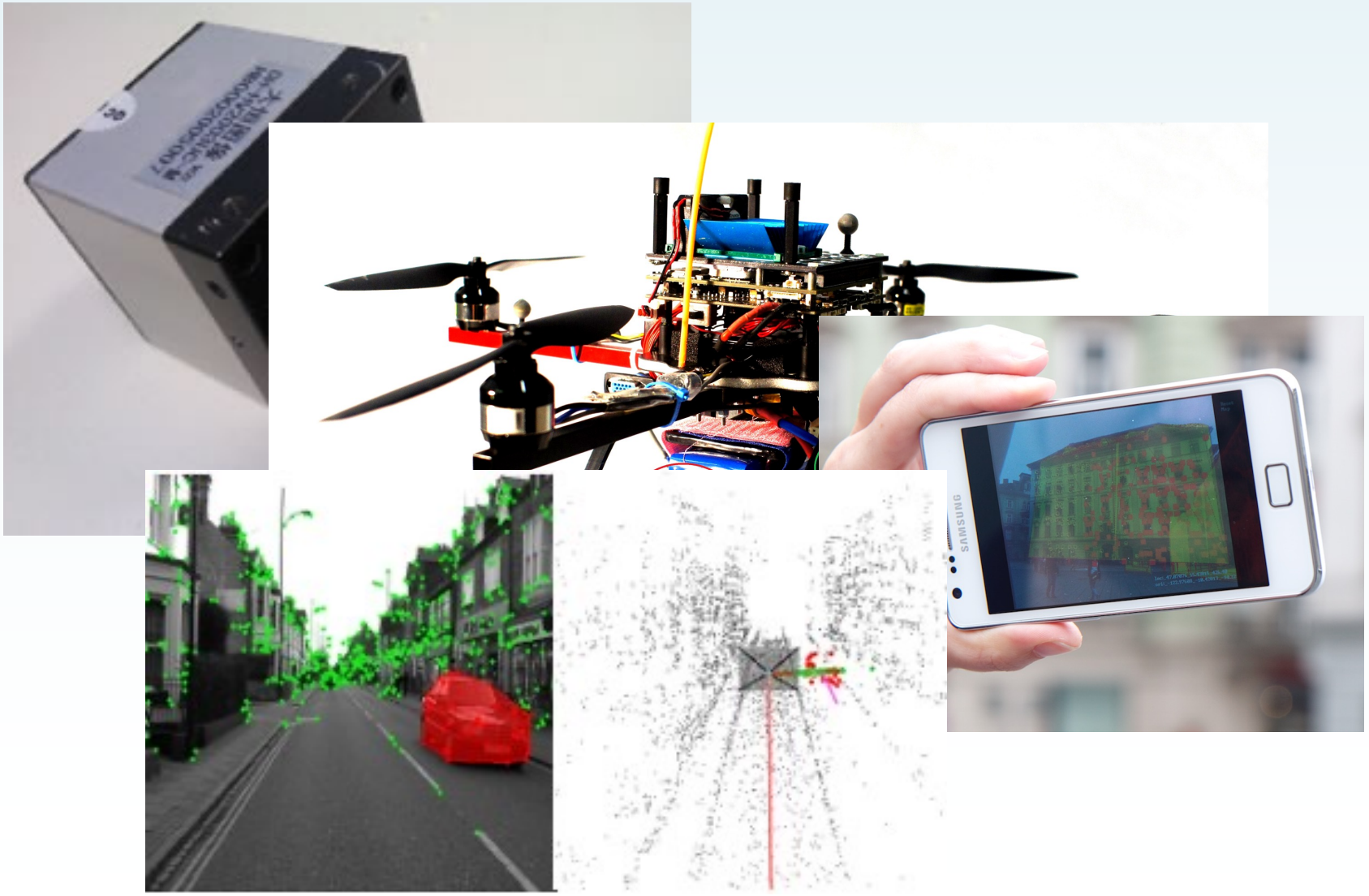
Robotic Vision and Navigation
Part III Visual SLAM

Lourdes Agapito

Email:
l.agapito@ucl.ac.uk

Slide acknowledgements: some slides have been adapted from S. Julier, D. Scaramuzza, J. Engel, R. Newcombe

Motivation: Monocular Camera SLAM



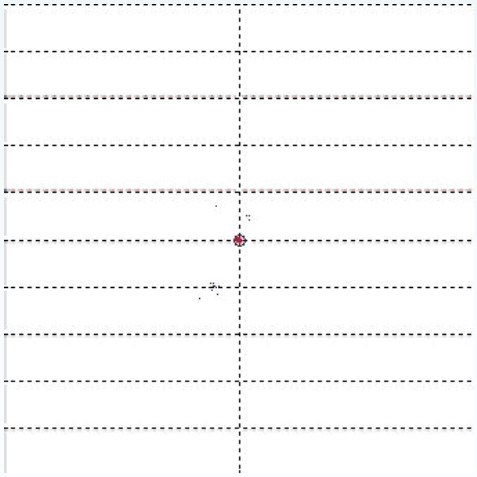
What is Visual Odometry (VO) ?

VO is the process of incrementally estimating the pose of the vehicle by examining the changes that motion induces on the images of its onboard cameras

input

Image sequence (or video stream)
from one or more cameras attached to a moving vehicle

output



$$R_0, R_1, \ldots, R_i$$

$$t_0, t_1, \ldots, t_i$$

Camera trajectory (3D structure is a plus):

Slide by Davide Scaramuzza – https://sites.google.com/scarabotx

Assumptions

- Sufficient illumination in the environment
- Dominance of static scene over moving objects
- Enough texture to allow apparent motion to be extracted
- Sufficient scene overlap between consecutive frames



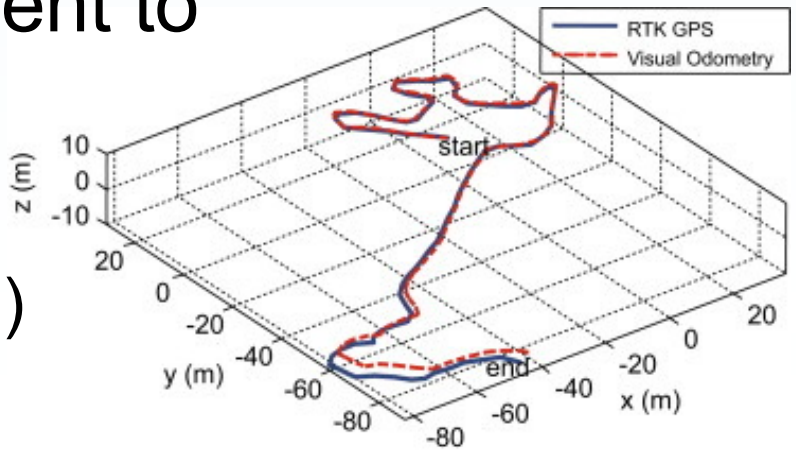


Why VO ?

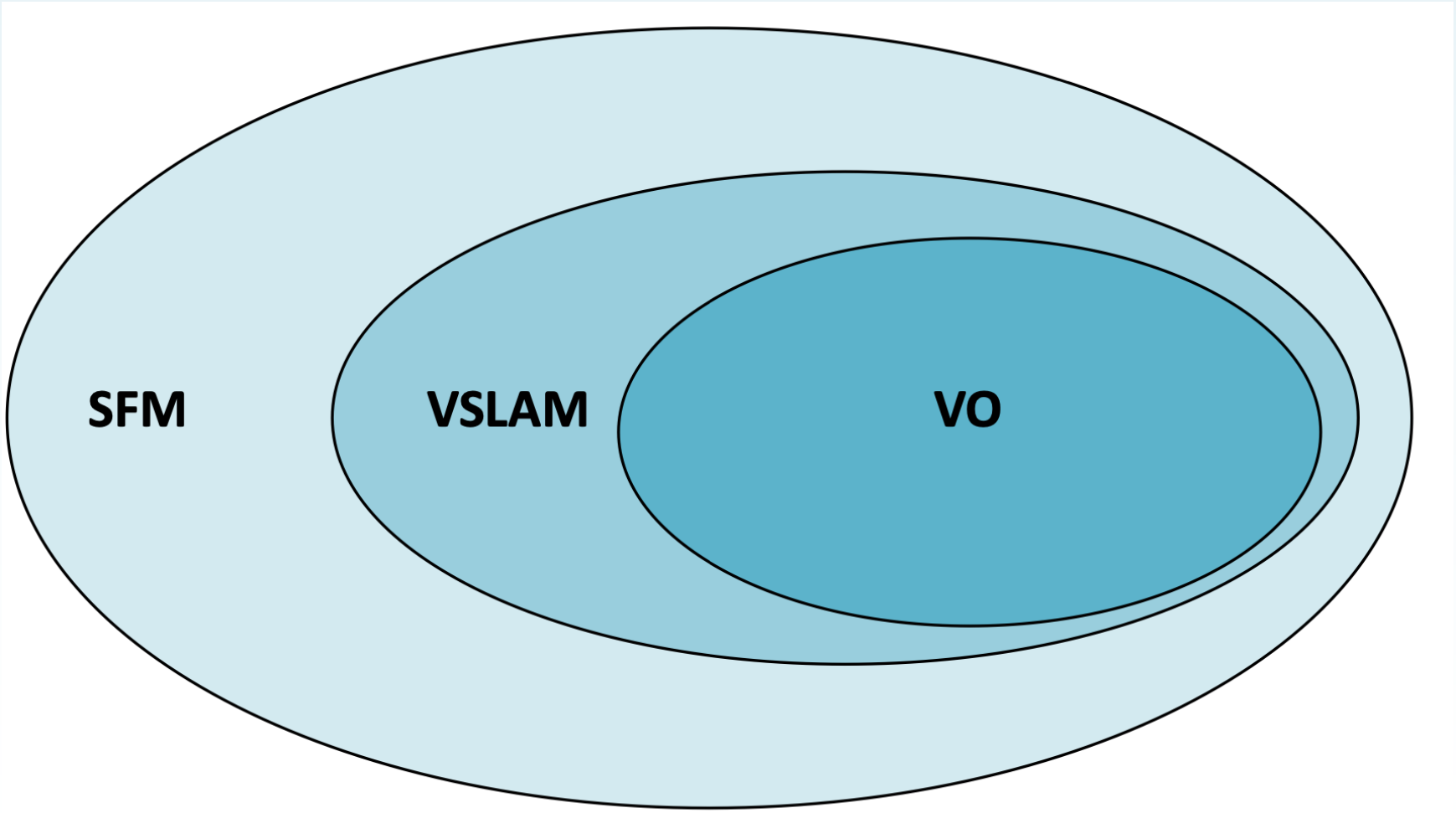
- Contrary to wheel odometry, VO is not affected by wheel slip in uneven terrain or other adverse conditions.

- More accurate trajectory estimates compared to wheel odometry (relative position error 0.1% − 2%)

- VO can be used as a complement to
  - wheel odometry
  - GPS
  - inertial measurement units (IMUs)
  - laser odometry

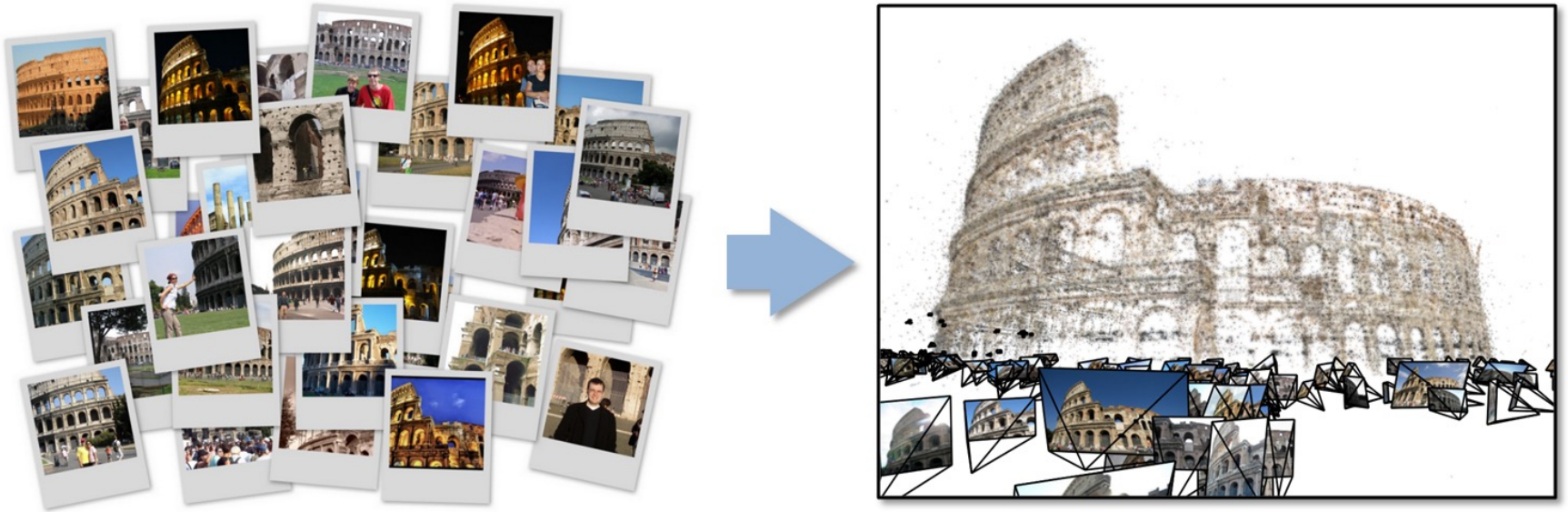


Visual Odometry vs Visual SLAM vs Structure from Motion



SfM from Unordered Internet Images

- Building 3D models from large, unordered image collections
  - [Snavely06], [Li08], [Agarwal09], [Frahm10], Microsoft's PhotoSynth, …



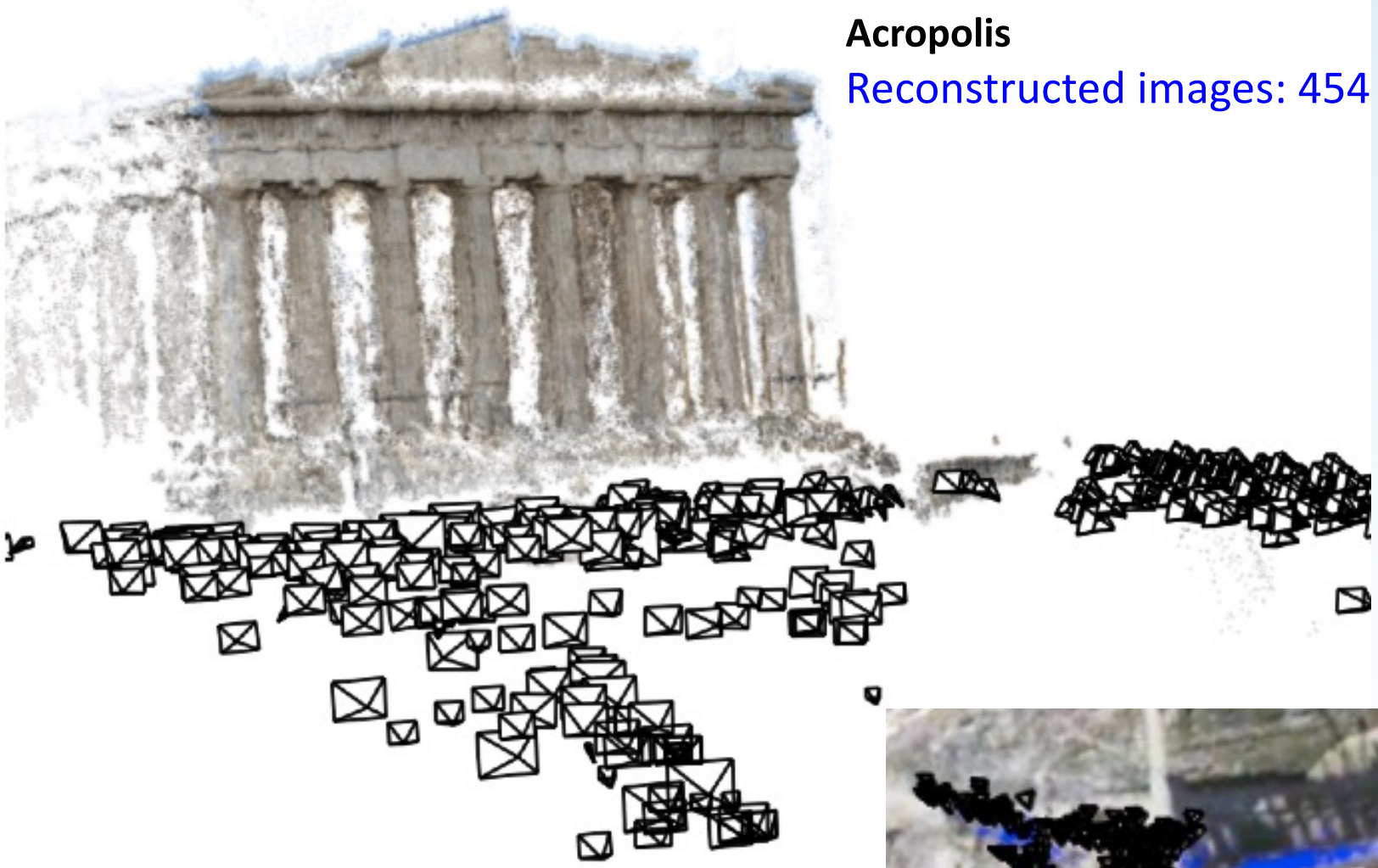
- SfM is a key part of these reconstruction pipelines

https://www.cs.cornell.edu/projects/bigsfm/

[Slide from authors]

Acropolis

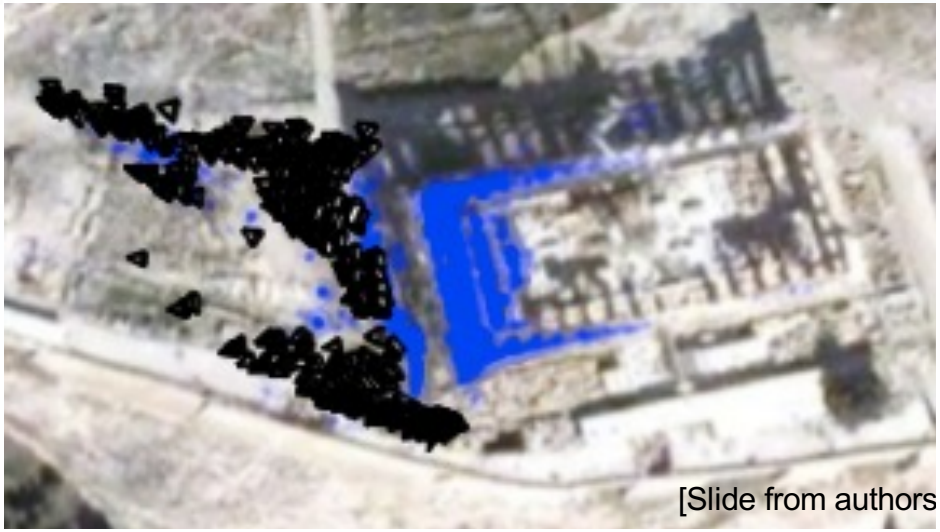
Reconstructed images: 454



Videos

http://youtu.be/kxtQqYLRaSQ

https://www.youtube.com/watch?v=i7ierVkXYa8



[Slide from authors

Reconstruction pipeline

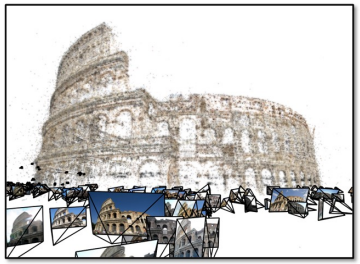
https://www.cs.cornell.edu/projects/bigsfm/

Feature detection

Feature matching
Find scene points seen by multiple cameras

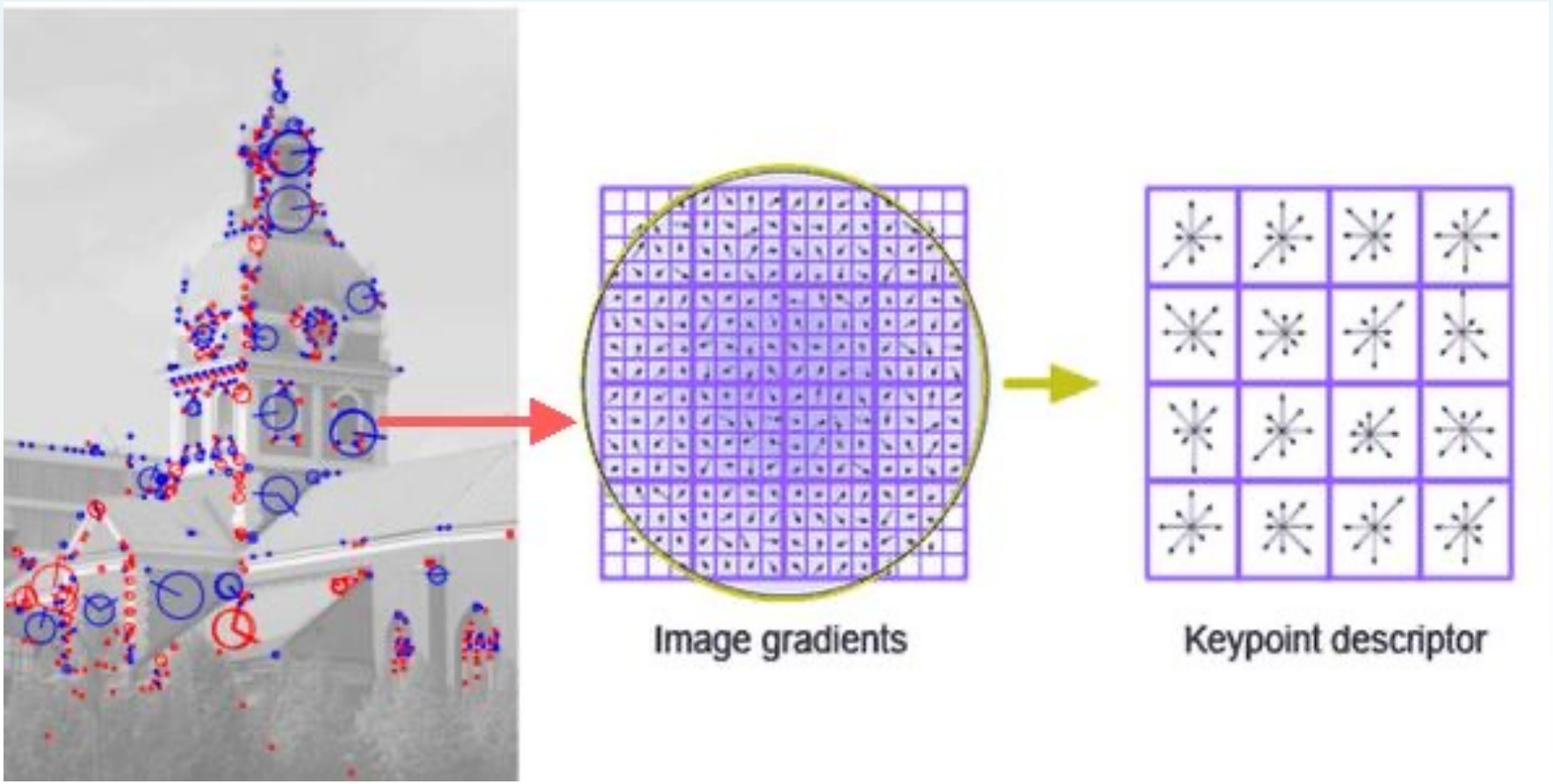
Initialization
Robustly estimate camera poses and/or scene points

Bundle adjustment
Refine camera poses **R, T** and scene structure **P**

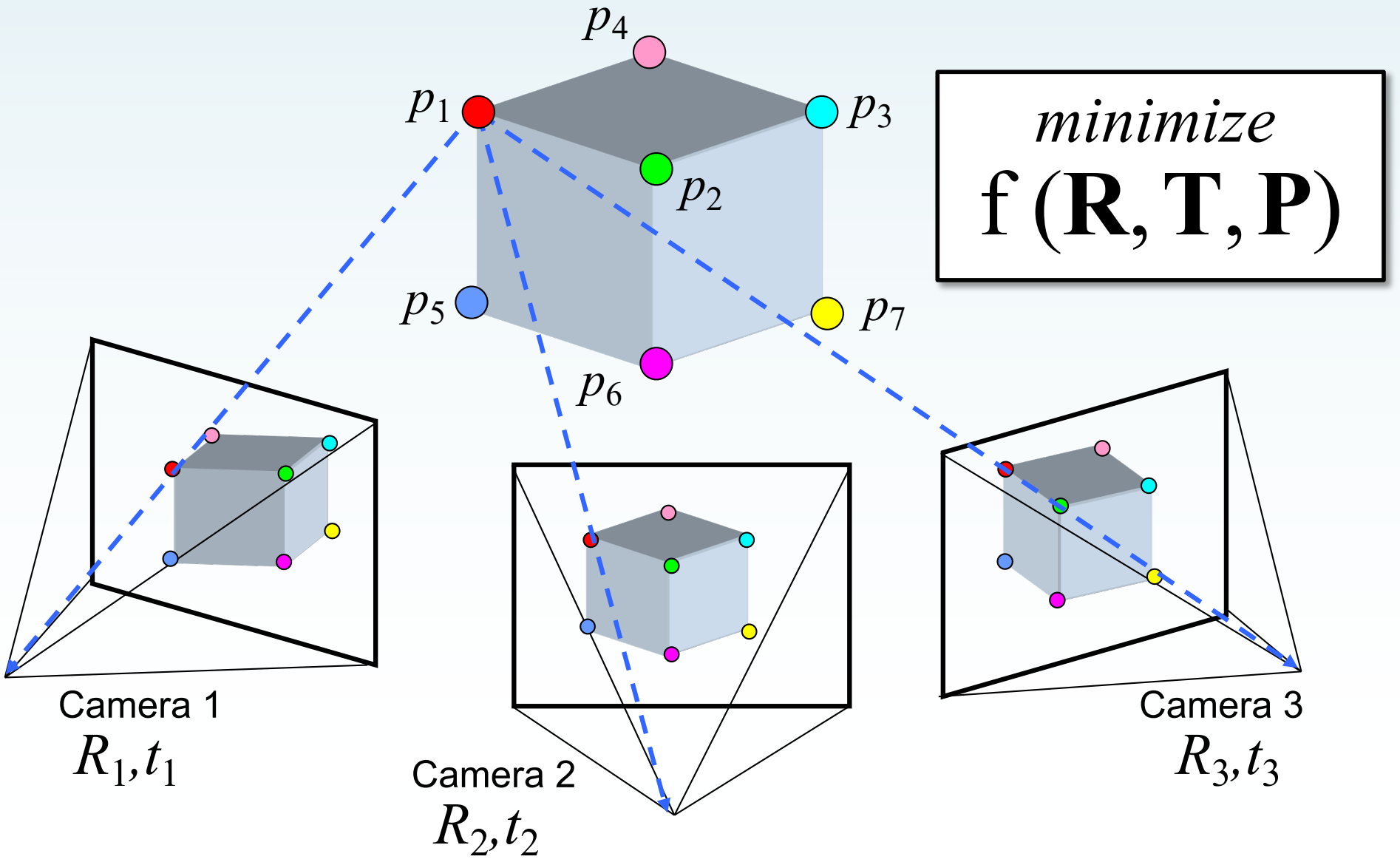


[Slide from authors

SIFT Features



Structure from Motion – Bundle Adjustment



Visual Odometry: Problem Formulation

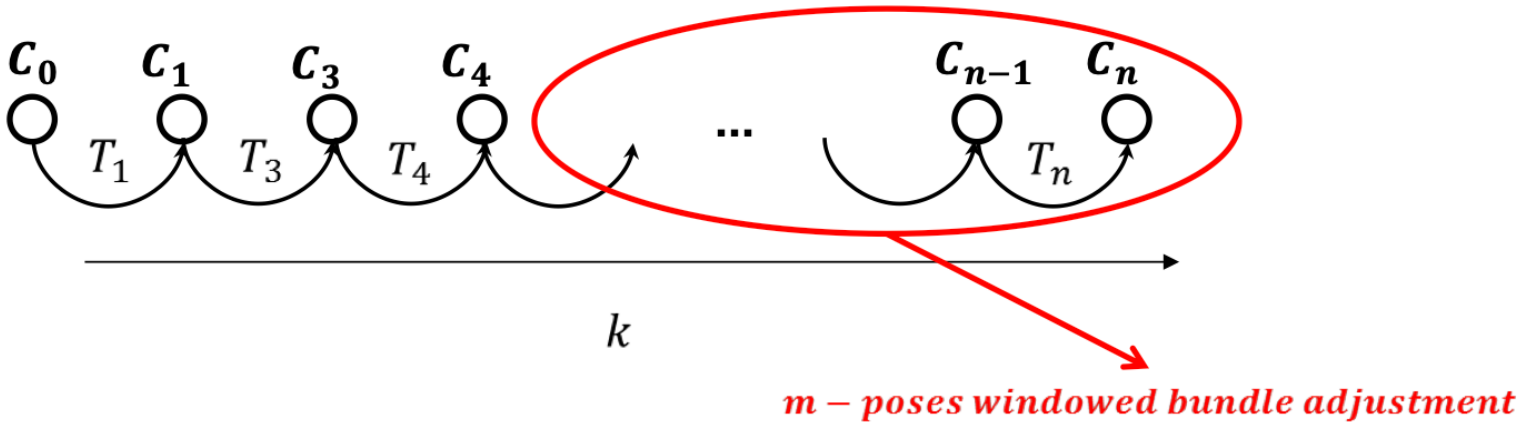
1. Compute the relative motion T_k from images I_{k-1} to image I_k

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix}$$

2. Concatenate them to recover the full trajectory

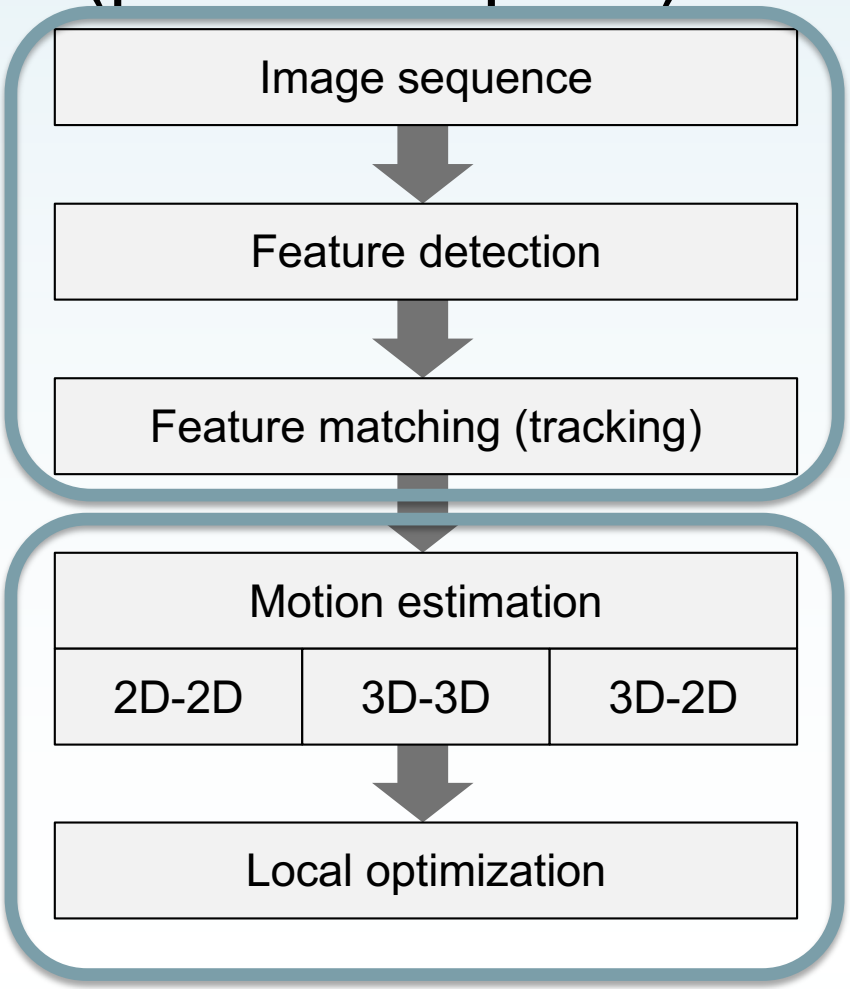
$$C_n = C_{n-1} T_n$$

3. An optimization over the last m poses can be done to refine locally the trajectory (Pose-Graph or Bundle Adjustment)



VO Flow Chart

- VO computes the camera path incrementally (pose after pose)



Front End

Back End

SIFT features tracks

Slide adapted from Davide Scaramuzza – https://sites.google.com/scarabotx

Front-End vs Back-End in VO/SLAM

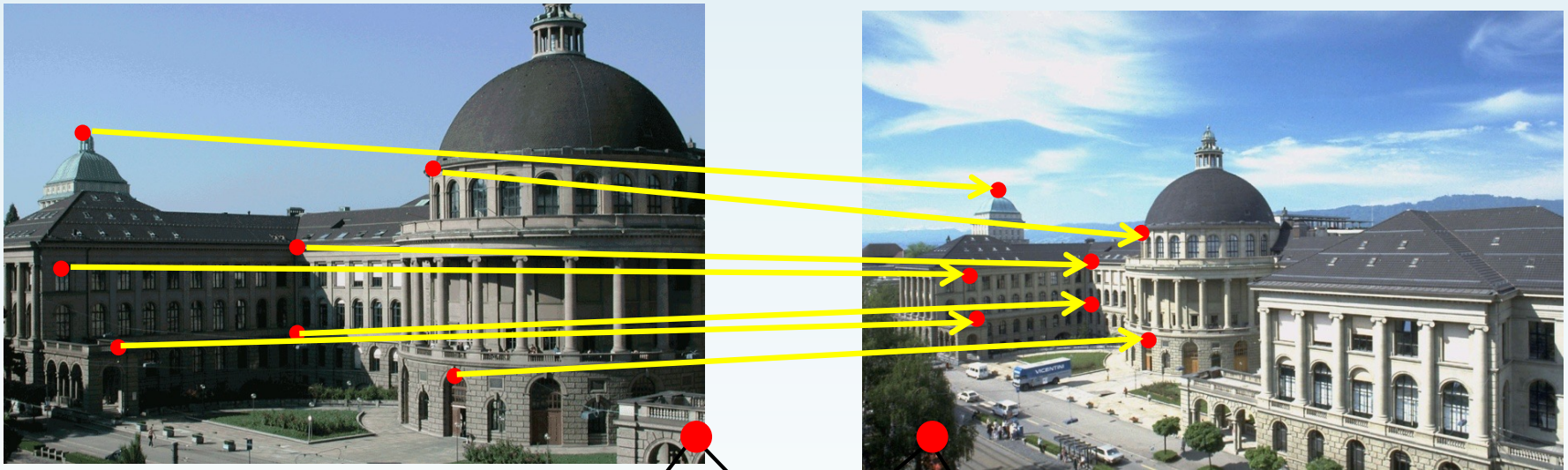
- The *front-end* is responsible for:

  - Feature extraction, matching and outlier rejection
  - Loop closure detection

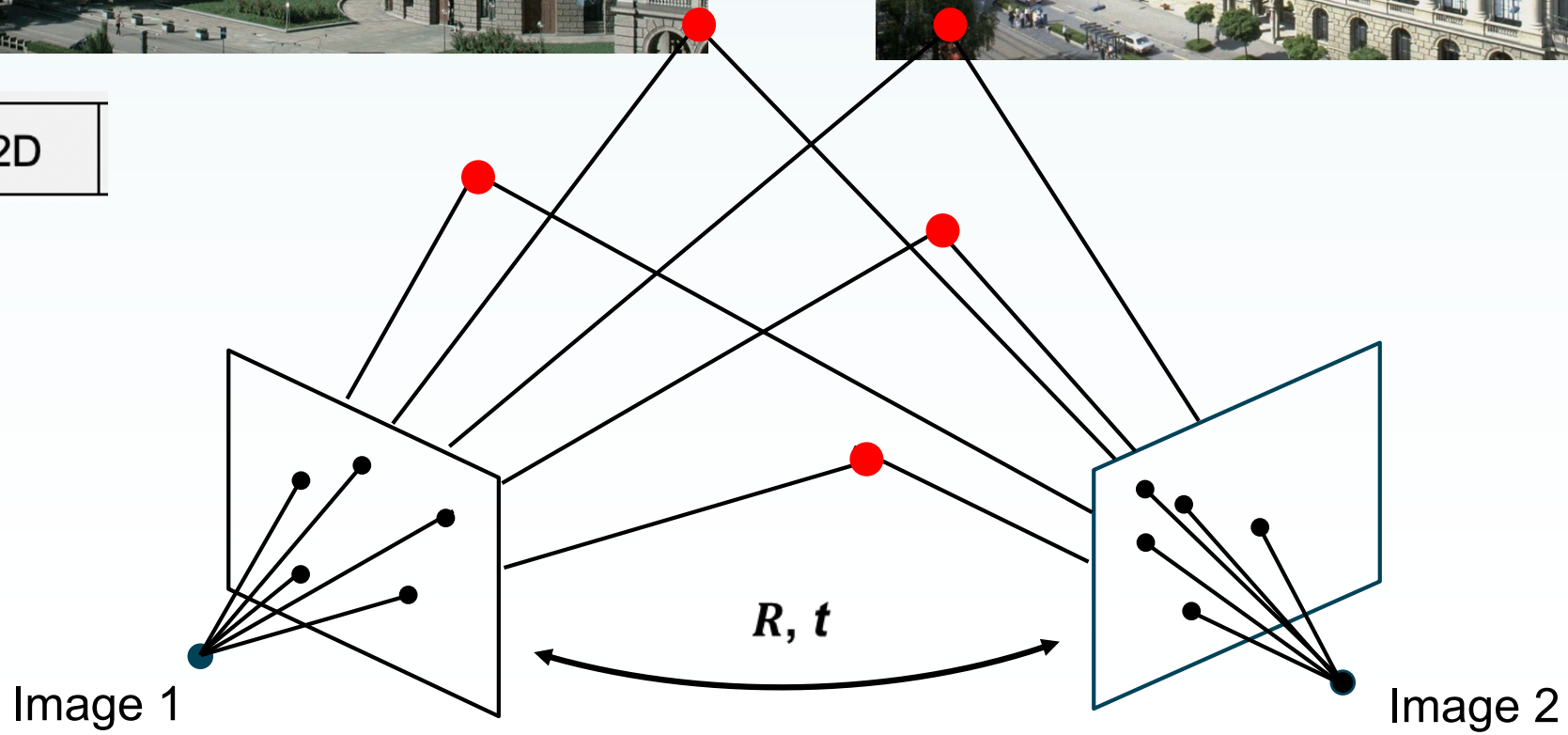
- The *back-end* is responsible for:

  - Camera pose + 3D structure optimization (e.g. factor-graph, Kalman filter, bundle-adjustment)

Relative motion estimation: 2-view geometry



2D-2D



2D-to-2D

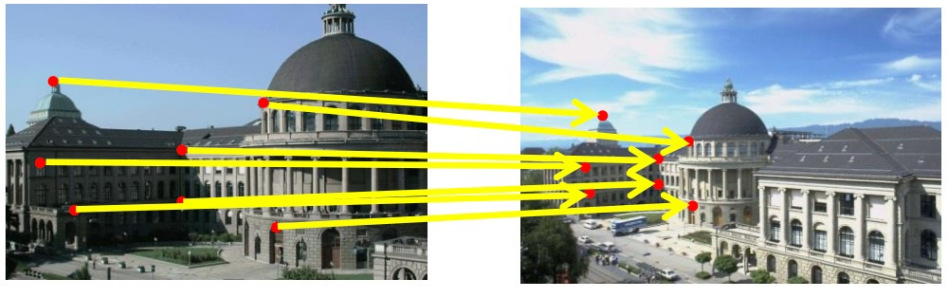
Motion estimation		
2D-2D	3D-2D	3D-3D

Motion from Image Feature Correspondences

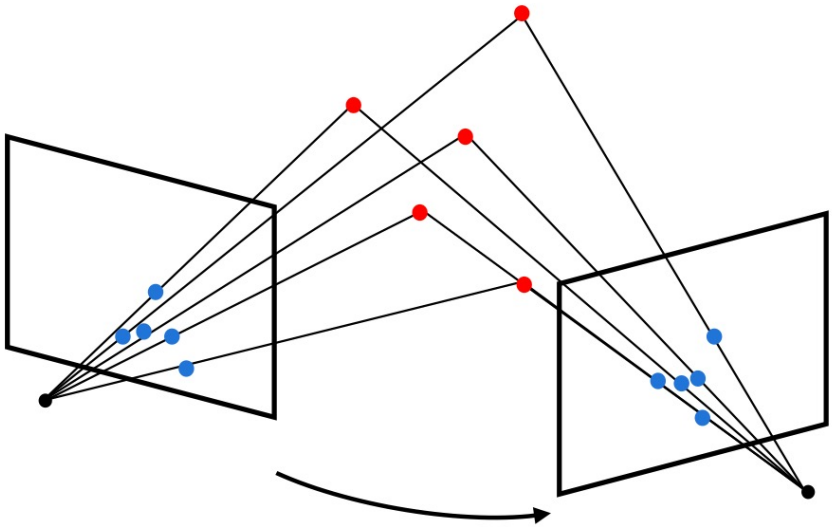
- Both feature points f_{k-1} and f_k are specified **in 2D**
- The minimal-case solution involves **5-point** correspondences
- The solution is found by minimizing the reprojection error:

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = \arg\min_{X^i, C_k} \sum_{i,k} \| p_k^i - g(X^i, C_k) \|^2$$

- Popular algorithms: 8- and 5-point algorithms [Hartley'97, Nister'06]



I_{k-1} I_k



3D-to-2D

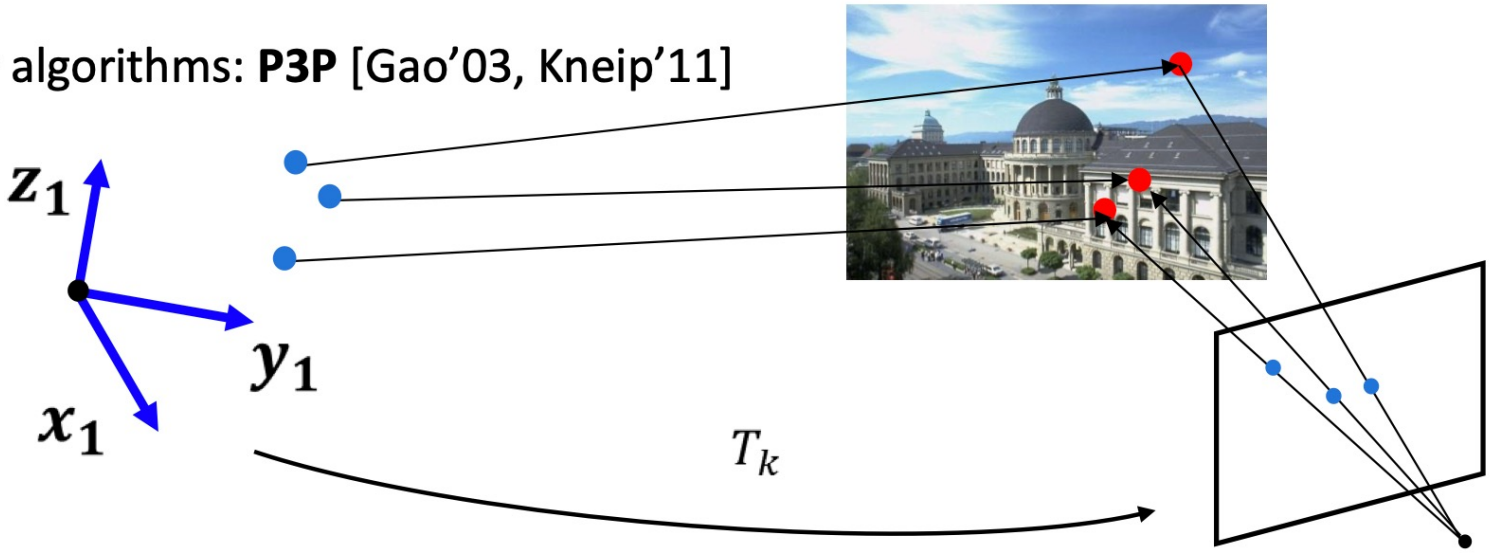
Motion estimation		
2D-2D	**3D-2D**	3D-3D

Motion from 3D Structure and Image Correspondences

- f_{k-1} is specified in 3D and f_k in **2D**
- This problem is known as *camera resection* or PnP (perspective from *n* points)
- The minimal-case solution involves **3 correspondences** (+1 for disambiguating the 4 solutions)
- The solution is found by minimizing the reprojection error:

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = \arg\min_{T_k} \sum_i \| p_k^i - \hat{p}_{k-1}^i \|^2$$

- Popular algorithms: **P3P** [Gao'03, Kneip'11]



3D-to-3D

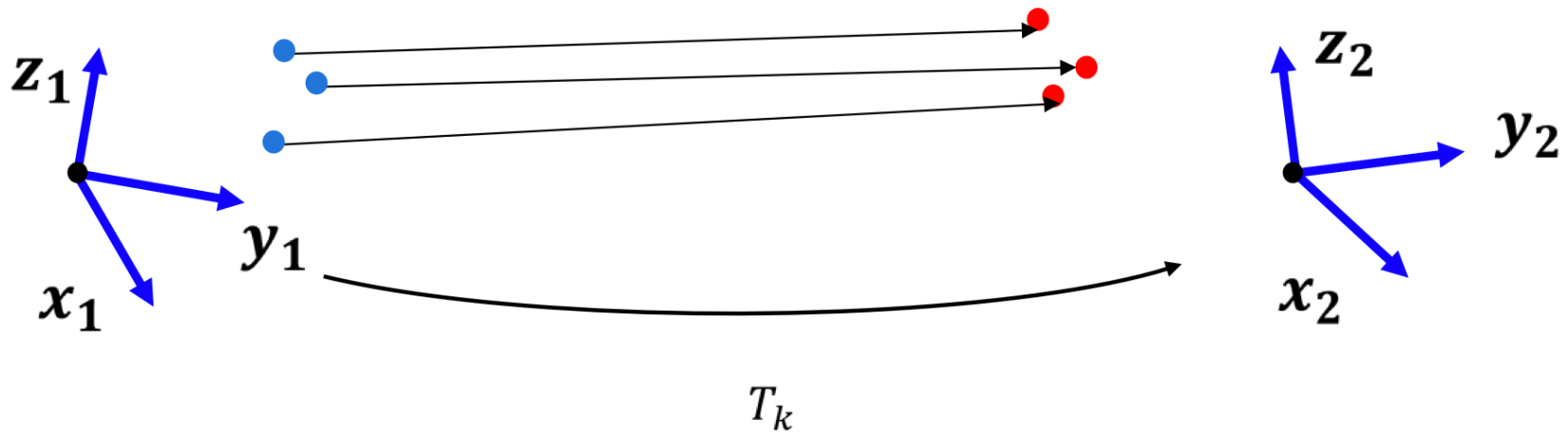
Motion estimation		
2D-2D	3D-2D	3D-3D

Motion from 3D-3D Point Correspondences (point cloud registration)

- Both f_{k-1} and f_k are specified **in 3D**. To do this, it is necessary to triangulate 3D points (e.g. use a stereo camera)
- The minimal-case solution involves **3 non-collinear correspondences**
- The solution is found by minimizing the 3D-3D Euclidean distance:

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = \arg\min_{X^i, C_k} \sum_{i,k} \| p_k^i - g(X^i, C_k) \|^2$$

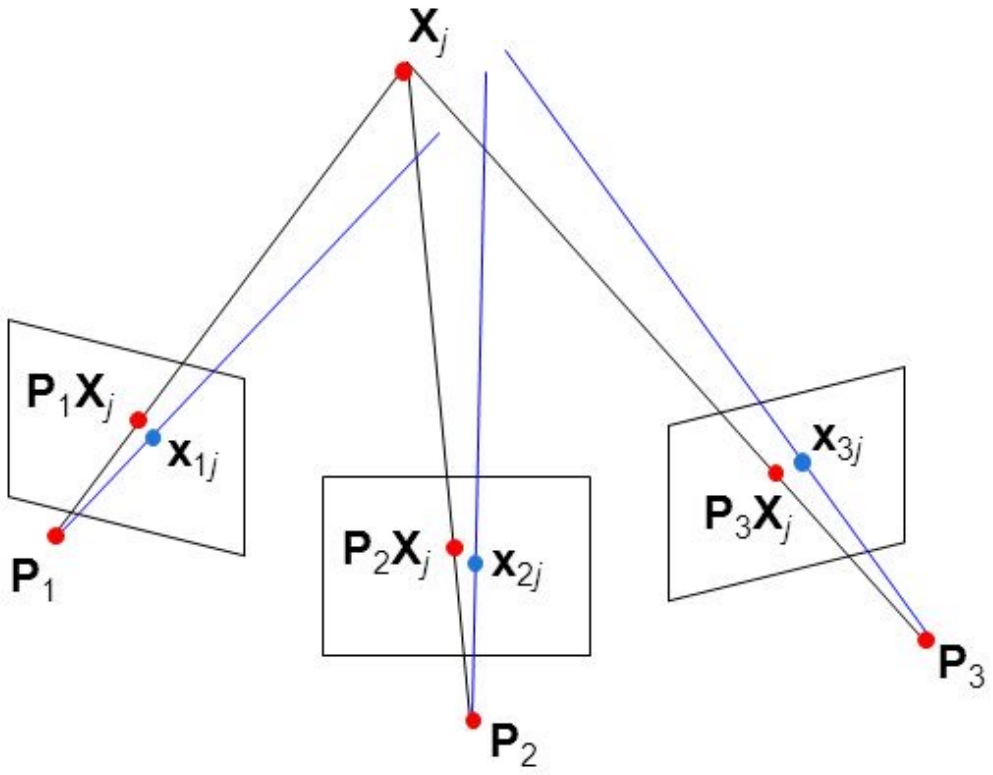
- Popular algorithm: [Arun'87] for global registration, ICP for local refinement or Bundle Adjustment (BA)



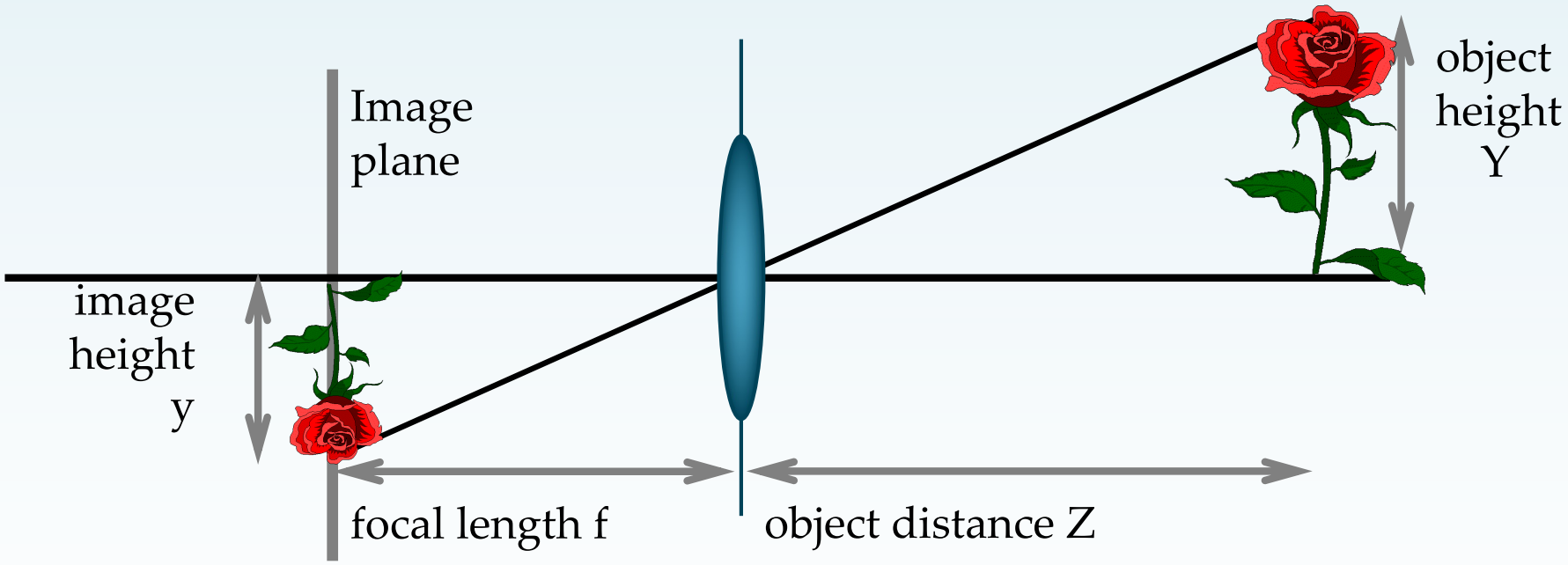
Bundle adjustment

- Non-linear method for refining structure and motion
- Minimizing reprojection error

$$E(\mathbf{P}, \mathbf{X}) = \sum_{i=1}^{m} \sum_{j=1}^{n} D\left(\mathbf{x}_{ij}, \mathbf{P}_i \mathbf{X}_j\right)^2$$



Mathematics of Perspective



$$\text{Similar triangles}: \quad \frac{y}{f} = \frac{Y}{Z}$$

$$\text{Projection}: \quad y = \frac{f}{Z}Y$$

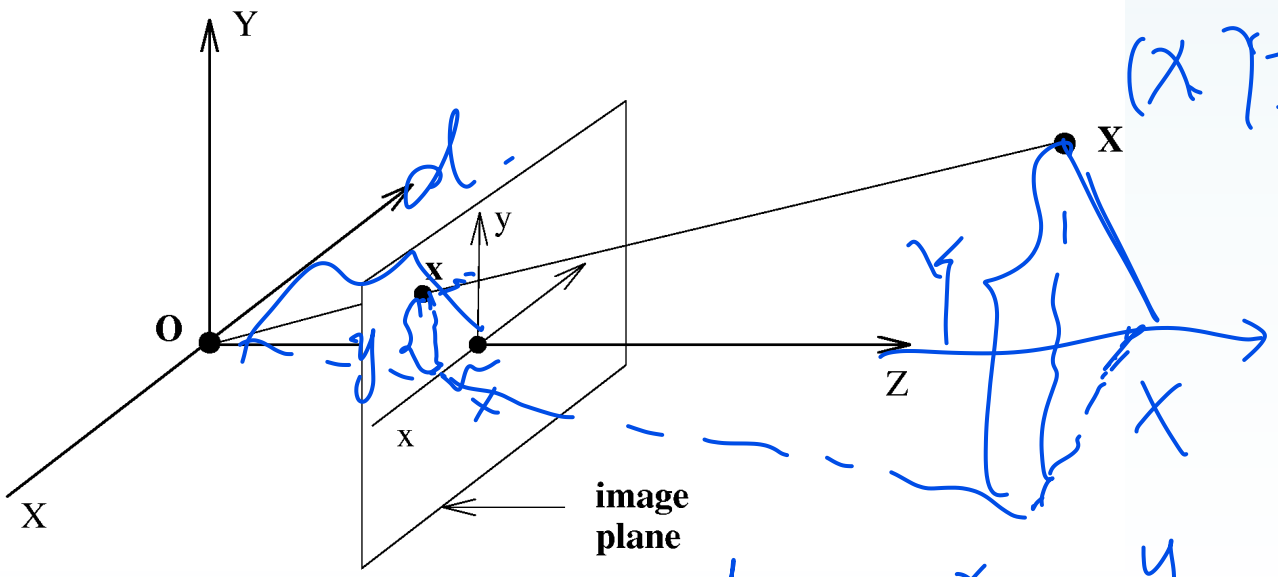
The 3x4 Projection Matrix

1. Camera Coordinate System: perspective projection.

$$\begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix} = \lambda \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix}$$

where

$\lambda = f / Z_c$.



The 3x4 Projection Matrix

This can be written as a linear mapping between homogeneous coordinates (the equation is only up to a scale factor):

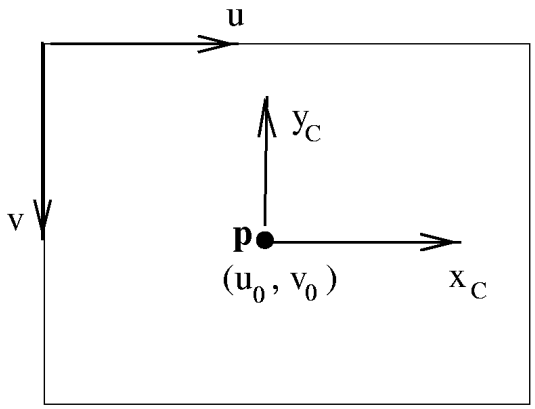
$$\begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

where a 3×4 **projection matrix** represents a map from 3D to 2D.

2. Image Coordinate System: (intrinsic/internal camera parameters)

$$k_u x_c = u - u_0$$
$$k_v y_c = v_0 - v$$

where the units of k are [pixels/length].



$$\mathbf{x}_i = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & -fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix} = \mathtt{K} \begin{bmatrix} x_c \\ y_c \\ f \end{bmatrix}$$

$\mathtt{K}$ is a 3×3 upper triangular matrix, called the **camera calibration matrix**:

$$\mathtt{K} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

where $\alpha_u = fk_u$, $\alpha_v = -fk_v$.

- There are four parameters:

  1. The scaling in the image x and y directions, α_u and α_v.

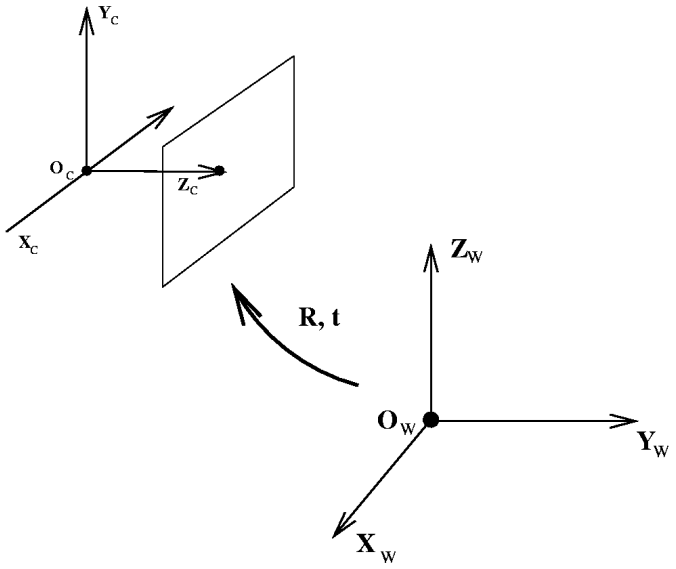
  2. The *principal point* (u_0, v_0), which is the point where the optic axis intersects the image plane.

  The *aspect ratio* is α_v/α_u.

3. World Coordinate System: (extrinsic/external camera parameters)

The Euclidean transformation between the camera and world coordinates is $\mathbf{X}_c = \mathtt{R}\mathbf{X}_w + \mathbf{t}$:

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$



$\mathtt{R}$: a 3×3 rotation matrix

$\mathbf{t}$: a 3×1 translation vector

Finally, concatenating the three matrices,

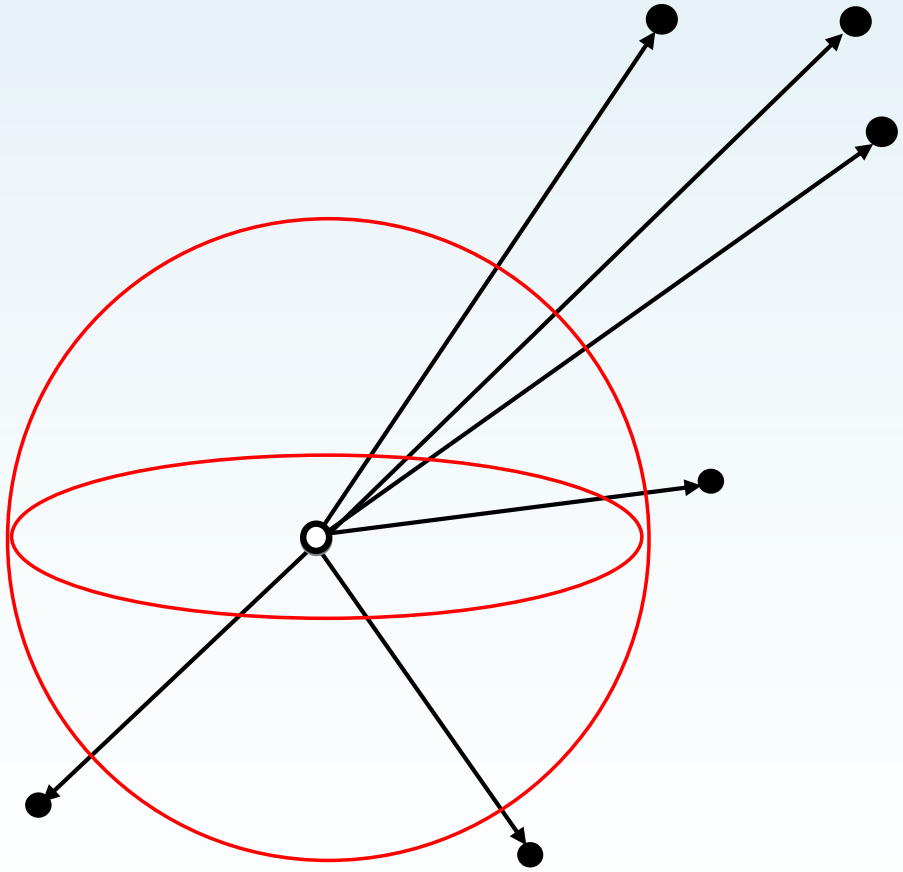
$$\mathbf{x} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathtt{K} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

$$= \mathtt{K} \begin{bmatrix} \mathtt{R} | \mathbf{t} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}$$

which defines the 3×4 projection matrix from Euclidean 3-space to an image:

$$\mathbf{x} = \mathtt{P} \begin{bmatrix} \mathbf{X} \\ 1 \end{bmatrix} \qquad \mathtt{P} = \mathtt{K} \begin{bmatrix} \mathtt{R} | \mathbf{t} \end{bmatrix}$$

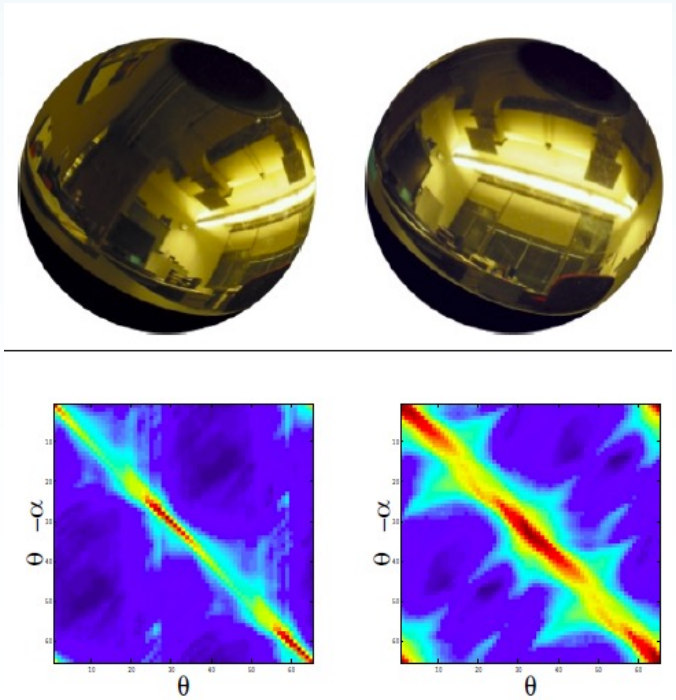
Spherical Model

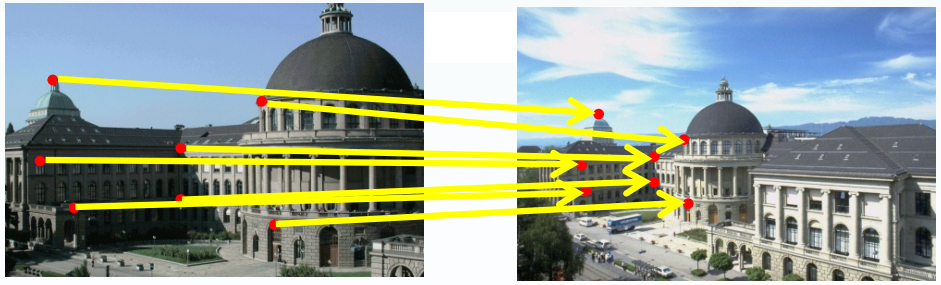


- Always possible after the camera has been calibrated!

There are two main approaches to compute the relative motion T_k

- Appearance-based methods use the intensity information of all the pixels in the two input images
- Feature-based methods only use salient and repeatable features extracted (or tracked) across the images





Makadia et al. «Correspondence-free structure from motion», IJCV'07

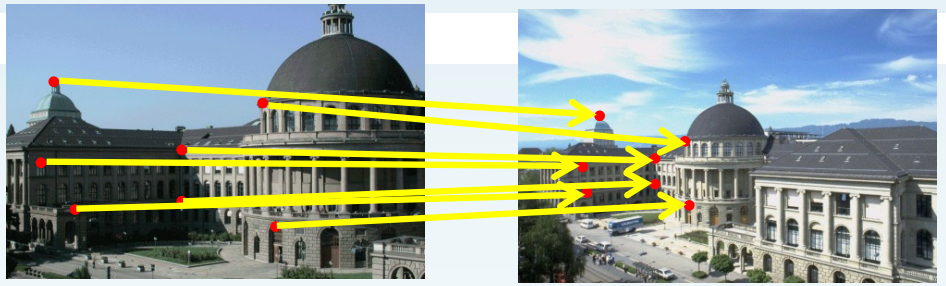
Motion Estimation

Depending on whether the feature correspondences f_{k-1} and f_k are specified in 2D or 3D, there are three different cases:

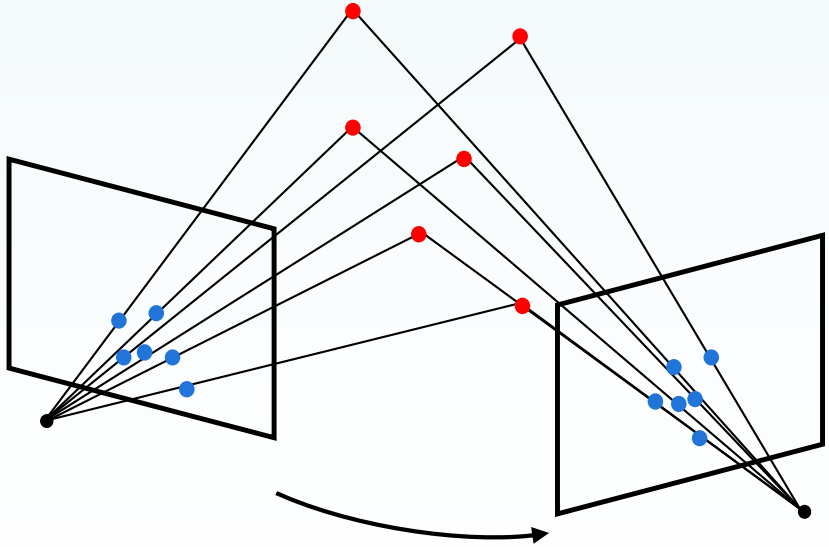
- **2D-to-2D**: both f_{k-1} and f_k are specified in 2D image coordinates
- **3D-to-3D**: both f_{k-1} and f_k are specified in 3D To do this, it is necessary to triangulate 3D points at each time instant, for instance, by using a stereo camera system
- **3D-to-2D**: f_{k-1} are specified in 3D and f_k are their corresponding 2D reprojections on the image I_k

2D-to-2D

Motion from Image Feature Correspondences

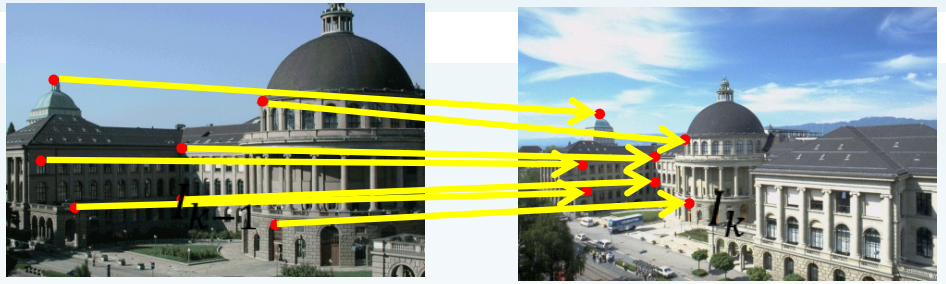


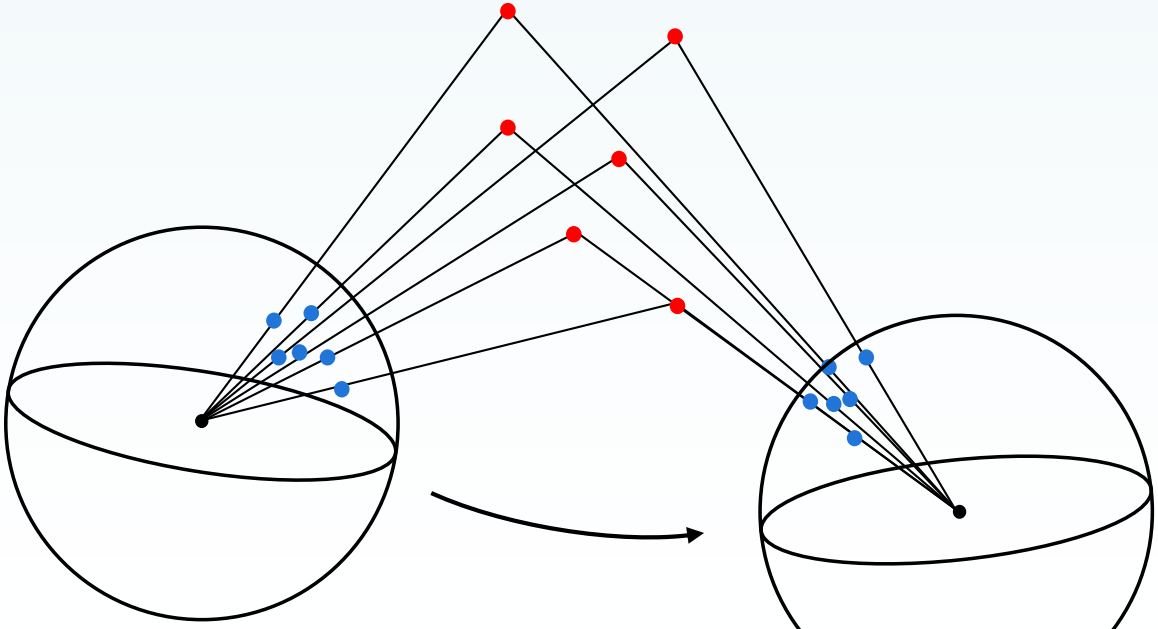
I_{k-1} I_k



$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = ?$$

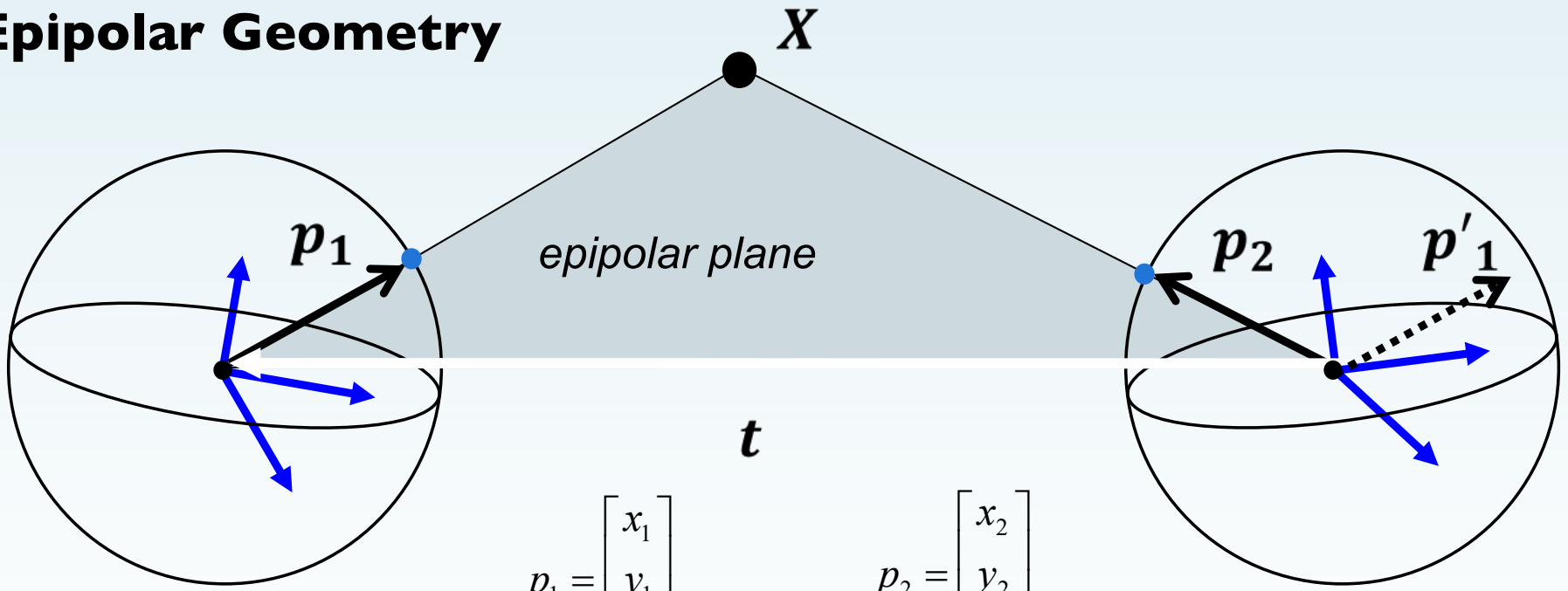
2D-to-2D
Motion from Image Feature Correspondences





$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = ?$$

2D-to-2D
Epipolar Geometry



$$p_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \qquad p_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$$

p_1, p_2, T are coplanar:

$$p_2^T \cdot (t \times p_1') = 0 \qquad \Rightarrow p_2^T \cdot (t \times (R p_1)) = 0$$

$$\Rightarrow p_2^{\mathrm{T}} [t]_\times R \, p_1 = 0 \qquad \Rightarrow p_2^T \, E \, p_1 = 0 \qquad \textit{Epipolar constraint}$$

$$\mathrm{E} = [t]_\times R \qquad \textit{essential matrix}$$

2D-to-2D

Epipolar Geometry

$$p_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \quad p_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$$ ***Image coordinates on the Unit sphere***

$$p_2^T \, E \, p_1 = 0$$ ***Epipolar constraint***

$$\mathrm{E} = [t]_\times R$$ ***Essential matrix***

$$[\boldsymbol{t}]_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

- The Essential Matrix can be computed directly from the image coordinates (using SVD).
- At least 5 points needed! [Kruppa, 1913]. The more points, the better!
- The Essential Matrix can be decomposed into $\boldsymbol{R}$ and $\boldsymbol{t}$ (again using SVD)

2D-to-2D

Computing the Essential Matrix

- The Essential matrix can be computed from 5 point correspondences using [Nister'2003] algorithm (5-*point algorithm)*
- *The 5-p algorithm* has become the standard for 2D-to-2D motion estimation, however, its implementation is not straightforward
- A simple and straightforward solution for $n \geq 8$ noncoplanar points is the Longuet-Higgins' *8-p algorithm*, which is summarized here:
- Let $p_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}$, $p_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}$ be the coordinates one feature correspondence
- $E = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} = \begin{bmatrix} e_{11} \\ \vdots \\ e_{33} \end{bmatrix}$
- ${p_2}^{\mathrm{T}} E p_1 = 0 \Rightarrow [\boldsymbol{x_1x_2}\ \boldsymbol{y_1x_2}\ \boldsymbol{z_1x_2}\ \boldsymbol{x_1y_2}\ \boldsymbol{y_1y_2}\ \boldsymbol{z_1y_2}\ \boldsymbol{x_1z_2}\ \boldsymbol{y_1z_2}\ \boldsymbol{z_1z_2}]E = 0$

  which can be solved with SVD

Relationship between the *Essential* and the *Fundamental* Matrices

A similar matrix can be defined for calibrated cameras, from

$$\mathbf{x}'^{\top}\left(\mathtt{K}'^{-\top}[\mathbf{t}]_{\times}\mathtt{R}\mathtt{K}^{-1}\right)\mathbf{x}=0$$

and using the calibration matrices to relate the image point in pixels to the point in the camera coordinate system:

$$\mathbf{x}=\mathtt{K}\mathbf{x}_c \qquad \mathbf{x}'=\mathtt{K}'\mathbf{x}'_c$$

defines the **Essential Matrix**, $\mathtt{E}$ by $\mathbf{x}'^{\top}_{\mathbf{c}}\mathtt{E}\mathbf{x}_{\mathbf{c}}=0$, where

$$\mathtt{E}=[\mathbf{t}]_{\times}\mathtt{R}$$

The essential matrix is the algebraic representation of epipolar geometry for known calibration. It is related to the fundamental matrix by

$$\mathtt{F}=\mathtt{K}'^{-\top}\mathtt{E}\mathtt{K}^{-1}$$

Computing F: Number of correspondences

We have seen that $\mathtt{F}$ can be computed from the internal and relative external camera parameters as $\mathtt{F} = \mathtt{K}'^{-\top}[\mathbf{t}]_{\times}\mathtt{R}\mathtt{K}^{-1}$. Here we show that $\mathtt{X}$ can be computed directly from image point correspondences.

Given perfect image points (no noise) in general position. Each point correspondence $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ generates one constraint on $\mathtt{F}$.

$$\begin{bmatrix} x'_i & y'_i & 1 \end{bmatrix} \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0$$

Properties of the Fundamental Matrix

If $\mathbf{x}$ and $\mathbf{x}'$ are corresponding image points, then

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} \begin{bmatrix} & \mathbf{F} & \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0$$

i.e. $\mathbf{x}'^{\top}\mathbf{F}\mathbf{x} = 0$, where $\mathbf{F}$ is the 3×3 **fundamental** matrix of maximum rank 2.

This can be rearranged as $\mathtt{A}\mathbf{f} = \mathbf{0}$ where $\mathtt{A}$ is a $n \times 9$ measurement matrix, and $\mathbf{f}$ is the fundamental matrix represented as a 9-vector.

$$\begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \\ f_9 \end{bmatrix} = \mathbf{0}$$

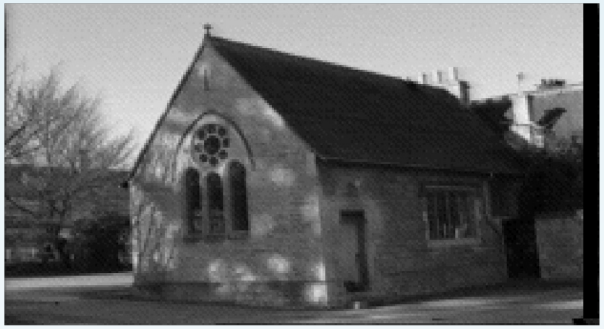
- **8 or more points in general position**: The dimension of the null-space of $\mathtt{A}$ is one. $\mathbf{f}$, and consequently $\mathtt{F}$, is determined uniquely up to scale.

- **7 points in general position**: The dimension of the null-space is two.

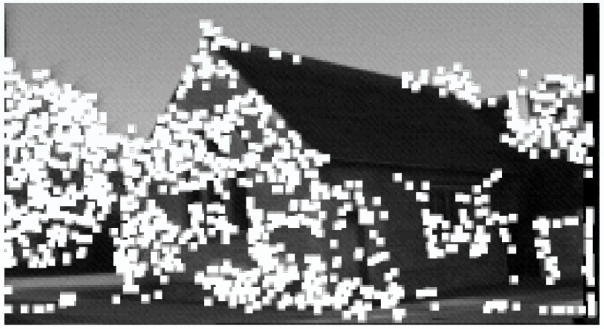
  Suppose the 9-vectors $\mathbf{u}_1$ and $\mathbf{u}_2$, corresponding to the matrices $\mathtt{U}_1$ and $\mathtt{U}_2$ respectively, span the null-space. Then, up to scale, there is a one-parameter family of matrices: $\alpha\mathtt{U}_1 + (1-\alpha)\mathtt{U}_2$. Imposing $\det \mathtt{F} = 0$

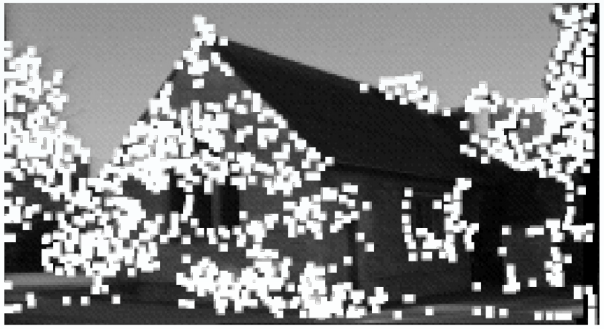
$$\det|\alpha\mathtt{U}_1 + (1-\alpha)\mathtt{U}_2| = 0$$

  which is a cubic in α, and thus has either one or three real solutions.



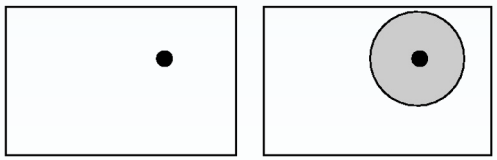
Compute corners (typically 200-300 per image):





Correspondences are computed automatically and robustly in three stages:

1. **Unguided matching:** obtain a small number of *seed* matches using a local search and normalised cross-correlation with a conservative threshold.



1. **Compute epipolar geometry:** use seed matches to robustly compute $\mathtt{F}$ using RANSAC (Torr).

Matches consistent with $\mathtt{F}$

2D-to-2D Algorithm

Algorithm 1: VO from 2D-to-2D correspondeces

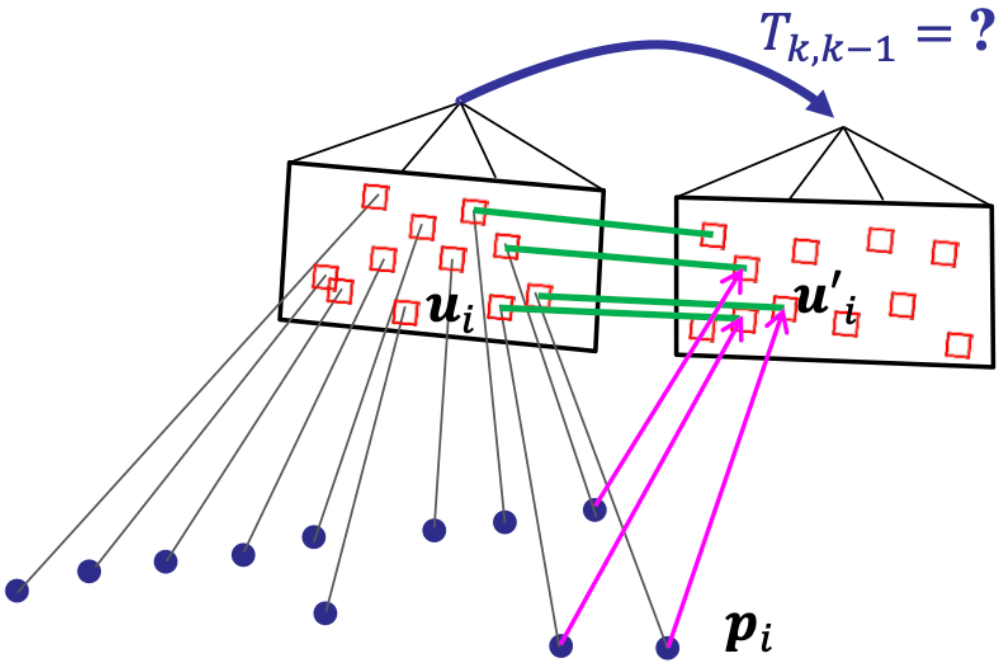
1 Capture new frame I_k
2 Extract and match features between I_{k-1} and I_k,
3 Compute essential matrix for image pair I_{k-1}, I_k
4 Decompose essential matrix into R_k and t_k, and form T_k
5 Compute relative scale and rescale t_k accordingly
6 Concatenate transformation by computing $C_k = C_{k-1}T_k$
7 Repeat from 1

Motion Estimation

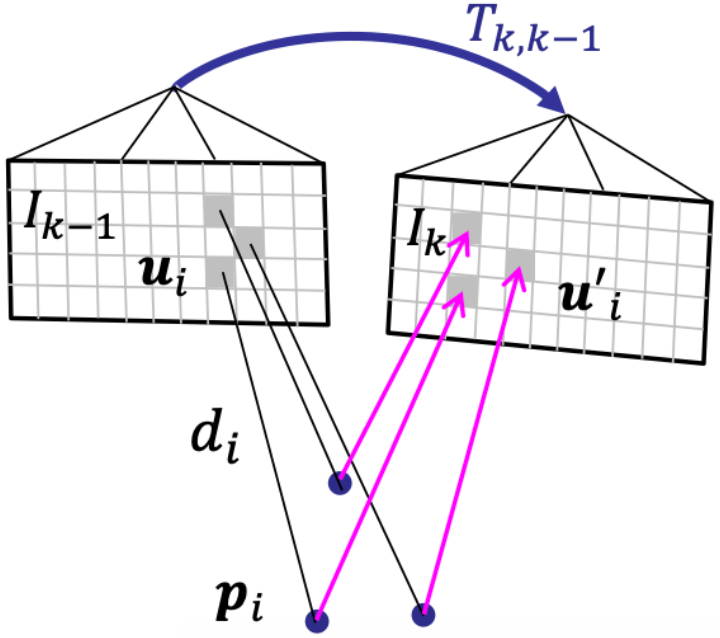
Depending on whether the feature correspondences f_{k-1} and f_k are specified in 2D or 3D, there are three different cases:

- **2D-to-2D**: both f_{k-1} and f_k are specified in 2D image coordinates
- **3D-to-3D**: both f_{k-1} and f_k are specified in 3D To do this, it is necessary to triangulate 3D points at each time instant, for instance, by using a stereo camera system
- **3D-to-2D**: f_{k-1} are specified in 3D and f_k are their corresponding 2D reprojections on the image I_k

Feature-based methods

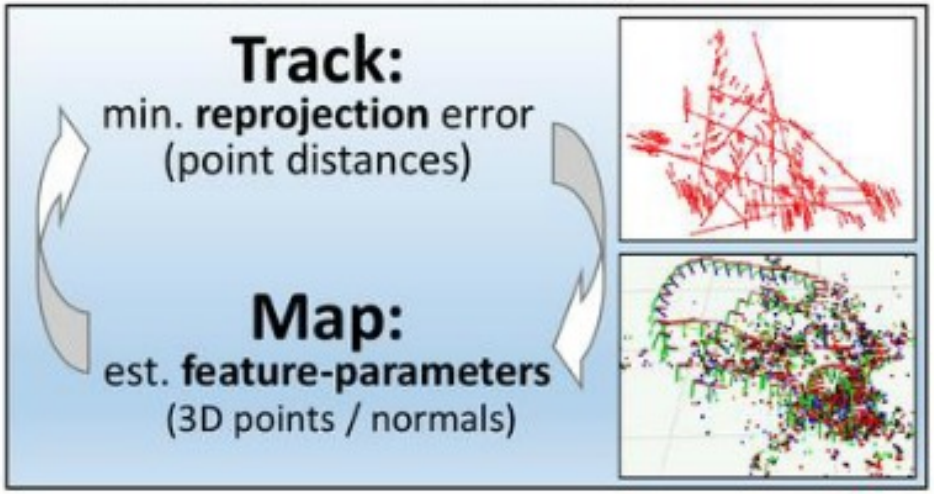


Direct methods



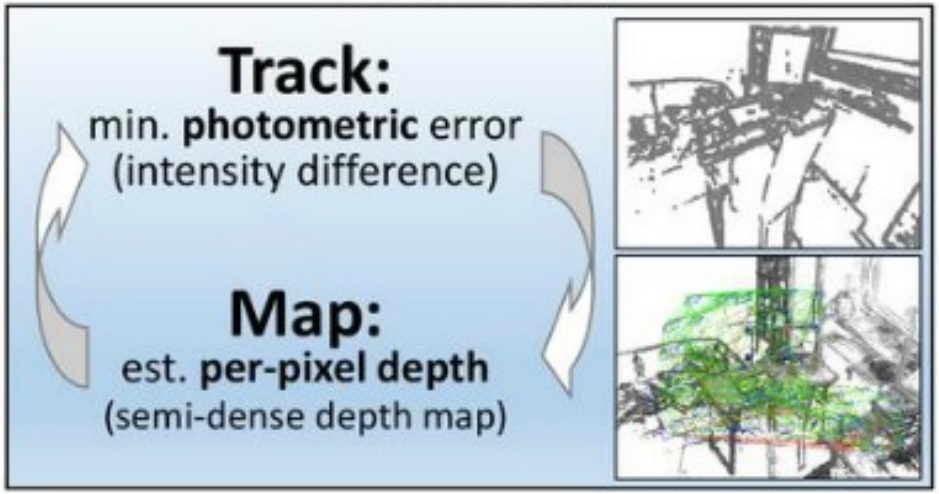
Keypoint-Based

abstract images to feature observations



Direct Dense SLAM

keep full image

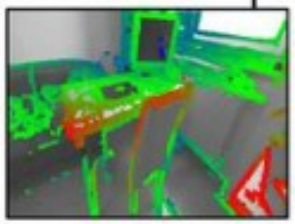


Direct Methods minimise photometric cost

$$E(\boldsymbol{\xi}) = \sum_{\mathbf{x} \in \Omega_{\mathrm{KF}}} \left(I_{\mathrm{KF}}(\mathbf{x}) - I(\omega(\mathbf{x}, D_{\mathrm{KF}}(\mathbf{x}), \boldsymbol{\xi})) \right)^2 =: \|\mathbf{r}(\boldsymbol{\xi})\|_2^2$$

Camera Pose in $\mathfrak{se}(3)$

KF image

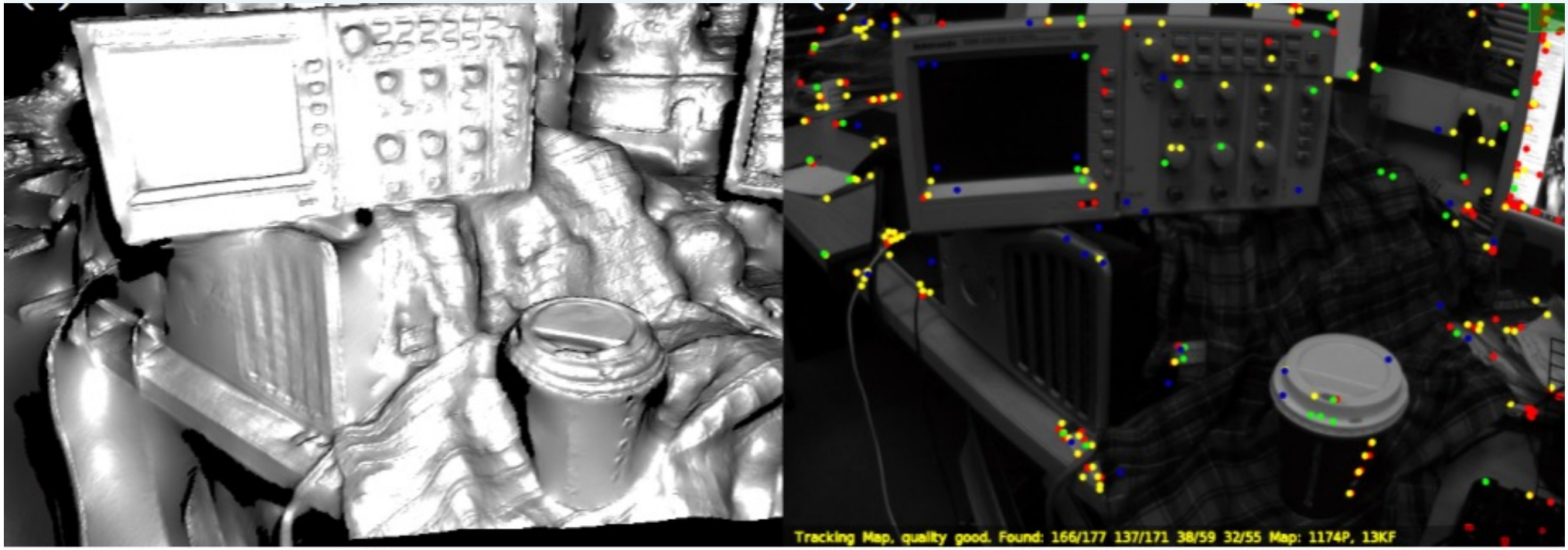


KF depth

back-warped new frame

- minimize using **Gauss-Newton** Algorithm (≈ forward-compositional Lucas-Kanade)

Why direct methods?



DTAM– **Dense**
Newcombe et al.
ICCV'11

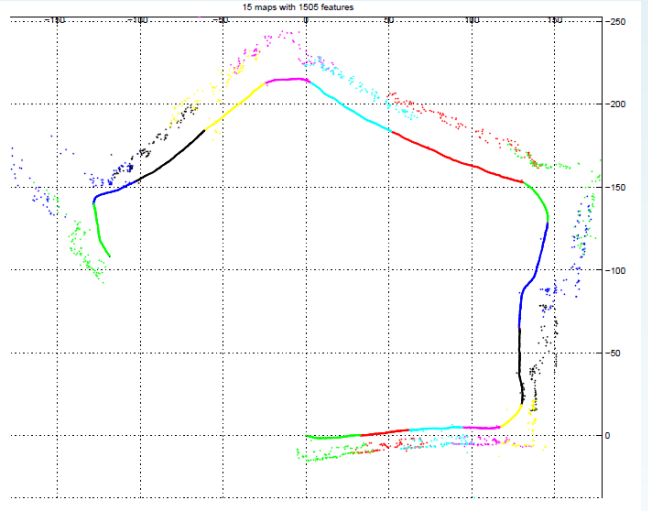
PTAM– **Sparse**
Klein and Murray,
ISMAR'07

Visual Odometry vs. Visual SLAM

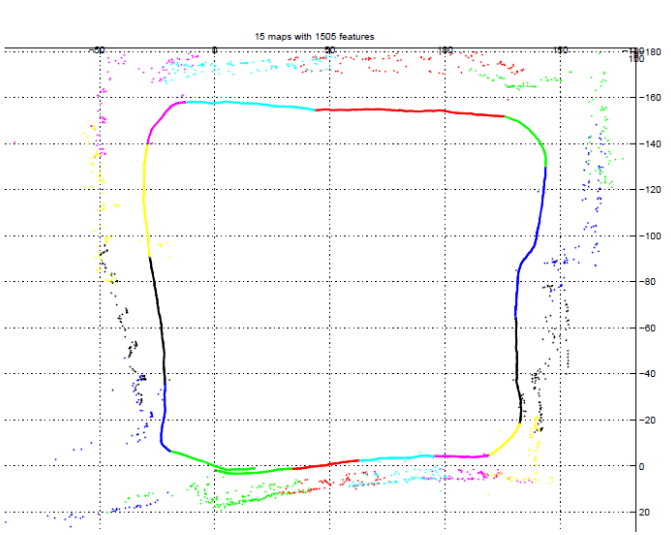
- Main difference between Visual Odometry (VO) and Visual SLAM (V-SLAM) is that while VO cares mostly about **local** estimation of the **camera trajectory**, V-SLAM also estimates an accurate globally consistent map.
- VO focuses on estimating the 3D motion of the camera (trajectory) sequentially (as a new frame arrives) and in real time.
- Instead, V-SLAM also focuses on estimating a map of the 3D environment.
- Bundle adjustment can be used in VO to refine the local estimate of the trajectory, but it is ”optional”.
- Often said that the difference between VO and V-SLAM is that V-SLAM incorporates loop closure.
- Both are strongly related to Structure from Motion (SFM)

VO vs. Visual SLAM (2/2)

- VO only aims to the local consistency of the trajectory
- SLAM aims to the global consistency of the trajectory and of the map
- VO can be used as a building block of SLAM
- VO is SLAM before closing the loop!
- The choice between VO and V-SLAM depends on the tradeoff between performance and consistency, and simplicity in implementation.
- VO trades off consistency for real-time performance, without the need to keep track of all the previous histor of the camera.



Visual odometry



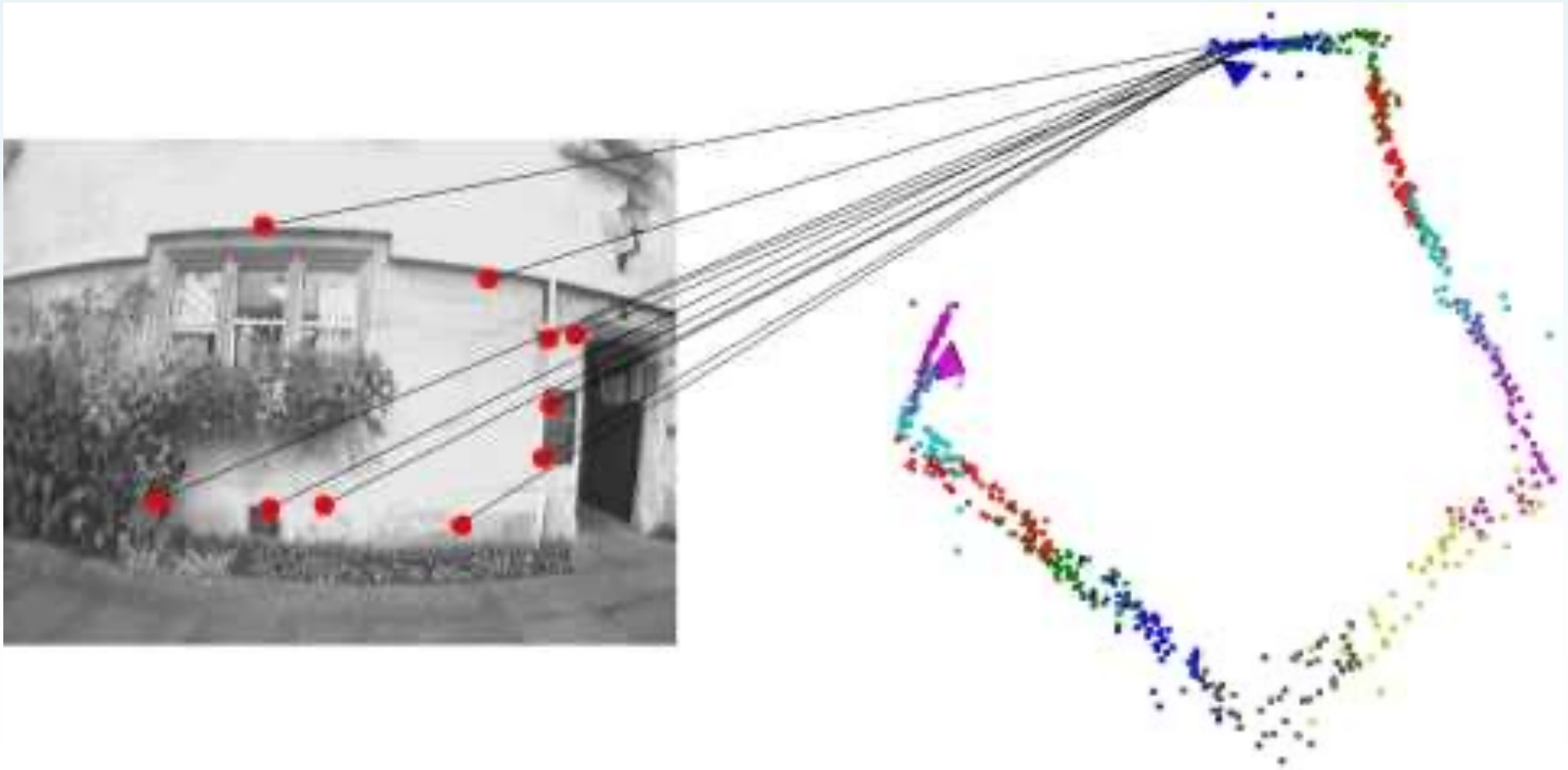
Visual SLAM

Image courtesy of Clemente et al. RSS'07

Example Loop Closing Problem

New College Data Set

Example Loop Closing Problem



Example Loop Closing Problem

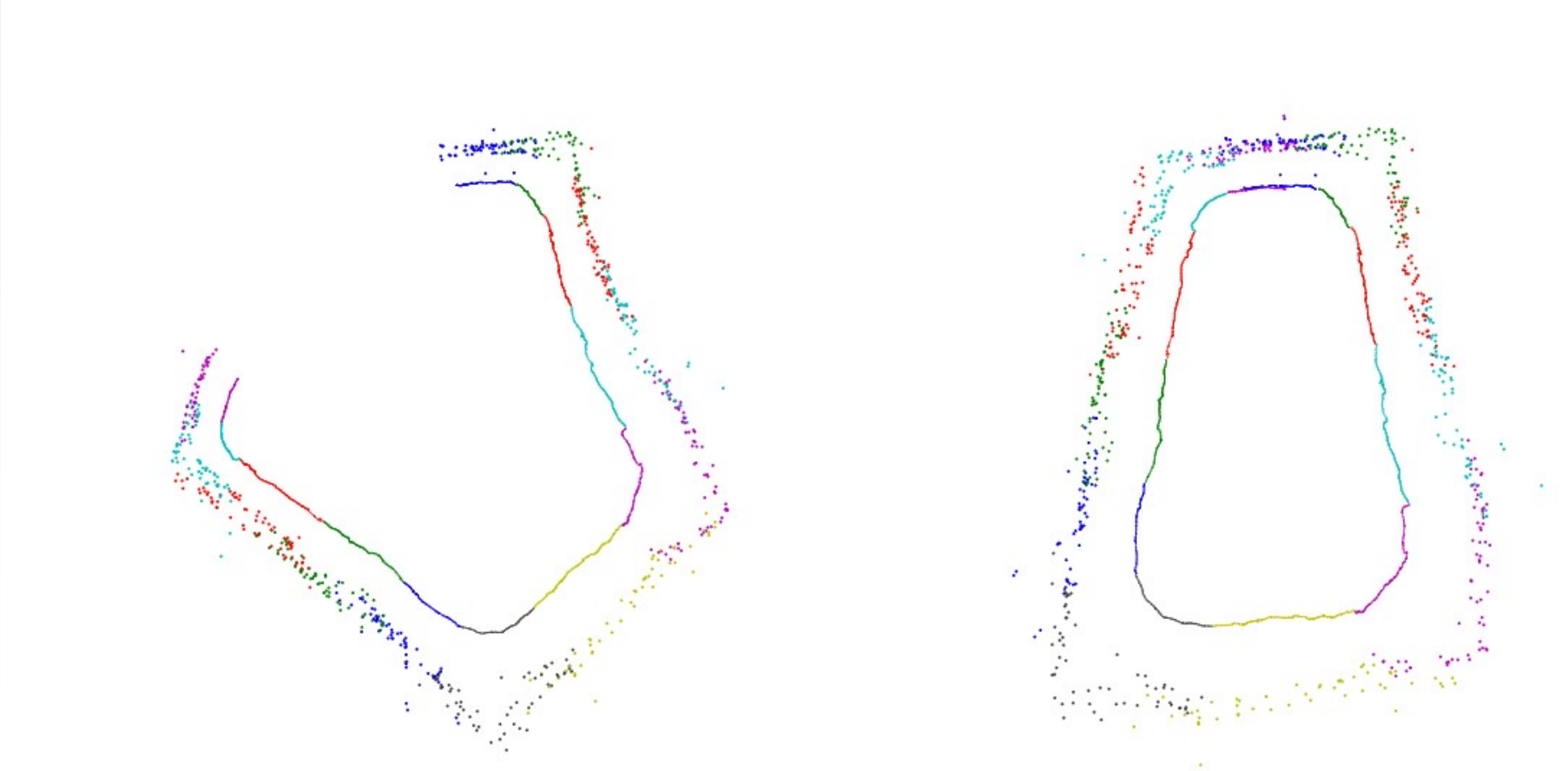


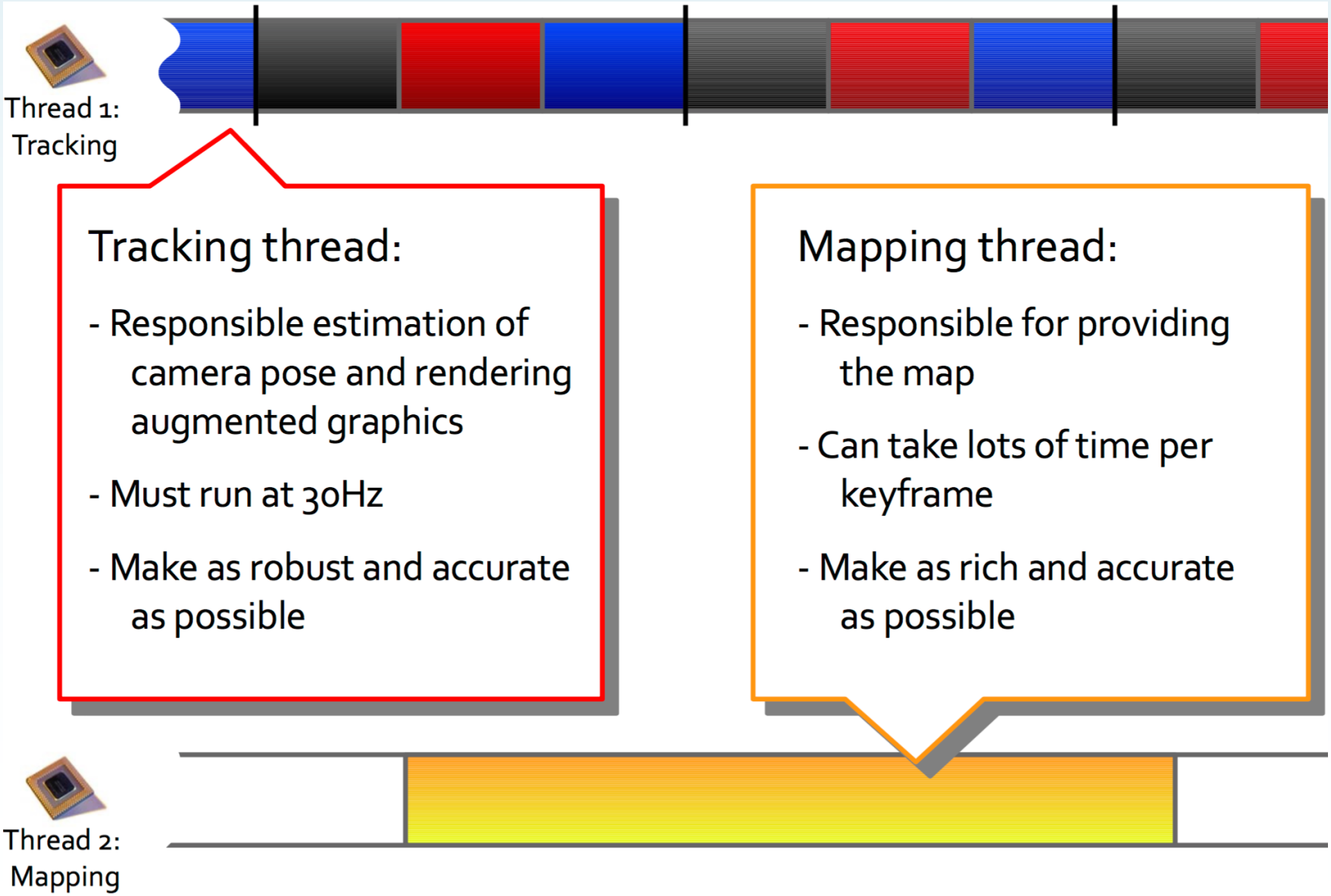
Table 1 List of different visual SLAM system. Non-filter-based approaches are highlighted in a gray color.

Year	Name	Method	Type	Reference
2003	Real-time simultaneous localization and mapping with a single camera	filter	indirect	Davison (2003)
2004	Simultaneous localization and mapping using multiple view feature descriptors	filter	indirect	Meltzer et al (2004)
2004	Real-Time 3D SLAM with Wide-Angle Vision	filter	indirect	Davison et al (2004)
2005	Real-Time Camera Tracking Using a Particle Filter	filter	indirect	Pupilli and Calway (2005)
2005	CV-SLAM	filter	indirect	Jeong and Lee (2005)
2006	Real-time Localization and 3D Reconstruction	non-filter	indirect	Mouragnon et al (2006)
2006	Scalable Monocular SLAM	filter	indirect	Eade and Drummond (2006)
2006	Real-Time Monocular SLAM with Straight Lines	filter	indirect	Smith et al (2006)
2007	Monocular-vision based SLAM using Line Segments	filter	indirect	Lemaire and Lacroix (2007)
2007	MonoSLAM	filter	indirect	Davison et al (2007)
2007	Parallel Tracking and Mapping (PTAM)	non-filter	indirect	Klein and Murray (2007)
2007	Monocular SLAM as a Graph of Coalesced Observations	filter	indirect	Eade and Drummond (2007)
2007	Mapping Large Loops with a Single Hand-Held Camera	filter	indirect	Clemente et al (2007)
2007	Dimensionless Monocular SLAM	filter	indirect	Civera et al (2007)
2008	A Square Root UKF for visual monoSLAM	filter	indirect	Holmes et al (2008)
2008	An Efficient Direct Approach to Visual SLAM	non-filter	direct	Silveira et al (2008)
2008	Efficient View-Based SLAM Using Visual Loop Closures	filter	indirect	Mahon et al (2008)
2008	Large-Scale SLAM Building Conditionally Independent Local Maps: Application to Monocular Vision	filter	indirect	Pinies and Tardos (2008)
2009	Towards a robust visual SLAM approach: Addressing the challenge of lifelong operation	filter	indirect	Hochdorfer and Schlegel (2009)
2009	Use a Single Camera for Simultaneous Localization And Mapping with Mobile Object Tracking in dynamic environments	filter	indirect	Migliore et al (2009)
2010	On Combining Visual SLAM and Visual Odometry	filter	indirect	Williams and Reid (2010)
2010	Scale Drift-Aware Large Scale Monocular SLAM	non-filter	indirect	Strasdat et al (2010a)
2010	Live dense reconstruction with a single moving camera	non-filter	hybrid	Newcombe and Davison (2010)
2010	Monocular SLAM with locally planar landmarks via geometric rao-blackwellized particle filtering on Lie groups	filter	indirect	Kwon and Lee (2010)

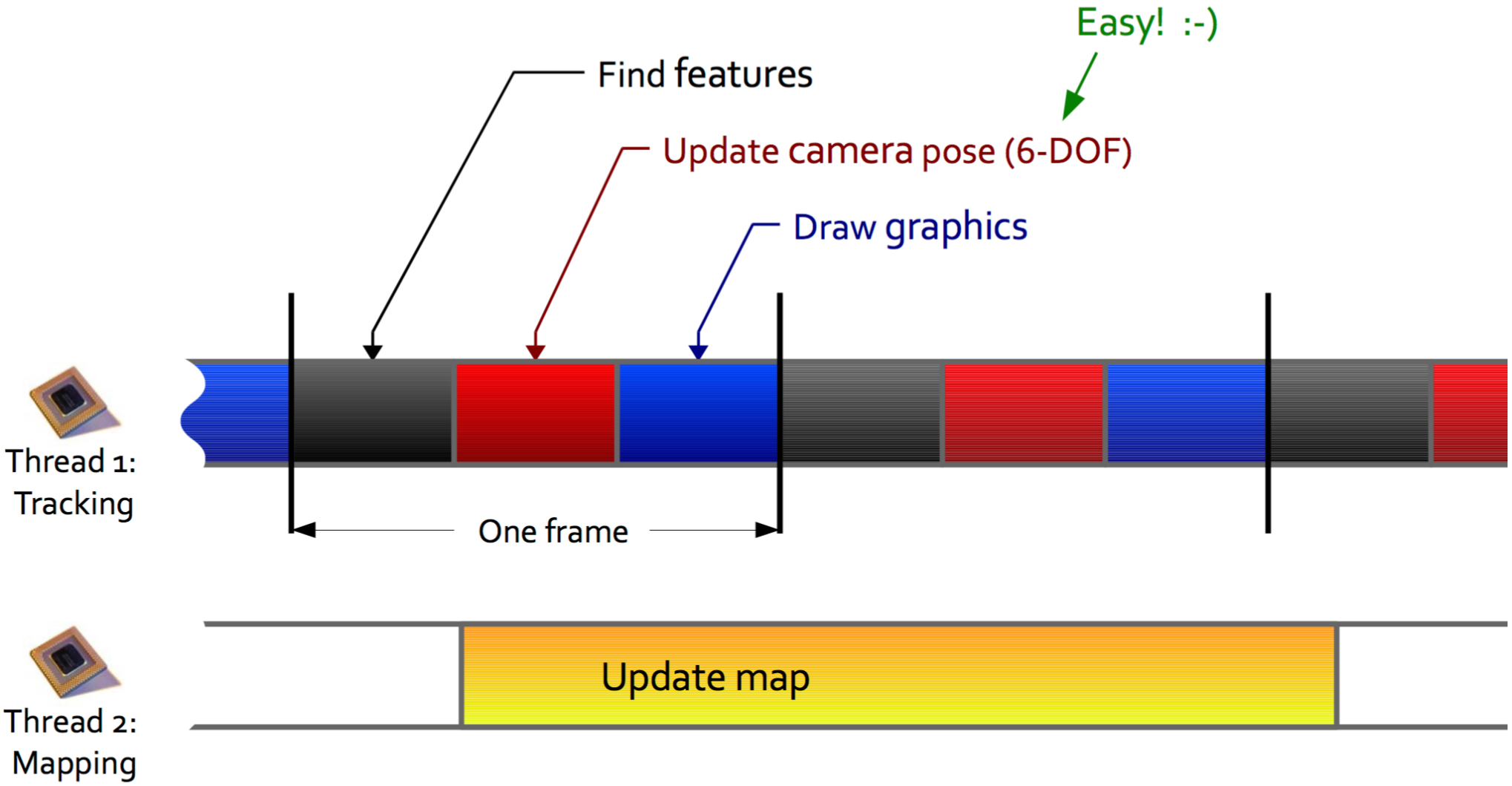
Visual SLAM

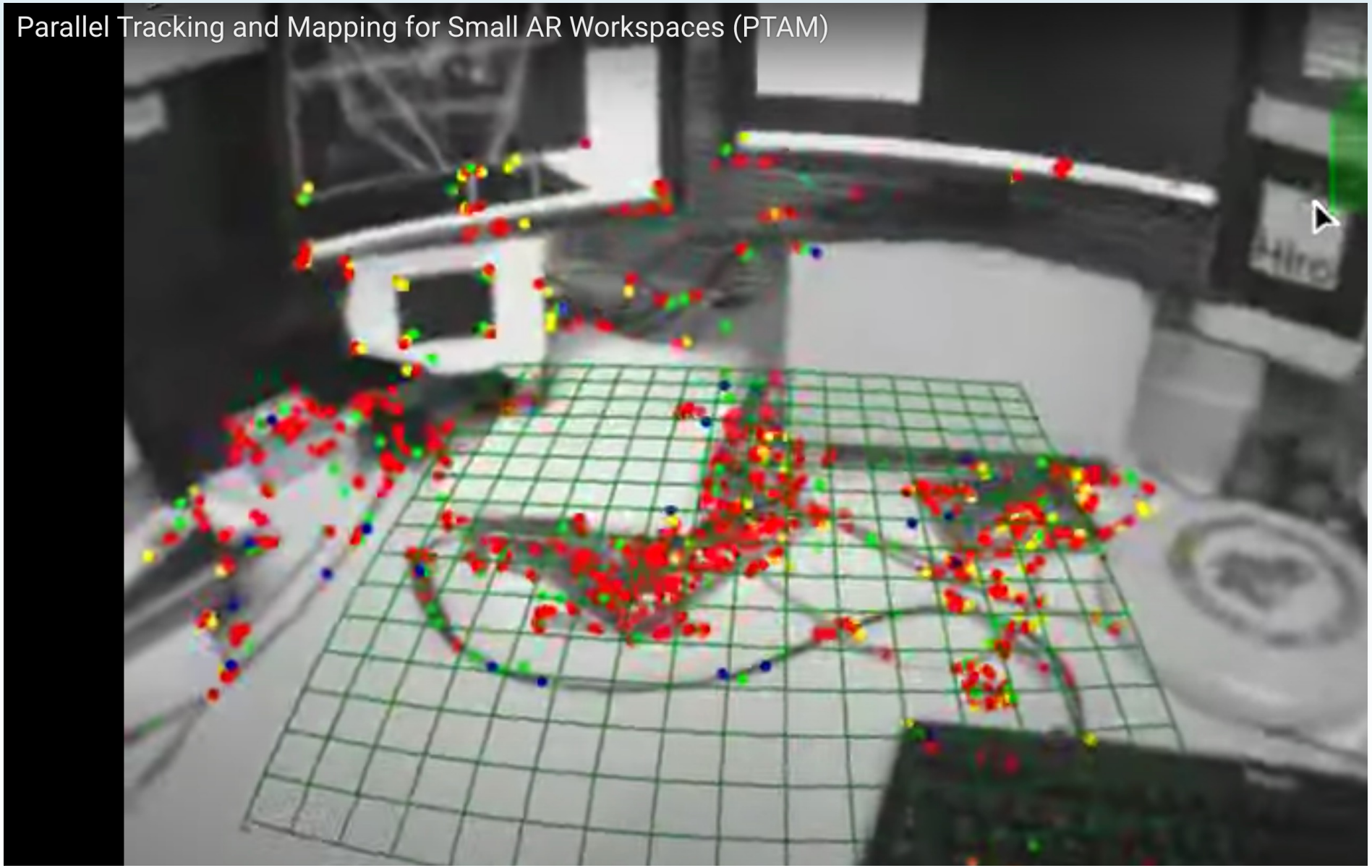
2011	Dense Tracking and Mapping (DTAM)	non-filter	direct	Newcombe et al (2011)
2011	Omnidirectional dense large-scale mapping and navigation based on meaningful triangulation	non-filter	direct	Pretto et al (2011)
2011	Continuous localization and mapping in a dynamic world (CD SLAM)	non-filter	indirect	Pirker et al (2011)
2011	Online environment mapping	non-filter	indirect	Lim et al (2011)
2011	Homography-based planar mapping and tracking for mobile phones	non-filter	indirect	Pirchheim and Reitmayr (2011)
2013	Robust monocular SLAM in Dynamic environments (RD SLAM)	non-filter	indirect	Tan et al (2013)
2013	Handling pure camera rotation in keyframe-based SLAM (Hybrid SLAM)	non-filter	indirect	Pirchheim et al (2013)
2013	Monocular Vision SLAM for Indoor Aerial Vehicles	filter	indirect	Celik and Somani (2013)
2014	Visual SLAM for Handheld Monocular Endoscope	filter	indirect	Grasa et al (2014)
2014	Real-time camera tracking using a particle filter combined with unscented Kalman filters	filter	indirect	Lee (2014)
2014	Semi-direct Visual Odometry (SVO)	non-filter	hybrid	Forster et al (2014)
2014	Large Scale Direct monocular SLAM (LSD SLAM)	non-filter	direct	Engel et al (2014)
2014	Deferred Triangulation SLAM (DT SLAM)	non-filter	indirect	Herrera et al (2014)
2014	Real-Time 6-DOF Monocular Visual SLAM in a Large Scale Environment	non-filter	indirect	Lim et al (2014)
2015	StructSLAM: Visual SLAM With Building Structure Lines	filter	indirect	Zhou et al (2015)
2015	Robust large scale monocular Visual SLAM	non-filter	indirect	Bourmaud and Megret (2015)
2015	ORB SLAM	non-filter	indirect	Mur-Artal et al (2015)
2015	Dense Piecewise Parallel Tracking and Mapping (DPPTAM)	non-filter	direct	Concha and Civera (2015)

PTAM


Thread 1:
Tracking
Tracking thread:
- Responsible estimation of camera pose and rendering augmented graphics
- Must run at 30Hz
- Make as robust and accurate as possible
Mapping thread:
- Responsible for providing the map
- Can take lots of time per keyframe
- Make as rich and accurate as possible
Thread 2:
Mapping

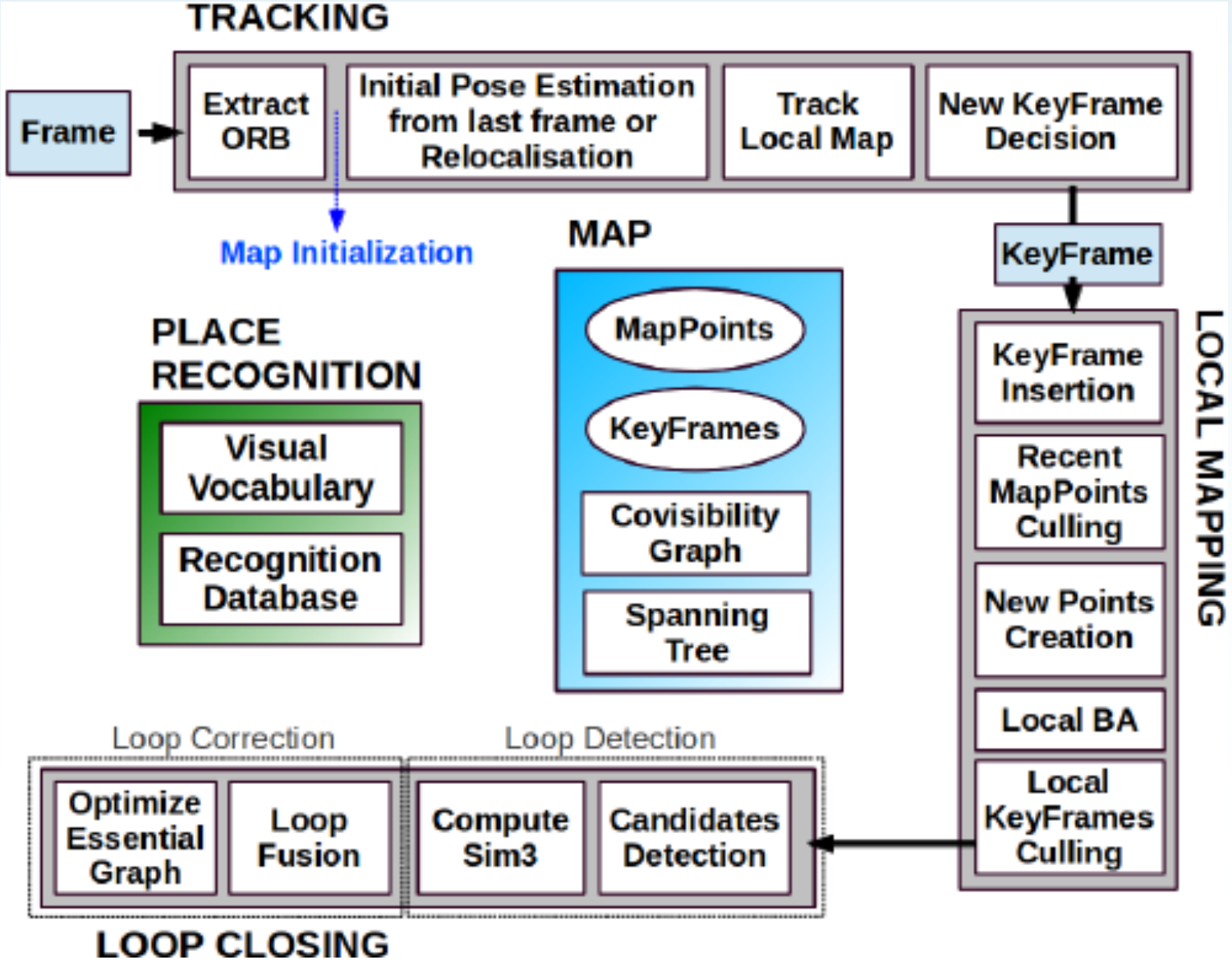
PTAM


Easy! :-)
Find features
Update camera pose (6-DOF)
Draw graphics
Thread 1:
Tracking
One frame
Update map
Thread 2:
Mapping

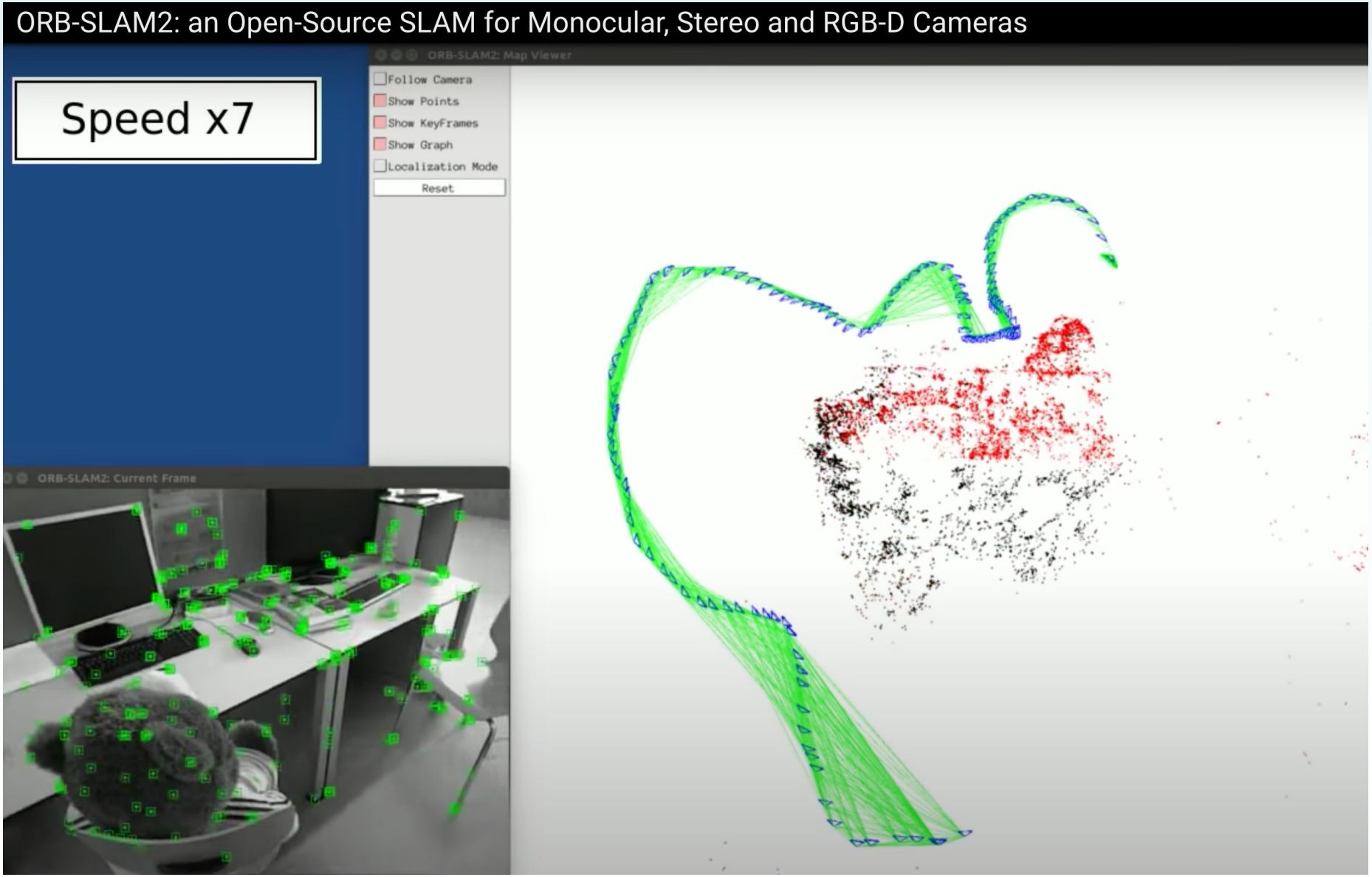


https://www.youtube.com/watch?v=F3s3M0mokNc

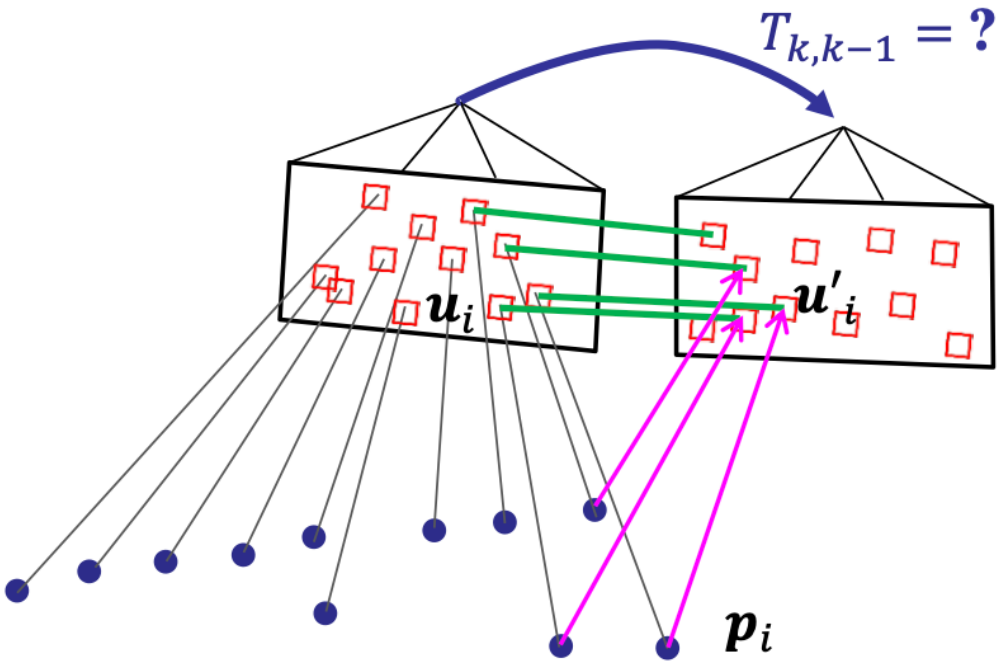
ORB-SLAM https://www.youtube.com/watch?v=ufvPS5wJAx0



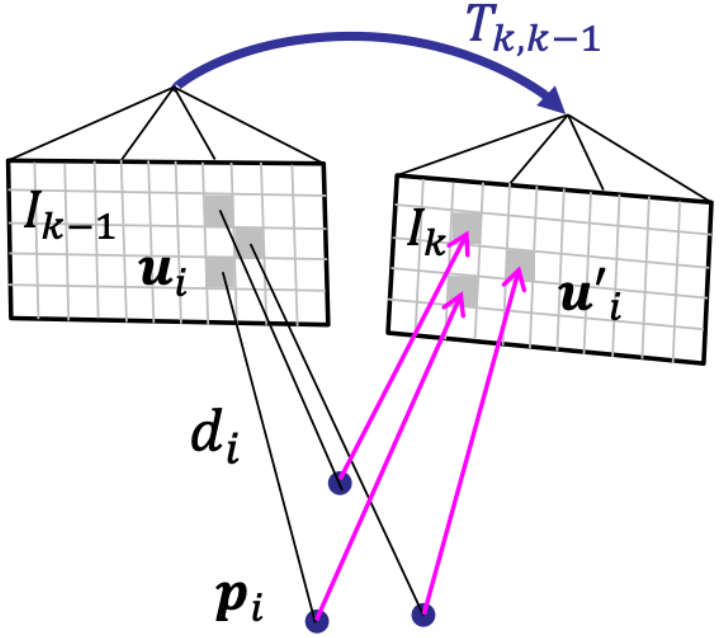
ORB-SLAM https://www.youtube.com/watch?v=ufvPS5wJAx0



Feature-based methods

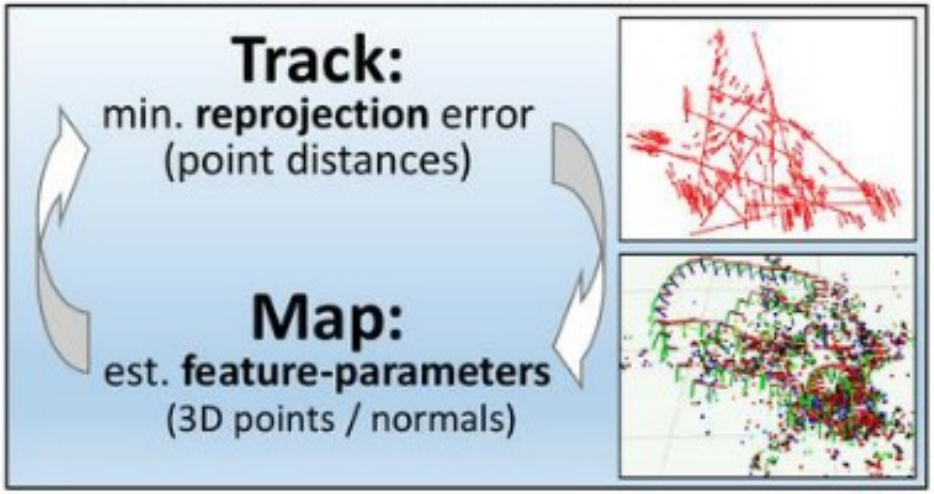


Direct methods



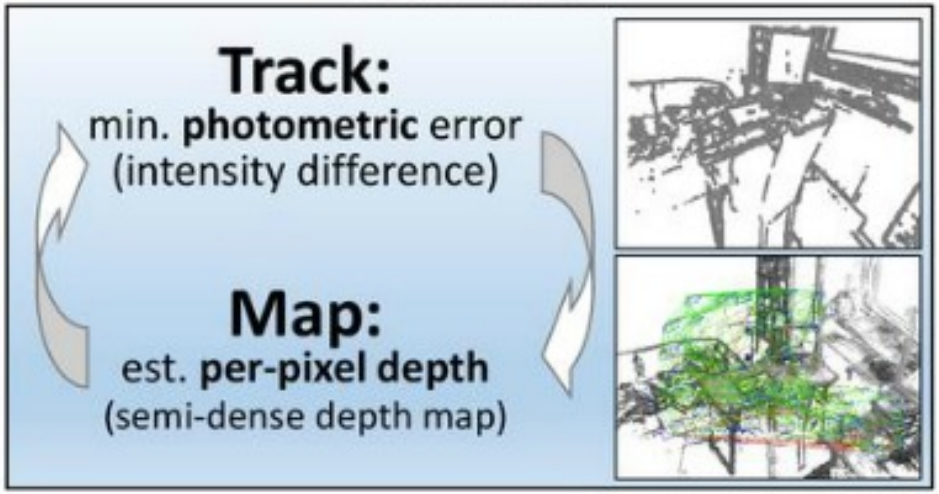
Keypoint-Based

abstract images to feature observations



Direct Dense SLAM

keep full image

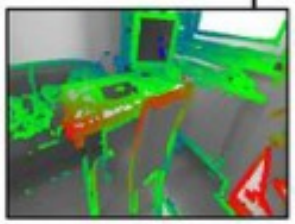


Direct Methods minimise photometric cost

$$E(\boldsymbol{\xi}) = \sum_{\mathbf{x} \in \Omega_{\mathrm{KF}}} \left(I_{\mathrm{KF}}(\mathbf{x}) - I(\omega(\mathbf{x}, D_{\mathrm{KF}}(\mathbf{x}), \boldsymbol{\xi})) \right)^2 =: \|\mathbf{r}(\boldsymbol{\xi})\|_2^2$$

Camera Pose in $\mathfrak{se}(3)$

KF image

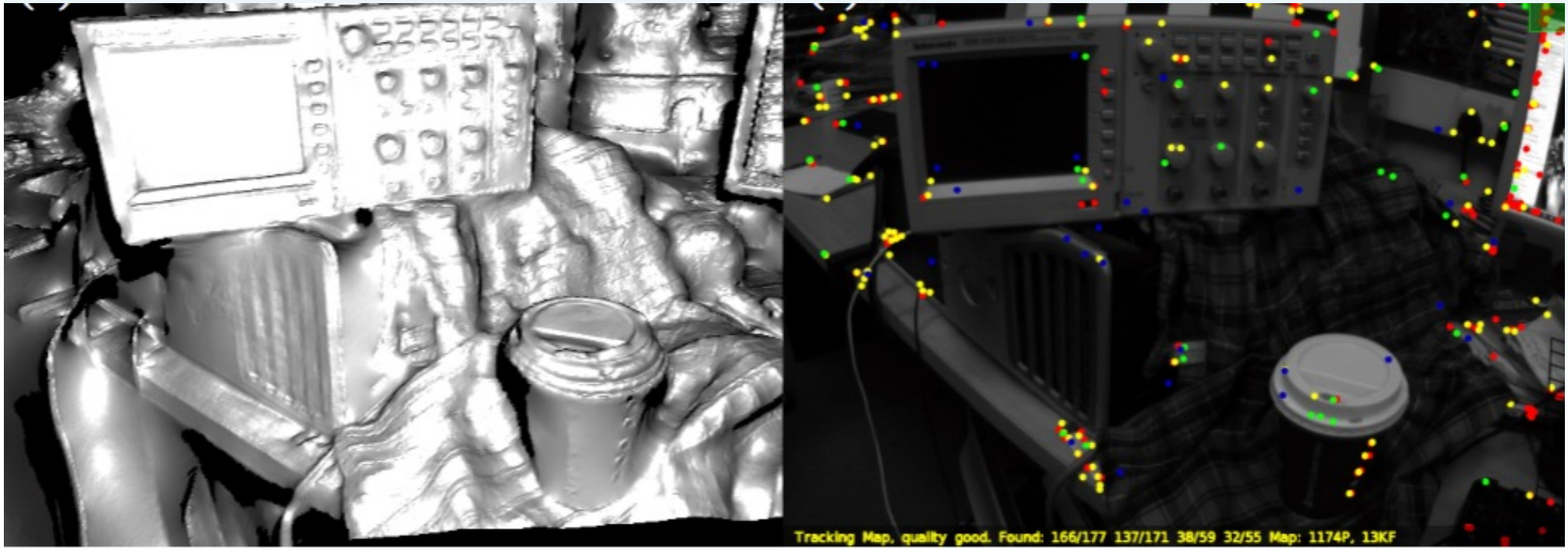


KF depth

back-warped new frame

- minimize using **Gauss-Newton** Algorithm (≈ forward-compositional Lucas-Kanade)

Why direct methods?

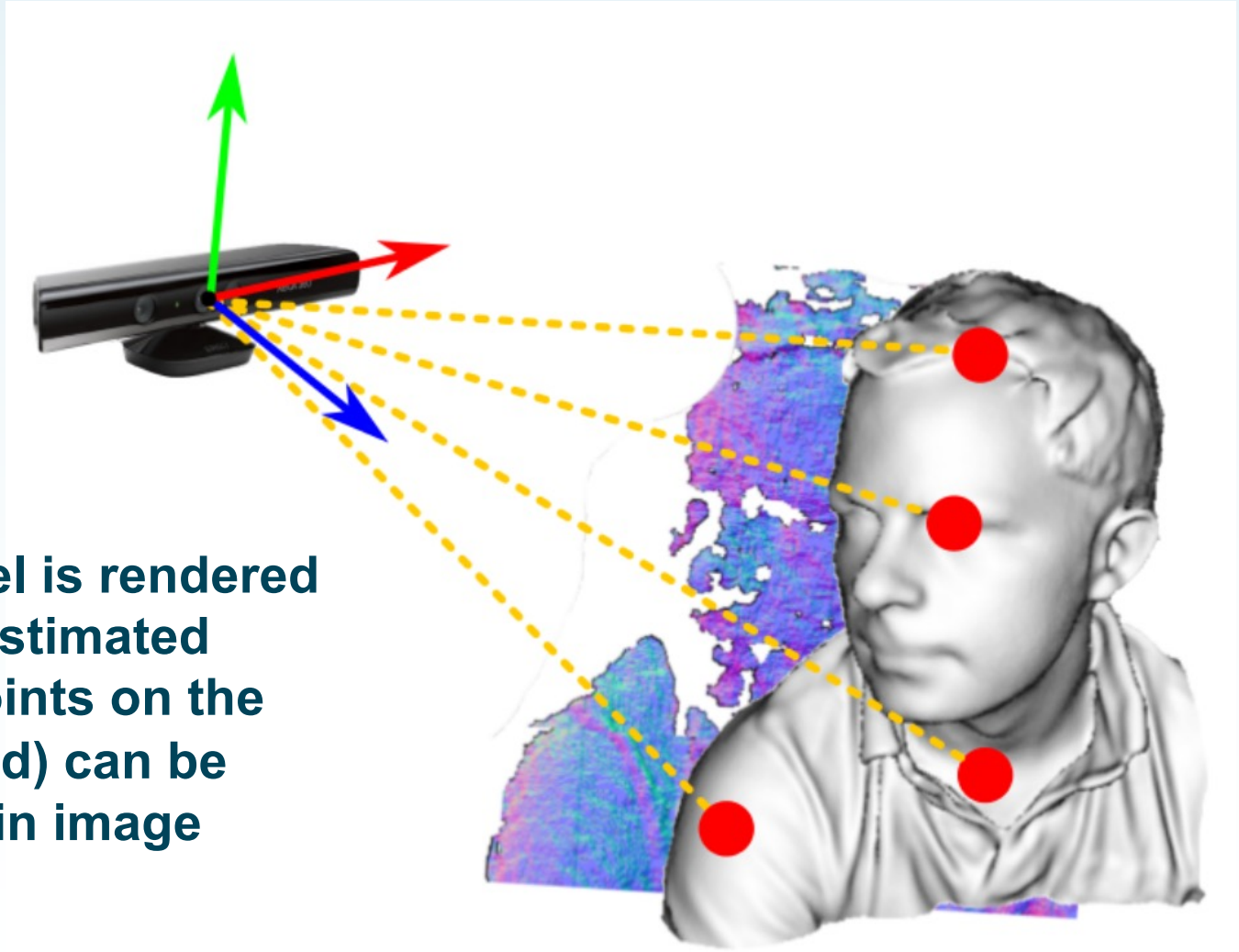


DTAM– **Dense**
Newcombe et al.
ICCV'11

PTAM– **Sparse**
Klein and Murray,
ISMAR'07

KinectFusion -- Depth Only Camera Tracking (ICP)

Newcombe et al. ISMAR'11

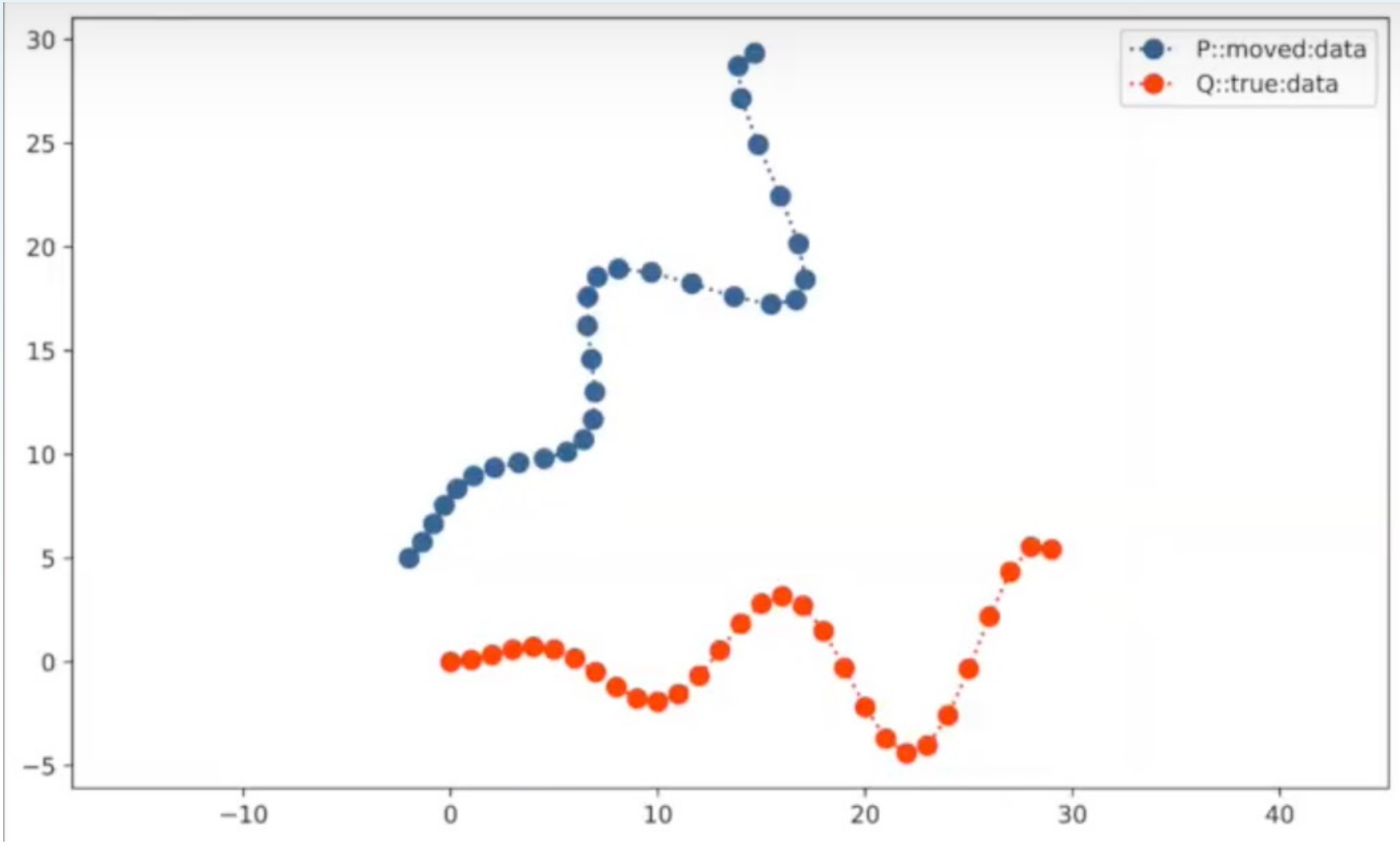


The model is rendered into the estimated frame. Points on the model (red) can be sampled in image space.

Slide acknowledgement R. Newcombe

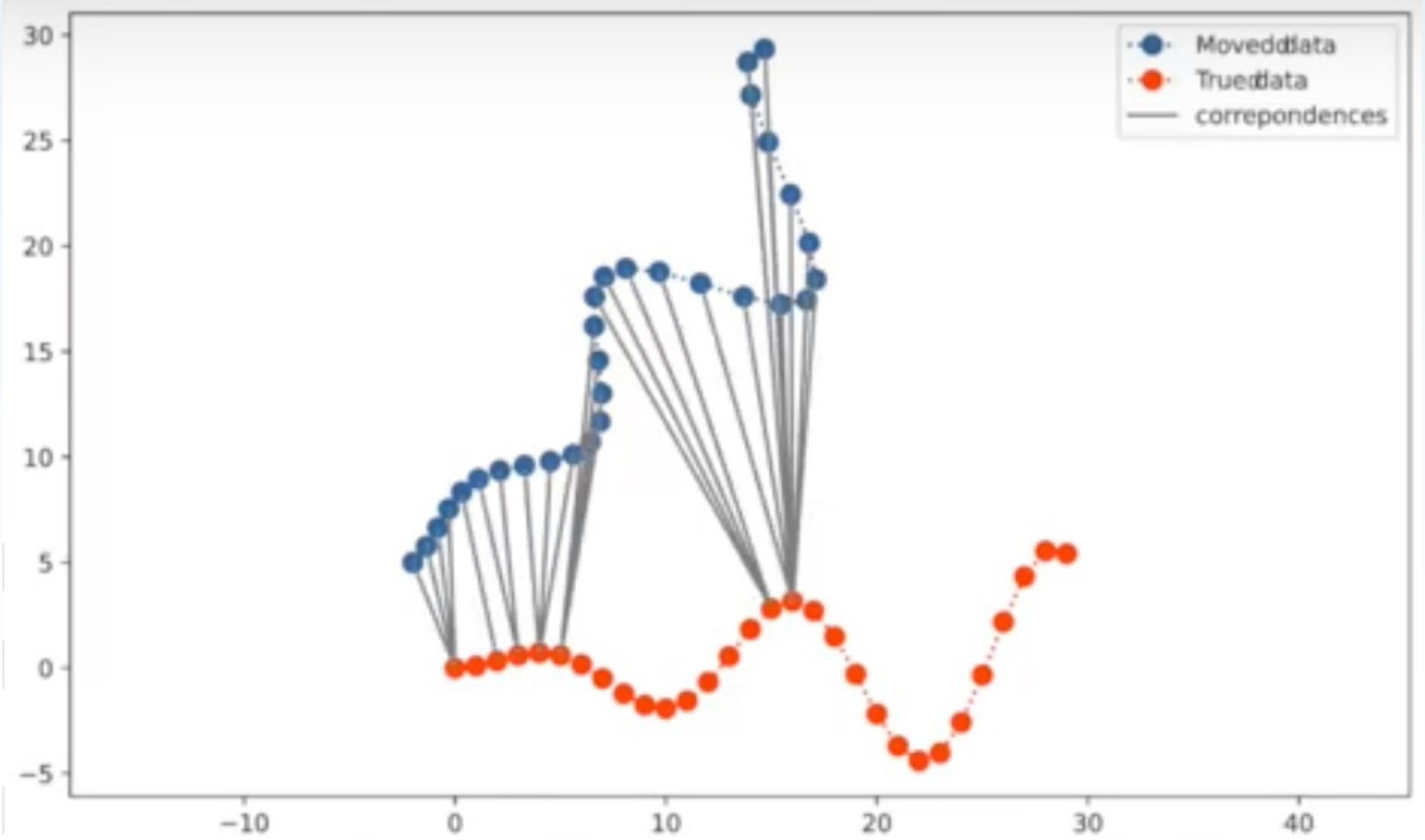
https://www.youtube.com/watch?v=KOUSSIKUJ-A

Iterative Closest Point



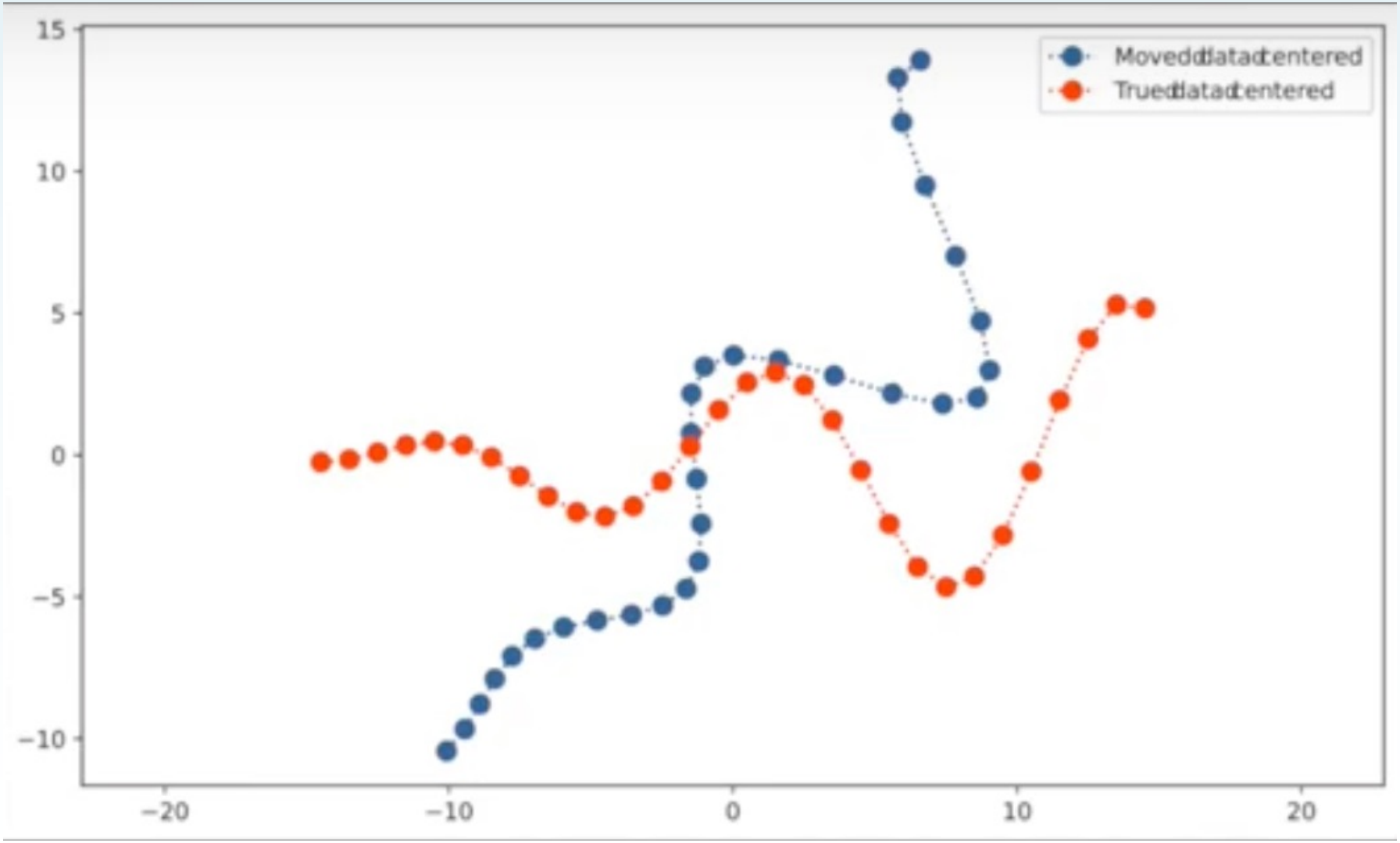
We want to align two point clouds (estimate R and T)

ICP – Step1: Data Association



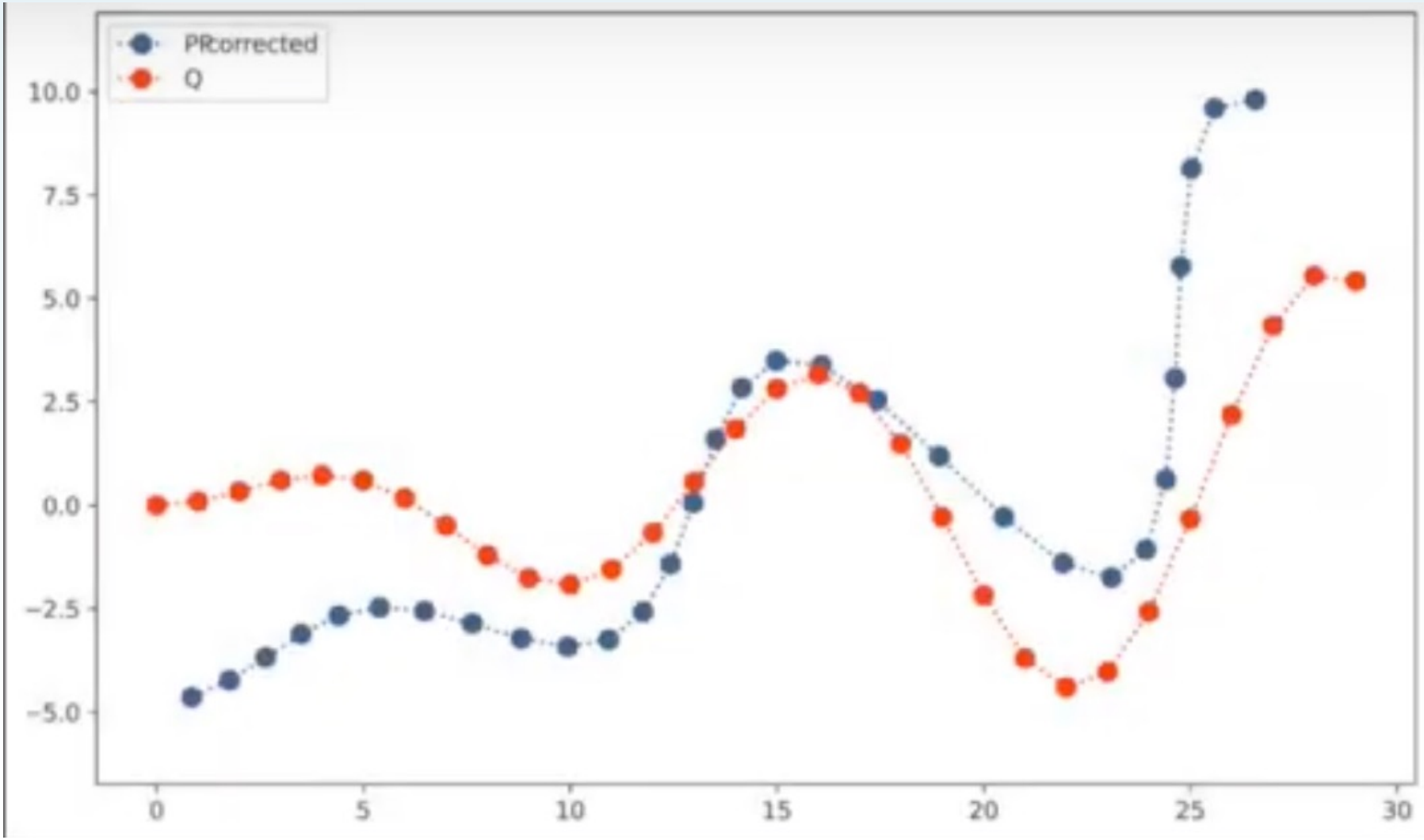
Find closest point via Nearest Neighbours

ICP – Compute Transformation

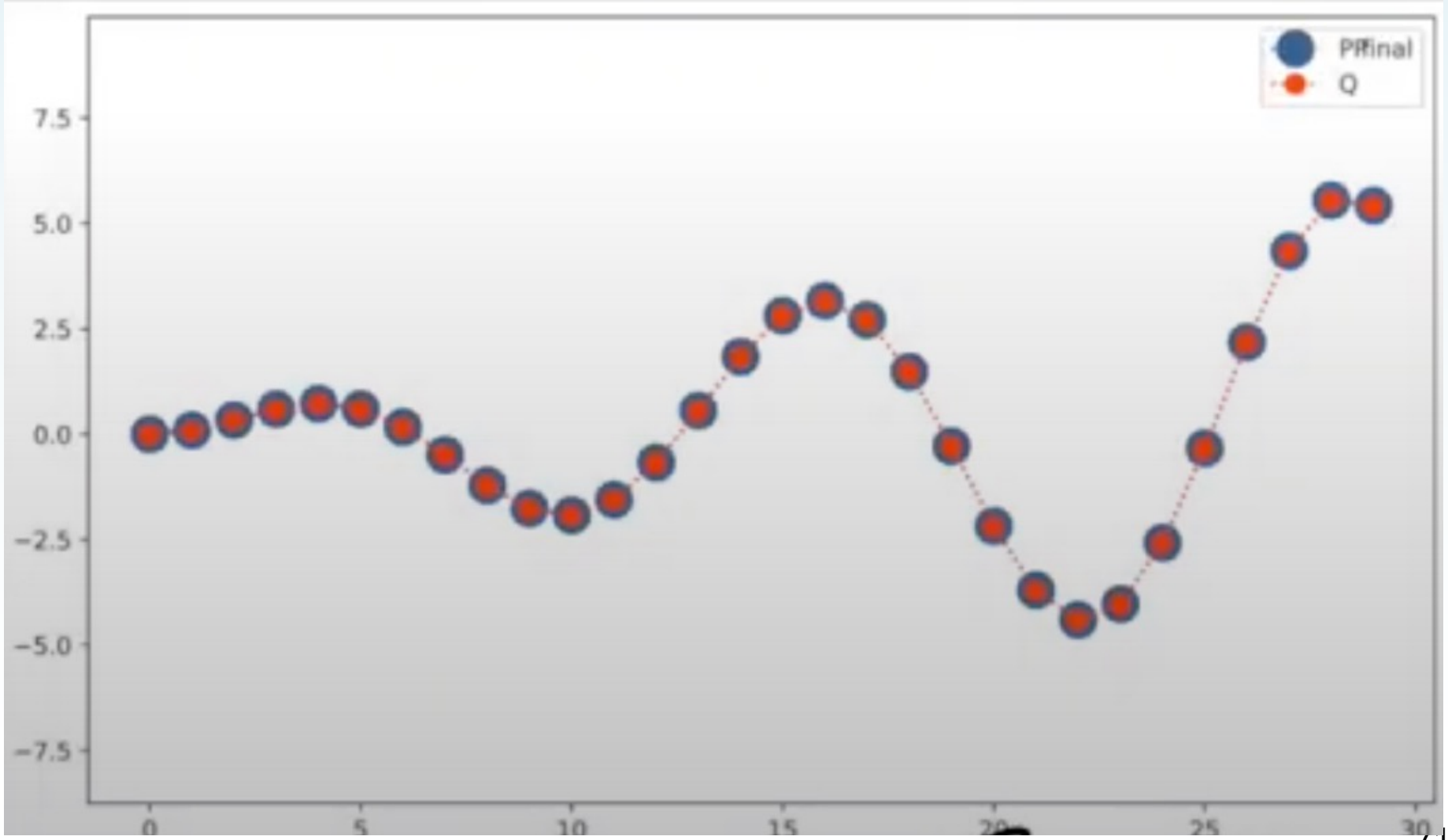


1. Align Centroids 2. Compute rotation via SVD

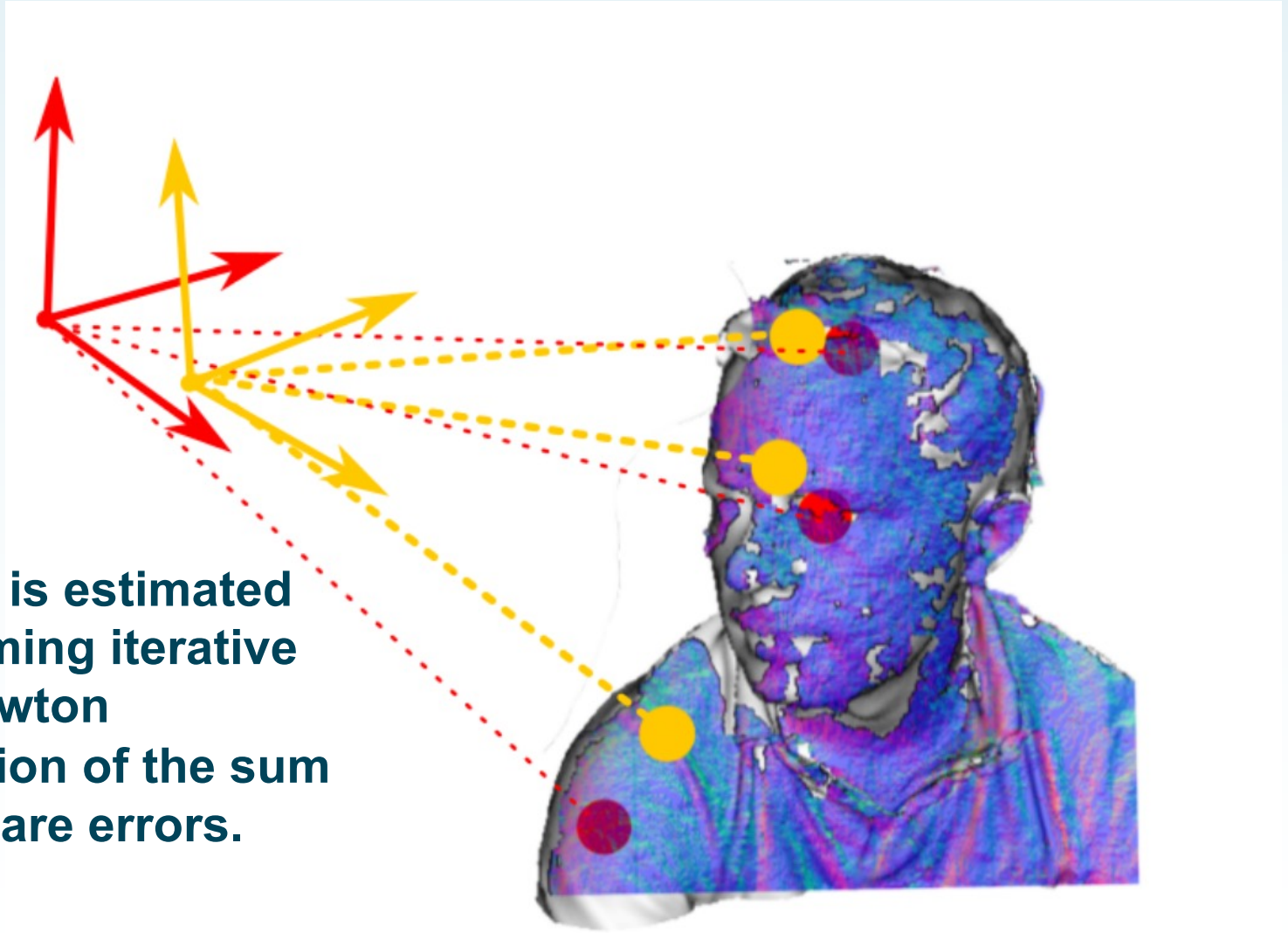
ICP – Iterate Steps 1 and 2



ICP – Until alignment error is minimized



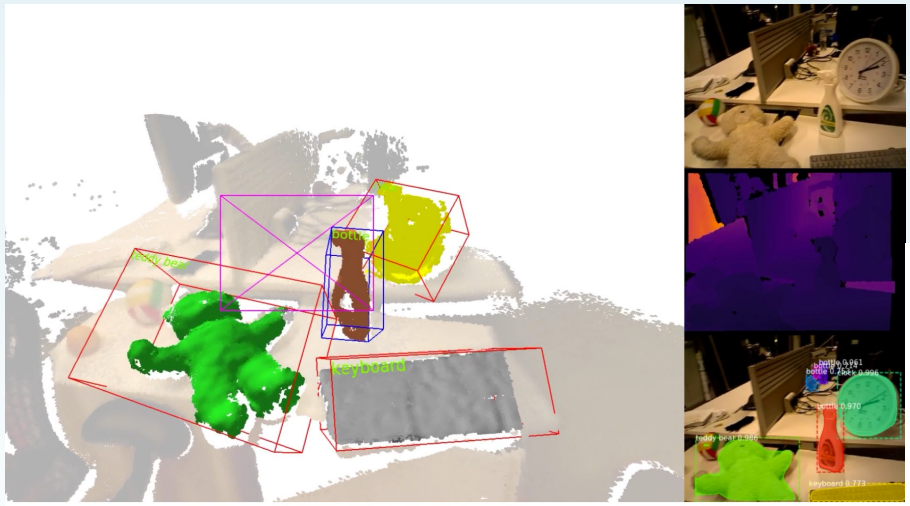
KinectFusion -- Depth Only Camera Tracking (ICP)



New pose is estimated by performing iterative Gauss-Newton minimisation of the sum of the square errors.

Slide acknowledgement R. Newcombe

Recognition meets 3D Reconstruction



Semantic priors

MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects

Martin Runz, Maud Buffier, Lourdes Agapito ISMAR'18

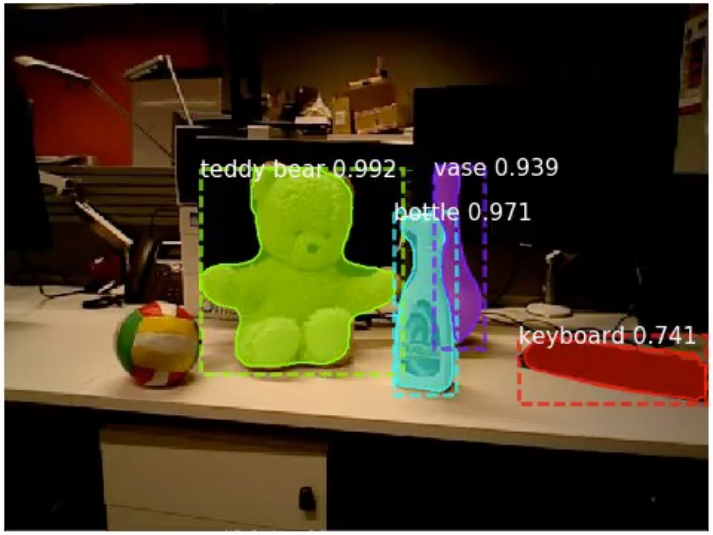
Semantic + shape priors

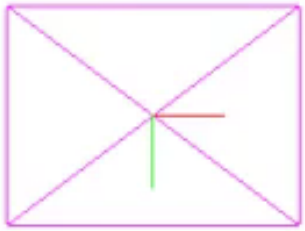
FroDO: From Detections to Objects

Runz et al. CVPR'20

Input

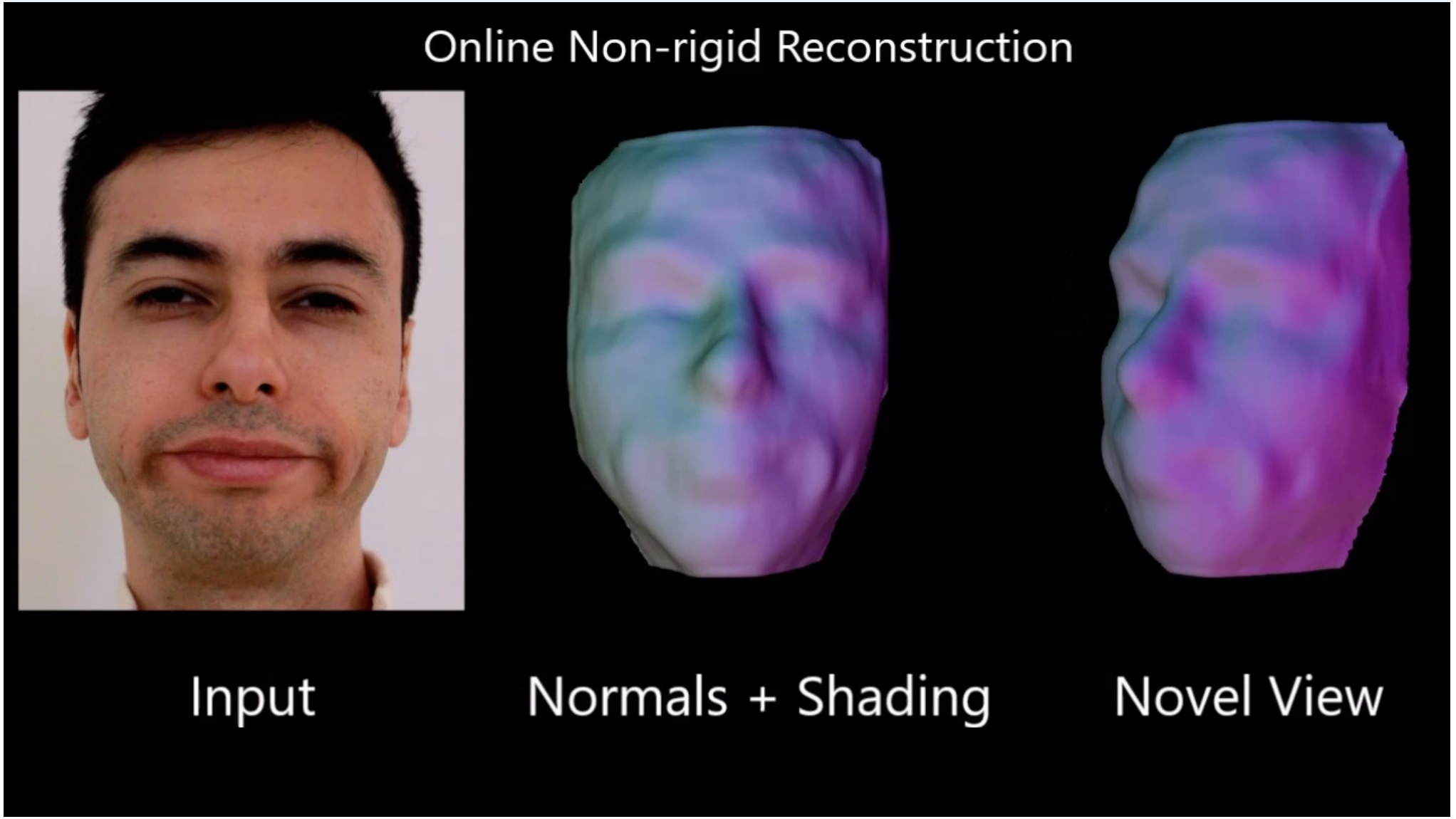
Contribution - Object Based Dynamic SLAM





Monocular Dense Direct and Deformable

with R. Yu, N. Campbell and C. Russell (ICCV'15)



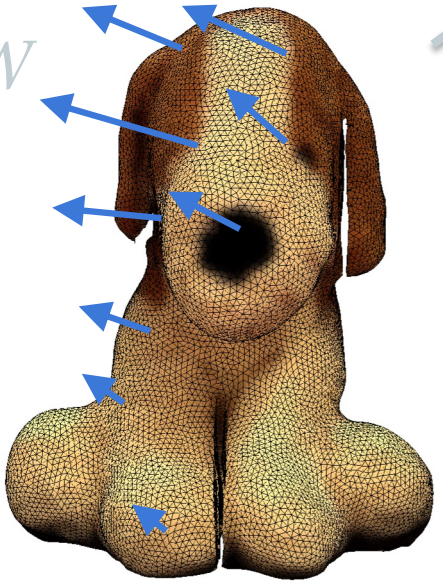
Monocular Dense Direct and Deformable

current live frame

with R. Yu, N. Campbell and C. Russell (ICCV'15)

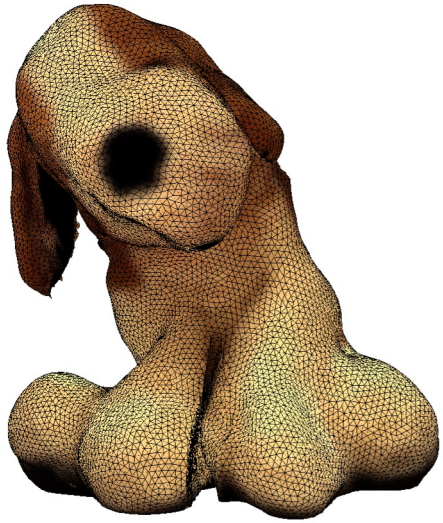
w

3D warp field



w

template



$$E_{(w,R,t)} = E_{data} + E_{reg}$$