Computer Vision and Mixed Reality

Computer Vision for Augmented Reality

Chapter 4

Pedro Mendes Jorge

Introduction

- **Computer vision** for AR is concerned with electronically perceiving and understanding imagery from camera sensors that can inform the AR system about the user and the surrounding environment.

- The objective of optical tracking is to **determine the pose** of an object in the real world relative to a camera.

- Registration and tracking in AR necessitates **real-time** approaches.

Augmented Reality Applications

- Marker Tracking
- Multiple-Camera Infrared Tracking
- Natural Feature Tracking by Detection

Marker Tracking

- Marker tracking is a simple approach that introduces a basic camera representation, contour-based shape detection, pose estimation from a homography, and nonlinear pose refinement.

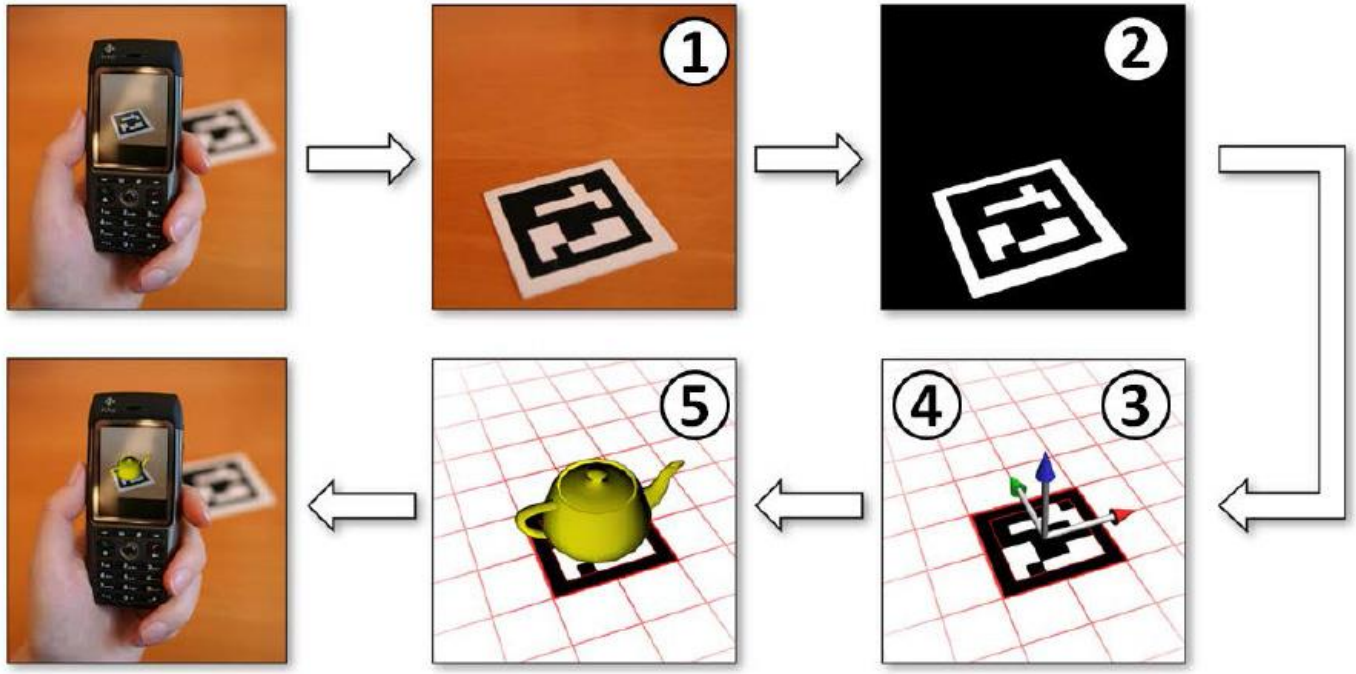
- Marker tracking is computationally inexpensive and can deliver useful results even with rather poor cameras.

- Detecting the four corners of a flat marker in an image from a single calibrated camera delivers just enough information to recover the pose of the camera relative to the marker.

Marker Tracking

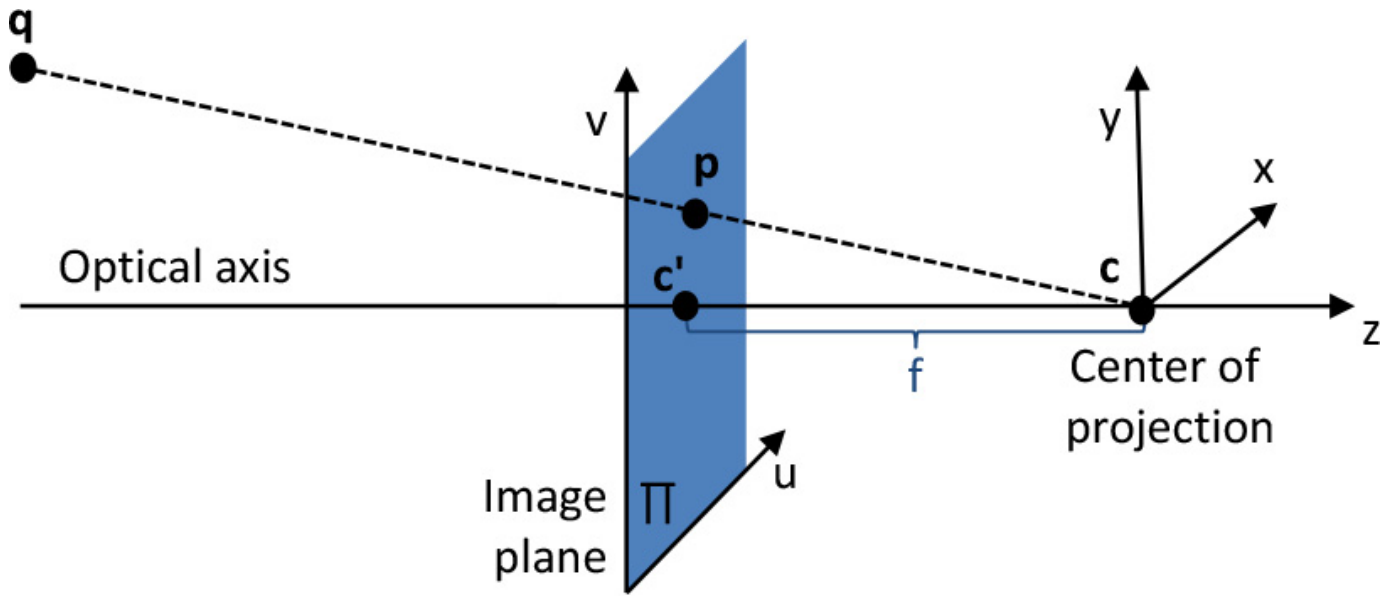
Marker tracking pipeline

1. Capturing an image using a camera with a known mathematical representation;
2. Marker detection by searching for quadrilateral shapes;
3. Pose estimation from a homography;
4. Pose refinement by nonlinear reprojection error minimization;
5. AR rendering with the recovered camera pose;



Marker Tracking

Camera Representation – Pinhole Model



$$\begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix} \propto \mathbf{M} \begin{bmatrix} q_x \\ q_y \\ q_z \\ 1 \end{bmatrix}$$

The perspective projection depends on **internal and external camera parameters**.

$\mathbf{M}$ has 11 degrees of freedom (11DOF), one less than the number of matrix elements.

Perspective Projection

The **internal parameters** describe the geometric properties of the camera itself in a 3 × 3 **matrix $\mathbf{K}$**, the camera calibration matrix.

The **external parameters** describe the pose of the camera relative to the origin of the world coordinate system in a 3 × 4 matrix **$[\mathbf{R} \mid \mathbf{t}]$**, which is itself composed of a 3 × 3 **rotation matrix $\mathbf{R}$** and a **translation vector t.**

$$\mathbf{M} = \mathbf{K}\,[\mathbf{R} \mid \mathbf{t}]$$

Internal Parameters – Camera Calibration

Camera calibration means determining its internal parameter matrix $\mathbf{K}$.

$$\mathbf{K} = \begin{bmatrix} f_u & s & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix}$$

- f_u and f_v describe the focal length of the camera, scaled by the size of a pixel in the directions *u* and *v*, respectively;
- c_u and c_v describe the offset in image coordinates for the principal point c.
- s is the skew factor and needed only if the image directions u and v are not perpendicular and is often 0 for common cameras.

Camera calibration is done once in an offline step, and afterward $\mathbf{K}$ is assumed to be fixed. This assumption implies that no change to the lens—no zooming—is allowed.

External Parameters

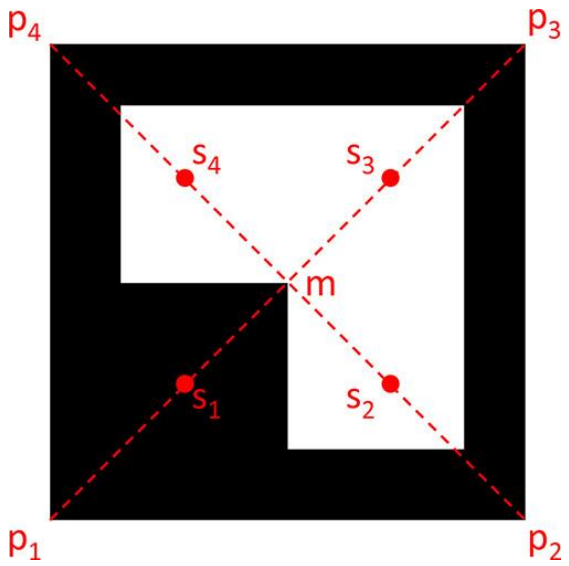
The 11DOF of the projection matrix $\mathbf{M}$ can be decomposed into the 5DOF for $\mathbf{K}$, 3DOF for $\mathbf{R}$, and 3DOF for $\mathbf{t}$.

The 3 × 3 matrix $\mathbf{R}$ is an orthogonal matrix, and its nine elements redundantly encode the 3DOF of orientation.

Pose estimation try to recover $\mathbf{R}$ and $\mathbf{t}$ with a tracking algorithm.

Marker Detection

Black square outline of a given thickness on white background.



In this simple example one quarter of the marker interior is covered in black to provide a unique orientation.

Marker Detection

Marker detection algorithm:

1. Contour extraction (example: thresholding + find contours).
2. Remove borders with a small number of points.
3. Polygonal approximation of the contour with exactly 4 corners (example: Convex Hull approximation).
4. Remove too close rectangles.

Marker Identification

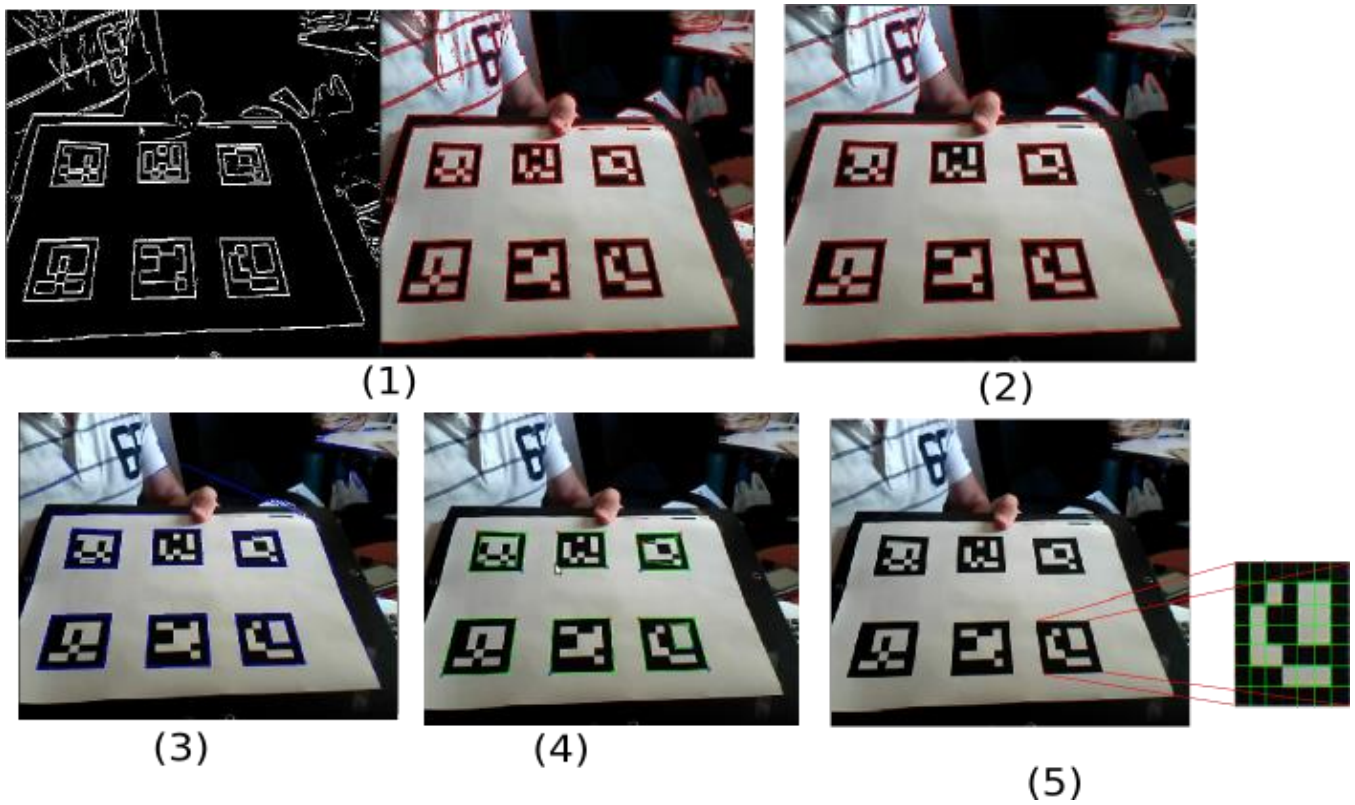
Marker identification algorithm:

1. Correct the projection perspective to obtain a frontal view of the rectangle using a homography.
2. Threshold the area using Otsu.
3. Identification of the internal code.

   The marker is divided in a $N \times N$ grid, of which the internal $(N-2) \times (N-2)$ cells contains ID information.
4. For the valid markers, refine corners using subpixel interpolation

Marker Detection and Identification

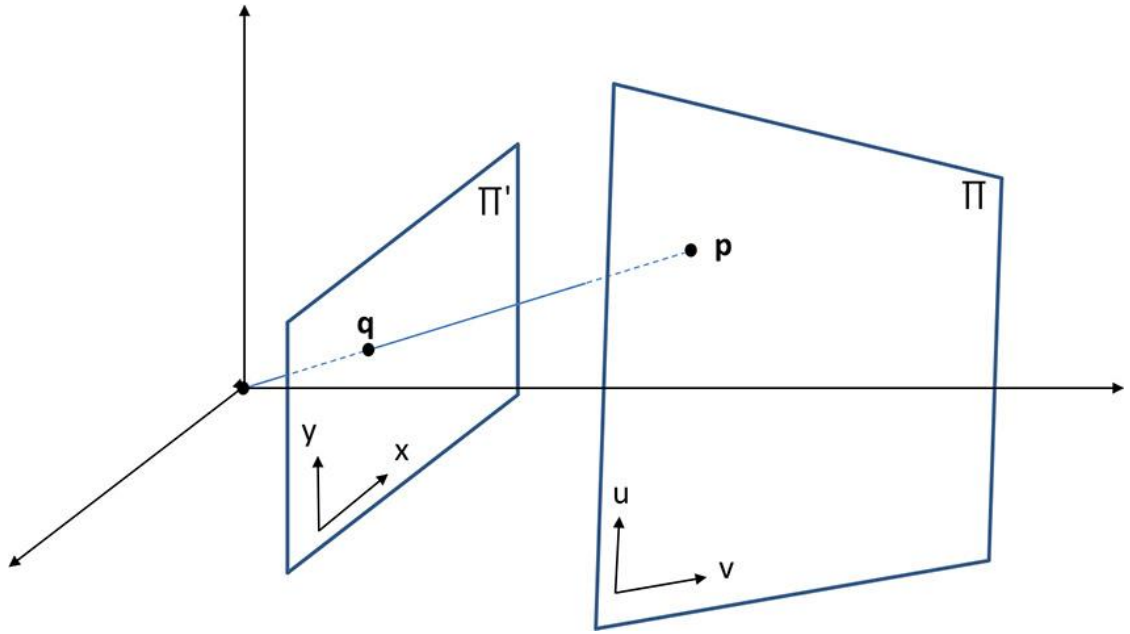
ArUco example



[3] ArUco: An efficient library for detection of planar markers and camera pose estimation

Pose Estimation from Homography

- The transformation between de marker plane in world coordinate system and its polygonal representation on the image plane is defined by a homography (plane to plane transformation).



Pose Estimation from Homography

- The marker corners in the world coordinate system have $q_z = 0$.
- The homography transformation is define by a 3x3 matrix $\mathbf{H}$.

$$\begin{bmatrix} p_u \\ p_v \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} q_x \\ q_y \\ 1 \end{bmatrix} = \mathbf{H} \begin{bmatrix} q_x \\ q_y \\ 1 \end{bmatrix}$$

- To estimate $\mathbf{H}$ (8 DOF) at least 4 corresponding points in both planes are needed.
- The estimation process evolves the solution of a homogeneous equation $A\mathbf{h} = 0$, where $\mathbf{h}$ has the h_{ij} elements of $\mathbf{H}$.
- To solve these equations a SVD or LSE methods can be used.

Pose Estimation from Homography

From the homography matrix $\mathbf{H}$ it is possible to factorize in $\mathbf{R}$ and $\mathbf{t}$.

$$\begin{aligned} p = \mathbf{M}q &= \mathbf{K}[\mathbf{R}|\mathbf{t}][q_x \quad q_y \quad q_z \quad 1]^T \\ &= \mathbf{K}[R_{C1}|R_{C2}|R_{C3}|t][q_x \quad q_y \quad 0 \quad 1]^T \\ &= \mathbf{K}[R_{C1}|R_{C2}|t][q_x \quad q_y \quad 1]^T \\ &= \mathbf{H}q' \end{aligned}$$

so

$$\mathbf{H} = \mathbf{K}[R_{C1}|R_{C2}|t]$$

$$\mathbf{K^{-1}H} = [R_{C1}|R_{C2}|t]$$

As $\mathbf{R}$ is an orthogonal matrix

$$R_{C3} = R_{C1} \text{x}\, R_{C2}$$

Pose Refinement

- Pose estimation cannot always be computed directly from imperfect point correspondences with the desired accuracy.
- Pose estimation can be refined by iteratively minimizing the **reprojection error**.
- The displacement of the known points $\mathbf{q}_i$ in 3D, projected using $\mathbf{K}[\mathbf{R}|\ \mathbf{t}]$, from its known image location $\mathbf{p}_i$, can be minimized.

$$\arg\min_{\mathbf{R},\mathbf{t}} \sum_i (\mathbf{K}[\mathbf{R}|\ \mathbf{t}]\mathbf{q}_i - \mathbf{p}_i)^2$$

- This kind of quadratic minimization problem can be solved iteratively.

Multiple-Camera Infrared Tracking

- In general, the known points in the world will not be constrained to a plane, as assumed on tracking of flat markers.
- For tracking arbitrary objects, we require **general pose estimation**.
- Determining the camera pose from 2D–3D correspondences between known points $\mathbf{q}_i$ in world coordinates and their projections $\mathbf{p}_i$ in image coordinates.

Multiple-Camera Infrared Tracking – Example

Infrared tracking system designed to track rigid body markers composed of four or more retro-reflective spheres.

- It is used an **outside-in setup** with multiple infrared cameras.
- A minimum of two cameras in a known configuration — **a calibrated stereo camera rig** — is required.
- Multiple viewing angles (additional cameras) will improve the tracking quality and the working volume.
- In practice, four cameras set up in the corners of a laboratory space are a popular configuration.

Multiple-Camera Infrared Tracking – Example

The stereo camera tracking pipeline consists of the following steps:

1. Blob detection in all images to locate the spheres of the rigid body markers.
2. Establishment of point correspondences between blobs using epipolar geometry between the cameras.
3. Triangulation to obtain 3D candidate points from the multiple 2D points.
4. Matching of 3D candidate points to 3D target points.
5. Determination of the target's pose using absolute orientation.

Multiple-Camera Infrared Tracking – Blob Detection

The targets/markers are composed of four or five spheres covered with retro-reflective foil in a known rigid structure.

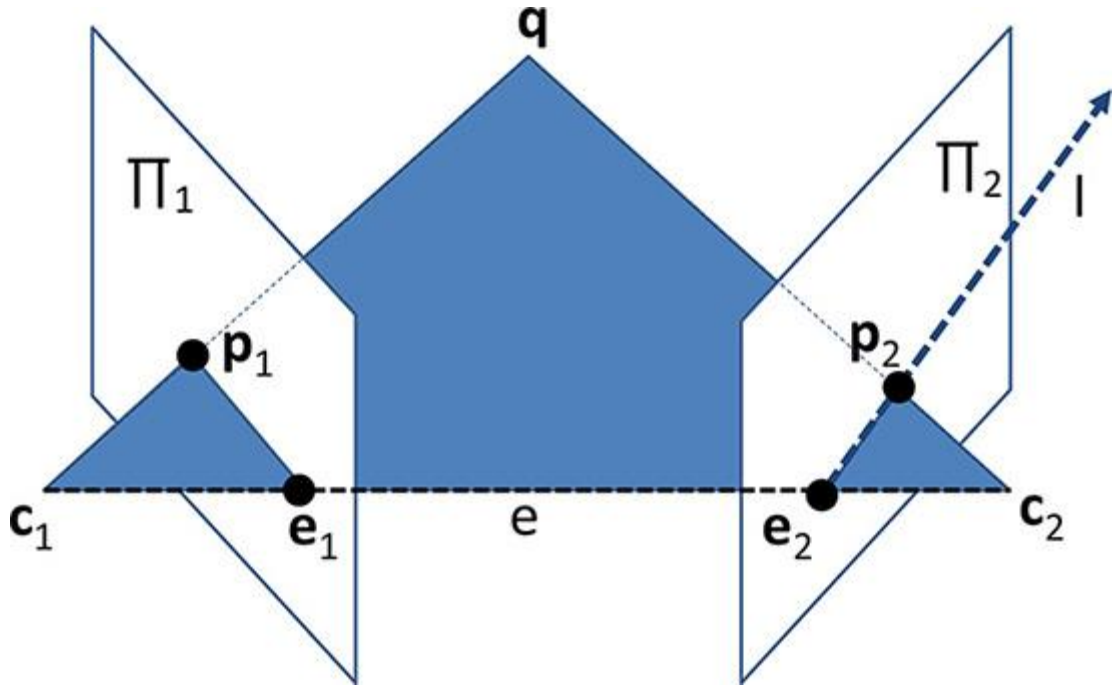
The cameras capture images of infrared light reflected from the spheres, resulting in images that include high-intensity blobs for each sphere.

The binary images (simple thresholding) are scanned for connected regions consisting of white pixels.

Regions that are too small or too elongated are rejected.

The centroids are computed and returned as candidate points.

Point Correspondences – Epipolar Geometry



- The candidate 2D points in the two images Π_1 and Π_2 can be related using epipolar lines.
- **e (baseline)**, $\mathbf{e}_1$ and $\mathbf{e}_2$ **(epipoles)** defines the **epipolar geometry** of two cameras with centers $\mathbf{c}_1$, $\mathbf{c}_2$ and image planes Π_1, Π_2.
- A 3D point $\mathbf{q}$ projects to $\mathbf{p}_1 \in \Pi_1$ and $\mathbf{p}_2 \in \Pi_2$.
- The baseline (e) from $\mathbf{c}_1$ to $\mathbf{c}_2$ intersects Π_1 at the epipole $\mathbf{e}_1$ and Π_2 at the epipole $\mathbf{e}_2$.

Point Correspondences – Epipolar Geometry

Tracking with stereo cameras usually involves establishing correspondences for a given point $\mathbf{q}$ in the two images.

This means we know that there is a 2D point $\mathbf{p}_1 \in \Pi_1$, which is the projection of $\mathbf{q}$, but we do not know where $\mathbf{q}$ is.

We know that $\mathbf{p}_2$ must lie on the epipolar line $\mathbf{l}$ passing through $\mathbf{e}_2$.

Point Correspondences – Epipolar Line

We are considering a stereo camera system, which has already been calibrated.

The world coordinate system is chosen so that it coincides with the camera coordinate system of Π_1.

Both the internal parameters $\mathbf{K}_1$, $\mathbf{K}_2$ of the cameras and the external parameters $[\mathbf{R}|\mathbf{t}]$ relating the second to the first camera are known.

Point Correspondences – Epipolar Line

The epipolar line is defined as

$$\mathbf{l} = \mathbf{F}.\mathbf{p}_1$$

where $\mathbf{F}$ is the **fundamental matrix.** The fundamental matrix is the algebraic representation of epipolar geometry.

$\mathbf{F}$ can be rewrite using **essential matrix** $\mathbf{E}$ as

$$\mathbf{F} = \mathbf{K}_2^{-T}\mathbf{E}\mathbf{K}_1^{-1}$$

$\mathbf{E}$ can be defined as

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

where $[\cdot]_\times$ is the skew-symmetric matrix.

Replacing the previous equations in the first, the epipolar line can be computed as

$$\mathbf{l} = \mathbf{K}_2^{-T}[\mathbf{t}]_\times \mathbf{R}\mathbf{K}_1^{-1}.\mathbf{p}_1$$

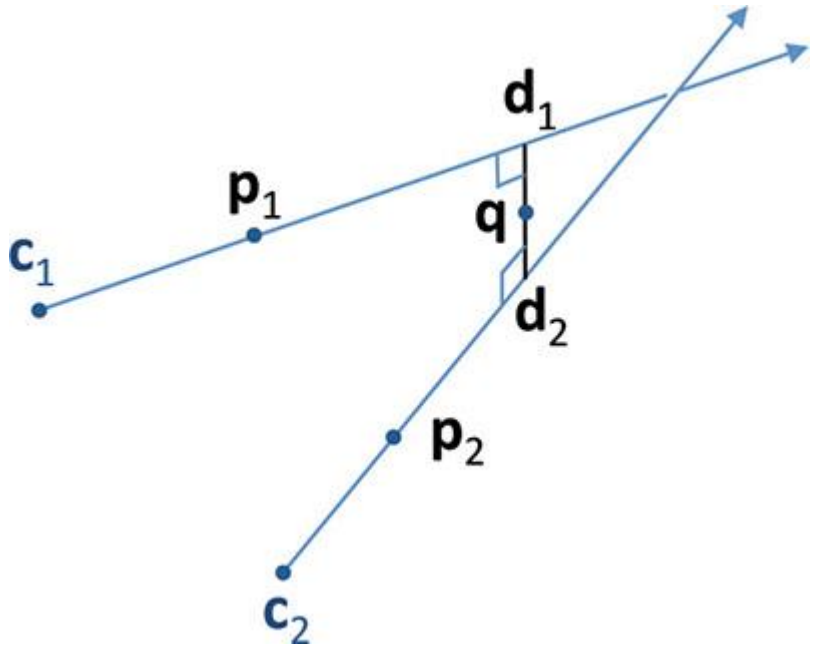
Point Correspondences

- To find the projection of **q** in Π_2, $\mathbf{p}_2$, we search along the epipolar line **l** for interest points that correspond to the scene observed at $\mathbf{p}_1$.
- Any candidate point in Π_2 closer than a threshold to **l** is considered for **triangulation**.
- Ideally, only a single candidate point in Π_2 is found.
- A useful verification step for the matching is to compute epipolar lines from the first camera to the second, and vice-versa, and retain only associations that are consistent in both directions.
- If no unique match can be found, the right data association must be determined from the structure of the target.

Triangulation from Two Cameras

Triangulation - compute the 3D point corresponding to the associated 2D points.

- Given $\mathbf{p}_1$ and $\mathbf{p}_2$, we can compute $\mathbf{q}$ by identifying intersecting rays originating at $\mathbf{c}_1$ and $\mathbf{c}_2$.
- Due to various calibration errors, the rays will not meet exactly.
- For two cameras, we can find the two closest distance points along the rays, $\mathbf{d}_1$ and $\mathbf{d}_2$, and compute their midpoint $\mathbf{q}$



Triangulation from Two Cameras

Closest Point to Two Rays

We start with two rays that come closest to each other at points $\mathbf{d}_1$ and $\mathbf{d}_2$:

$$d_1 = c_1 + t_1(p_1 - c_1) = c_1 + t_1 v_1$$
$$d_2 = c_2 + t_2(p_2 - c_2) = c_2 + t_2 v_2$$

Because the connecting line from $\mathbf{d}_1$ to $\mathbf{d}_2$ is perpendicular to both rays, the dot product of the connecting line and each ray must be zero:

$$v_1.(d_2 - d_1) = 0$$
$$v_2.(d_2 - d_1) = 0$$

After solving this linear system of equations for t_1 and t_2, the point $\mathbf{q}$ is finally obtained:

$$q = (c_1 + t_1 v_1 + c_2 + t_2 v_2)/2$$

Triangulation from More Than Two Cameras

- Cannot rely on a midpoint computation like the previous example for two cameras.
- We have a calibrated multi-camera setup: for each camera, **the internal calibration matrix** and the **external parameters** relating it to the world coordinate system defined by the first camera.
- Projection matrix $\mathbf{M}$ for each camera can be computed.
- The 3D point with the minimum error given the multiple 2D observations can be computed with the same method used in the estimation of the homography matrix.

Matching Targets Consisting of Spherical Markers

- The **candidate points** $\mathbf{q}_i$ obtained from the triangulation step must be matched to the **target points** $\mathbf{r}_j$.
- It is considered that $j \leq i$; that is, there may be more candidate points than target points because of ambiguous observations (spurious candidate points and/or spheres that are occluded).
- The association from candidate points to target points is resolved using the known **geometric structure of the target**.
- The distance between any two points and the angles of the triangle formed by any three points should yield a **unique signature**.

Matching Targets Consisting of Spherical Markers

Matching Algorithm (Pintaric and Kaufmann, 2008)

1. Select all permutations of *j* points out of the *i* candidate points and compute the signature of this permutation.

2. The signatures are then compared to the target signature, and permutations with an error exceeding a threshold are rejected.

3. Among the remaining permutations, the one with the lowest error is elected.

Absolute Orientation

- The target points $\mathbf{r}_i$ are specified in a **reference coordinate system**.
- Given the association between the observed points $\mathbf{q}_i$ and the target points $\mathbf{r}_i$, we would like to compute the pose $[\mathbf{R}|\mathbf{t}]$ of the observed target relative to the reference coordinate system.
- The pose $[\mathbf{R}|\mathbf{t}]$ can be estimated with the Horn Algorithm (Horn, 1987).
- This approach must be refined by minimizing the least squares error, while considering all measurements.
- With the pose $[\mathbf{R}|\mathbf{t}]$ of a single target, it can be used for AR rendering.
- Several targets can be handled at the same time, as long as their signatures are sufficiently different to discriminate them.

Absolute Orientation – Horn Algorithm

- At least three points are required.
- The translation $\mathbf{t}$ 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 $\mathbf{R}$, we assume the origin at $\mathbf{q}_1$ and the x-axis $\mathbf{x}$ aligned with the vector from $\mathbf{q}_1$ to $\mathbf{q}_2$. The y-axis $\mathbf{y}$ is orthogonal to $\mathbf{x}$ and lies in the plane given by $\mathbf{q}_1$, $\mathbf{q}_2$ and $\mathbf{q}_3$. The z-axis $\mathbf{z}$ is the cross-product of x and y.

$$\mathbf{x} = \underline{N}(\mathbf{q}_2 - \mathbf{q}_1)$$
$$\mathbf{y} = \underline{N}\left((\mathbf{q}_3 - \mathbf{q}_1) - \left(((\mathbf{q}_3 - \mathbf{q}_1) \cdot \mathbf{x})\mathbf{x}\right)\right)$$
$$\mathbf{z} = \mathbf{y} \times \mathbf{x}$$

where $\underline{N}(\mathbf{x}) = \mathbf{x}/\|\mathbf{x}\|$ defines the normalization of a vector.

Absolute Orientation – Horn Algorithm (cont.)

- The **3 × 3** matrix $[\mathbf{x}|\mathbf{y}|\mathbf{z}]$ defines a rotation from the measurement coordinate system to the intermediate coordinate system.
- We compute an equivalent rotation $[\mathbf{x}_r|\mathbf{y}_r|\mathbf{z}_r]$ from the reference coordinate system to the intermediate coordinate system.
- The desired rotation matrix is simply the product of the second rotation with the inverse of the first:

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

Natural Feature Tracking by Detection

- In more **practical situations** we would like to use **natural feature tracking** to determine the **camera pose from observations** in the image **without** instrumenting the environment with **markers**.
- First a suitable digital model is reconstructed by scanning the physical environment, and then the tracking model is matched at runtime to observations from the camera.
- The wide availability of mobile devices with built-in cameras makes this the preferred tracking hardware for mobile AR.
- Depth cameras can also be used for natural feature tracking.

Natural Feature Tracking by Detection

- Monocular tracking (single camera) implies that the objective of tracking is to determine correspondences between 2D points in the camera image and known 3D points in the world.

- The correspondence 2D-3D can be done with:
  - **Sparse matching:** try to find a correspondence for a small but sufficient number of salient interest points selected from the image;
  - **Dense matching:** try to find a correspondence for every pixel in an image.

Natural Feature Tracking by Detection – Sparse Matching

Sparse Matching

- Tracking models consists of **sparse interest points** that are **easier to produce**.
- These models are rather **compact** and can be **efficiently stored** and processed for **matching**.
- Sparse interest points are **handled independently**, is robust to occlusion or changes in illumination.
- The discrete nature of interest points provides **resilience** against background clutter.

Natural Feature Tracking by Detection – Dense Matching

Dense Matching

- Can better deal with difficult environments (objects with poor texture, repetitive structures, and reflective surfaces).
- Matching many redundant image points can also better accommodate image noise.
- Dense models matching increase computational cost for handling dense image points.

Natural Feature Tracking by Detection

- The camera pose is determined from matching interest points in **every frame anew**, without relying on **prior** information gleaned from previous frames.
- The interest points are represented by **descriptors**, which are data structures designed for quick and reliable matching.
- Descriptors are created for the interest points found in the new camera image and matched with the ones in the tracking model.

Natural Feature Tracking by Detection

A typical pipeline for **tracking by detection** of sparse interest point 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

Natural Feature Tracking by Detection - Interest Point Detection

What are the right features to use?

Those that can be matched reliably!

- The area around the interest point should be **visually distinct**.
- Interest points should be **sufficiently textured**, with **high-contrast** intensity changes occurring within a small local neighborhood and forming **reliably identifiable structures**, such as **corners** or **T-junctions**, or **circular blobs**.
- Ideally an interest point should **not be part of a repetitive structure**.
- Interest points should be relatively **uniformly distributed in the image**.

Natural Feature Tracking by Detection - Interest Point Detection

- The interest point selection should be **repeatable**, the detection algorithm should select the same interest points **independent** of observation parameters such as **viewpoint** and **illumination condition**.
- Interest point detection should be **robust to rotation**, **scale**, **perspective transformation**, **and lighting changes**.
- The model should be **not too sparse** (so that a reliable result can be computed) **or too dense** (so that the system can handle the computations in real time).

Natural Feature Tracking by Detection - Interest Point Detection

There are many approaches that fulfill some or many of these requirements:

- **Harris corner detection**;
- **Shi-Tomasi corner detector ("good feature to track")**;
- **SIFT (Scale-Invariant Feature Transform)**;
- SURF (Speeded-Up Robust Features);
- **FAST (Features from Accelerated Segment Test)**;
- BRIEF (Binary Robust Independent Elementary Features);
- ORB (Oriented FAST and Rotated BRIEF);

(just to mention the ones that are implemented in OpenCV, https://docs.opencv.org/master/db/d27/tutorial_py_table_of_contents_feature2d.html)

Interest Point Detection – Harris Corners

Detecting a **corner** point means that there must be a **strong gradient both in the horizontal and vertical directions**.

Harris corner detection is based on the auto-correlation of the partial derivatives (I_x, I_y) of the image I.

$$\mathbf{A}(x,y) = \begin{bmatrix} \sum_{i,j} I_x(x+i,y+j)^2 & \sum_{i,j} I_x(x+i,y+j)I_y(x+i,y+j) \\ \sum_{i,j} I_x(x+i,y+j)I_y(x+i,y+j) & \sum_{i,j} I_y(x+i,y+j)^2 \end{bmatrix}$$

The principal curvature — the strength of the image gradients — is expressed in the eigenvalues λ_1, λ_2 of $\mathbf{A}$.

- If both eigenvalues are **small**, there is no significant curvature, and the region is **uniform**.
- If **one** eigenvalue is **large**, then we are examining an **edge region**.
- If **both** eigenvalues are **large**, we have found a **corner** region.

Interest Point Detection – Harris Corners (cont.)

- The computation of the eigenvalues is costly!
- Harris corner detector uses a scoring function ρ that can be expressed with the trace of $\mathbf{A}$ rather than the eigenvalues themselves:

$$\rho(x,y) = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$$
$$= \det[\mathbf{A}(x,y)] - k.\mathit{trace}[\mathbf{A}(x,y)]^2$$

- Non-maximum suppression is performed in an 8-connected neighborhood
- Candidates with a response lower than a predefined percentage of the maximum response are removed

Interest Point Detection – Shi-Tomasi

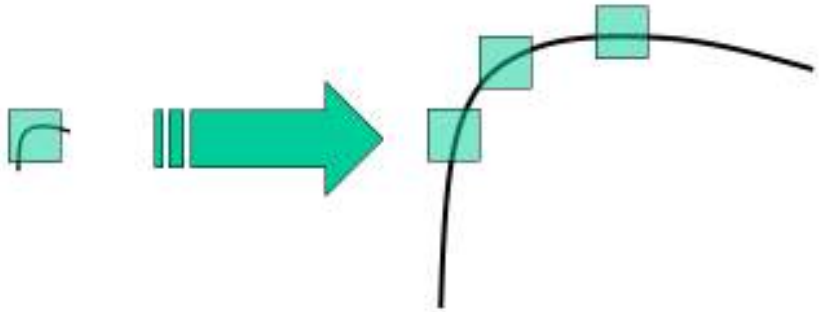
- Shi-Tomasi method ("good features to track") is similar to Harris corners.
- However, rely on a score that is more costly to compute, because it requires access to the eigenvalues.
- This score requires that both eigenvalues be larger than a percentage τ of the maximum response:

$$\rho(x, y) = \min(\lambda_1, \lambda_2)$$

$$\rho(x, y) > \tau . \max_{x,y}(\lambda_1, \lambda_2)$$

Interest Point Detection – SIFT

- Harris corners are **not invariant** against changes in **scale**, so they do not perform well when the camera is translated along the viewing direction.



- **SIFT** (Scale-Invariant Feature Transform) addresses this problem by detecting local extrema of the image filtered with Differences of Gaussian (DOG), a method that operates in the scale space obtained from an image pyramid.

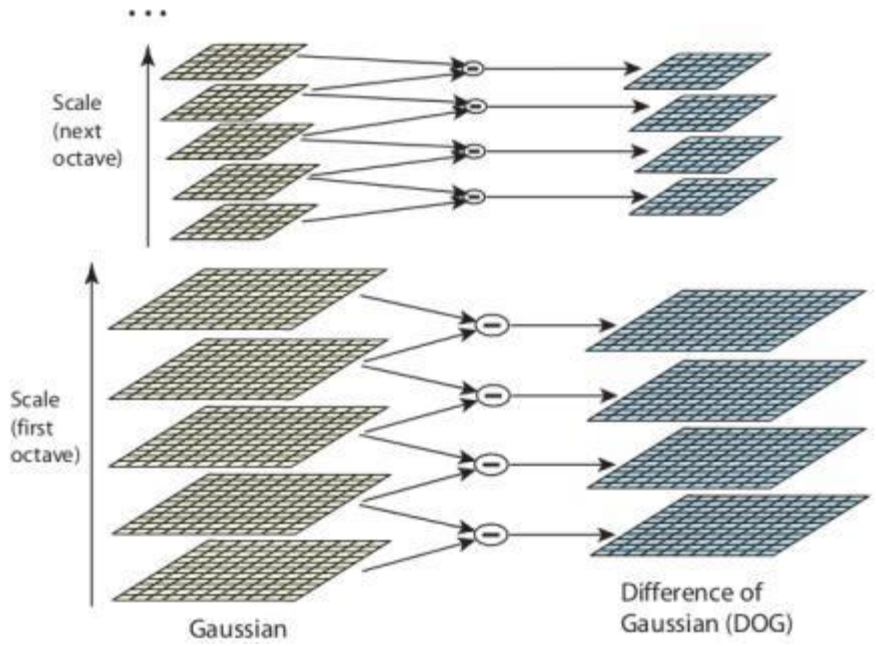
Interest Point Detection – SIFT (cont.)

- An image pyramid $I_d, d \in [0 \ldots N]$ is built from the convolution of image I with Differences of Gaussian filters G at different scales given by σ and k $(k > 1)$ – σ acts as a scale parameter.
- This image pyramid is computed as a sequence of image convolutions and differencing:

$$G_\sigma(x, y) = \frac{e^{\frac{-x^2+y^2}{2\sigma^2}}}{2\pi\sigma^2}$$

$$I_d = I * G_{k^{d+1}\sigma} - I * G_{k^d\sigma}$$

  - gaussian kernel with low σ gives high value for small corner
  - gaussian kernel with high σ fits well for larger corner

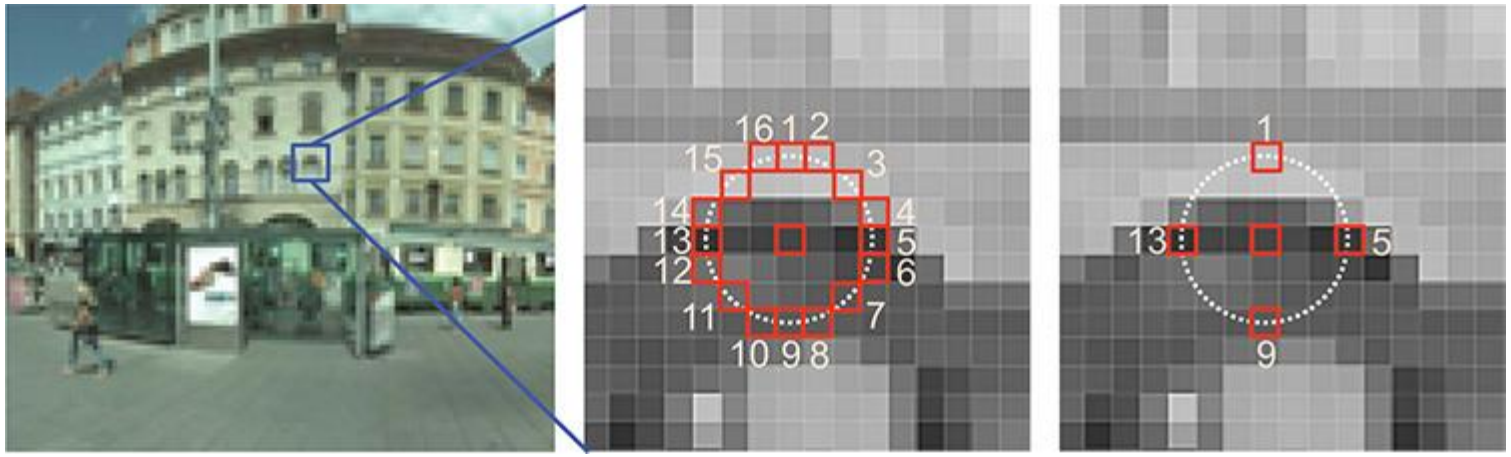


Interest Point Detection – SIFT (cont.)

- Once the pyramid is computed, images are searched for local extrema over scale and space which gives us a list of (x, y, σ) values which means there is a potential **keypoint** at (x, y) at σ scale.
- One pixel in an image is compared with its 8 neighbors as well as 9 pixels in next scale and 9 pixels in previous scales.
- Once a potential keypoint is found, its exact location is determined by quadratic minimization (Taylor series).
- Candidates with weak contrast or edges are suppressed.
- Orientation is assigned to each keypoint to achieve invariance to image rotation.
- The keypoint descriptor is created.
- However, the Gaussian convolution makes SIFT computation expensive

Interest Point Detection – FAST

- **FAST** (Features from Accelerated Segment Test) detector is highly suitable for real-time video processing applications (for example, mobile AR).
- It is used a discretized circle centered on the candidate point and is classified as a corner, if there exists a contiguous arc of pixels with sufficient contrast to the center pixel, which covers up to three quarters of a circle.
- A tradeoff must be made when choosing *N*, the number of contiguous pixels (arc), and the contrast threshold value d.



Interest Point Detection – FAST12 Algorithm

- FAST12 uses a circle consisting of 16 pixels $s_i, i \in [1 \dots 16]$.
- A corner is found, if 12 contiguous pixels are all lighter or darker than the center by a threshold d.
- Rapid rejection can be determined by first testing s_1(top), s_9 (bottom), s_{13} (left), and s_5 (right).
- Because FAST detection itself does not determine the strength of the interest point, the score ρ is used, which considers the pixels darker than the center, S^D, and the pixels lighter than the center, S^L:

$$S^D = \{s_i(x,y) | s_i(x,y) \leq I(x,y) - d\}$$

$$S^L = \{s_i(x,y) | s_i(x,y) \geq I(x,y) + d\}$$

$$\rho(x,y) = max\left(\sum_{s_i \in S^D} |I(x,y) - s_i| - d\,,\ \sum_{s_i \in S^L} |s_i - I(x,y)| - d\right)$$

Descriptor Creation

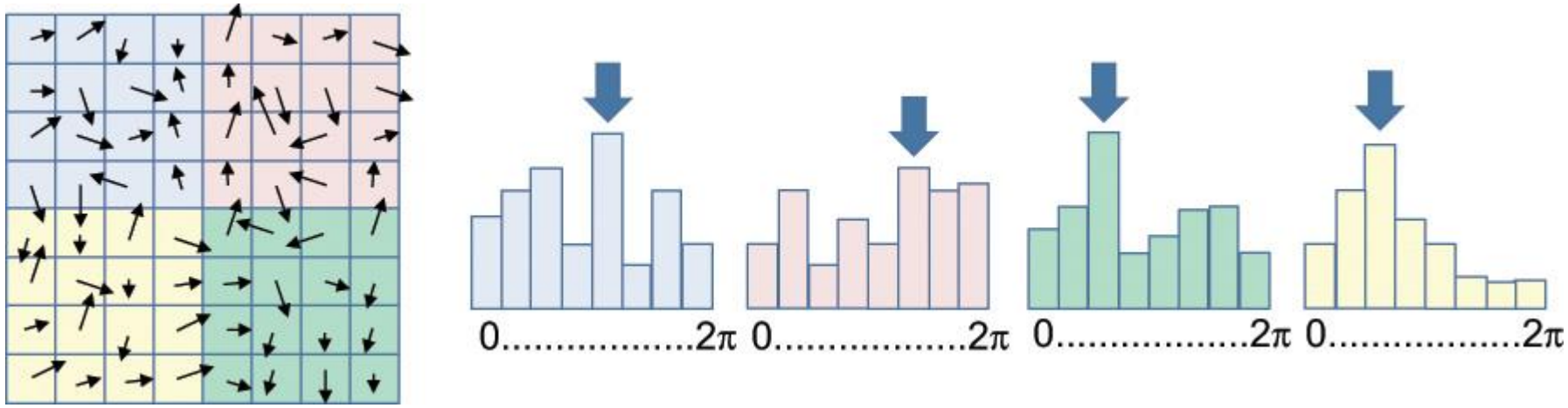
- After an interest point has been selected, **a descriptor must be computed** — a **data structure** suitable for matching the interest point to the tracking model or other images.
- Ideally, the descriptor should be **unique for every point** on the tracking model, but **identical for arbitrary viewpoints and lighting conditions**.
- A good descriptor captures the **texture of the local neighborhood**, while being largely **invariant to changes in illumination, scale, rotation, and affine transformations**.

Descriptor Creation – SIFT Example

- A patch of $N \times N$ neighborhood pixels around the keypoint is taken.
- The orientation θ and magnitude g of the gradient from the patch are computed.
- The image patch is subdivided into a $K_x \times K_y$ grid, and a separate weighted orientation histogram with K_b bins is computed.
- The peak of the histogram is chosen as the descriptor orientation.
- The descriptor is taken to be the feature vector resulting from the concatenation of the $K_x \times K_y \times K_b$ bins.
- The vector is normalized to minimize the influence of illumination variation.

Descriptor Creation – SIFT Example (cont.)

The standard SIFT descriptor has $N = 16, K_x = K_y = 4$, $K_b = 8$ and the feature vector has $4 \times 4 \times 8 = 128$ dimensions.



Example with $N = 8, K_x = K_y = 2$, $K_b = 8$ and descriptor dimension of $2 \times 2 \times 8 = 32$.

Descriptor Matching

- Given the descriptors of **interest points** detected in the **image**, we must try to find the **best match** from **interest points** in the tracking **model**.
- The **simplest matching score** of two descriptors is their **Euclidean distance**.
- For a given descriptor from the image, the descriptor in the tracking model with the **smallest distance** represents the **best match**.
- This match should be **unique**; thus, if the ratio of the smallest distance and the second smallest distance is larger than some threshold (typically set at 80%), the interest point is discarded.

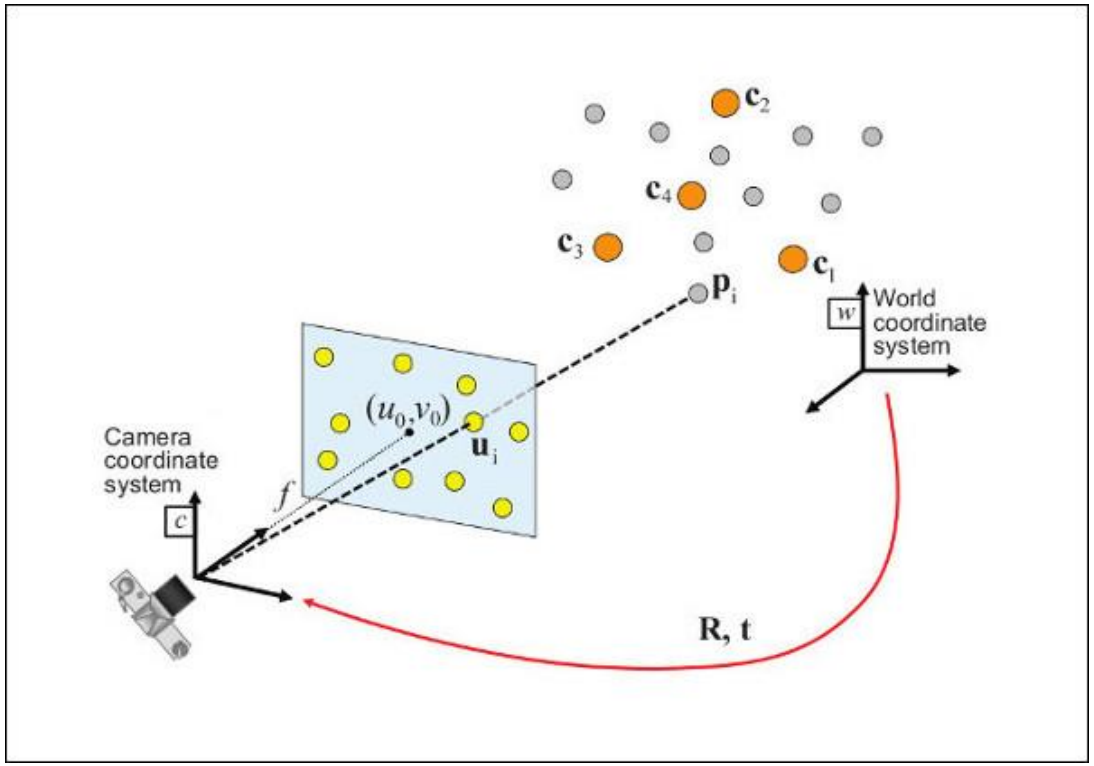
Descriptor Matching

- If the detected keypoints are too many for an exhaustive search, **heuristic search structures** should be used.
- A typical approach is to rely on hierarchical search structures such as **k-d trees**.
- Any **approximate search structure** can lead to **incorrect matching** results, producing outliers that affect the pose computation.
- Robust and efficient **outlier removal** can be used, for example, use orientation information, such as SIFT.
  - It can be checked if all matched interest points in the image have a consistent orientation with respect to the target model.

Perspective-n-Point Pose

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \vec{0} & 1 \end{bmatrix} \begin{bmatrix} q_x \\ q_y \\ q_z \\ 1 \end{bmatrix}$$

- We know the pixel $[u\ v]^T$ in the image plane, the internal camera matrix $\mathbf{K}$ and the 3D point $\left[q_x\ q_y q_z\right]^T$ in the world coordinate system (**WCS**).
- We must estimate the transformation $[\mathbf{R}|\mathbf{t}]$ between the **WCS** and camera coordinate system (**CCS**) that projects the 3D point into the know pixel.



Perspective-n-Point Pose

- The problem of recovering the 6DOF pose of a calibrated camera from N 2D–3D point correspondences is known as the Perspective-n-Point (PnP) problem, where n denotes the number of correspondences.
- Given that every correspondence provides two constraints, a minimum of three 2D–3D correspondences is necessary for six constraints matching the 6DOF.
- With only **three points**, this **P3P** algorithm delivers up to four ambiguous solutions.
- A fourth point correspondence is required to determine the unique solution.

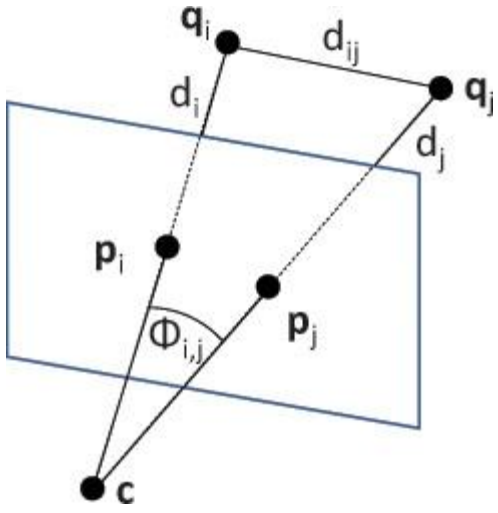
Perspective-n-Point Pose – P3P algorithm

1. Select any two points $\mathbf{q}_i$, $\mathbf{q}_j$ from the given three points.
2. Create a triangle $\mathbf{cq}_i\mathbf{q}_j$ from the two selected points and the (unknown) camera center $\mathbf{c}$.
3. Determine $d_{ij} = |\mathbf{q}_j - \mathbf{q}_i|$ and $\cos\Phi_{i,j}$:

$$\cos\Phi_{i,j} = \underline{N}(\mathbf{q}_i - \mathbf{c}).\underline{N}(\mathbf{q}_j - \mathbf{c})$$

4. It is possible to set up equations with the unknowns $d_i = |\mathbf{q}_i - \mathbf{c}|$ and $d_j = |\mathbf{q}_j - \mathbf{c}|$

$$d_{ij}^2 = d_i^2 + d_j^2 - 2d_i d_j \cos\Phi_{i,j}$$



Perspective-n-Point Pose – P3P algorithm (cont.)

5. Every pair (3) yields one polynomial equation and together they form a set of three equations in d_1, d_2, d_3. This set can be solved in closed form with at most four solutions.

   In practice, a unique solution can be easily obtained with four points: The P3P problem is solved for each subset consisting of three points out of the four points, and the common solution is retained.

6. Compute the position of $\mathbf{q}_i$ in camera coordinates, $\mathbf{q}_i^C$:

$$\mathbf{q}_i^C = c + d_i . \underline{N}(\mathbf{p}_i - \mathbf{c})$$

7. Compute the pose $[\mathbf{R}|\mathbf{t}]$ by aligning $\mathbf{q}_i$ and $\mathbf{q}_i^C$, using absolute orientation (see the "Absolute Orientation" section).
8. Perform nonlinear refinement of the pose (see the "Pose Refinement" section).

Robust Pose Estimation

- A larger number of data points correspondences is generally desirable, as it improves the numerical optimization.
- However, increases the probability of being present outliers (wrong 2D-3D point correspondences) which will contaminate the result.
- It is preferred to find a good initialization from a data set, even if it contains outliers, and then converge the results to obtain an accurate solution.
- The goal is to select an all-inlier subset from a contaminated set of data points.

RANdom SAmpling Consensus (RANSAC)

1. Select randomly the minimum number of points required to determine the model parameters.
2. Solve for the parameters of the model.
3. Determine how many points from the set of all points fit with a predefined tolerance ε (inliers).
4. If the fraction of the number of inliers over the total number points in the set exceeds a predefined threshold τ, re-estimate the model parameters using all the identified inliers and terminate.
5. Otherwise, repeat steps 1 through 4 (maximum of N times).

Based on [5]

Robust Pose Estimation – RANSAC

- In step 2, for camera poses estimation, P3P is often used, since it requires only three 2D–3D point correspondences.
- In step 3, for the remaining point correspondences, compute the residual error, assuming the camera pose computed from the three chosen points.
- A data point with a residual smaller than a threshold counts as an inlier.
- RANSAC terminates if either enough inliers are found, or a maximum number of iterations is reached.

Bibliography

Based on:

[1] - Augmented Reality: Principles and Practice, 1st Edition by Dieter Schmalstieg and Tobias Hollerer, June 2016, Pearson Education.

[2] - ArUco: a minimal library for Augmented Reality applications based on OpenCV - https://www.uco.es/investiga/grupos/ava/node/26

[3] - Multiple View Geometry in Computer Vision, 2nd edition, Richard Hartley, Andrew Zisserman, 2004, Cambridge University Press.

[4] - OpenCV (Open-Source Computer Vision Library), https://opencv.org/

[5] - Overview of the RANSAC Algorithm, Konstantinos G. Derpanis, May 13, 2010.

Some Algebra

Cross product

Of particular interest are 3 × 3 skew-symmetric matrices.

If $\mathbf{a} = [a_1 \quad a_2 \quad a_3]^T$ and $\mathbf{b} = [b_1 \quad b_2 \quad b_3]^T$ are two 3-vectors, the cross product $\mathbf{a} \times \mathbf{b}$ is the vector that is perpendicular (orthogonal) to both $\mathbf{a}$ and $\mathbf{b}$, with a direction given by the right-hand rule and a magnitude equal to the area of the parallelogram that the vectors span.

The cross product is defined by the formula

$$\mathbf{a} \times \mathbf{b} = [a_2 b_3 - a_3 b_2 \quad a_3 b_1 - a_1 b_3 \quad a_1 b_2 - a_2 b_1]^T$$

If one defines a corresponding skew-symmetric matrix from $\mathbf{a}$ as follows:

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

The cross product can be given by

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b} = (\mathbf{a}^T [\mathbf{b}]_\times)^T$$

Some Definitions

- **Auto-correlation** describes how similar an image $I(x, y)$ is to a shifted version of itself considering an image patch $I\left(x \pm W_x/2, \mathrm{y} \pm W_y/2\right)$ of dimension $W_x \times W_y$.
- **Trace of a $n \times n$ square matrix A**, denoted $Tr(A)$, is defined to be the sum of elements on the main diagonal (from the upper left to the lower right) of A, i.e., $Tr(A) \equiv \sum_{i=1}^{n} a_{ii}$.