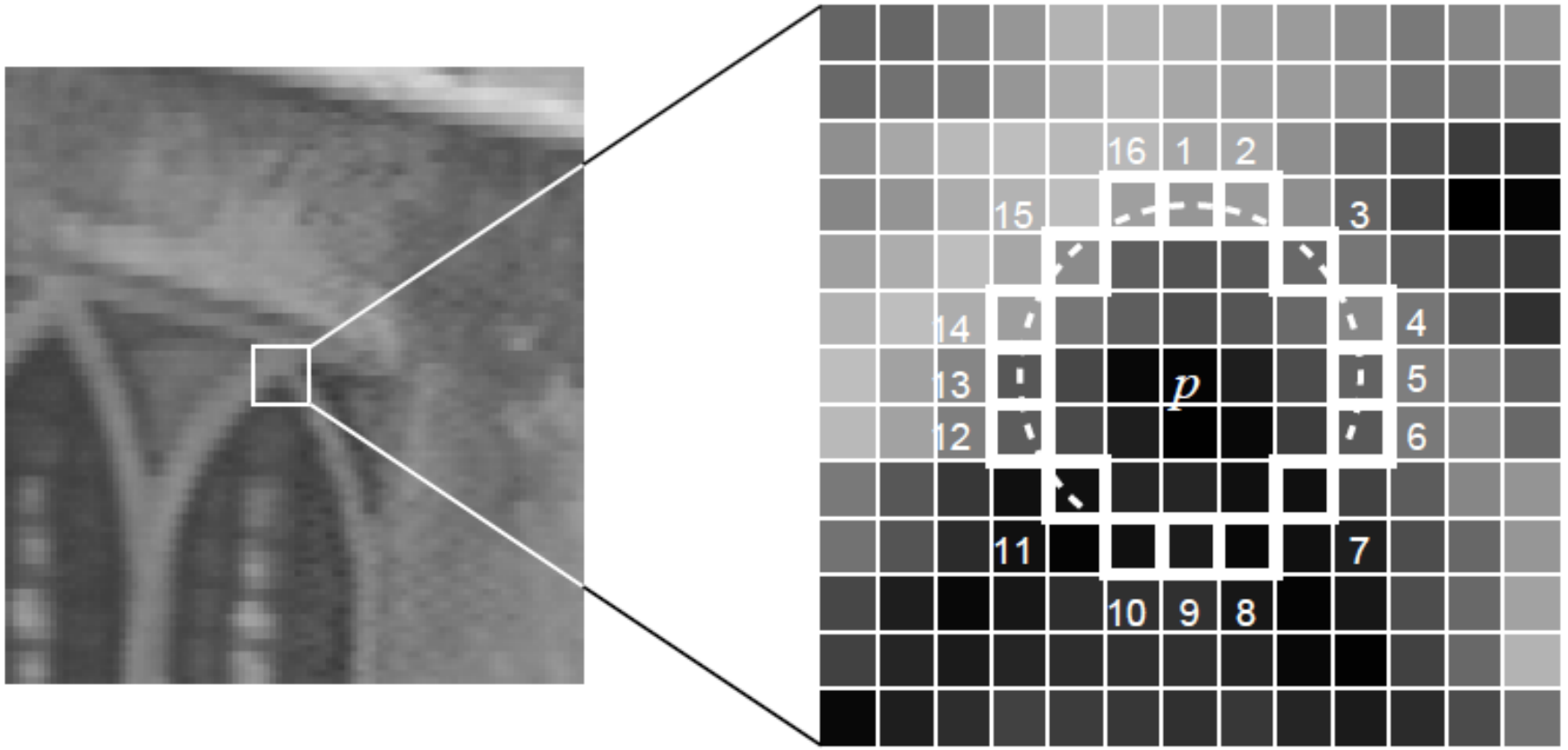
Note: this is invariant to rotation by definition.



FAST (-ER) Corners [Rosten et. al., PAMI2010]

Test: All points in circular segment brighter/darker by ***T*** than center (***p***)

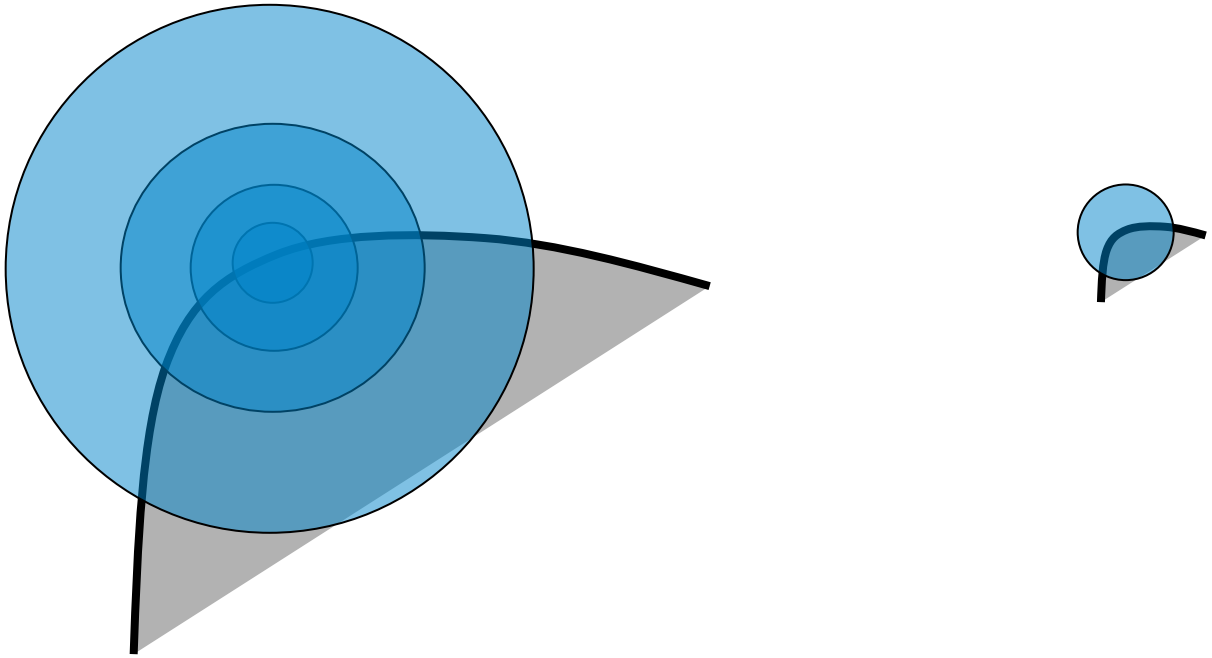
Very fast: in the order of 100 Mega-pixel/second



Scale Invariant Detection

Consider regions (e.g. circles) of different sizes around a point.

Corresponding regions will look the same in image space, when the appropriate scale-change is applied



Excursion: Spatial Filters

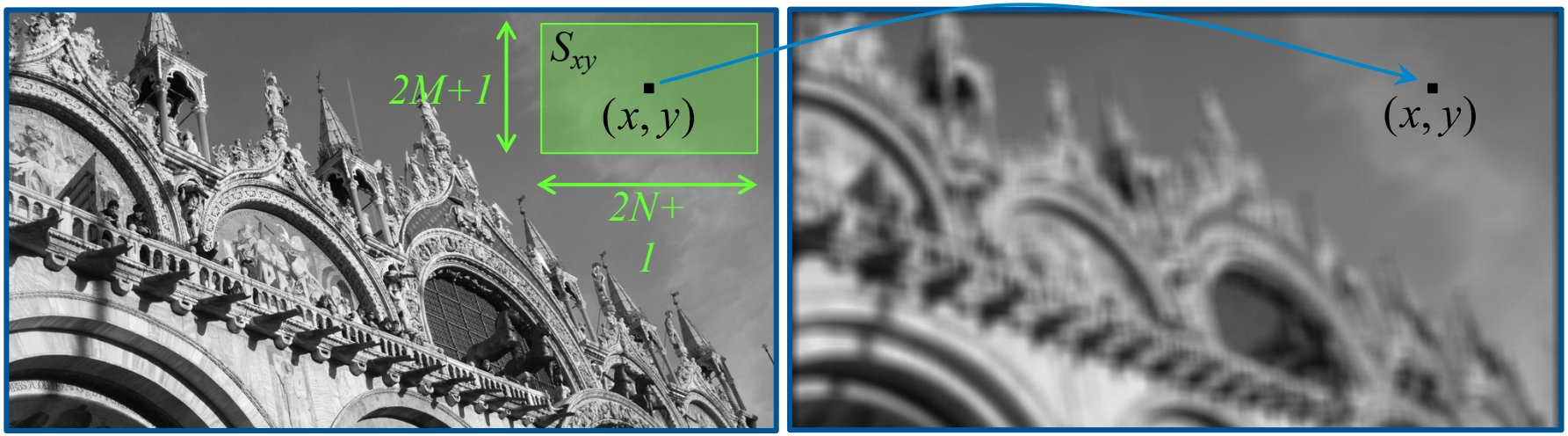


Image I

Filtered Image $J = F(I)$

$$J(x,y) = \frac{\sum_{(r,c)\in S_{xy}} I(r,c)}{(2M+1)(2N+1)}$$

Equivalent to **Convolution** of (windowed) **Kernel** *k* with the image:

$$J(x,y) = k(x,y) * I(x,y)$$

Blob Detection: LoG and DoG

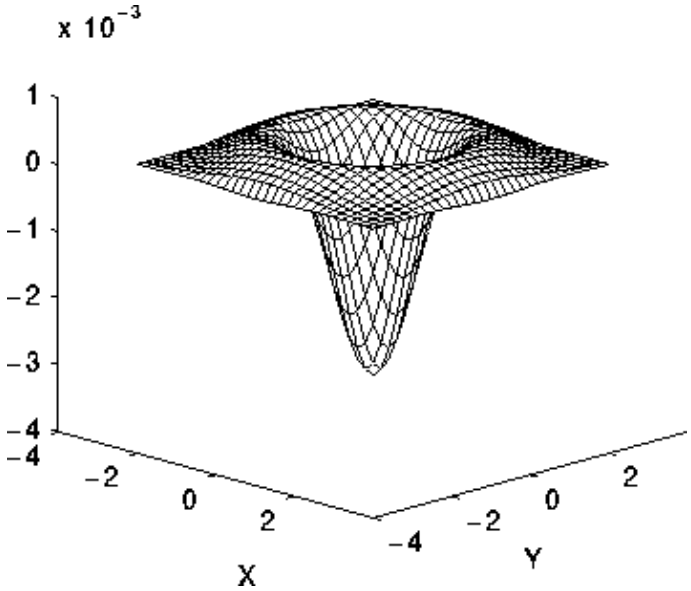
Scale(t)-normalized Laplacian of Gaussian (LoG):

– Image smoothed by a Gaussian kernel:

$$L = g(x, y, t) * I(x, y)$$

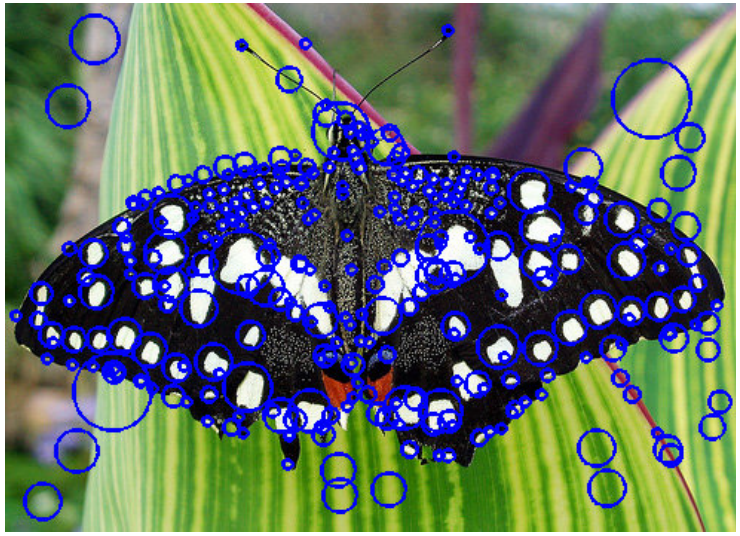
– Apply Laplacian operator:

$$\nabla^2_{\text{norm}} L = t\left(\frac{\partial^2 L}{\partial x^2} + \frac{\partial^2 L}{\partial y^2}\right)$$



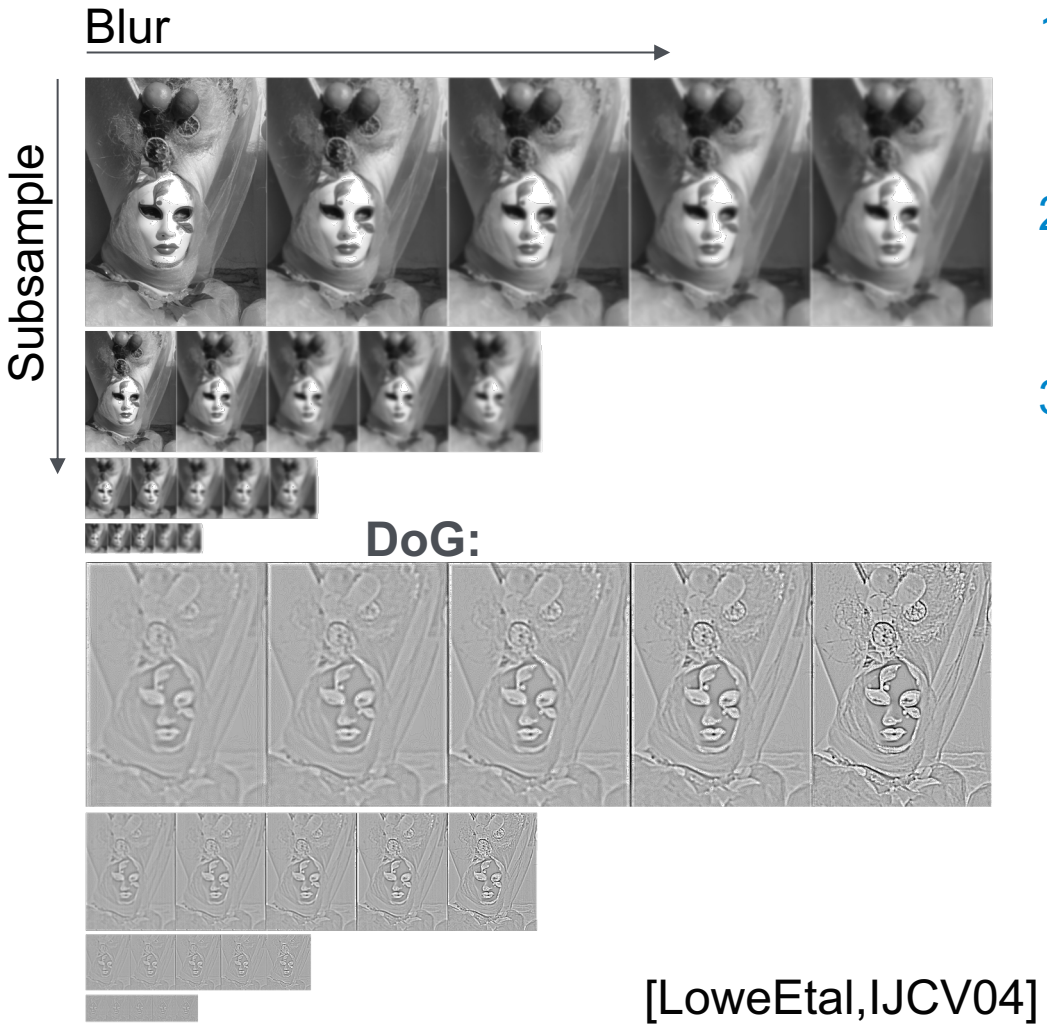
Approximation:
Difference of Gaussians (DoG)
SIFT detector [Lowe et.al., IJCV04]

$$\Delta L = L(x, y, t) - L(x, y, kt)$$



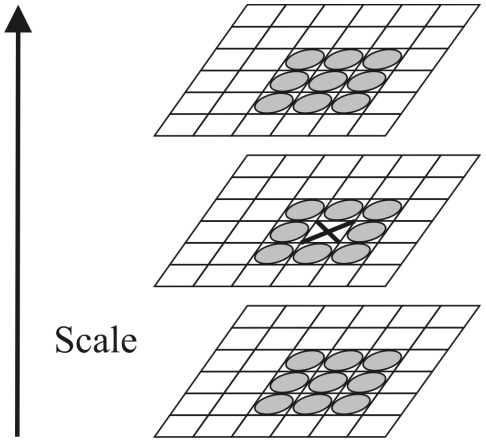
Micah Taylor, www.kixor.net, accessed June 2012

SIFT Detector (Keypoint Location + Scale)



[LoweEtal,IJCV04]

1. Scale-space pyramid: subsample and blur original image
2. Difference of Gaussians (DoG) pyramid: subtract successive smoothed images
3. Keypoints: local extrema in the DoG pyramid



SIFT: Orientation Assignment

1. Find ‘orientation’ of keypoint to achieve **rotation invariance**
2. Sample intensities around the keypoint
3. Compute a histogram of orientations of intensity gradients
4. Peaks in histogram: dominant orientations
5. Keypoint orientation = histogram peak
6. If there are multiple candidate peaks, construct a different keypoint for each such orientation

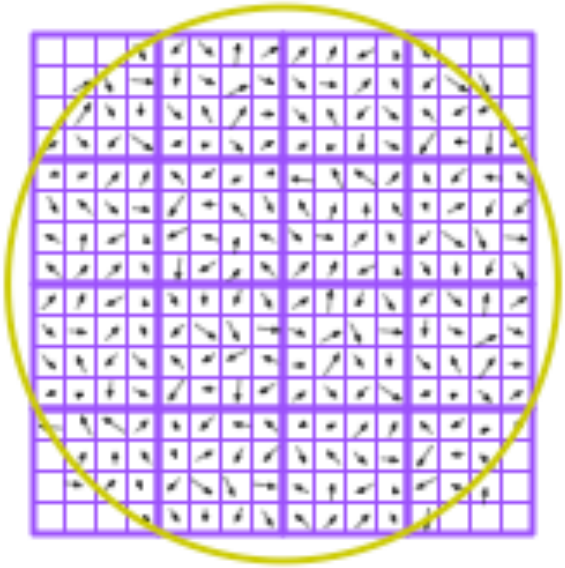
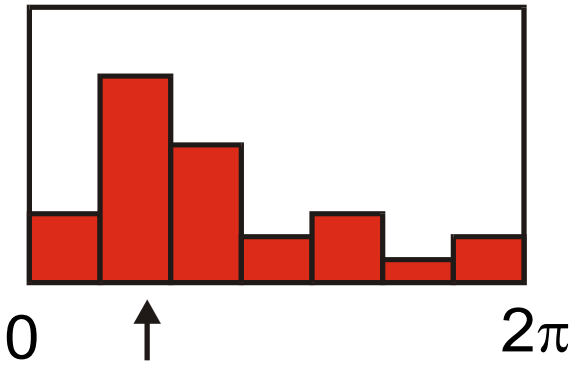


Image gradients



SIFT Descriptor

- We want it to be tolerant/invariant to certain changes, but yet very discriminative
- Describe gradient orientations relative to the keypoint orientation

- Divide keypoint neighborhood into 4×4 regions and compute orientation histograms along 8 directions
- SIFT descriptor: concatenation of all 4×4×8 (=128) values
- **Matching**: L_2-distance between these descriptor vectors

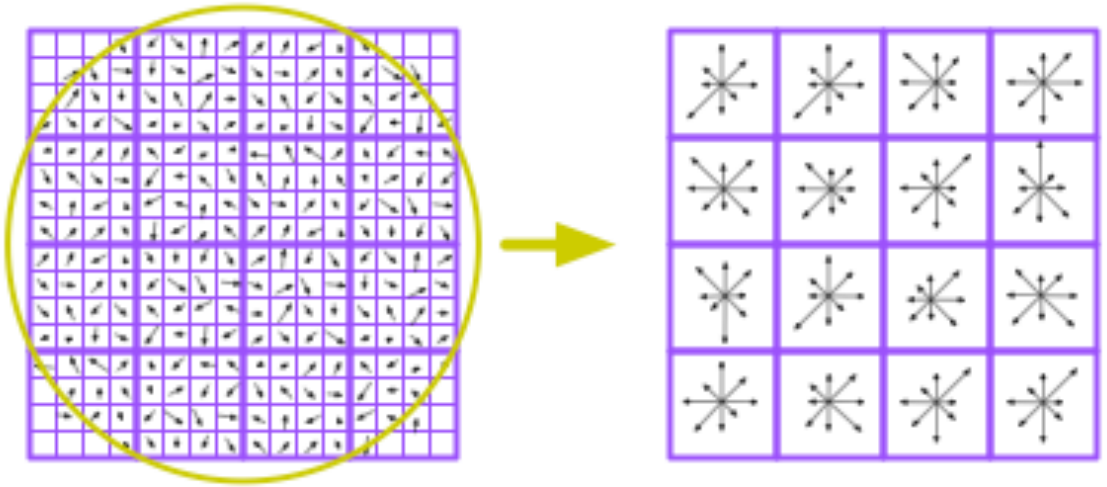
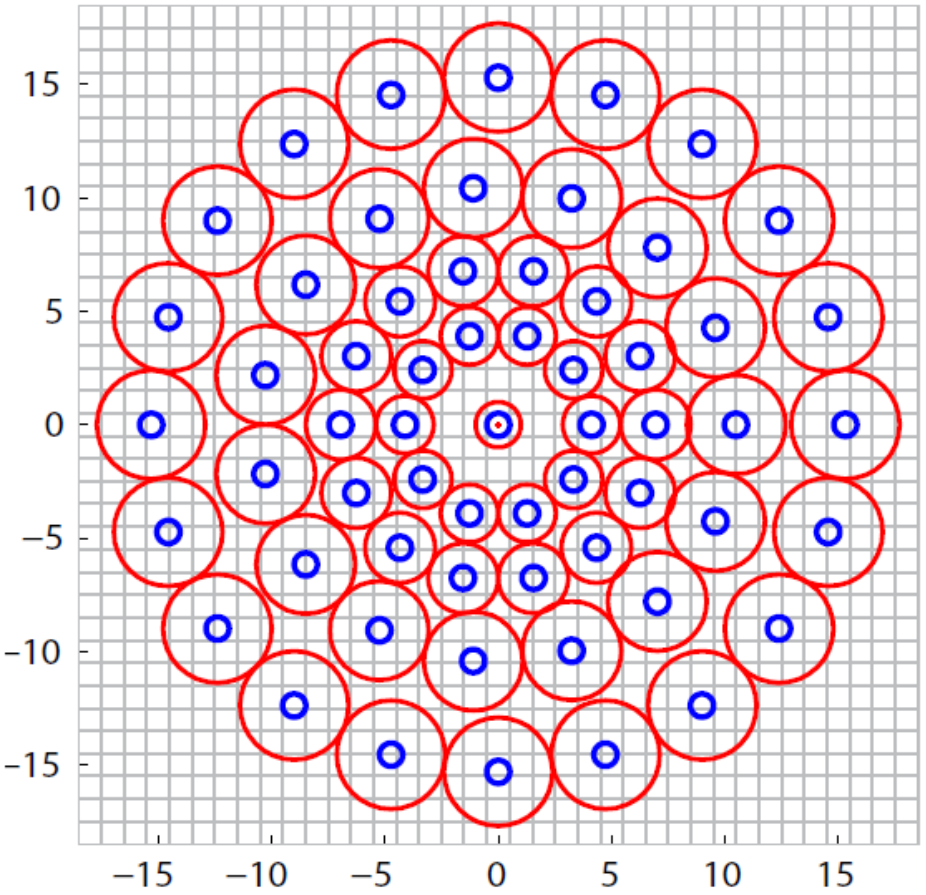


Image gradients

Keypoint descriptor

BRISK Descriptor [Leutenegger et. al ICCV 2011]

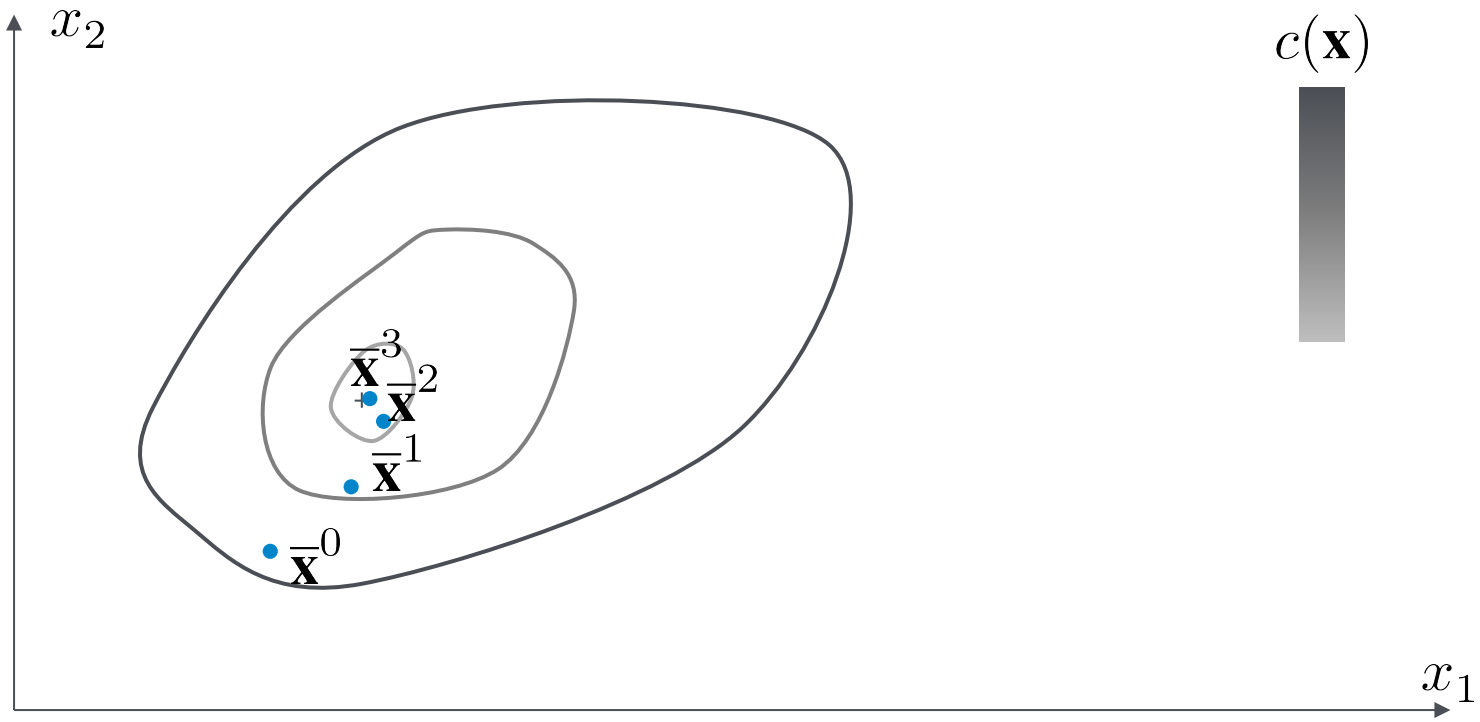
- Pattern used to access intensities in a keypoint neighborhood
- Red circles indicate the size of the smoothing kernel applied.
- Scaled and rotated versions stored in a look-up table
- Sampling point locations and smoothing is configurable
- Used for orientation assignment and brightness comparisons for **binary descriptor**
- Detection and descriptor speed: ≈10 times faster than SURF
- Ultra-fast **matching** using Hamming distance (xor descriptors and count ones)



BRISK sampling pattern

How To Solve Nonlinear Least Squares Problems

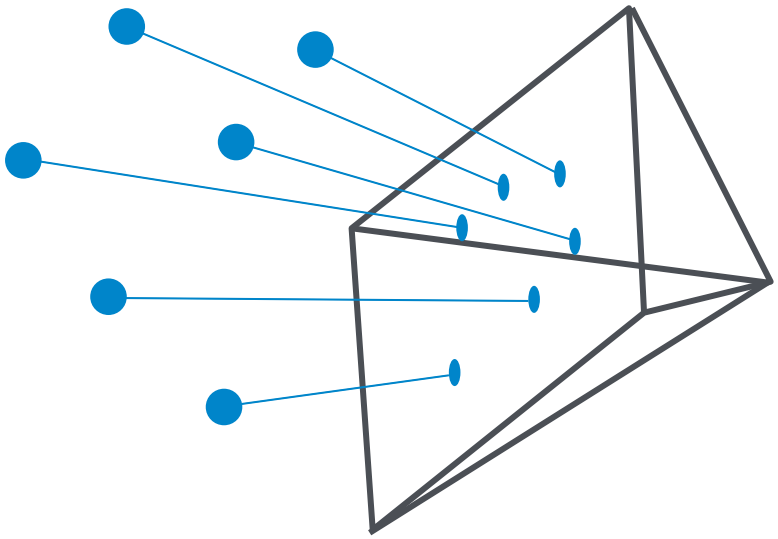
Iterative cost minimisation starting at an initial guess for $\mathbf{x}$:



But how do we find this initial guess?

3D-2D: The PnP Problem

Perspective n-Point Problem (also: Absolute Pose Problem): *n* 3D points are known and their 2D projections are observed in an image (known intrinsics)

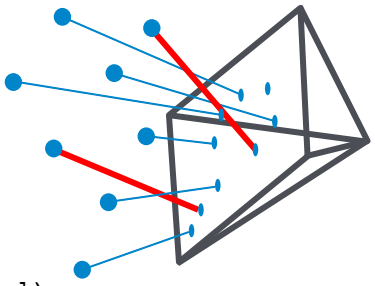


Find: Solution for the full camera pose **directly** (i.e. no iterative solving).

Solutions: different approaches for various values for *n*…

- State-of-the-art examples:
- UPnP (includes multi-camera setups) [L. Kneip, H. Li, Y. Seo, ECCV'14]
- EPnP [V. Lepetit, F. Moreno-Noguer, P. Fua IJCV'08]

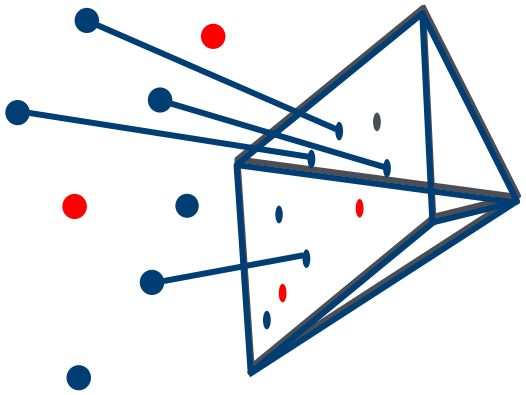
RANSAC: RANdom SAmpling Consensus



Problem: some amount of **wrong 3D-2D correspondences** called **outliers** may have a catastrophic effect PnP solution…

Find: Outliers and correct pose. **A Solution**: RANSAC.

```
function PnPRansac(points_3d, detections_2d, n, k, t, d)
besterr = inf
while iterations < k:
  maybeinliers <- select n points at random
  compute PnP solution (maybemodel) using maybeinliers
  for each point not in S:
    if point projection fits maybemodel with error < t:
      add point to alsoinliers
  if no. elements in alsoinliers > d
    inliers <- maybeinliers and alsoinliers
    compute bettermodel from inliers
    err <- mean reprojection error
    if err < besterr:
      bestfit = bettermodel
      besterr = thiserr
return [bestfit,inliers]
```



Stereo Triangulation

How do we initialise 3D points now that we know our camera pose?

Given: two corresponding 2D - points in 2 views (known poses), i.e. **two rays**.
The problem is that in practice, 3D lines won't intersect exactly…

Find: a good hypothesis for the respective 3D point **directly**.

A (fast) solution: Midpoint Method

1. Find the points along the ray with minimum distance to each-other:

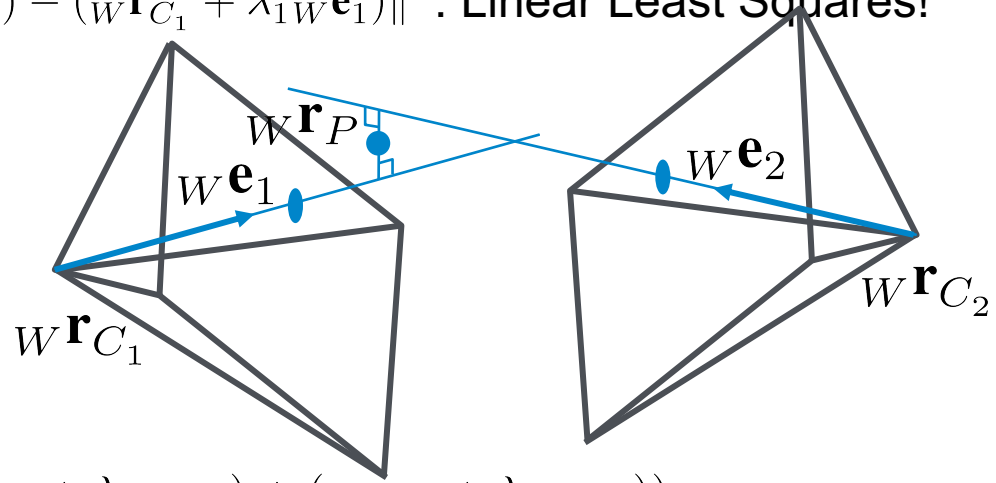
   $\boldsymbol{\lambda} = \begin{bmatrix}\lambda_1 \\ \lambda_2\end{bmatrix} = \operatorname{argmin} \left\| ({}_W\mathbf{r}_{C_2} + \lambda_2 {}_W\mathbf{e}_2) - ({}_W\mathbf{r}_{C_1} + \lambda_1 {}_W\mathbf{e}_1) \right\|^2$: Linear Least Squares!

   Solve the Normal Equations:

   $\mathbf{A}\boldsymbol{\lambda} = \mathbf{b},$

   $$\mathbf{A} = \begin{bmatrix} 1 & -{}_W\mathbf{e}_1^T \, {}_W\mathbf{e}_2 \\ -{}_W\mathbf{e}_1^T \, {}_W\mathbf{e}_2 & 1 \end{bmatrix},$$

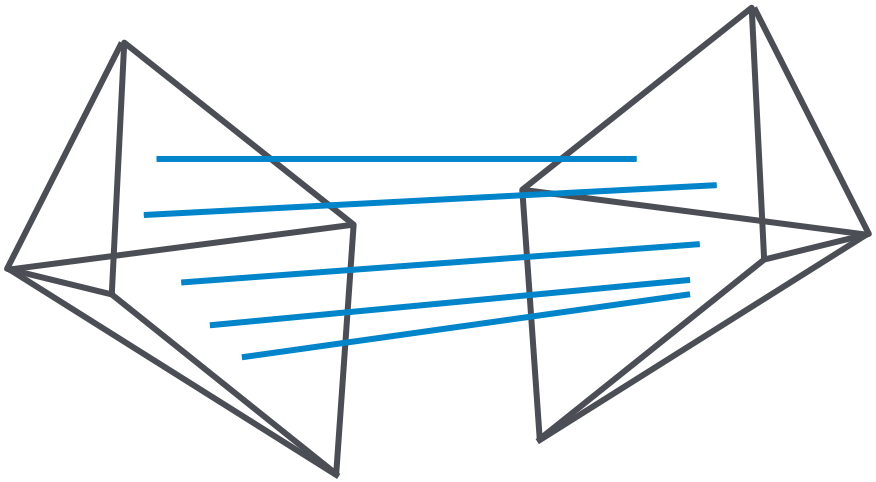
   $$\mathbf{b} = \begin{bmatrix} \mathbf{e}_1^T ({}_W\mathbf{r}_{C_2} - {}_W\mathbf{r}_{C_1}) \\ -\mathbf{e}_2^T ({}_W\mathbf{r}_{C_2} - {}_W\mathbf{r}_{C_1}) \end{bmatrix}.$$



2. Pick the mid-point ${}_W\mathbf{r}_P = \frac{1}{2}(({}_W\mathbf{r}_{C_2} + \lambda_2 {}_W\mathbf{e}_2) + ({}_W\mathbf{r}_{C_1} + \lambda_1 {}_W\mathbf{e}_1))$.

2D-2D RANSAC

How about if we don't know any 3D points (e.g. monocular SLAM initialisation)?



A great library to that implements all those 3D-2D and 2D-2D algorithms (with RANSAC):
http://laurentkneip.github.io/opengv/

[9] L. Kneip, P. Furgale, "OpenGV: A unified and generalized approach to real-time calibrated geometric vision", Proc. of The IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China. May 2014

Various algorithms to be used with RANSAC:

- 5-point algorithm by Stewenius
- 5-point algorithm by Nister
- 5-point algorithm to solve for rotations directly by Kneip
- 6-point algorithm by Henrik Stewenius for generalised relative pose
- 7-point algorithm
- 8-point algorithm by Longuet-Higgins
- 17-point algorithm by Hongdong Li

Place Recognition

How should we find out that we see something previously seen?

A popular approach: **Bag of Visual Words**.

Training

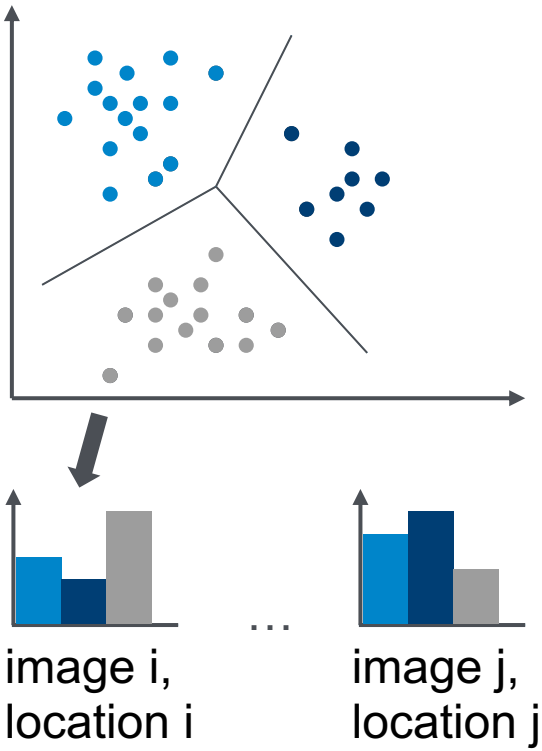
1. Build a **Vocabulary** from training descriptors by clustering close descriptors into «words».

Runtime – for each keyframe:

1. Detect **keypoints** and extract **descriptors**.
2. Build **histogram of visual words**, and perform some normalisation/re-weighting
3. Database **query**: compute histogram similiarity score for each location in the database.
4. If it you find a match:

   Apply individual **keypoints matching** and **verify** geometrically using some RANSAC which also gives you the **relative pose**.

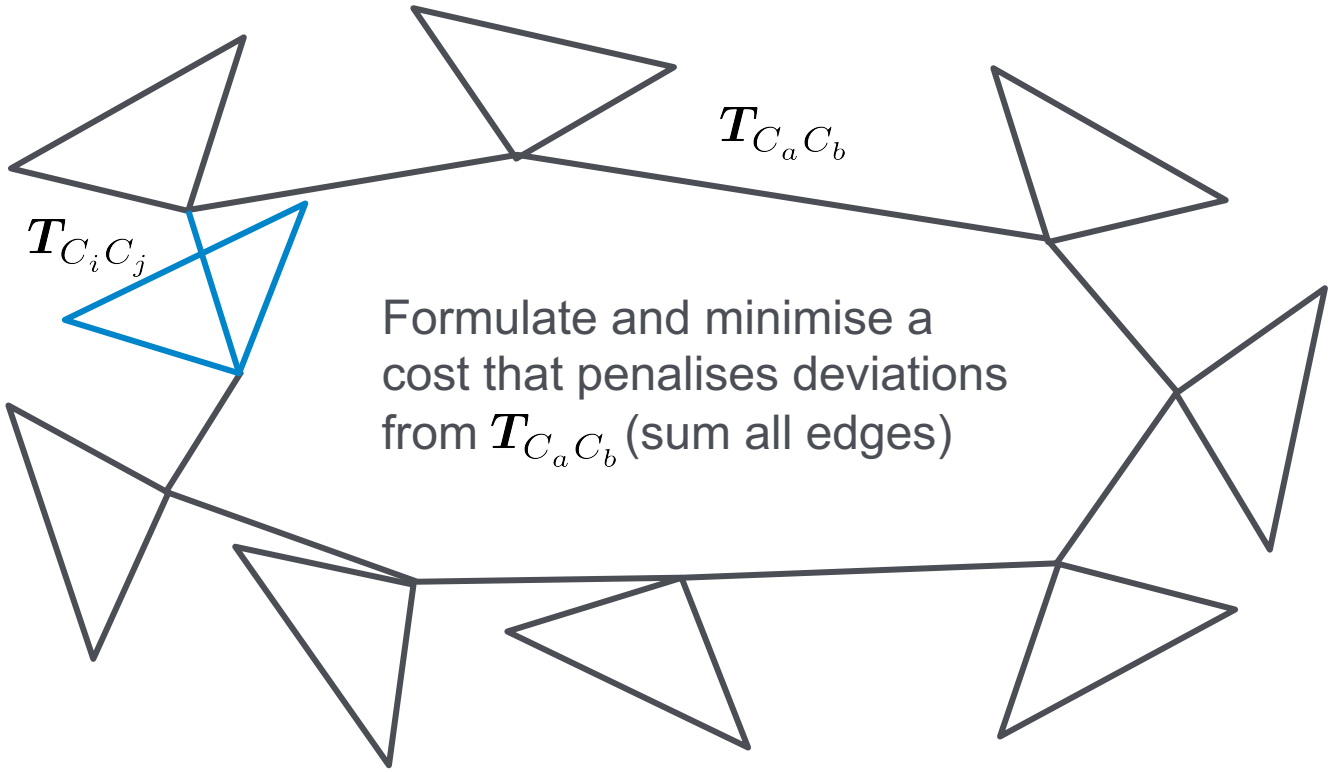
   Else insert the place into the database.



Pose Graph Optimisation

If trajectory / map needs to stay consistent after loop closure: bend it into place.

Example: Pose Graph Optimisation (Relaxation)



Dense SLAM Overview

Depth Cameras

Input: RGB image + Depth image

Principles to obtain depth image:

- Project pattern from baseline
- Passive stereo
- Time of Flight (ToF)

Typical SLAM algorithm:

1. **Track**: align RGB (and D) image with map to compute 6D pose.
2. **Map**: fuse new RGB-D into map.

Mono Cameras

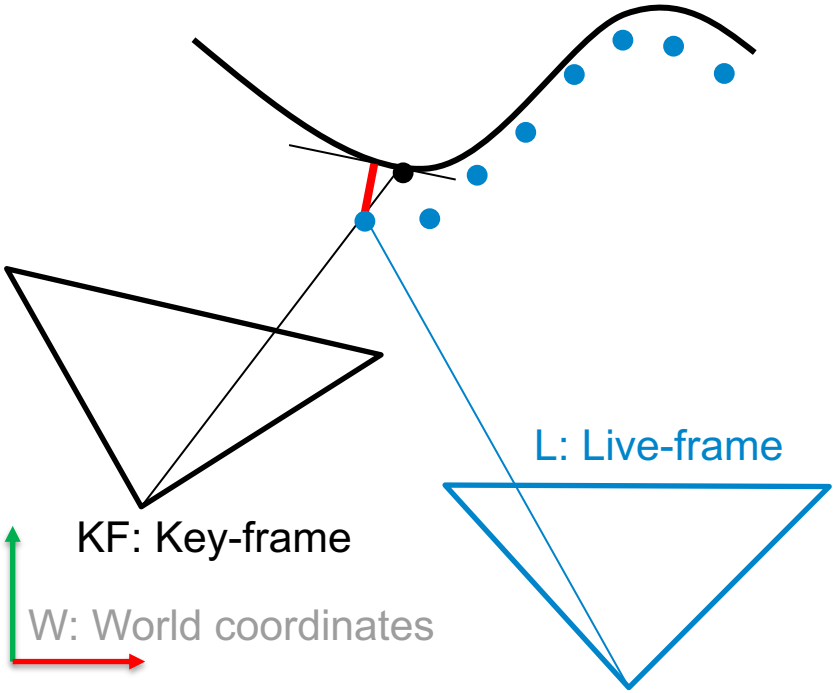
Input: one image at the time only!

Typical SLAM algorithm:

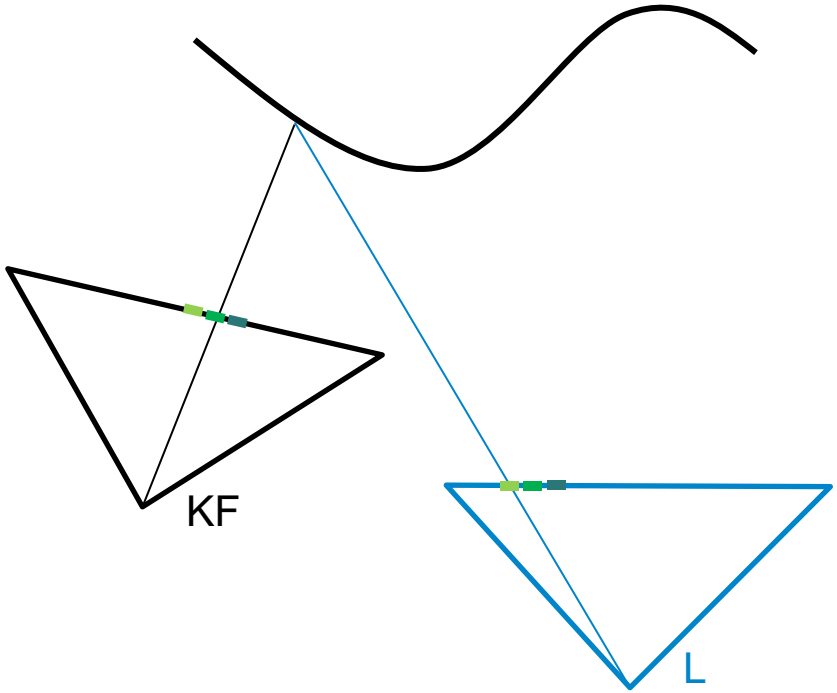
1. **Track**: align image with map to compute 6D pose.
2. **Map**: fuse new data.
   Problem: we don't know depth…
   Options:
   - Compute depth maps from successive poses and then do the same as in RGB-D SLAM
   - Optimise the existing map directly to match the image(s)

Dense SLAM: Tracking

ICP error (geometric distance):



Photometric error (colour difference):



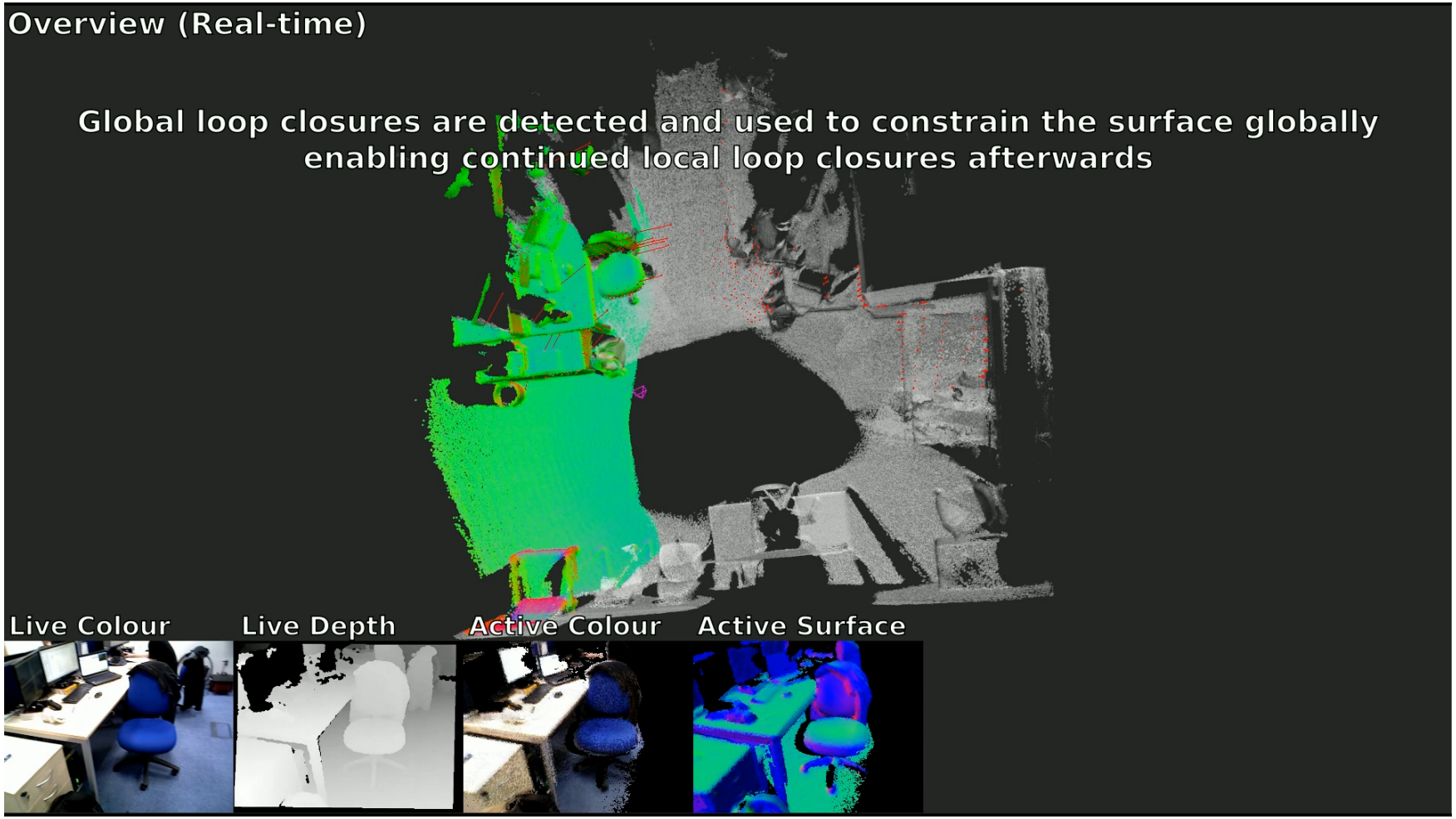
$$\mathbf{u}_{\mathrm{KF}} = \pi(\boldsymbol{T}_{WC_{\mathrm{KF}}}^{-1}\, \boldsymbol{T}_{WC_{\mathrm{L}}}\, (\pi^{-1}(\mathbf{u}_{\mathrm{L}}, D_{\mathrm{L}}[\mathbf{u}_{\mathrm{L}}])))$$

$$e_{\mathrm{ICP}} = {}_{W}\mathbf{n}_{\mathrm{KF}}[\mathbf{u}_{\mathrm{KF}}] \cdot (\boldsymbol{T}_{WC_{\mathrm{L}}}\, {}_{C_{\mathrm{L}}}\mathbf{v}_{\mathrm{L}}[\mathbf{u}_{\mathrm{L}}] - {}_{W}\mathbf{v}_{\mathrm{KF}}[\mathbf{u}_{\mathrm{KF}}])$$

$$\mathbf{u}_{\mathrm{L}} = \pi(\boldsymbol{T}_{WC_{\mathrm{L}}}^{-1}\, \boldsymbol{T}_{WC_{\mathrm{KF}}}\, (\pi^{-1}(\mathbf{u}_{\mathrm{KF}}, D_{\mathrm{KF}}[\mathbf{u}_{\mathrm{KF}}])))$$

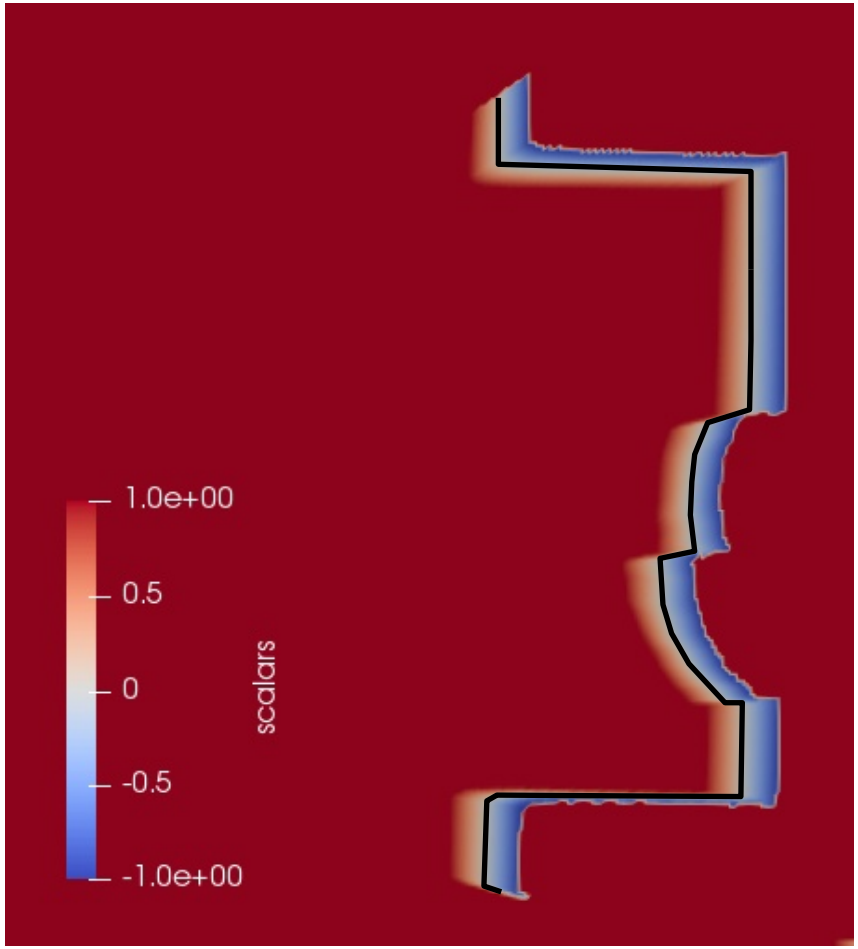
$$e_{\mathrm{p}} = I_{\mathrm{KF}}[\mathbf{u}_{\mathrm{KF}}] - I_{\mathrm{L}}[\mathbf{u}_{\mathrm{L}}]$$

ElasticFusion (Again)



Dense Volumetric Mapping

- Space represented as **truncated signed distance field (TSDF)** or Log-Odds **occupancy values**
- **Surfaces** represented implicitly as **zero-crossings**
- Volume may be encoded in an **octree** to exploit environment's sparsity.



Any Questions?

See you on Thursday at 1pm for the practical.