Inside-out (Visual) Tracking

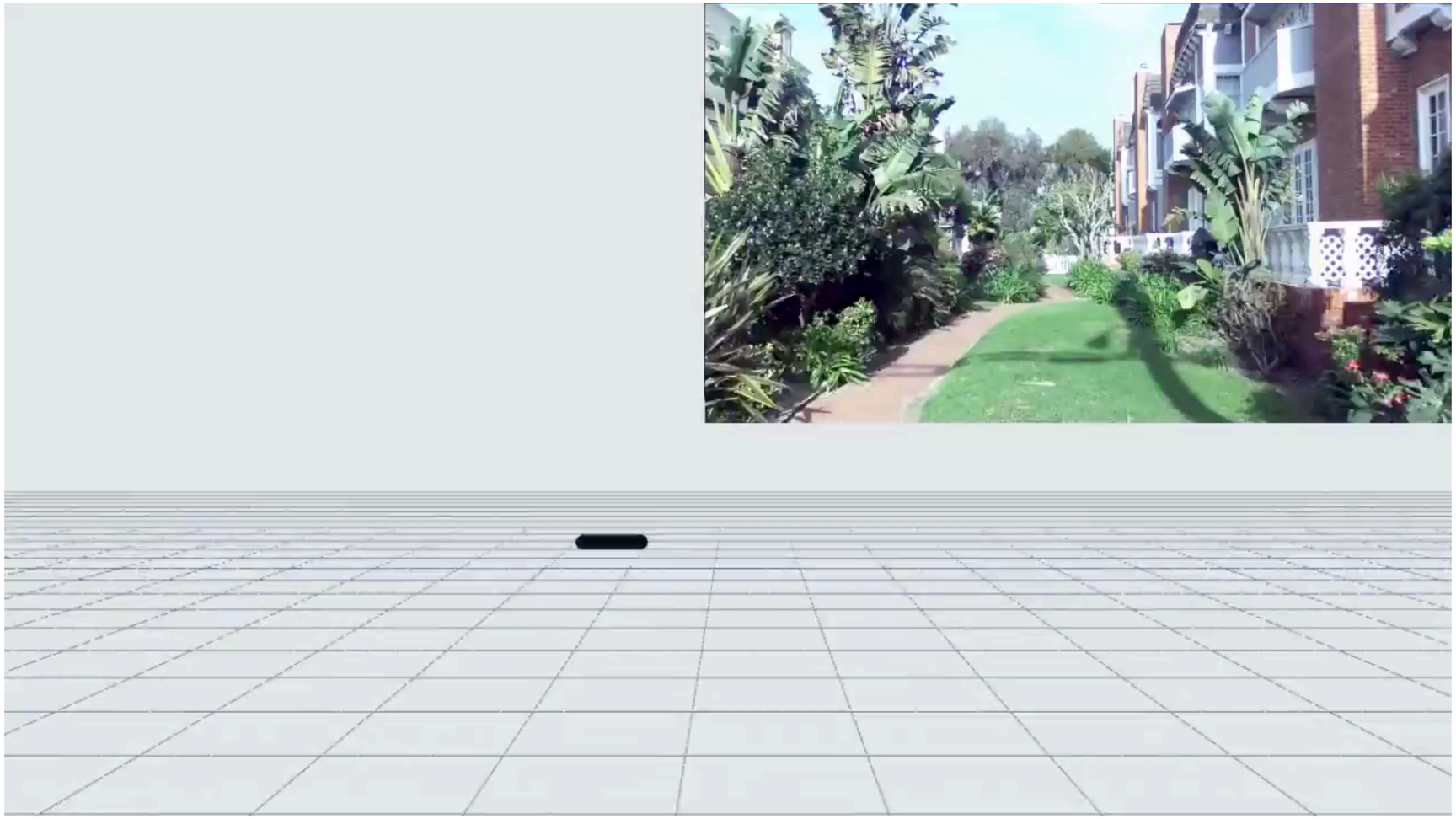
Avinash Sharma
Center for Visual IT, KCIS, IIIT Hyderabad

Content is largely borrowed from vSLAM tutorial by Prof. L. Freda (University of Rome)

INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

CVIT

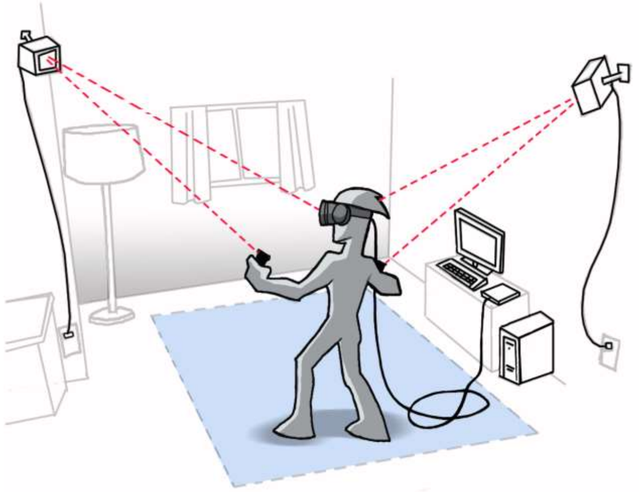
Positional Tracking



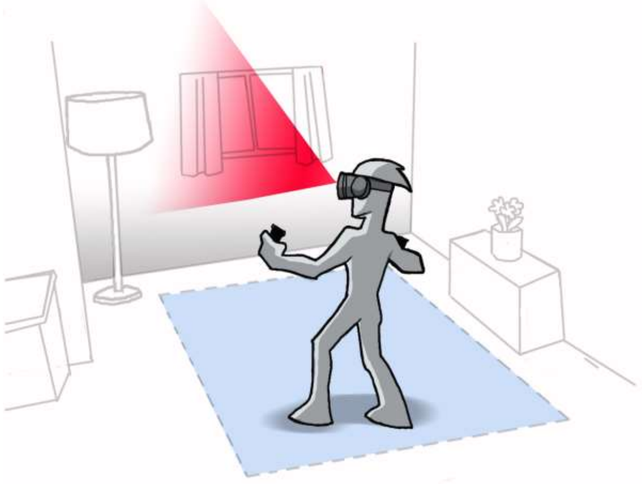
https://www.stereolabs.com/

Positional Tracking

- **Outside-in Tracking**: External sensors, cameras, or markers are required.



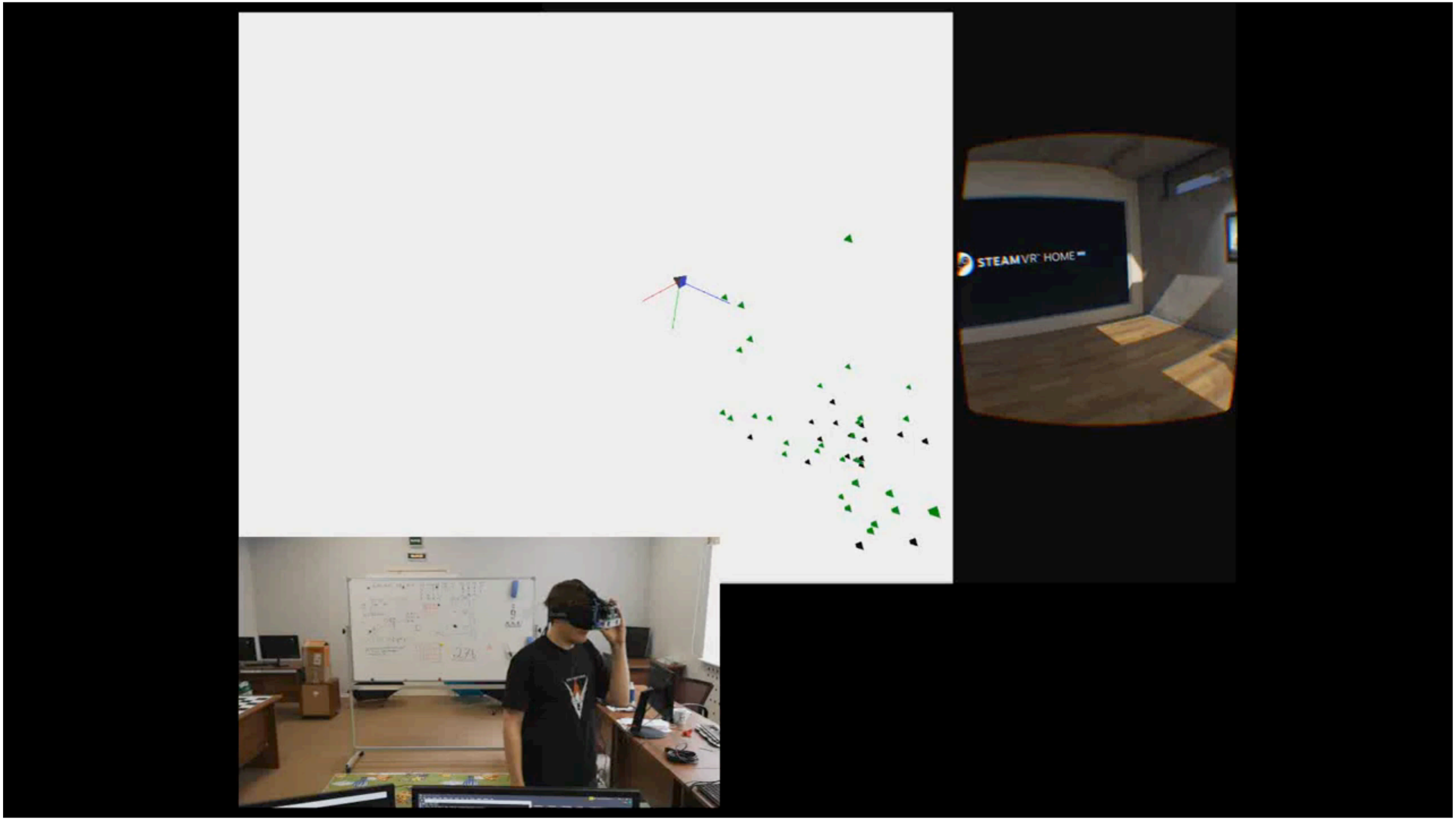
- **Inside-out Tracking**: Camera or sensor is located on HMD, no need for other external devices to do tracking.



Inside-out (Visual) Tracking

- **Unknown space**
- **Uncontrolled camera (6DoF motion)**
- **Real-time**
- **Drift-free**

https://www.youtube.com/watch?v=6S1la9SXntM



https://www.youtube.com/watch?v=6S1la9SXntM

Inside-out (Visual) Tracking

Two key steps:

1. **Mapping the unknown 3D world**
2. **Localizing the ego camera pose**

Doing it together:
“visual Simultaneous Localization and Mapping (vSLAM)”

vSLAM

Mapping – "What does the world look like?"
Integration of the information gathered with sensors into a given representation.

Localization – "Where am I?"
Estimation of the robot pose relative to a map. Typical problems:
(i) pose tracking, where the initial pose of the vehicle is known
(ii) global localization, where no a priori knowledge about the starting position is given.

Simultaneous localization and mapping (**SLAM**)
Build a map while at the same time localizing the robot within that map. The *chicken and egg problem*: A good map is needed for localization while an accurate pose estimate is needed to build a map.

Visual SLAM: SLAM by using *visual* sensors such as monocular cameras, stereo rigs, RGB-D cameras, DVS, etc

vSLAM System Overview

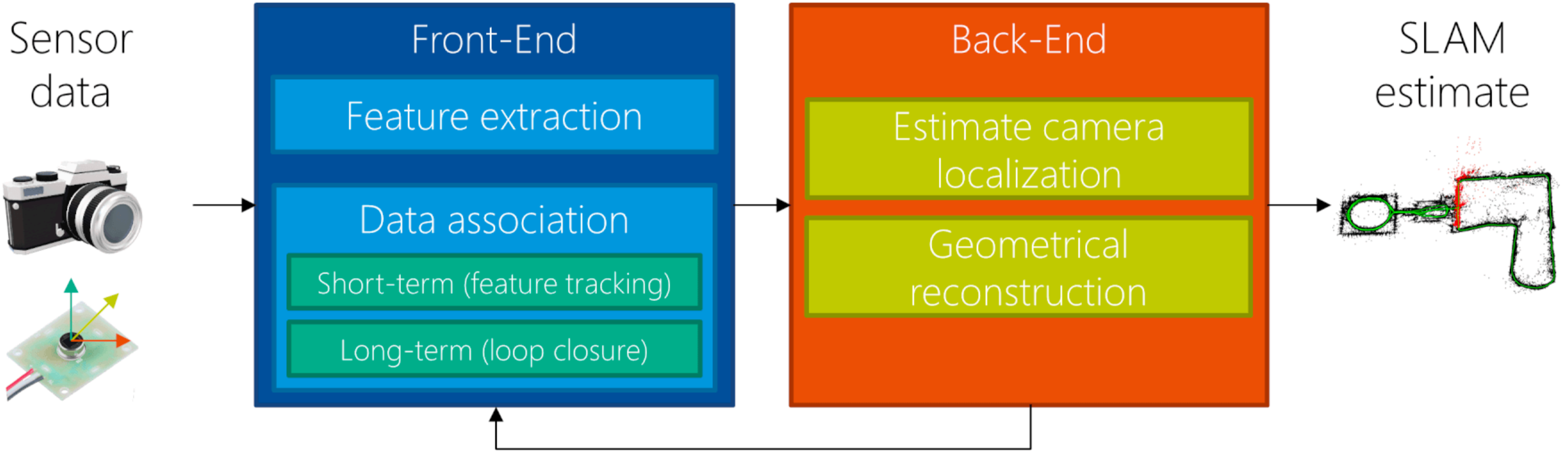


Diagram based on: Cadena, Cesar, et al. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age." *IEEE Transactions on Robotics* 32.6 (2016): 1309-1332.
SLAM estimate: R. Mur-Artal, J. M. M. Montiel and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," in *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147-1163, Oct. 2015.

vSLAM System Overview

1. **Sensor data:** Usually includes the camera, accelerometer and gyroscope. It might be augmented by other sensors like GPS, light sensor, depth sensors, etc.

2. **Front-End:** The first step is feature extraction. These features also need to be associated with landmarks – keypoints with a 3D position, also called *map points*. In addition, map points need to be tracked in a video stream.
   Long-term association reduces drift by recognizing places that have been encountered before (loop closure).

3. **Back-End:** Takes care of establishing the relationship between different frames, localizing the camera (pose model), as well as handling the overall geometrical reconstruction. Some algorithms create a sparse reconstruction (based on the keypoints). Others try to capture a dense 3D point cloud of the environment.

4. **SLAM estimate:** The result containing the tracked features, their locations & relations, as well as the camera position within the world.

vSLAM System Overview

Multi-Trajectory Pose Correspondences using Scale-Dependent Topological Analysis of Pose-Graphs

Sayantan Datta Avinash Sharma K Madhava Krishna

International Institute of Information Technology, Hyderabad

Why vSLAM

Why using a camera?

- Vast information
- Extremely low Size, Weight, and Power (SWaP) footprint
- Cheap and easy to use
- Passive sensor

Challenge

- We need power efficiency for truly capable always-on tiny devices; or to do much more with larger devices

Question

- How does the human brain achieve always-on, dense, semantic vision with very limited power?

Visual Odometry (VO)

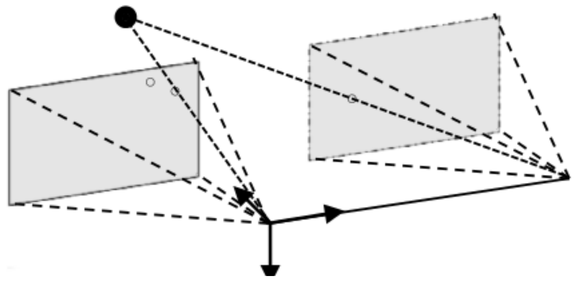
- An agent is moving through the environment and taking images with a rigidly-attached camera system at discrete times k
- In case of a **monocular system**, the set of images taken at times k is denoted by

$$I_{l,0:n} = \{I_0, ..., I_n\}$$

- In case of a **stereo system**, the set of images taken at times k is denoted by

$$I_{l,0:n} = \{I_{l,0}, ..., I_{l,n}\}$$
$$I_{r,0:n} = \{I_{r,0}, ..., I_{r,n}\}$$

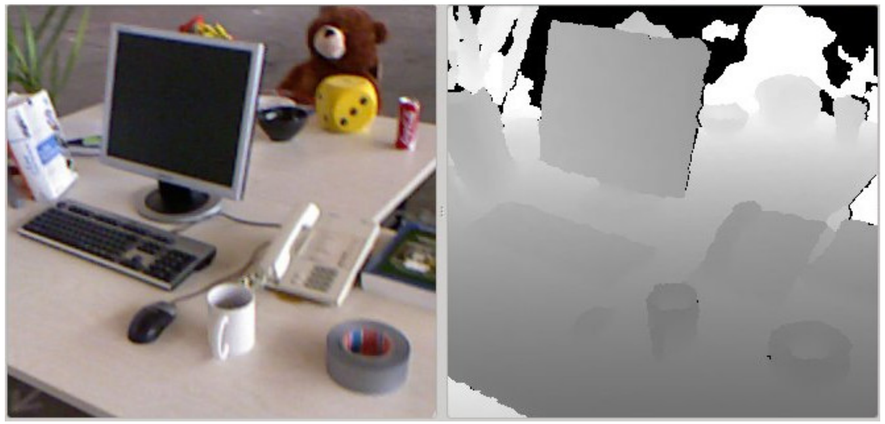
In this case, without loss of generality, the coordinate system of the left camera can be used as the origin



Visual Odometry (VO)

- In case of a **RGB-D camera**, the set of images taken at times k is denoted by

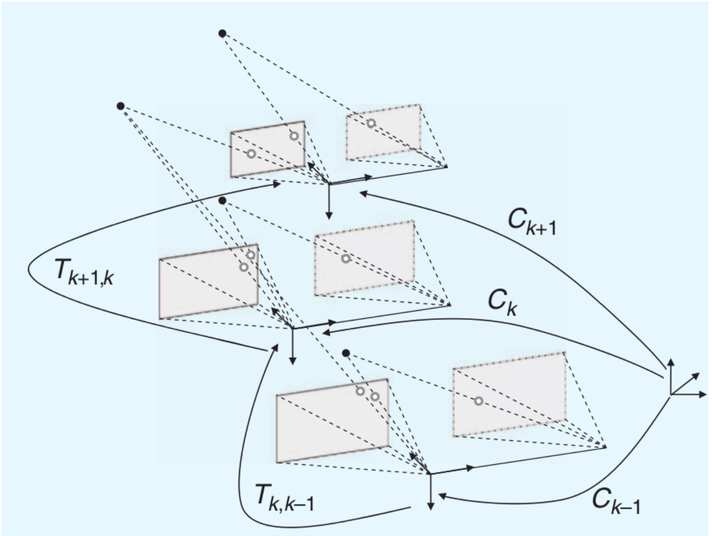
$$I_{0:n} = \{I_0, ..., I_n\}$$
$$D_{0:n} = \{D_0, ..., D_n\}$$



- Two camera positions at adjacent time istants $k-1$ and k are related by the rigid body transformation $T_k = \begin{bmatrix} R_{k-1,k} & t_{k-1,k} \\ 0 & 1 \end{bmatrix}$
- The set $T_{1:n} = \{T_1, ..., T_n\}$ contains all the subsequent motions

Visual Odometry (VO)

- The set of camera pose $C_{0:n} = \{C_0, ..., C_n\}$ contains the transformations of the camera w.r.t. the initial coordinate frame at $k = 0$



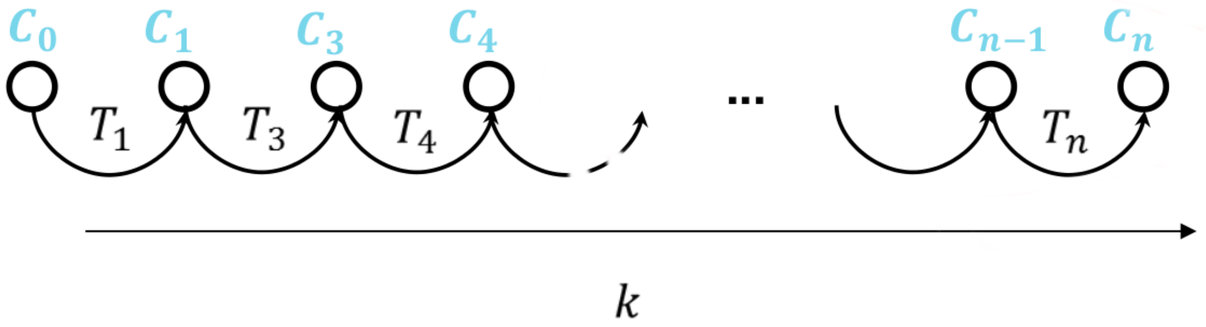
- The current camera pose C_n can be computed by concatenating all the transformations $T_{1:k}$, therefore

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

with C_0 being the camera pose at the instant $k = 0$, which can be arbitrarily set by the user

Visual Odometry (VO)

- The main task of VO is to compute the relative transformations T_k from images I_k and I_{k-1} and then to concatenate these transformation to recover the full trajectory $C_{0:n}$ of the camera
- This means that VO recovers the path incrementally, pose after pose



Visual Odometry (VO)

Usual assumptions about the environment

- 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

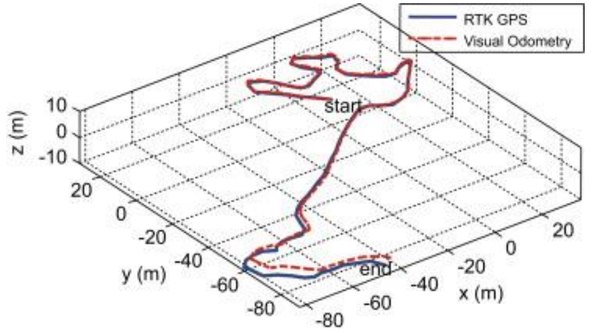
Are these examples OK?



Visual Odometry (VO)

Advantages of Visual odometry

- 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
- In GPS-denied environments, such as underwater and aerial, VO has utmost importance

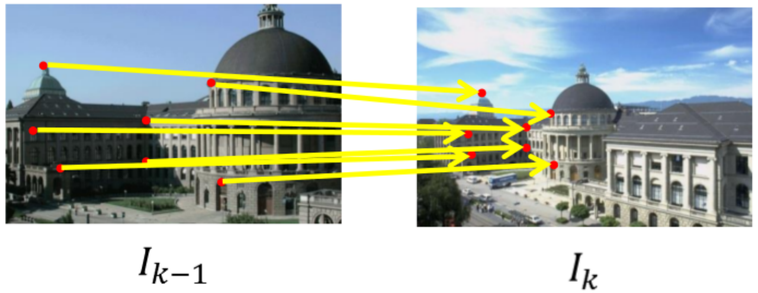


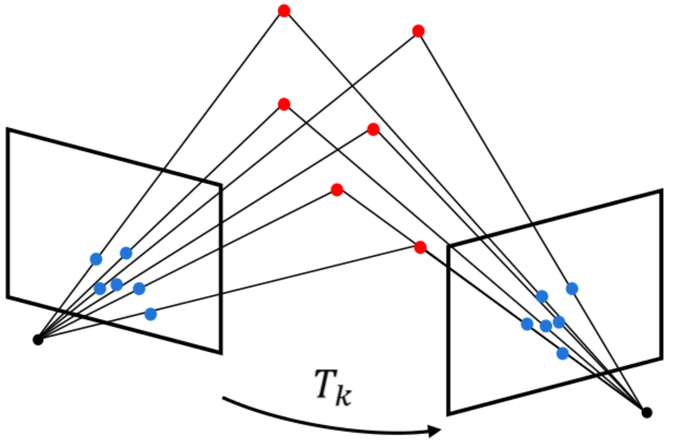
Visual Odometry Pipeline

Visual odometry (VO) feature-based

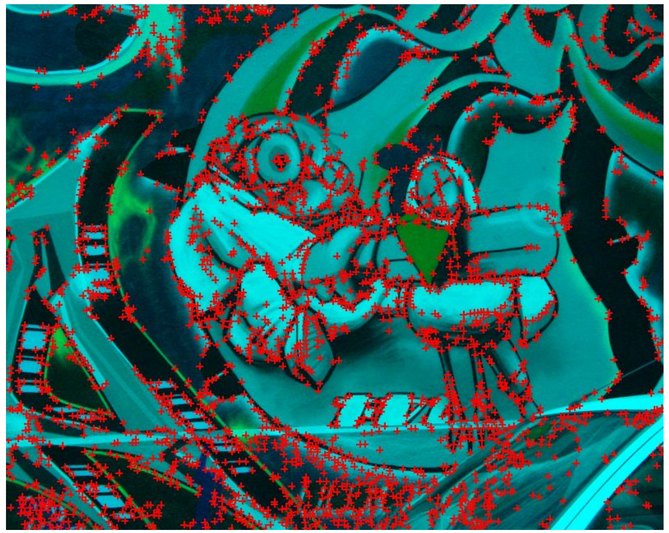
Overview

1. Feature detection
2. Feature matching/tracking
3. Motion estimation
4. Local optimization



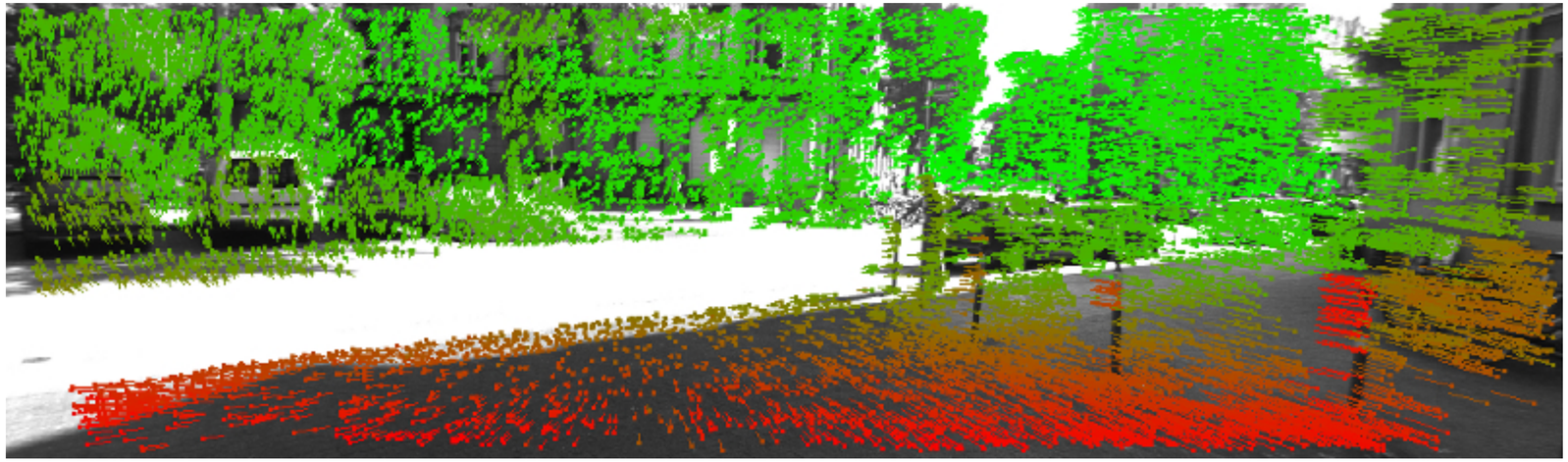


Visual Odometry Pipeline



Visual odometry (VO) feature-based
Assumption: camera is well calibrated

1. **Feature detection**:
   Detect a set of features f_k at time k
   (General idea: extract high-contrast areas in the image)



2. **Feature matching/Feature tracking**
   Find correspondences between set of features f_{k-1} , f_k
   - tracking: locally search each feature (e.g. by prediction and correlation)
   - matching: independently detect features in each image and find correspondences on the basis of a similarity metric (exploit descriptors such SURF, SIFT, ORB, etc)

Visual Odometry Pipeline

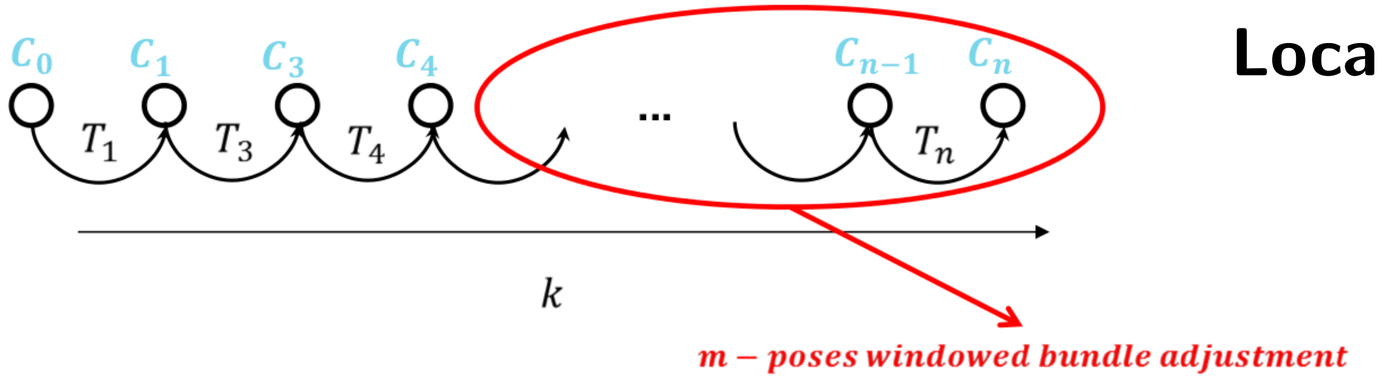
3. **Motion estimation**
   Compute transformation T_k between two images I_{k-1} and I_k from two sets of corresponding features f_{k-1} , f_k.
   Different algorithms depending on available sensor data:

   - *2-D to 2-D*: works on f_{k-1} , f_k specified in 2-D image coords
   - *3-D to 3-D*: works on X_{k-1} , X_k, sets of 3D points corresponding to f_{k-1} , f_k
   - *3-D to 2-D*: works on X_{k-1}, set of 3D points corresponding to f_{k-1}, and on f_k their corresponding 2-D reprojections on the image I_k

Type of correspondences	Monocular	Stereo
2D–2D	X	X
3D–3D		X
3D–2D	X	X

Visual Odometry Pipeline

4. An iterative refinement over last m poses can be **optionally** performed after motion estimation to obtain a more accurate estimate of the **local** trajectory



Local optimization

One has to minimize the following image reprojection error

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

where p_i^k is the i-th image point of the 3D landmark X_i measured in the k-th image and $g(X_i, C_k)$ is its image reprojection according to the current camera pose C_k

Visual Odometry Pipeline

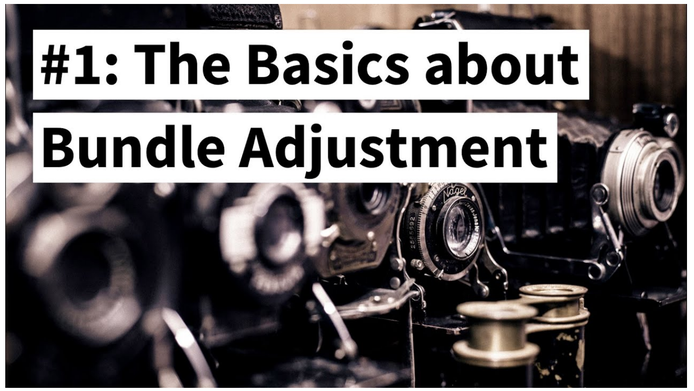
Bundle Adjustment

Least squares approach to estimating camera poses and 3D points

Key idea:

- Start with an initial guess
- Project the estimated 3D points into the estimated camera images
- Compare locations of the projected 3D points with measured (2D) ones
- Adjust to minimize error in the images

By Cyrill Stachniss

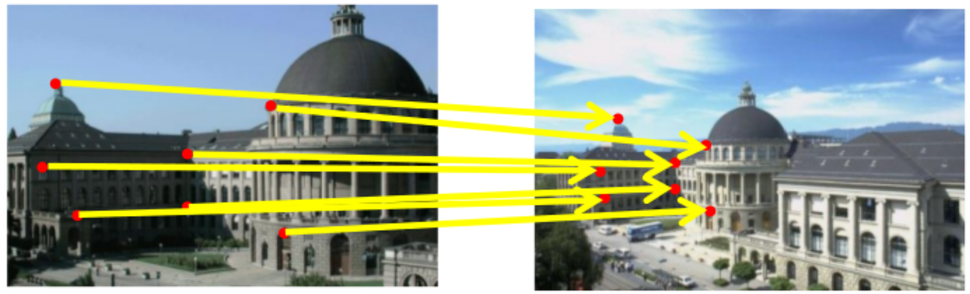


https://www.youtube.com/watch?v=sobyKHwgB0Y

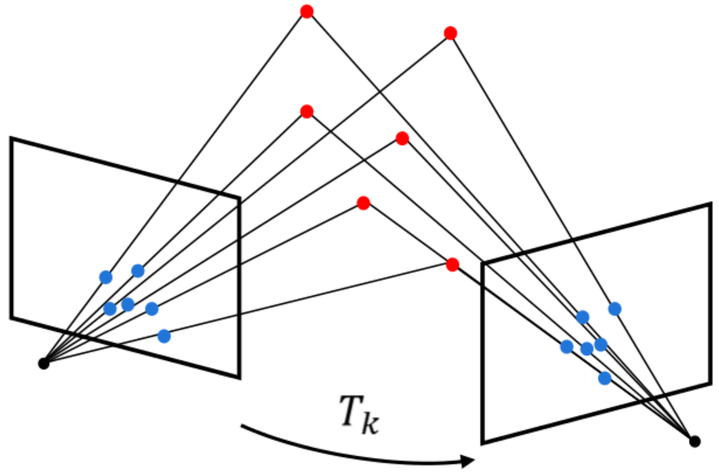
Visual Odometry Pipeline

VO from 2-D to 2-D (feature-based)

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).



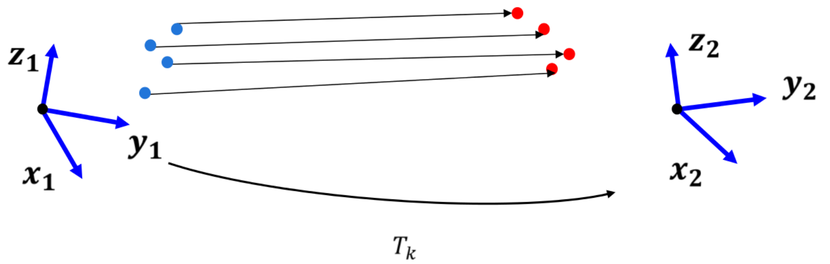
I_{k-1} I_k



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

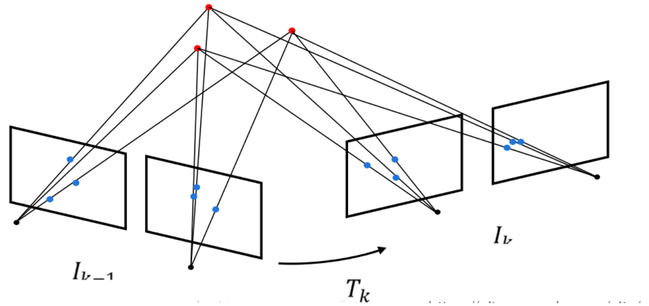
NOTE: The minimal-case solution involves 5-point correspondences

Visual Odometry Pipeline



VO from 3-D to 3-D (feature-based)

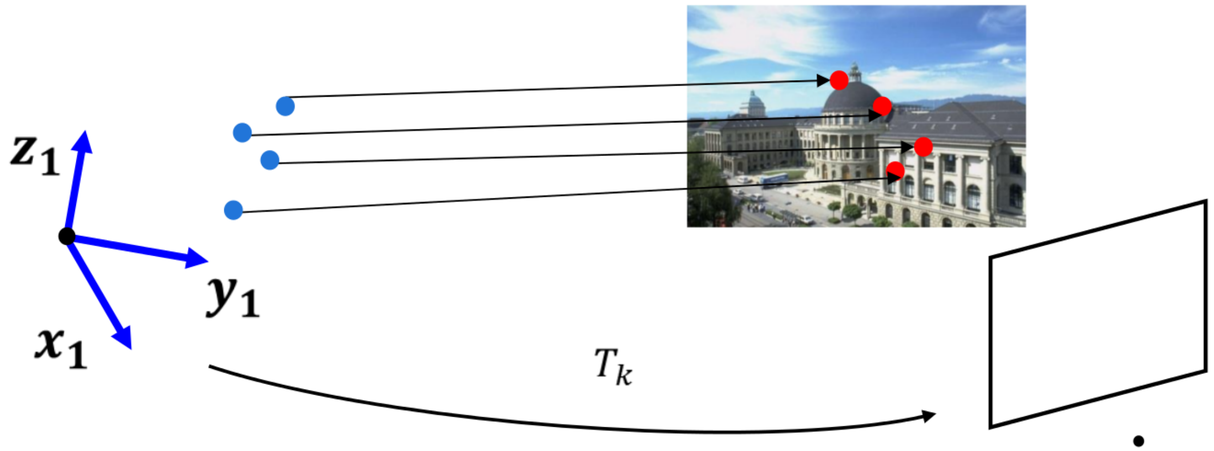
1. Capture two stereo image pairs $I_{l,k-1}$,$I_{r,k-1}$ and $I_{l,k}$,$I_{r,k}$
2. Extract and match features between $I_{l,k-1}$,$I_{l,k}$
3. Triangulate matched features for each stereo pair. Hence:
   $I_{l,k-1}$, $I_{r,k-1} \Rightarrow X_{k-1}$
   $I_{l,k}$, $I_{r,k} \Rightarrow X_k$
4. Compute T_k from 3-D features X_{k-1} and X_k
5. Concatenate transformation by computing $C_k = C_{k-1}T_k$
6. Repeat from 1).



$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = \arg\min_{T_k} \sum_i ||\tilde{X}_k^i - T_k \tilde{X}_{k-1}^i||$$

NOTE: The minimal-case solution involves 3 non-collinear correspondences

Visual Odometry Pipeline



VO from 3-D to 2-D (feature-based)

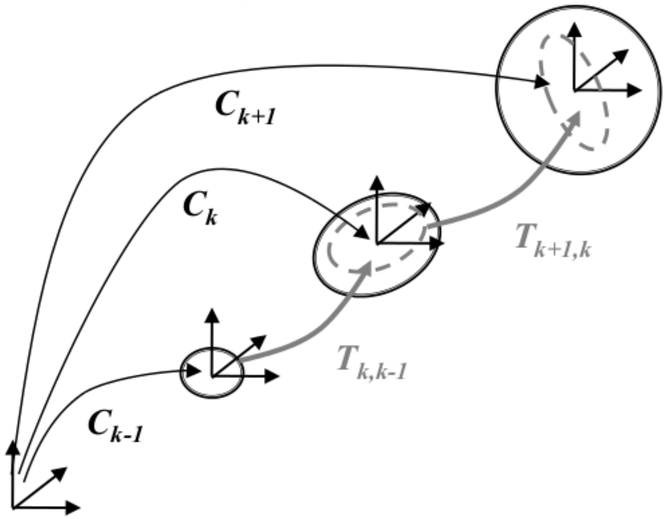
1. Do only once:
   - 1.1 Capture two frames I_{k-2} , I_{k-1}
   - 1.2 Extract and match features between them
   - 1.3 Triangulate features from I_{k-2} , I_{k-1} and get X_{k-1}
2. Do at each iteration:
   - 2.1 Capture new frame I_k
   - 2.2 Extract features and match with previous frame I_{k-1}
   - 2.3 Compute camera pose (PnP) from 3-D-to-2-D matches (between f_k and X_{k-1})
   - 2.4 Triangulate all new features between I_{k-1} and I_k and get X_k
   - 2.5 Iterate from 2.1

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

Visual Odometry (VO)

VO drift

1. The errors introduced by each new frame-to-frame motion accumulate over time
2. This generates a drift of the estimated trajectory from the real one



NOTE: the uncertainty of the camera pose at C_k is a combination of the uncertainty at C_{k-1} (black solid ellipse) and the uncertainty of the transformation $T_{k,k-1}$ (gray dashed ellipse)

VO vs SfM

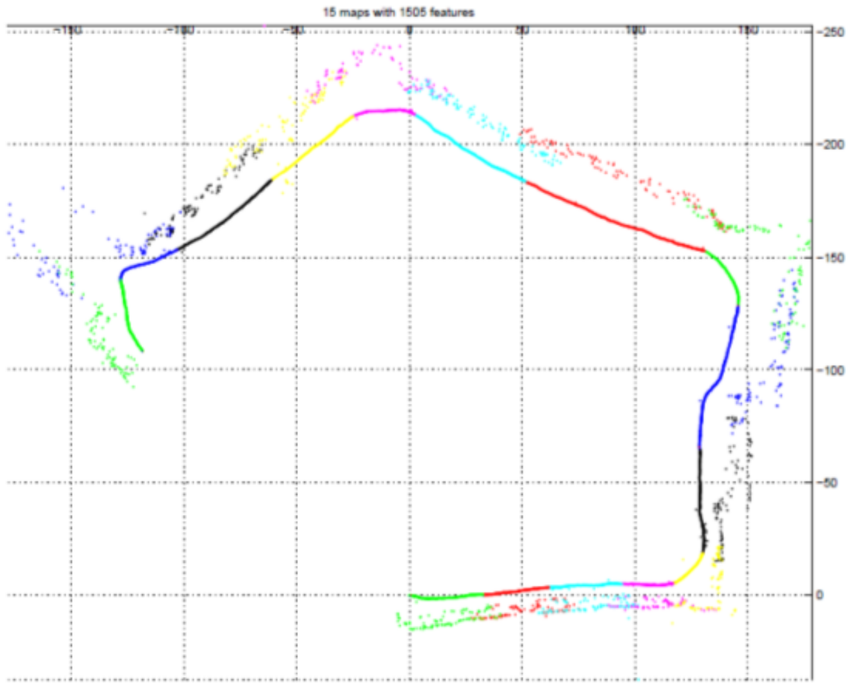
VO or SFM

- VO is a particular case of SFM
- VO focuses on estimating the 3D motion of the camera sequentially (as a new frame arrives) and in **real time**.
- Bundle adjustment can be used (but its optional) to refine the local estimate of the trajectory

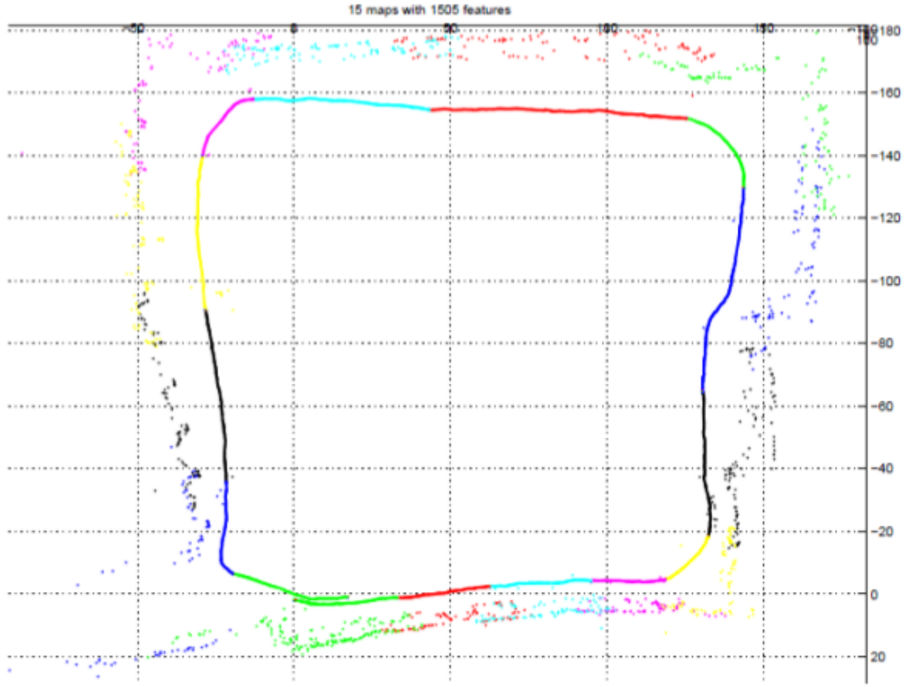
VO vs vSLAM

- The goal of **SLAM** in general is to obtain a **global** and **consistent** estimate of the robot **path** and the **map**. This is done by identifying **loop closures**. When a loop closure is detected, this information is used to reduce the drift in both the map and camera path (**global bundlel adjustment**)
- Conversely, **VO** aims at recovering a **path** incrementally, pose after pose, It can potentially use optimization only over the last m pose path (**windowed bundle adjustment**)

VO vs vSLAM



Before loop closing



After loop closing

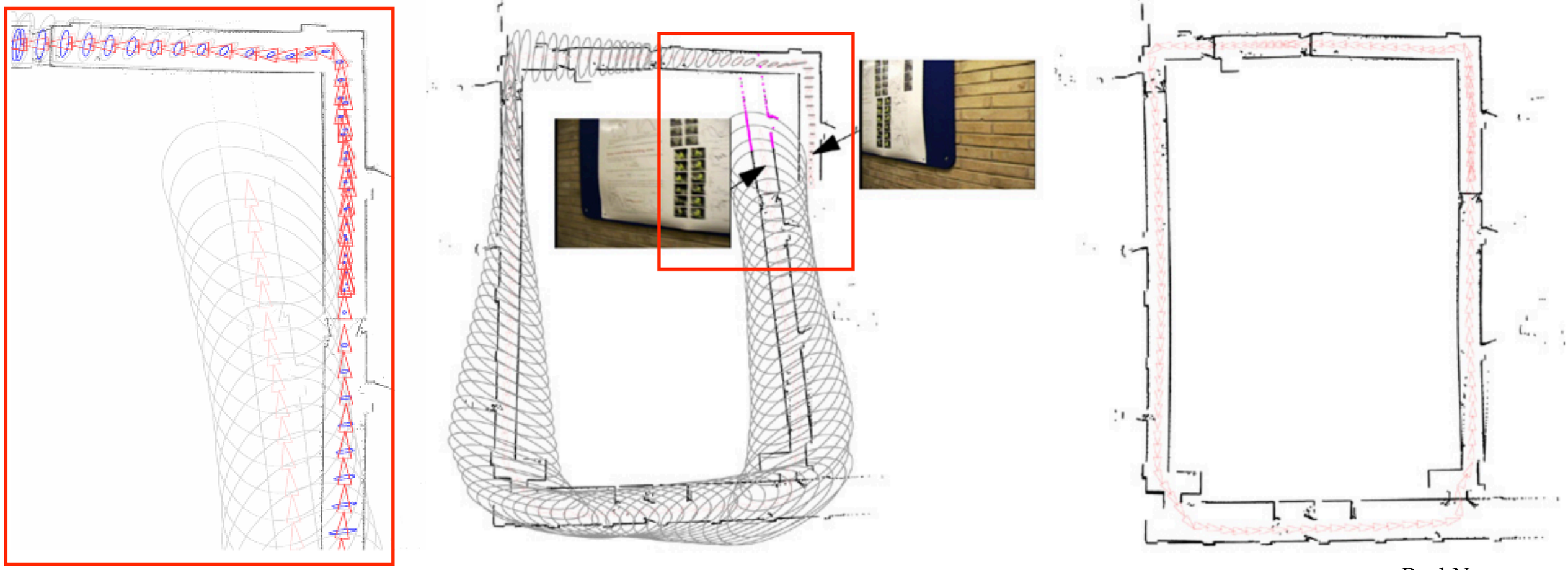
Image courtesy of Clemente et al. RSS'07

VO vs vSLAM

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

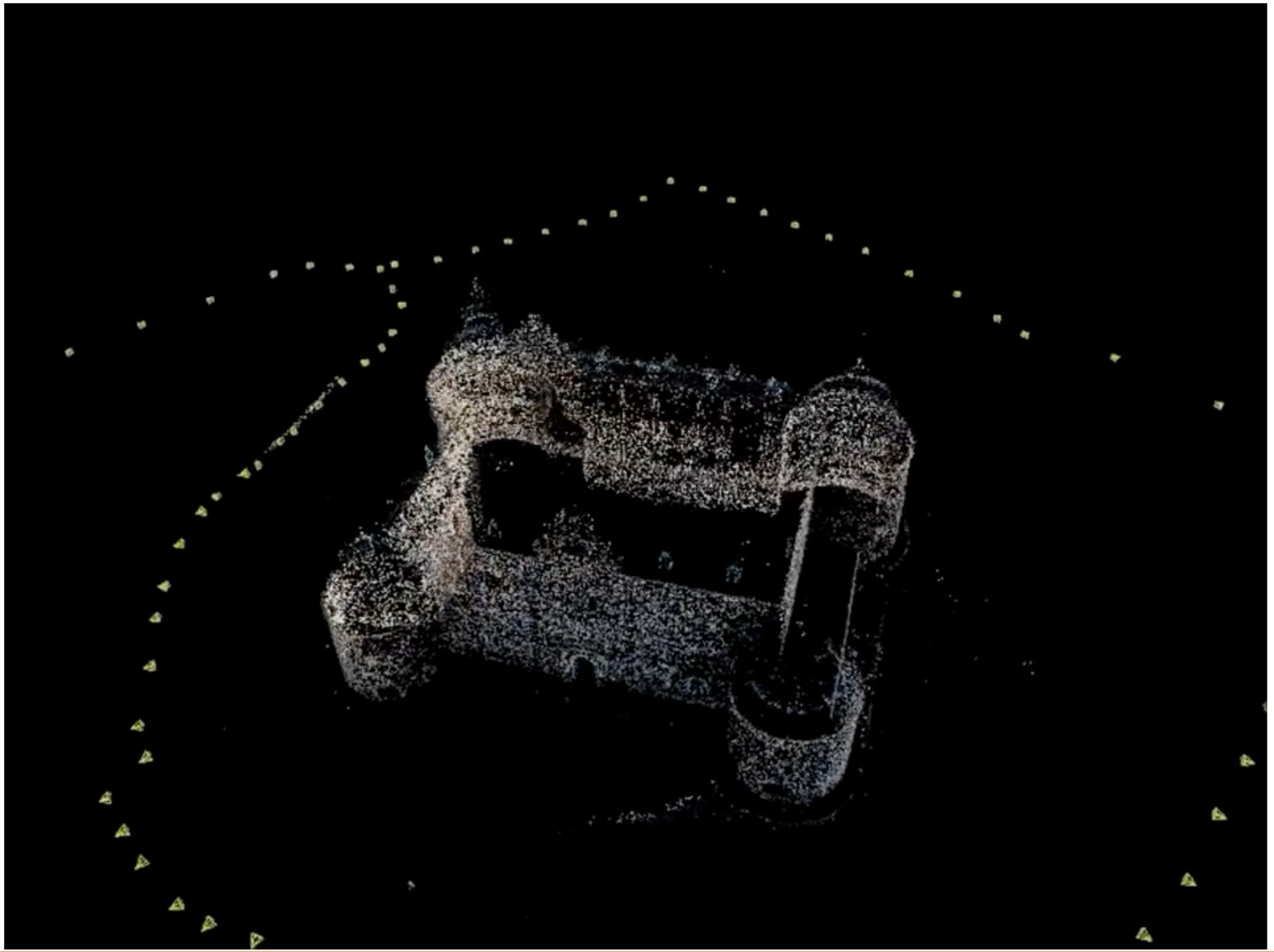
Loop Clousure

Recognizing an already mapped area, typically after a long exploration path.



Paul Newman

Structure from Motion (SfM)



ORB-vSLAM

Universidad Zaragoza

1542

Instituto Universitario de Investigación
en Ingeniería de Aragón
Universidad Zaragoza

ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras

Raúl Mur-Artal and Juan D. Tardós

raulmur@unizar.es tardos@unizar.es

Thank You !

Questions ?