Università di Genova
DIBRIS DIPARTIMENTO
DI INFORMATICA, BIOINGEGNERIA,
ROBOTICA E INGEGNERIA DEI SISTEMI

Augmented Reality

Lecture 12 – Tracking

Manuela Chessa – manuela.chessa@unige.it
Fabio Solari – fabio.solari@unige.it

Computer Vision for interaction in AR

- Now we consider computer vision algorithms for **interacting in AR**. We focus in particular on **visual tracking** and **3D** scene **reconstruction**.

- **Interaction** needs **real-time approaches**. This requirement is reflected in the subset of computer vision algorithms we consider here.

- They are also important for other interaction topics, such as *visual coherence, grasping, navigation, and collaboration*.

Computer Vision for interaction in AR

- <u>Marker Tracking</u>

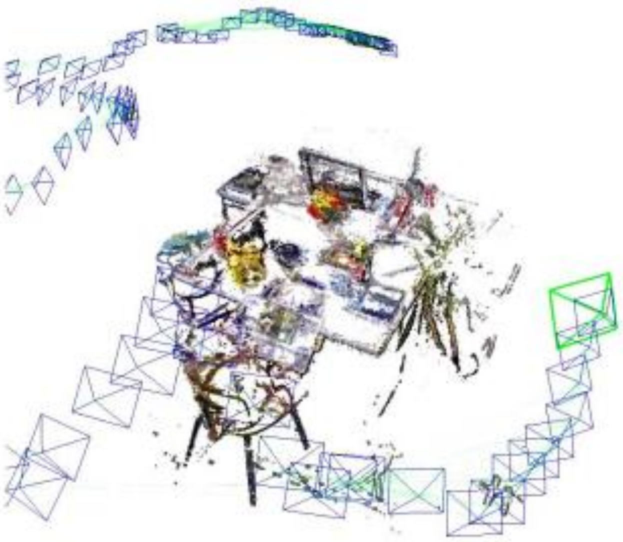
  moving camera

  (inside-out)

- <u>Natural Feature Tracking</u>

  moving camera

  (inside-out)

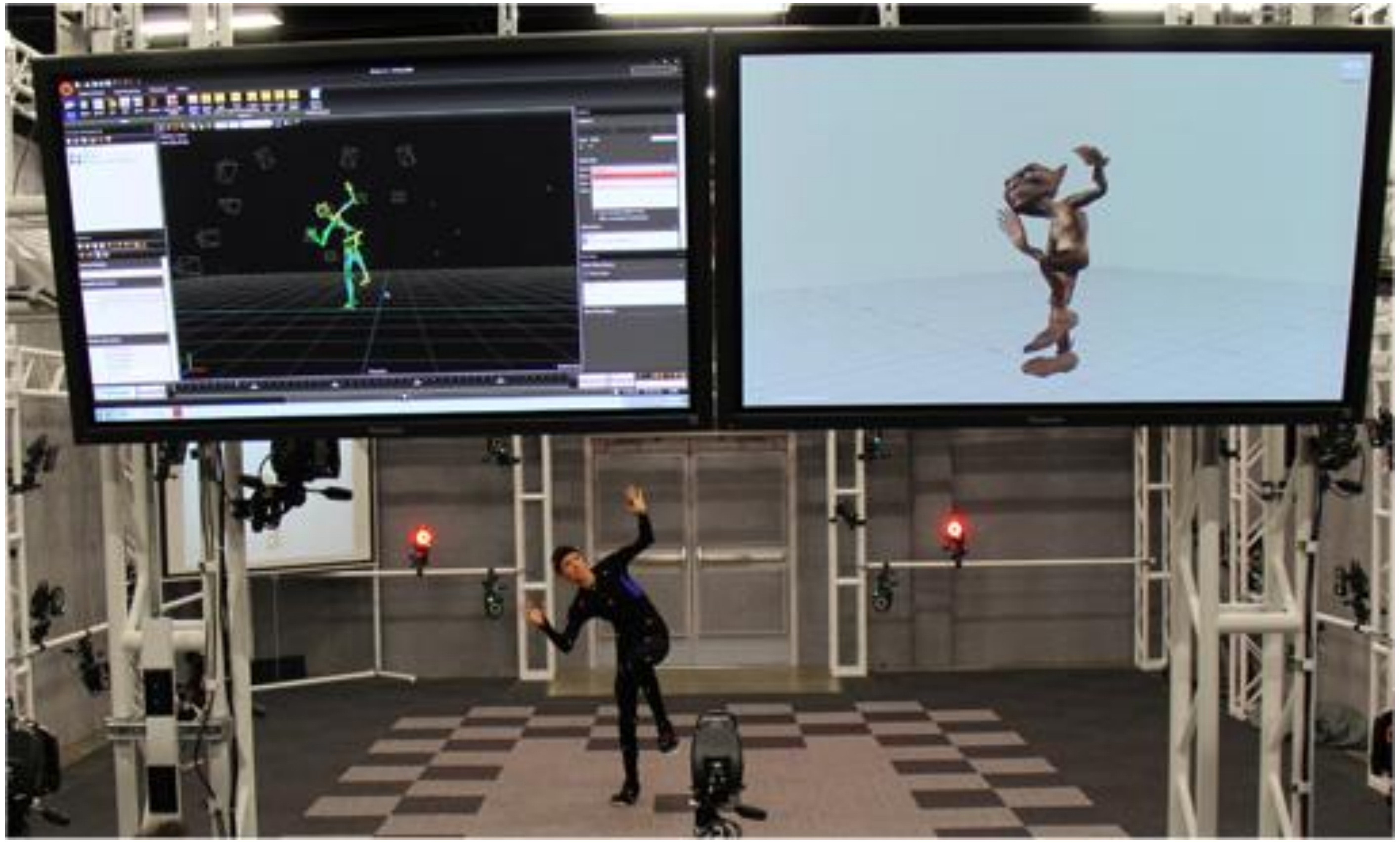


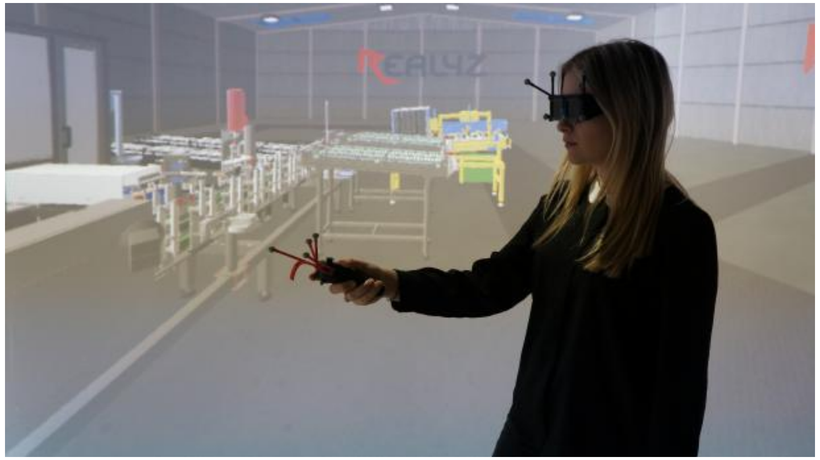
Computer Vision for interaction in AR

- Stereo (two) or <u>multiple cameras tracking</u>

  still camera rig, moving targets

  (outside-in)





Summary

- Marker Tracking *(moving camera)*
  - Homography (planar surfaces)
  - Pose Estimation from Homography
  - Pose Refinement
  - Note on camera calibration

- Stereo (multiple) camera tracking *(still camera rig, moving targets)*
  - Absolute Orientation

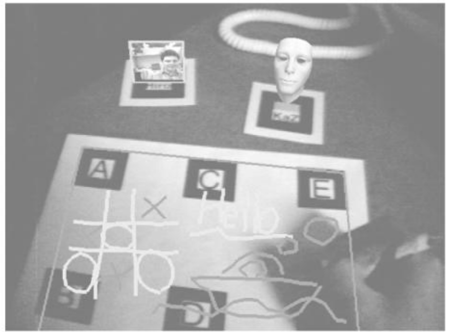
- Natural Feature Tracking *(moving camera)*
  - Tracking by detection
  - Incremental tracking
  - Simultaneous Localization and Mapping (SLAM)

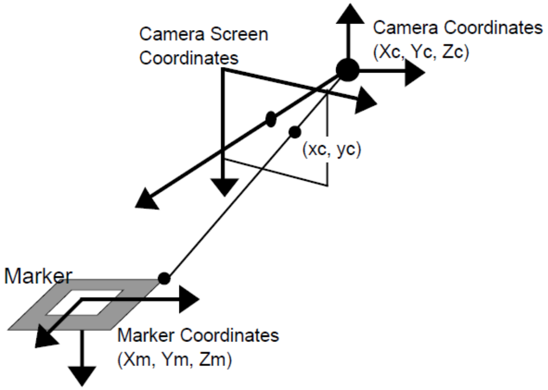
Università di Genova | DIBRIS DIPARTIMENTO DI INFORMATICA, BIOINGEGNERIA, ROBOTICA E INGEGNERIA DEI SISTEMI

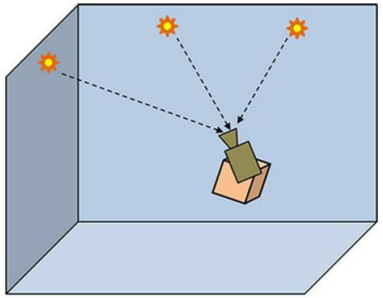
Marker Tracking (moving camera)

Marker Tracking

- **Tracking** (inside-out) black-and-white **fiducial markers** has been extremely popular ever since ARToolKit [Kato and Billinghurst, 1999] and ARToolKitPlus [Wagner and Schmalstieg, 2007].



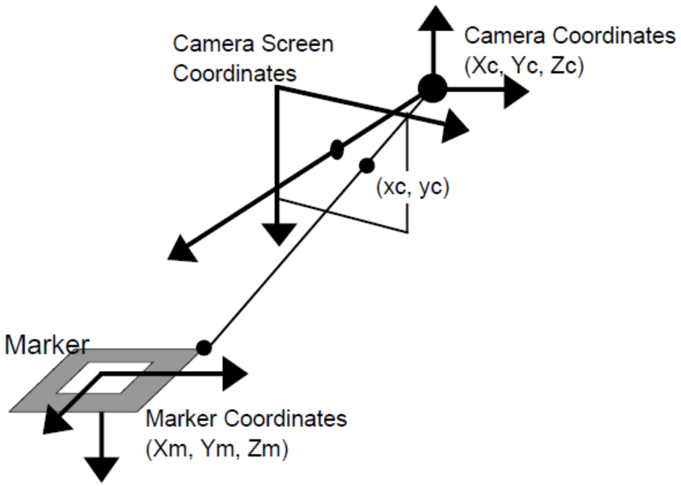


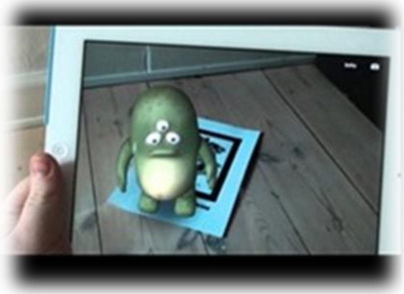


Kato, Hirokazu, and Mark Billinghurst. "Marker tracking and HMD calibration for a video-based augmented reality conferencing system." In Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99), pp. 85-94. IEEE, 1999.

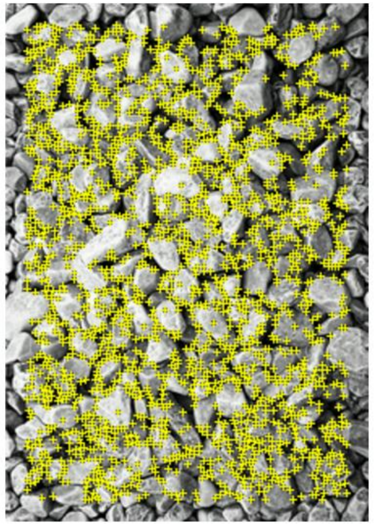
Marker Tracking

- Marker tracking is **computationally inexpensive** and can deliver useful results even with rather **poor cameras**.





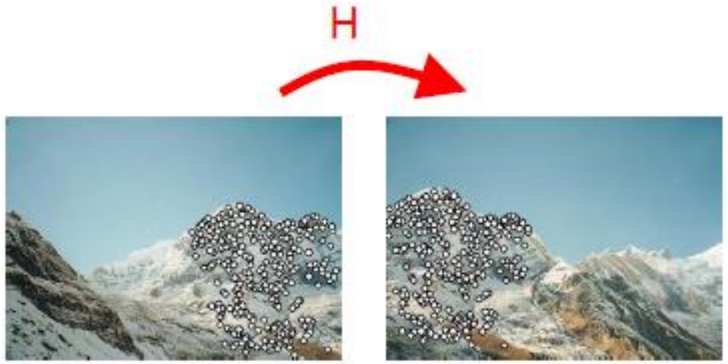
Example of a marker and the detected corners



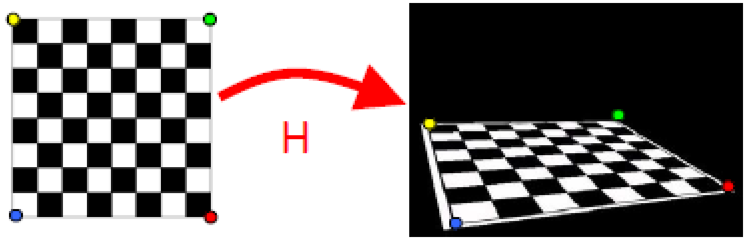
- Detecting *at least* **four corners** of a **flat marker** (*<u>in practical cases more than four points for robustness to noise</u>*) in an image from a single **calibrated camera** delivers just **enough information** to recover the **<u>pose</u>** of the camera relative to the marker (<u>homography</u>).

Points on a plane: Homography

- To estimate **homography** H (3×3 matrix) from *point correspondences between planar surfaces*
  - two images



  - **plane model image** and **viewed plane image**



Note: *estimation of F (or E) is degenerate for a planar scene.*
Camera calibration for coplanar points has a degenerate solution

Points on a plane: Homography

- General definition:

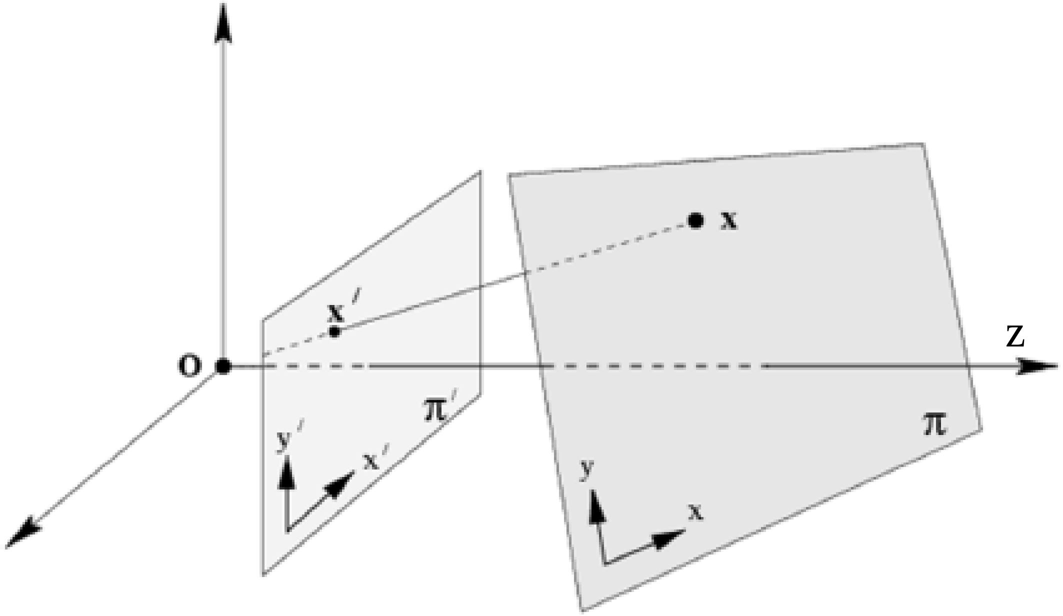
  A **homography** is a non-singular, line preserving, projective mapping H: P^n -> P^n

- It is represented by a square (n + 1)-dim matrix with $(n + 1)^2$-1 DOF

- In particular, we consider a

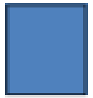
  mapping between planes

  (2D -> 2D, 3x3 matrix, 8 dof)

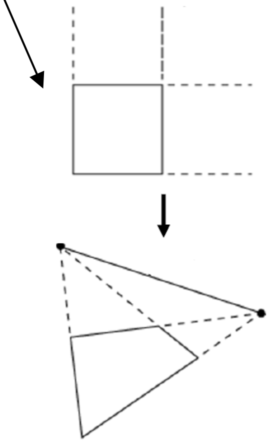


Homography: 2D transformation hierarchy

A square transforms to:

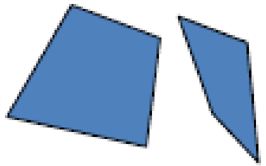


It changes homogeneous coordinate



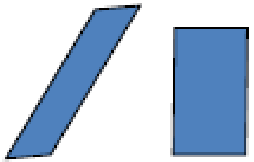
Projective
8dof

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$



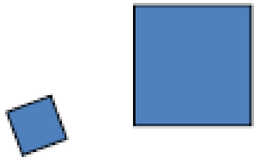
Affine
6dof

$$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$



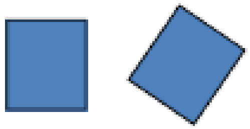
Similarity
4dof

$$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$



Euclidean
3dof

$$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

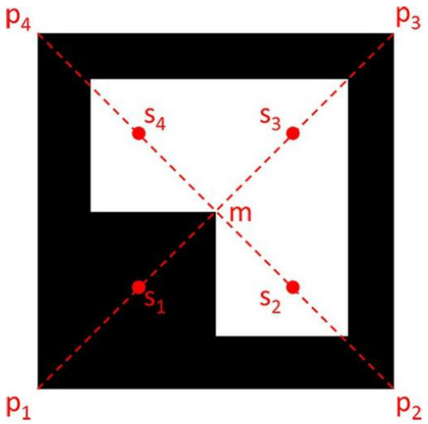


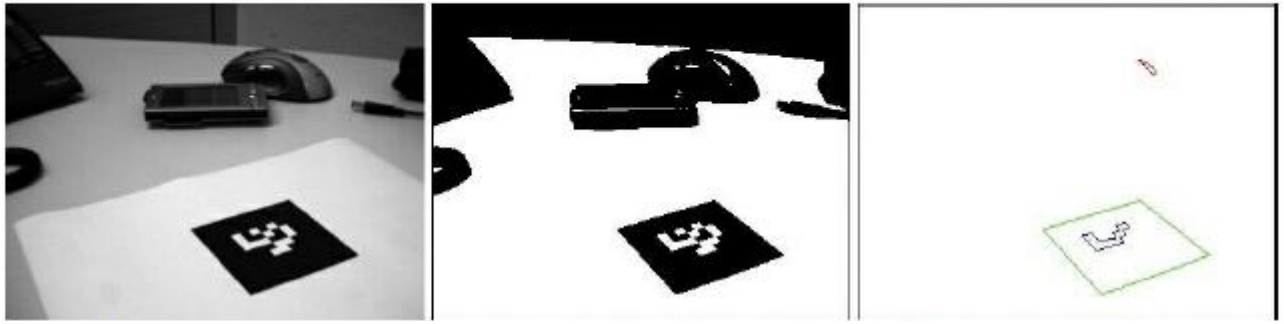
The marker tracking algorithm

1. **Image** acquired by using a **calibrated camera** (i.e., with known intrinsic matrix K)
2. **Marker detection** by searching for quadrilateral shapes (corners detection)
3. **Homography estimation** (from planar surfaces)
4. **Pose estimation** from a **homography**
5. **Pose refinement** by nonlinear reprojection error minimization
6. **AR rendering** with the recovered camera pose and registration (by allowing coherent **interaction**)

Marker Detection

- We assume a single input marker image



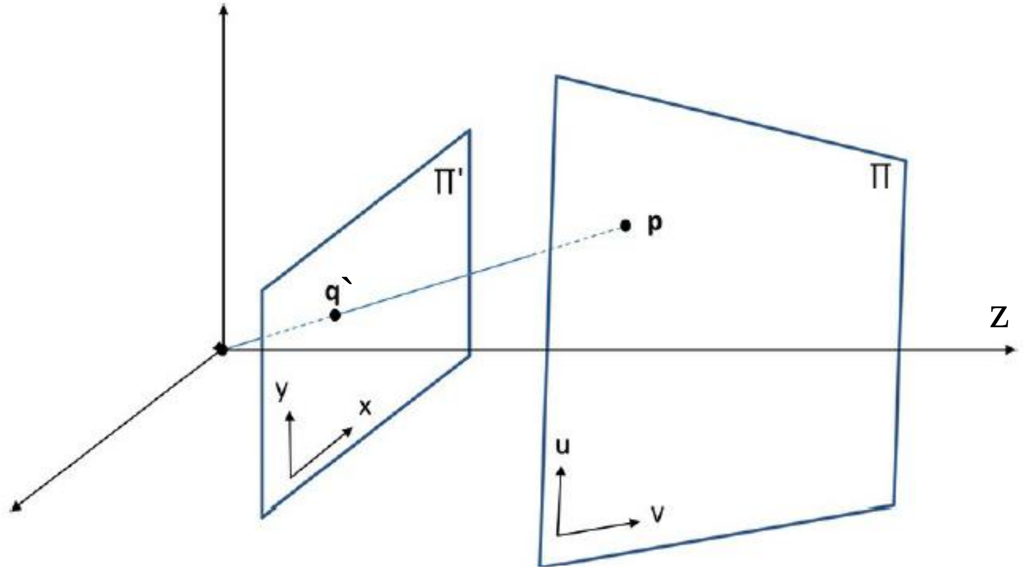


- We compute the **four corners** of the **<u>flat</u>** marker (by using image processing techniques, e.g. Harris corner detector, SIFT, ORB).

Note that is a flat marker: degenerate solution for camera calibration and 8-point algorithm.

Homography Estimation

- For tracking applications, <u>*one plane*</u> is the **plane model** (*as an image*), and the <u>*other plane*</u> is the **viewed plane** (*as an image*) that contains the **known points in the world**, such as the marker’s corners.



- Homogeneous 2D points **p** ϵ Π and **q′** ϵ Π′ can be related by a homography **H** as follows: $p = Hq'$

Homography Estimation

- We assume that the **marker** defines the **plane** Π': $\mathbf{q}_z = 0$ in **world coordinates**, and that marker corners have the coordinates $[0\ 0\ 0]^T$, $[1\ 0\ 0]^T$, $[1\ 1\ 0]^T$, and $[0\ 1\ 0]^T$.
- We can then express a 3D point $\mathbf{q}' \in \Pi'$ as a *homogeneous 2D point* $\mathbf{q}' = [q_x\ q_y\ 1]^T$.
- Mapping from one plane to another can be mathematically modeled as a **homography** defined by a 3 × 3 matrix **H**.

Homography Estimation

- Because the **four points** are constrained to lie in a **plane**, they can be expressed with only 2DOF.

- Consequently, a 3×3 matrix with 8DOF (the ninth element is the scale) is sufficient to relate them to the image plane.

- **H** can be estimated from 2D–2D correspondences using **direct linear transformation** (DLT).

Homography Estimation

- Because we are using homogeneous coordinates, two points are related by a homography only up to scale. *When interpreted as vectors, however, they point in the same direction.* Thus, the cross-product is zero.

- By using the notation of cross-product as a matrix multiplication

$$\mathbf{p}_\times = \begin{bmatrix} p_u \\ p_v \\ p_w \end{bmatrix}_\times = \begin{bmatrix} 0 & -p_w & p_v \\ p_w & 0 & -p_u \\ -p_v & p_u & 0 \end{bmatrix}$$

Homography Estimation

- From $\mathrm{p} = \mathrm{Hq}'$ we obtain:

$$\mathbf{p}_\times \begin{bmatrix} \mathbf{H}_{R1} \\ \mathbf{H}_{R2} \\ \mathbf{H}_{R3} \end{bmatrix} \mathbf{q}' = 0 \qquad \begin{bmatrix} 0 & -p_w & p_v \\ p_w & 0 & -p_u \\ -p_v & p_u & 0 \end{bmatrix} \begin{bmatrix} \mathbf{H}_{R1}\mathbf{q}' \\ \mathbf{H}_{R2}\mathbf{q}' \\ \mathbf{H}_{R2}\mathbf{q}' \end{bmatrix} = 0$$

$$\begin{bmatrix} 0 & -p_w \mathbf{H}_{R2}^T \mathbf{q}' & p_v \mathbf{H}_{R3}^T \mathbf{q}' \\ p_w \mathbf{H}_{R1}^T \mathbf{q}' & 0 & -p_u \mathbf{H}_{R3}^T \mathbf{q}' \\ -p_v \mathbf{H}_{R1}^T \mathbf{q}' & p_u \mathbf{H}_{R1}^T \mathbf{q}' & 0 \end{bmatrix} = 0$$

- Since $a^T b = b^T a$, we can rewrite the previous equations as

$$\begin{bmatrix} 0 & -p_w \mathbf{q}'^T & p_v \mathbf{q}'^T \\ p_w \mathbf{q}'^T & 0 & -p_u \mathbf{q}'^T \\ -p_v \mathbf{q}'^T & p_u \mathbf{q}'^T & 0 \end{bmatrix} \begin{bmatrix} \mathbf{H}_{R1}^T \\ \mathbf{H}_{R2}^T \\ \mathbf{H}_{R3}^T \end{bmatrix} = 0$$

Homography Estimation

- From **one 2D–2D correspondence**, we have now obtained three equations in the **unknown** coefficients of **H**.
- These **equations** are linearly dependent, so we retain only the **first two**.
- We require *a minimum* of **four** input **points** to determine **eight unknowns**.

- From N ($N>=4$) pairs $\mathbf{p}_i = [p_{i,u}\ p_{i,v}\ p_{i,w}]^T$ and $\mathbf{q}_i = [q_{i,u}\ q_{i,v}\ q_{i,w}]^T$, we set up a $2N \times 9$ matrix:

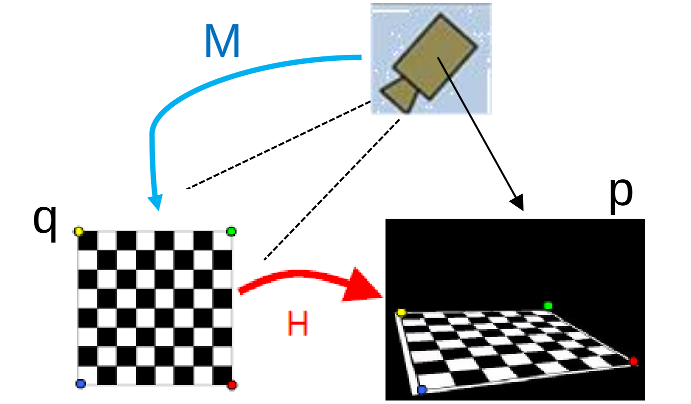
Homography Estimation

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 0 & -p_{1,w}q'^{T}_{1,u} & -p_{1,w}q'^{T}_{1,v} & -p_{1,w}q'^{T}_{1,w} & p_{1,v}q'^{T}_{1,u} & p_{1,v}q'^{T}_{1,v} & p_{1,v}q'^{T}_{1,w} \\ p_{1,w}q'^{T}_{1,u} & p_{1,w}q'^{T}_{1,v} & p_{1,w}q'^{T}_{1,w} & 0 & 0 & 0 & -p_{u,i}q'^{T}_{1,u} & -p_{u,i}q'^{T}_{1,v} & -p_{u,i}q'^{T}_{1,w} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & -p_{N,w}q'^{T}_{N,u} & -p_{N,w}q'^{T}_{N,v} & -p_{N,w}q'^{T}_{N,w} & p_{N,v}q'^{T}_{N,u} & p_{N,v}q'^{T}_{N,v} & p_{N,v}q'^{T}_{N,w} \\ p_{N,w}q'^{T}_{N,u} & p_{N,w}q'^{T}_{N,v} & p_{N,w}q'^{T}_{N,w} & 0 & 0 & 0 & -p_{N,u}q'^{T}_{N,u} & -p_{N,u}q'^{T}_{N,v} & -p_{N,u}q'^{T}_{N,w} \end{bmatrix}$$

- The homogeneous equation system **A h** = 0, which is overdetermined for $N > 4$, can be solved using singular value decomposition (SVD), i.e. *the eigenvector with smallest eigenvalue.*
- **Now we have the homography** estimation.
- We can exploit it ***to estimate the camera pose*****.**

Pose Estimation from Homography

- Recall that our **points** (q) **lie** in the **z-plane** and assume that matrix K (**intrinsic calibration matrix**) is **known**: we derive the 3D camera pose from H, since the third column $\mathbf{R}_{C3}$ of the rotation matrix **R** has no effect:



$$
\begin{aligned}
p &\propto Mq \qquad \text{Camera projection matrix M (3x4)} \\
&\propto K[R \mid t][q_x\ q_y\ q_z\ 1]^T \\
&\propto K[R_{C1} \mid R_{C2} \mid R_{C3} \mid t][q_x\ q_y\ 0\ 1]^T \\
&\propto K[R_{C1} \mid R_{C2} \mid t][q_x\ q_y\ 1]^T \\
&\propto Hq' \qquad \text{Homography matrix H (3x3)}
\end{aligned}
$$

Pose Estimation from Homography

- We see that $\mathbf{H} = \mathbf{K}[\mathbf{R}_{C1} \mid \mathbf{R}_{C2} \mid \mathbf{t}]$. Therefore, the <u>camera pose</u> can be computed from

$$\mathbf{H}^{\mathbf{k}} = \mathbf{K}^{-1}\,\mathbf{H}, \quad \text{thus } \mathbf{H}^{\mathbf{k}} = [\mathbf{R}_{C1} \mid \mathbf{R}_{C2} \mid \mathbf{t}]$$

by recovering the third column of the rotation matrix as the cross-product of the first two columns: $\mathbf{R}_{C1} \times \mathbf{R}_{C2}$

- However, the first two columns of $\mathbf{H}^{K}$ will usually not be truly orthonormal *because of noise* in the point correspondences, and we know them *up to a scale*. Therefore, a proper **normalization** needs to be **enforced**.

Pose Estimation from Homography

- To enforce a proper **normalization,** we consider

$$K^{-1}H = \lambda \left[r_1 r_2 t \right]$$

- r_1 and r_2 are unit vectors -> **find lambda**
- Use this **to compute t**
- Rotation matrices are **orthogonal** -> find r_3 as cross product between r_1 and r_2

- Thus, we have

$$P = K \left[r_1 \quad r_2 \quad (r_1 \times r_2) \quad t \right]$$

Pose Estimation from Homography

- Problem:
  - The vectors r1 and r2 might not yield the same lambda
- Solution:
  - Use the **average value**

- Problem:
  - The estimated rotation matrix might not be orthogonal
- Solution: **orthogonalize** R’
  - Obtain SVD -> $R=USV^T$
  - Set singular values S to 1 -> $R'=UV^T$

Pose Refinement

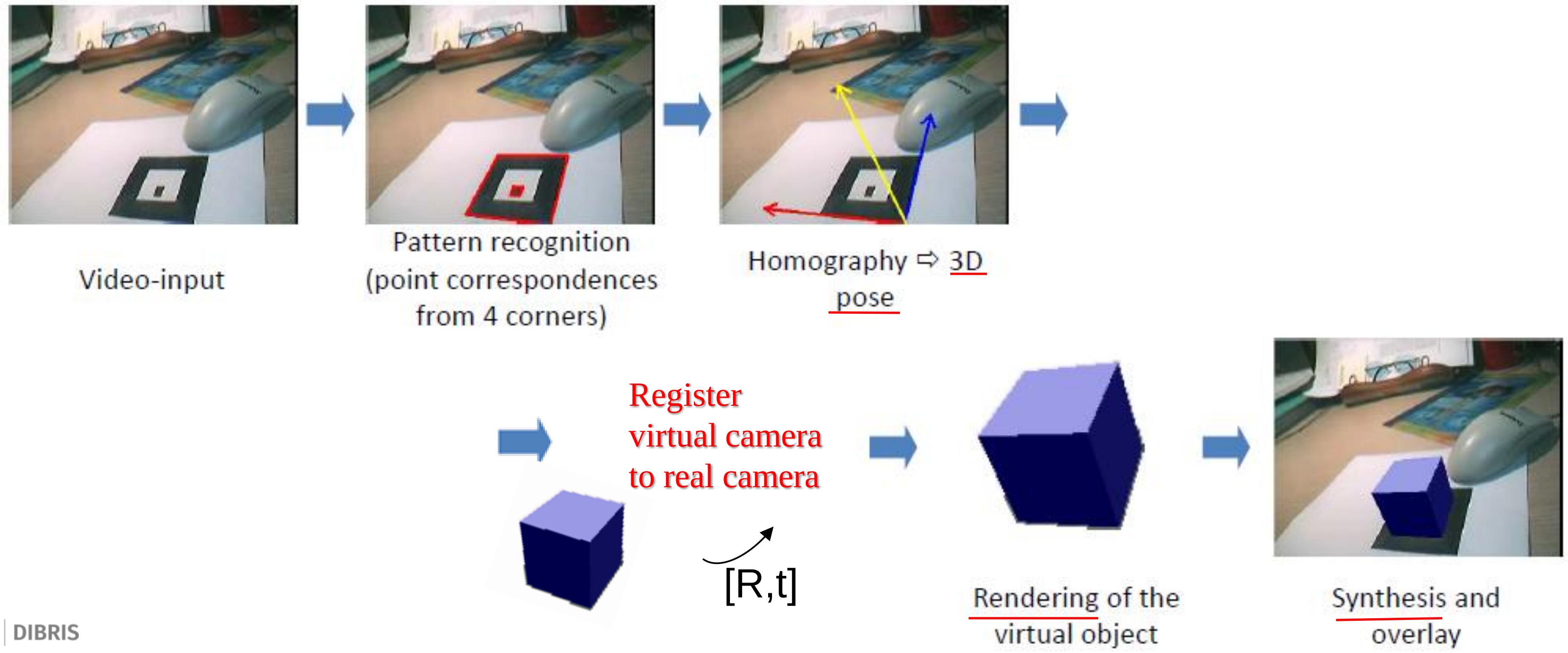
- Pose estimation cannot always be computed directly from imperfect point correspondences with the *desired accuracy*.
- Therefore, the pose estimation is **refined** by **iteratively** minimizing the **reprojection error.**
- When a **first estimate** of the camera **pose** is known, we **minimize** the **displacement** of the known points q_i in 3D, **projected** using [R| t], from its known image location p_i.

$$\operatorname*{argmin}_{R,t} \sum_i (K[R \mid t]q_i - p_i)^2$$

e.g. by using Levenberg-Marquardt method

Example: marker tracker for AR

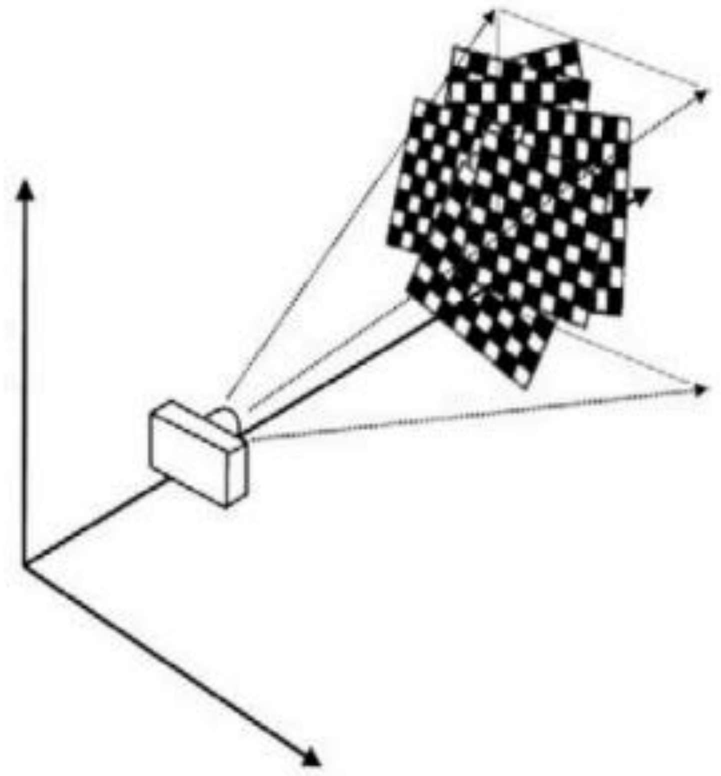
- Enables **augmentation** of 3D **real scenes**
  - **Register** virtual camera to real camera (*we know its pose*)
  - **Render** virtual scene
  - **Compose** with real image



Note: Camera calibration through homography

- We can calibrate a camera by considering **several images** of a **known <u>planar</u> pattern**, such as a checkerboard (*instead of a 3D calibration rig, as we have considered in a previous lecture*).

- We can follow the same approach used for the homography estimation, but now we have *to estimate the matrix K too*.
- Thus, **first** we estimate **H** by using the **4-point algorithm**, **then** we can obtain **two equations** that the calibration matrix **K** has to satisfy.
- Since K has 5 parameters, we need at least **three** different **images** of the **checkerboard** (*more images for robustnesss to noise*).

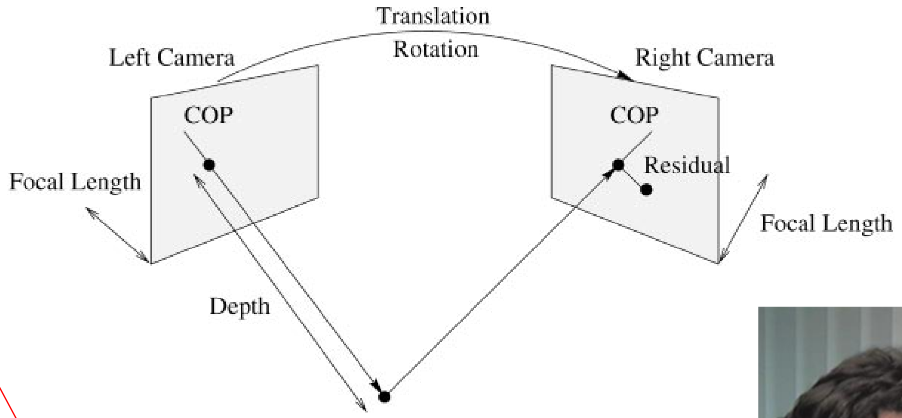


Università di Genova | DIBRIS DIPARTIMENTO DI INFORMATICA, BIOINGEGNERIA, ROBOTICA E INGEGNERIA DEI SISTEMI

Stereo (two) or multiple cameras tracking

Stereo (two) or multiple cameras tracking

- Tracking uses an **outside-in** setup with multiple (*infrared*) cameras [Dorfmüller 1999].

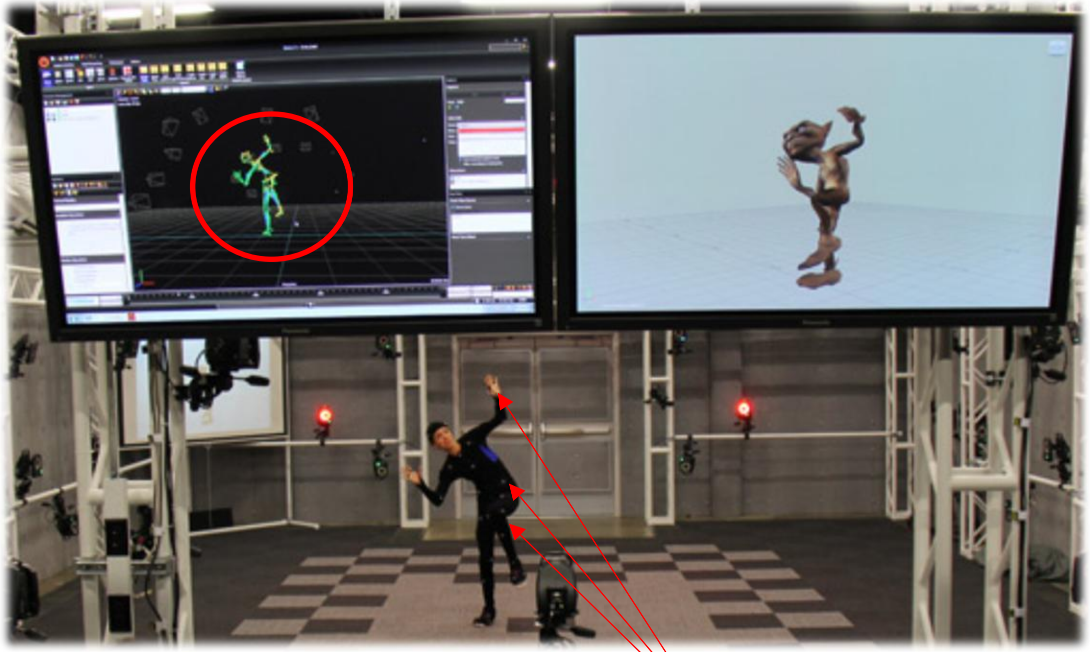


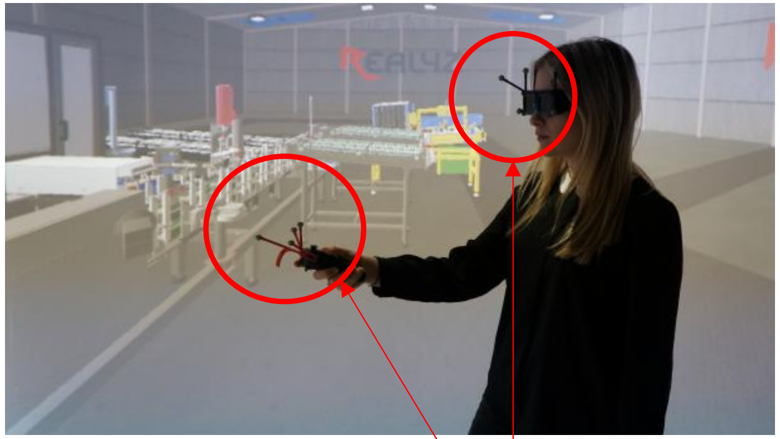
still camera rig, moving targets

Dorfmüller, Klaus. "Robust tracking for augmented reality using retroreflective markers." Computers & Graphics 23, no. 6 (1999): 795-800.

Stereo (two) or multiple cameras tracking

still camera rig, moving targets





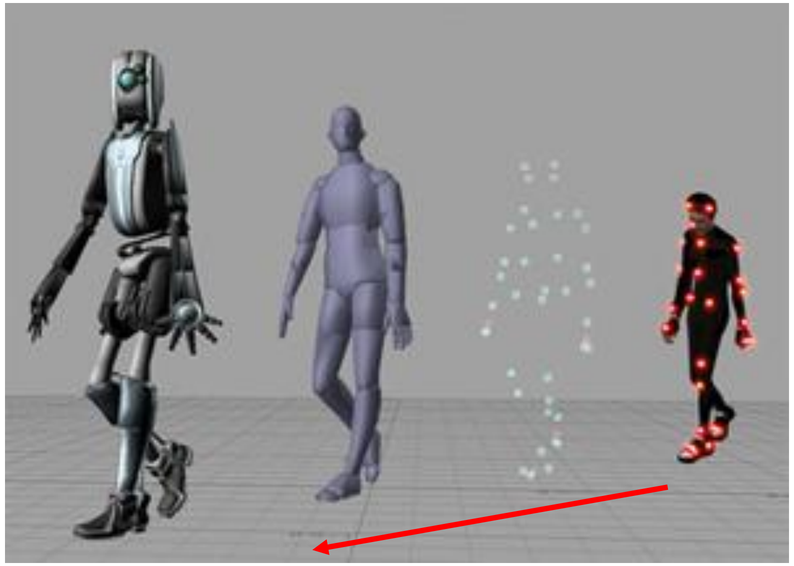
tracking of points

tracking of objects

- A *minimum of two cameras (but in actual cases from 8 to 16, or more, cameras)* in a known configuration, i.e. a **calibrated stereo camera rig** *(<u>epipolar geometry</u>)*, is required.

Stereo (two) or multiple cameras tracking

- For tracking **arbitrary objects** (*not points*), we require **general pose estimation**, which addresses the problem of *determining the camera pose from 2D–3D correspondences*.
- We describe an infrared tracking technique designed to track rigid **body markers** composed of *four or more retro-reflective spheres*.



point tracking

object tracking

The stereo camera tracking algorithm

points

1. **Blob detection** in all images to locate the retro-reflective **body markers** (spheres)
2. Establishment of point correspondences between blobs by using **epipolar geometry** between the cameras
3. **Triangulation** to obtain 3D candidate points from the multiple 2D points

objects

4. **Matching** of 3D candidate points to 3D target points
5. Determination of the **target's pose** by using <u>absolute orientation</u> technique

The stereo camera tracking

- Blob detection
  - It is **simplified**, since the targets are composed of spheres covered with retro-reflective foil (moreover, for objects the four or five spheres are in a known rigid structure).

- Establishing Point Correspondences
  - The candidate 2D points p1 and p2 in the two images can be related using **epipolar lines**. Since the system is **calibrated**, we can use the **Essential matrix**.

- Triangulation from two Cameras
  - We can perform a **3D reconstruction** from multiple 2D points, since the system is **calibrated**. This is enough for *points*.

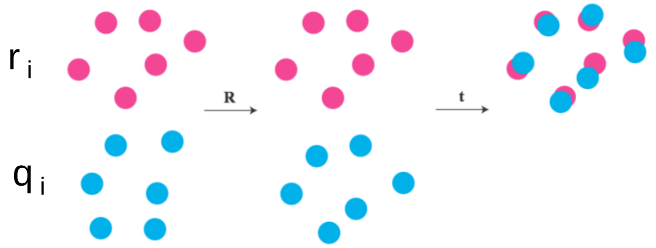
  - We can use *constraints* on the detected points to stabilize the skeleton reconstruction

The stereo camera tracking

- <u>Object tracking</u>: matching targets consisting of spherical markers (i.e., ***matching objects***)
  - there may be more candidate points than target points because of ambiguous observations.
  - The association from candidate points to target points is resolved using the **known geometric structure of the target** (*e.g. the distance between any two points and the angles of the triangle formed by any three points should yield a unique signature*).
  - Determination of the **target's pose** by using absolute orientation technique.

The stereo camera tracking

- Absolute Orientation
  - After candidate point association, we are left with **two sets of corresponding points**, the *observed points* q_i and the *target points* r_i.
  - The **target** points are specified in a **reference coordinate system**, and we would like to compute the **pose** [R | t] of the **observed target** relative to the reference coordinate system.
  - It requires at least **three points** (e.g., *the absolute orientation* using the method described by Horn [1987])
  - The **centroid** of the three points can be used to determine the **translation** from the **reference** coordinate system to the **measurement** coordinate system.
  - The **rotation** is computed from two parts.
  - First, we define a **rotation** from the measurement coordinate system into an **intermediate coordinate system** defined by the q_i .
  - Second, we do the same for r_i. Finally, we **concatenate** the two rotations to obtain R.



The stereo camera tracking

- Absolute Orientation
  - The translation **t** is determined from the difference of the centroids

$$\mathbf{q}^c = (\mathbf{q}_1 + \mathbf{q}_2 + \mathbf{q}_3)/3$$

$$\mathbf{r}^c = (\mathbf{r}_1 + \mathbf{r}_2 + \mathbf{r}_3)/3$$

$$\mathbf{t} = \mathbf{q}^c - \mathbf{r}^c$$

  - To compute the rotation **R**, we assume the origin at q_1 and the x-axis x aligned with the vector from q_1 to q_2

$$x = N(q_2 - q_1)$$

  - The y-axis y is orthogonal to x and lies in the plane given by q_1, q_2 and q_3

$$y = N((q_3 - q_1) \times x)$$

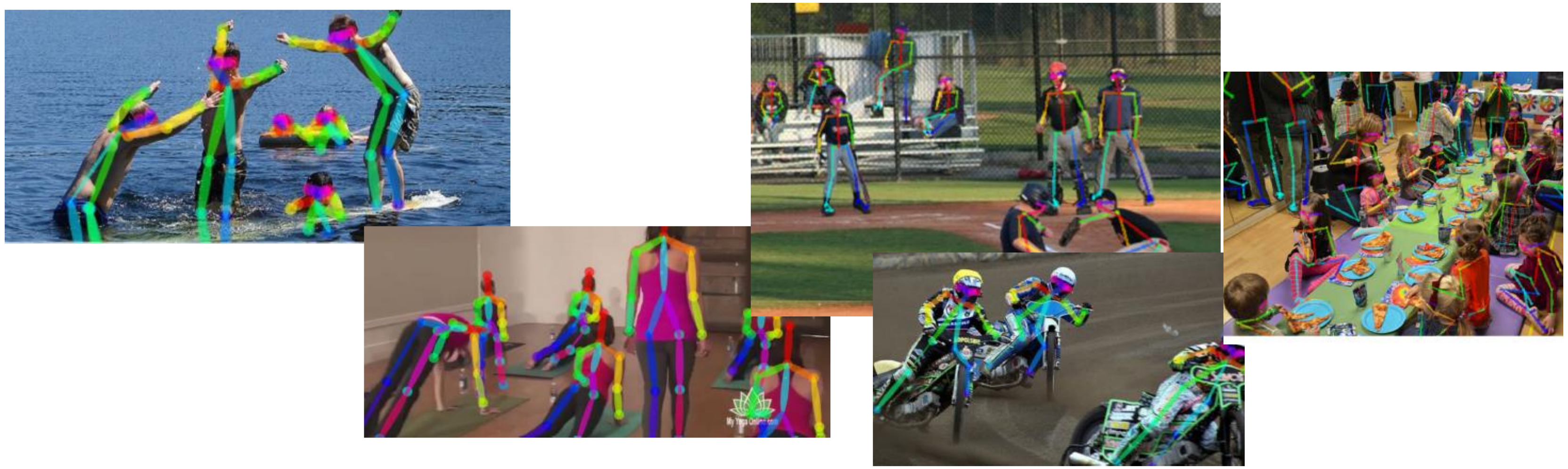
  - The z-axis z is the cross product of x and y

$$z = y \times x$$

  - The 3 × 3 matrix [x | y | z] defines a rotation *from the measurement coordinate system to the intermediate coordinate system*. We compute an equivalent rotation $[x_r \mid y_r \mid z_r]$ from the reference coordinate system.
  - The desired rotation matrix **R** is simply the product of the second rotation with the inverse of the first

$$\mathbf{R} = [\mathbf{x}_r \mid \mathbf{y}_r \mid \mathbf{z}_r][\mathbf{x} \mid \mathbf{y} \mid \mathbf{z}]^T$$

Human pose: OpenPose algorithm



Cao, Zhe, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. "OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields." IEEE transactions on pattern analysis and machine intelligence 43, no. 1 (2019): 172-186.

https://github.com/CMU-Perceptual-Computing-Lab/openpose

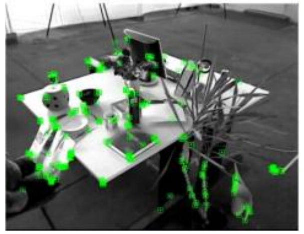
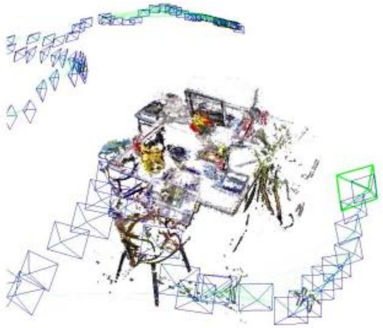
Università di Genova | DIBRIS DIPARTIMENTO DI INFORMATICA, BIOINGEGNERIA, ROBOTICA E INGEGNERIA DEI SISTEMI

Natural Feature Tracking

Natural Feature Tracking

- In the previous two case studies, we have considered *artificial markers*.
- Here, we introduce the use of **natural feature tracking** (***inside-out***) to determine the **camera pose** from observations in the image **without** instrumenting the environment with **markers**.

- We consider monocular **tracking** with a **single camera**.

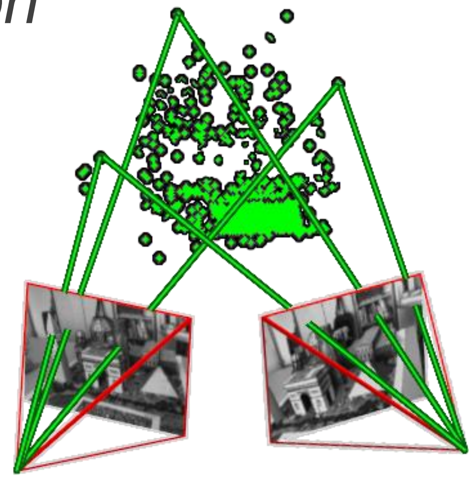
- First, we describe **tracking by detection**: *the camera pose is determined from matching interest points (corners) in every frame anew, without relying on prior information gleaned from previous frames.*

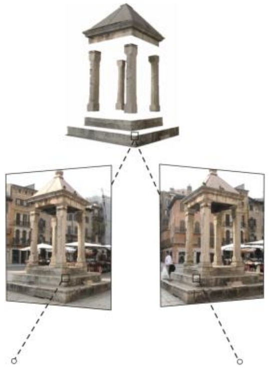
Natural feature tracking algorithm: tracking by detection

- A typical pipeline for **tracking by detection** of **sparse interest points (corners)** consists of five stages:
  1. Interest point detection
  2. Descriptor creation
  3. Descriptor matching
  4. Perspective-n-Point camera pose determination
  5. *Robust pose estimation*

Corners, SIFT, ORB

Essential matrix estimation (*calibrated camera*)





Natural feature tracking algorithm: tracking by detection

- A typical pipeline for **tracking by detection** of **sparse interest point** consists of five stages:

  …

  5. Robust pose estimation $\Big\}$ random sampling consensus (RANSAC)

RANSAC:

- To estimate the model parameters **x** from a randomly chosen subset of the data points.
- For every one of the remaining, potentially many, point correspondences, we compute the residual error, assuming the camera pose computed.
- A data point with a residual smaller than a threshold counts as an inlier. If the ratio of inliers to outliers is not sufficient, the procedure is repeated.

Natural feature tracking algorithm: incremental Tracking

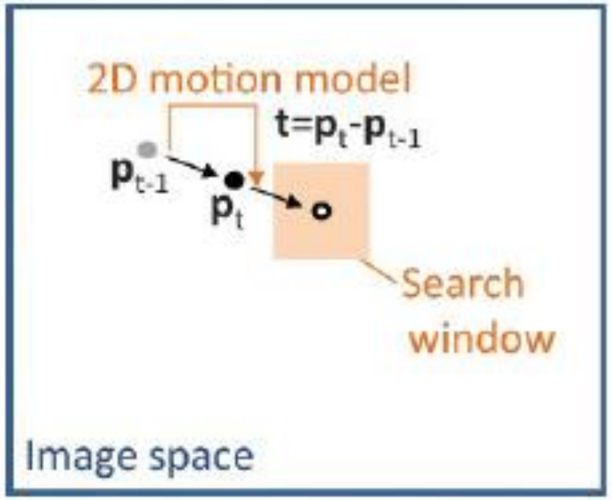
- Since **interaction** in AR requires **real-time update rates**, *neither the camera pose nor the projection of feature points to the image will change drastically from one frame to the next.*
- Tracking by detection ignores this **coherence**, *so that the tracking problem becomes harder to solve than necessary.*
- A tracking system that uses **information from a previous step** is said to use **incremental tracking** or *recursive tracking*.

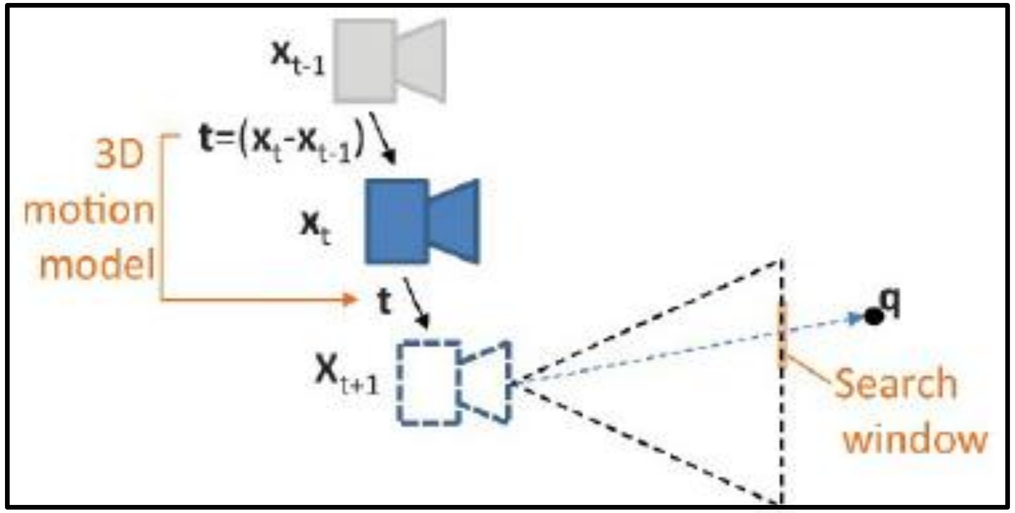
- If the last tracking iteration was successful, there is good reason to believe that *we can be successful again by searching for the inliers from the last frame* and searching **close to their last known positions**.

- Incremental tracking requires two components:
  1. an incremental search component;
  2. an interest point matching component.

Incremental Tracking

Active search





In the simplest case, if **no 3D tracking model** is available, a motion model can be obtained purely from the 2D positions of interest points (optic flow).

If a **3D tracking model** can be obtained, a corresponding 3D motion model will usually give a better result.

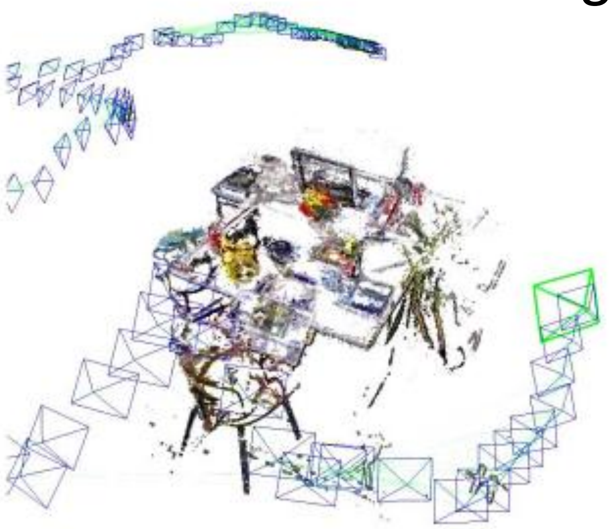
Incremental Tracking

- The classic approach to *incremental tracking* is the Kanade-Lucas-Tomasi (**KLT**) tracker, which extracts **keypoints** from an initial image and then tracks them using **optical flow**.

- Hierarchical Search:
  - Even a medium-quality video stream of 640 × 480 pixel resolution at 30 Hz from a handheld camera often contains hundreds of features
  - Naive tracking would require very large search windows, making the tracking *computationally expensive*
  - It is usually sufficient to employ a simple (*two-level*) **image pyramid**
  - Only a **small number** of strong **features** (e.g., 20–30 features) is tracked at this resolution using the predicted camera pose from the motion model (moreover **small** search **window**)
  - The **camera pose** resulting from this coarse tracking step is **not yet accurate** enough, it is adequate for **initialization** of tracking at the **full resolution** by using a much smaller search window

Natural feature tracking algorithm: Simultaneous Localization and Mapping

- *We can extend the incremental tracking*: we consider the <u>**Simultaneous Localization and Mapping**</u> (**SLAM**) algorithm:
  - **Localization** (visual odometry) means continuous 6DOF (*rotation and translation*) tracking of a camera pose relative to an arbitrary starting point
  - **Mapping** is to create a map using consistent data association of observations to points in the scene (*useful if a point moves out of sight and is later reacquired, loop closure*)



Simultaneous Localization and Mapping (SLAM) algorithm

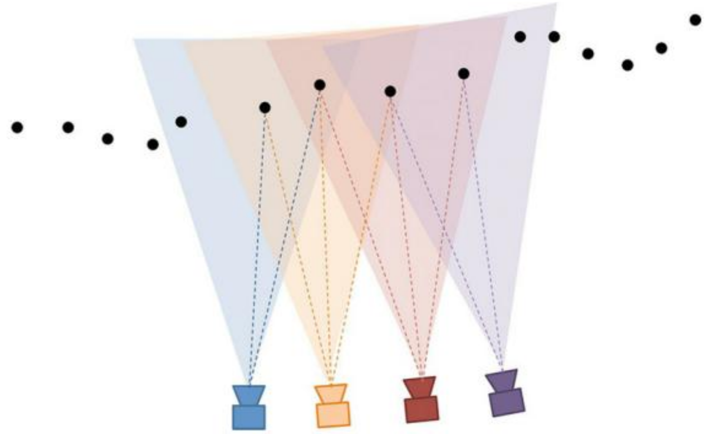
1. Detect **interest points** in a frame, *e.g. corners, SIFT, ORB*
2. **Track** the interest point in 2D from the previous frame, *e.g. using KLT*
3. Determine the **essential matrix** (*calibrated camera*) between the current and previous frames (*inside a RANSAC loop for refining it*)
4. Recover the *incremental* **camera pose** from the essential matrix
5. The essential matrix determines the **translation** part of the pose *only up to scale*, but it must be consistent throughout the tracked image sequence. Thus, 3D point locations are triangulated from multiple 3D observations of the same image feature over time (**bundle adjustment**)
6. Proceed to the next frame

Simultaneous Localization and Mapping

- Bundle Adjustment:
  - A naive visual odometry approach, such as that previously described, will likely accumulate *drift over time*. This problem can be attacked by minimizing the reprojection error

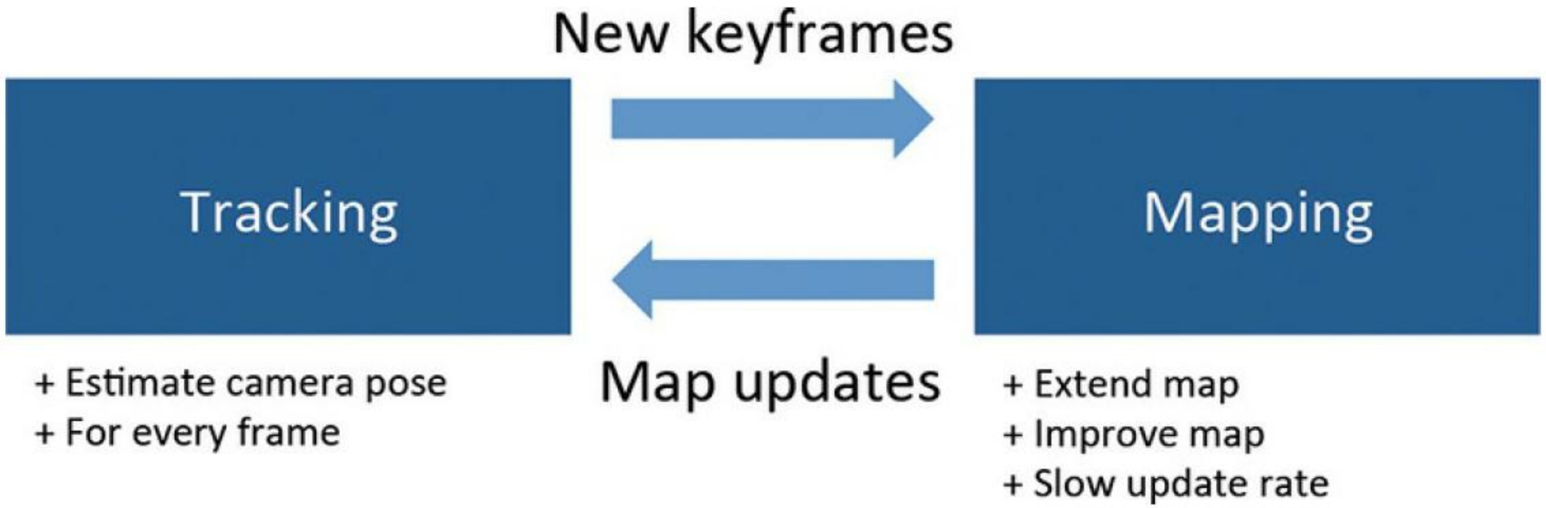
$$\arg\min_{X_k, q_i} \sum_k \sum_i \rho(\mathrm{K}\mathrm{P}_\mathrm{k}\mathrm{q}_\mathrm{i} - \mathrm{p}_{\mathrm{k,i}})$$

  - Moreover, the computation is limited to a certain spatial region



Simultaneous Localization and Mapping

- **Parallel tracking and mapping** (PTAM) [Klein and Murray 2007] is a more recent approach, which decouples the tracking from the mapping
- PTAM lets both tracking and mapping execute *in parallel threads*, but it permits them to have different update frequencies.

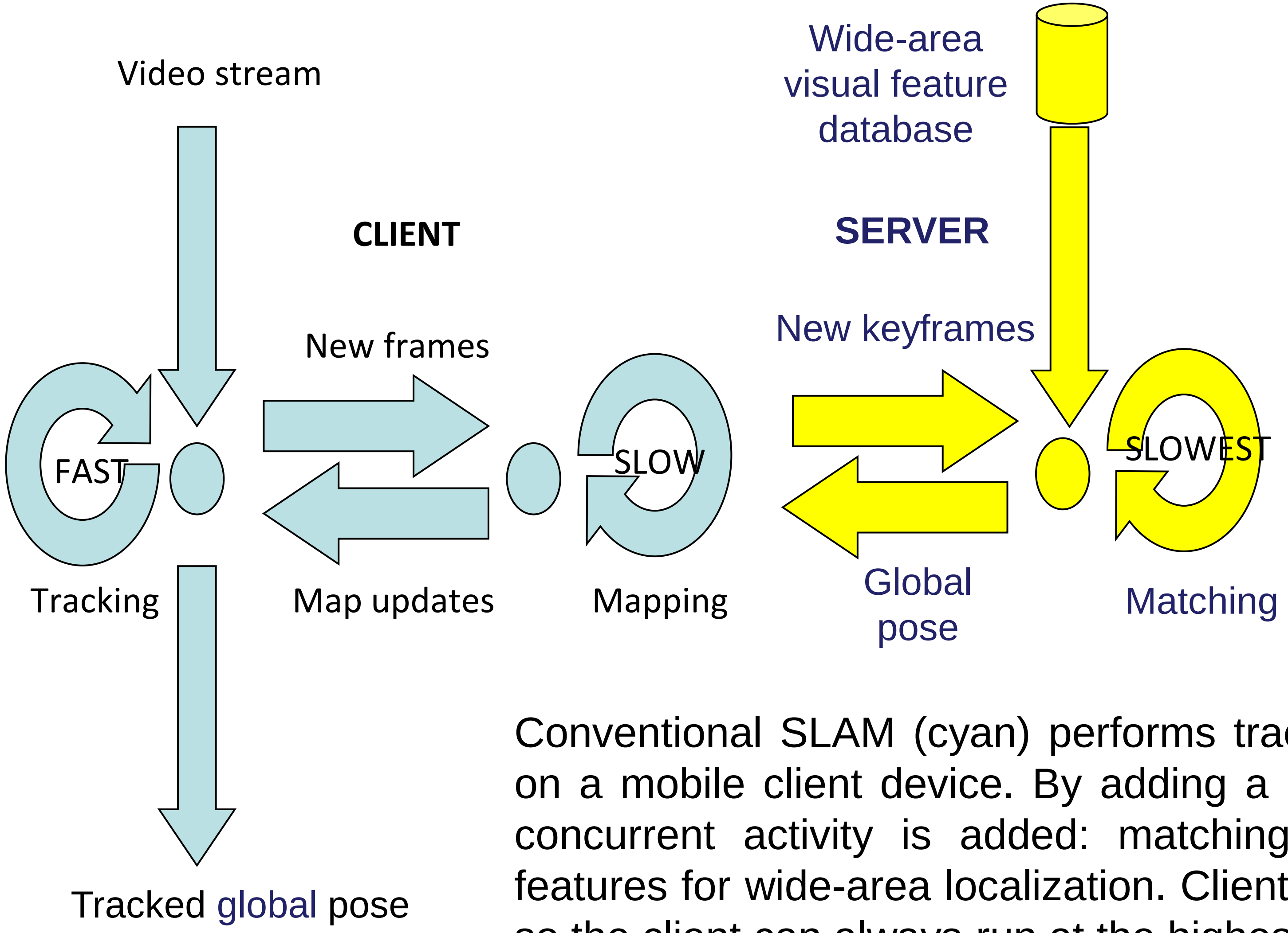


Keyframe is a frame from the video stream that represent a diverse camera poses

Outdoor Tracking

- *The tracking methods we have described so far are primarily intended for indoor use*.
- Outdoor tracking is generally more difficult than indoor tracking for the following reasons
  - **Mobility**. The user is free to go anywhere. The algorithm must run on mobile devices.
  - **Environment**. Many areas with poor or unusable textures. Variations can quickly make any tracking model outdated.
  - **Localization database**. The tracking model can grow very large.
  - **The user**. In general, *we cannot expect a naive user of an AR system to understand the system's operation in depth*.

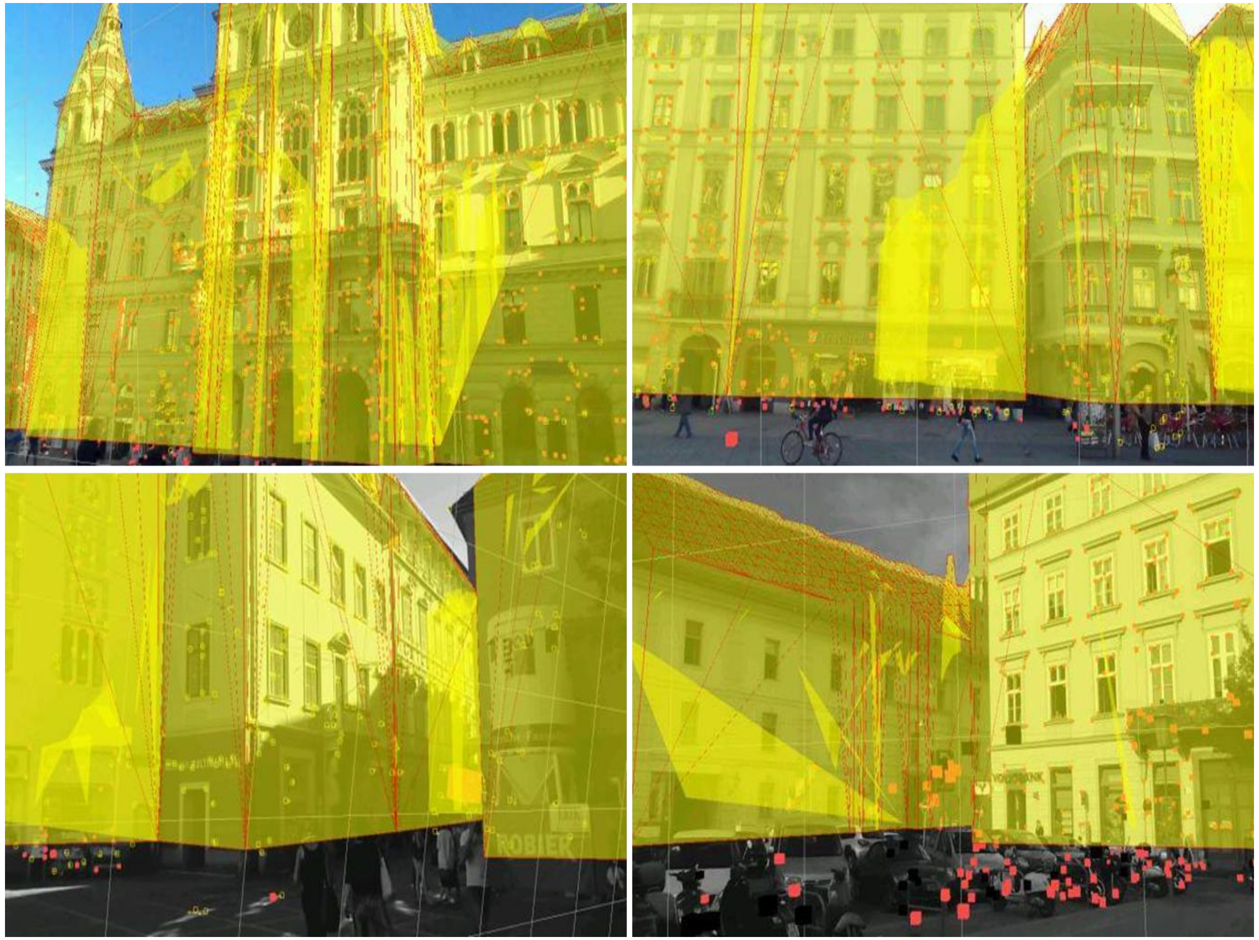
Outdoor Tracking



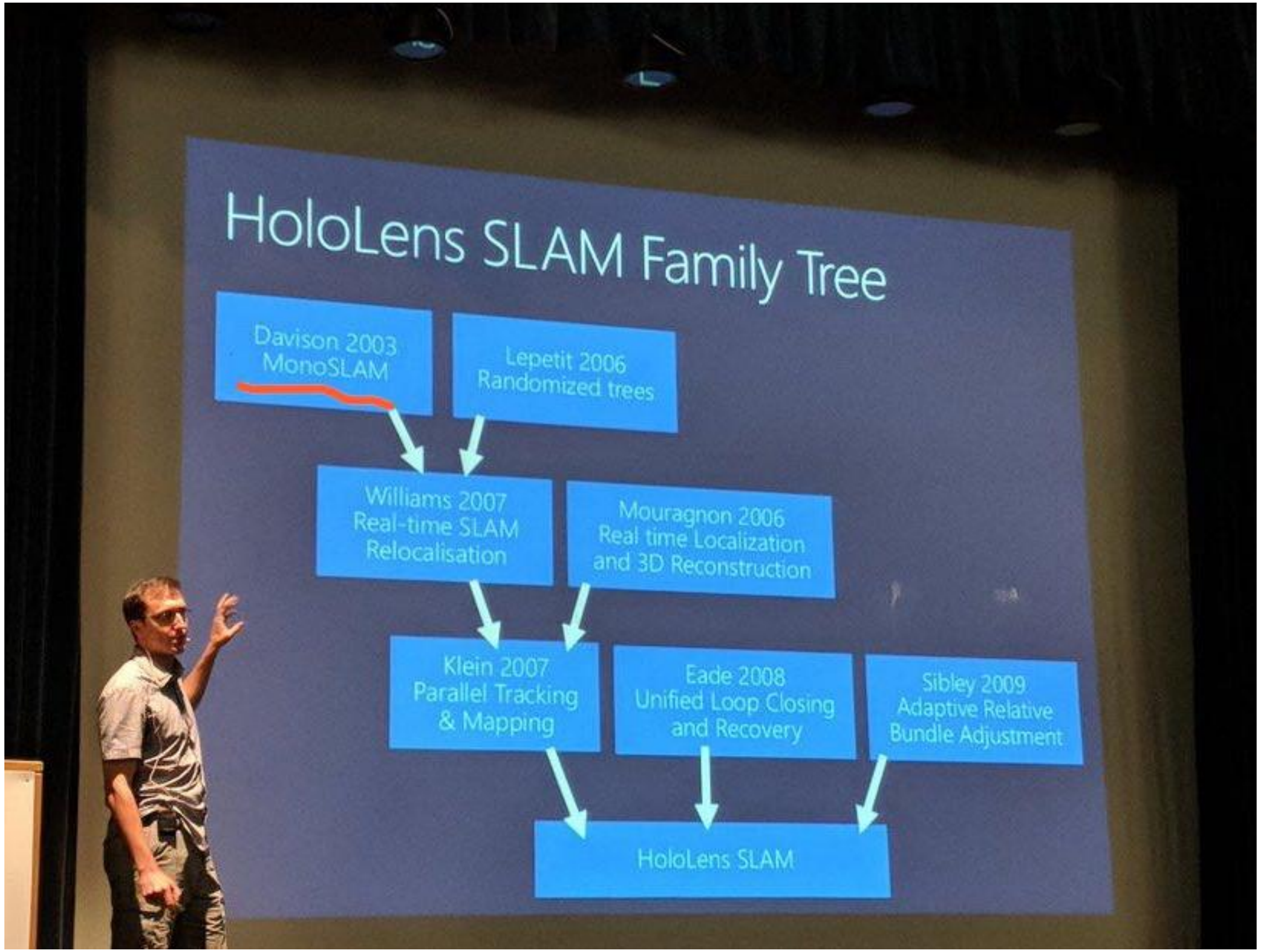
Conventional SLAM (cyan) performs tracking and mapping simultaneously on a mobile client device. By adding a localization server (yellow), a third concurrent activity is added: matching to a global database of visual features for wide-area localization. Client and server operate independently, so the client can always run at the highest frame rate.

Outdoor Tracking

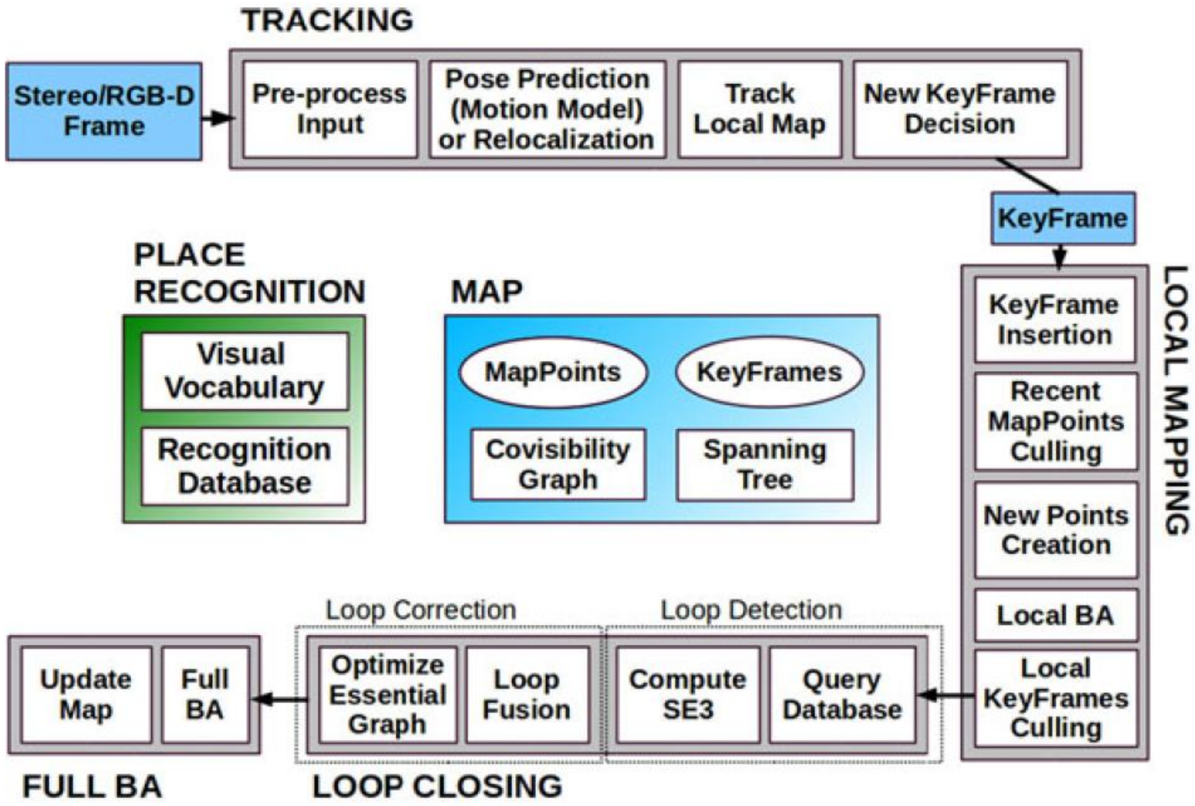
Multiple images from a sequence tracked with 6DOF SLAM on a client, while a localization server provides the global pose used to overlay the building outlines with transparent yellow structures.



Simultaneous Localization and Mapping



Simultaneous Localization and Mapping



Mur-Artal, Raul, and Juan D. Tardós. “ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras." IEEE transactions on robotics 33, no. 5 (2017): 1255-1262.