Homogeneous representation of 2D lines and points

- A 2D line is defined by $ax + by + c = 0$ i.e. a parametrization $\mathbf{l} = (a, b, c)$.

Homogeneous representation of 2D lines and points

- A 2D line is defined by $ax + by + c = 0$ i.e. a parametrization $\mathbf{l} = (a, b, c)$.
- However, $kax + kby + kc = 0$ corresponds to the same line, thus $\mathbf{l} = (ka, kb, kc), \forall k \in \mathbb{R} \setminus \{0\}$

Homogeneous representation of 2D lines and points

- A 2D line is defined by $ax + by + c = 0$ i.e. a parametrization $\mathbf{l} = (a, b, c)$.
- However, $kax + kby + kc = 0$ corresponds to the same line, thus $\mathbf{l} = (ka, kb, kc), \forall k \in \mathbb{R} \setminus \{0\}$
- A 2D point (x, y) lies on a line (a, b, c) if $ax + by + c = 0$.

Homogeneous representation of 2D lines and points

- A 2D line is defined by $ax + by + c = 0$ i.e. a parametrization $\mathbf{l} = (a, b, c)$.
- However, $kax + kby + kc = 0$ corresponds to the same line, thus $\mathbf{l} = (ka, kb, kc), \forall k \in \mathbb{R} \setminus \{0\}$
- A 2D point (x, y) lies on a line (a, b, c) if $ax + by + c = 0$.
- This may be expressed as $(x, y, 1)^T \cdot (a, b, c) = (x, y, 1)^T \cdot \mathbf{l} = 0$.

Homogeneous representation of 2D lines and points

- A 2D line is defined by $ax + by + c = 0$ i.e. a parametrization $\mathbf{l} = (a, b, c)$.
- However, $kax + kby + kc = 0$ corresponds to the same line, thus $\mathbf{l} = (ka, kb, kc), \forall k \in \mathbb{R} \setminus \{0\}$
- A 2D point (x, y) lies on a line (a, b, c) if $ax + by + c = 0$.
- This may be expressed as $(x, y, 1)^T \cdot (a, b, c) = (x, y, 1)^T \cdot \mathbf{l} = 0$.
- $\forall k \in \mathbb{R} \setminus \{0\}, (kx, ky, k)^T \cdot \mathbf{l} = 0$ if and only if $(x, y, 1)^T \cdot \mathbf{l} = 0$.

Homogeneous representation of 2D lines and points

- A 2D line is defined by $ax + by + c = 0$ i.e. a parametrization $\mathbf{l} = (a, b, c)$.
- However, $kax + kby + kc = 0$ corresponds to the same line, thus $\mathbf{l} = (ka, kb, kc), \forall k \in \mathbb{R} \setminus \{0\}$
- A 2D point (x, y) lies on a line (a, b, c) if $ax + by + c = 0$.
- This may be expressed as $(x, y, 1)^T \cdot (a, b, c) = (x, y, 1)^T \cdot \mathbf{l} = 0$.
- $\forall k \in \mathbb{R} \setminus \{0\}, (kx, ky, k)^T \cdot \mathbf{l} = 0$ if and only if $(x, y, 1)^T \cdot \mathbf{l} = 0$.
- $\forall k \in \mathbb{R} \setminus \{0\}$, we denote thus (kx, ky, k) as the homogeneous representation of the 2D point (x, y).

Homogeneous representation of 2D lines and points

- A 2D line is defined by $ax + by + c = 0$ i.e. a parametrization $\mathbf{l} = (a, b, c)$.
- However, $kax + kby + kc = 0$ corresponds to the same line, thus $\mathbf{l} = (ka, kb, kc), \forall k \in \mathbb{R} \setminus \{0\}$
- A 2D point (x, y) lies on a line (a, b, c) if $ax + by + c = 0$.
- This may be expressed as $(x, y, 1)^T \cdot (a, b, c) = (x, y, 1)^T \cdot \mathbf{l} = 0$.
- $\forall k \in \mathbb{R} \setminus \{0\}, (kx, ky, k)^T \cdot \mathbf{l} = 0$ if and only if $(x, y, 1)^T \cdot \mathbf{l} = 0$.
- $\forall k \in \mathbb{R} \setminus \{0\}$, we denote thus (kx, ky, k) as the homogeneous representation of the 2D point (x, y).
- An arbitrary homogeneous $\mathbf{x} = (x_1, x_2, x_3)$ corresponds to the 2D point $(x_1/x_3, x_2/x_3)$.

Homogeneous representation of 2D lines and points

- A 2D line is defined by $ax + by + c = 0$ i.e. a parametrization $\mathbf{l} = (a, b, c)$.
- However, $kax + kby + kc = 0$ corresponds to the same line, thus $\mathbf{l} = (ka, kb, kc), \forall k \in \mathbb{R} \setminus \{0\}$
- A 2D point (x, y) lies on a line (a, b, c) if $ax + by + c = 0$.
- This may be expressed as $(x, y, 1)^T \cdot (a, b, c) = (x, y, 1)^T \cdot \mathbf{l} = 0$.
- $\forall k \in \mathbb{R} \setminus \{0\}, (kx, ky, k)^T \cdot \mathbf{l} = 0$ if and only if $(x, y, 1)^T \cdot \mathbf{l} = 0$.
- $\forall k \in \mathbb{R} \setminus \{0\}$, we denote thus (kx, ky, k) as the homogeneous representation of the 2D point (x, y).
- An arbitrary homogeneous $\mathbf{x} = (x_1, x_2, x_3)$ corresponds to the 2D point $(x_1/x_3, x_2/x_3)$.
- Result : the point $\mathbf{x}$ lies on the line $\mathbf{l}$ if and only if $\mathbf{x}^T\mathbf{l} = 0$.

Homogeneous representation of 2D lines and points

- A 2D line is defined by $ax + by + c = 0$ i.e. a parametrization $\mathbf{l} = (a, b, c)$.
- However, $kax + kby + kc = 0$ corresponds to the same line, thus $\mathbf{l} = (ka, kb, kc), \forall k \in \mathbb{R} \setminus \{0\}$
- A 2D point (x, y) lies on a line (a, b, c) if $ax + by + c = 0$.
- This may be expressed as $(x, y, 1)^T \cdot (a, b, c) = (x, y, 1)^T \cdot \mathbf{l} = 0$.
- $\forall k \in \mathbb{R} \setminus \{0\}, (kx, ky, k)^T \cdot \mathbf{l} = 0$ if and only if $(x, y, 1)^T \cdot \mathbf{l} = 0$.
- $\forall k \in \mathbb{R} \setminus \{0\}$, we denote thus (kx, ky, k) as the homogeneous representation of the 2D point (x, y).
- An arbitrary homogeneous $\mathbf{x} = (x_1, x_2, x_3)$ corresponds to the 2D point $(x_1/x_3, x_2/x_3)$.
- Result : the point $\mathbf{x}$ lies on the line $\mathbf{l}$ if and only if $\mathbf{x}^T\mathbf{l} = 0$.
- Result : the intersection of two lines $\mathbf{l}$ and $\mathbf{l}'$ is the point $\mathbf{x} = \mathbf{l} \times \mathbf{l}'$.

Homogeneous representation of 2D lines and points

- A 2D line is defined by $ax + by + c = 0$ i.e. a parametrization $\mathbf{l} = (a, b, c)$.
- However, $kax + kby + kc = 0$ corresponds to the same line, thus $\mathbf{l} = (ka, kb, kc), \forall k \in \mathbb{R} \setminus \{0\}$
- A 2D point (x, y) lies on a line (a, b, c) if $ax + by + c = 0$.
- This may be expressed as $(x, y, 1)^T \cdot (a, b, c) = (x, y, 1)^T \cdot \mathbf{l} = 0$.
- $\forall k \in \mathbb{R} \setminus \{0\}, (kx, ky, k)^T \cdot \mathbf{l} = 0$ if and only if $(x, y, 1)^T \cdot \mathbf{l} = 0$.
- $\forall k \in \mathbb{R} \setminus \{0\}$, we denote thus (kx, ky, k) as the homogeneous representation of the 2D point (x, y).
- An arbitrary homogeneous $\mathbf{x} = (x_1, x_2, x_3)$ corresponds to the 2D point $(x_1/x_3, x_2/x_3)$.
- Result : the point $\mathbf{x}$ lies on the line $\mathbf{l}$ if and only if $\mathbf{x}^T\mathbf{l} = 0$.
- Result : the intersection of two lines $\mathbf{l}$ and $\mathbf{l}'$ is the point $\mathbf{x} = \mathbf{l} \times \mathbf{l}'$.
- Result : the line through two points $\mathbf{x}$ and $\mathbf{x}'$ is $\mathbf{l} = \mathbf{x} \times \mathbf{x}'$.

Some quick vector operations

$$\mathbf{x} \times \mathbf{y} = \mathbf{x}_\times \cdot \mathbf{y} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{vmatrix} = \begin{pmatrix} x_2 y_3 - x_3 y_2 \\ x_3 y_1 - x_1 y_3 \\ x_1 y_2 - y_1 x_2 \end{pmatrix}$$

$$\mathbf{x}_\times = \begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix}$$

Mixed product : $\mathbf{x}^T(\mathbf{y} \times \mathbf{z}) = |\mathbf{x} \;\; \mathbf{y} \;\; \mathbf{z}|$ (the volume of the parallelepiped defined by the three vectors)

Singular value decomposition

Theorem (SVD) :

Let $\mathbf{A}$ be an $m \times n$ matrix. $\mathbf{A}$ may be expressed as :

$$\mathbf{A} = \mathbf{U\Sigma V}^T = \sum_{i=1}^{\min(m,n)} \sigma_i U_i V_i^T$$

where $\mathbf{\Sigma}$ is a $m \times n$ diagonal matrix with $\sigma_i = \mathbf{\Sigma}_{ii} \geq 0$, and $\mathbf{U}$ $(m \times m)$ and $\mathbf{V}$ $(n \times n)$ are composed of orthornormal columns

Singular value decomposition

Theorem (SVD) :

Let $\mathbf{A}$ be an $m \times n$ matrix. $\mathbf{A}$ may be expressed as :

$$\mathbf{A} = \mathbf{U\Sigma V}^T = \sum_{i=1}^{\min(m,n)} \sigma_i U_i V_i^T$$

where $\mathbf{\Sigma}$ is a $m \times n$ diagonal matrix with $\sigma_i = \mathbf{\Sigma}_{ii} \geq 0$, and $\mathbf{U}$ $(m \times m)$ and $\mathbf{V}$ $(n \times n)$ are composed of orthornormal columns

- The rank of $\mathbf{A}$ is the number of $\sigma_i > 0$

Singular value decomposition

Theorem (SVD) :

Let $\mathbf{A}$ be an $m \times n$ matrix. $\mathbf{A}$ may be expressed as :

$$\mathbf{A} = \mathbf{U\Sigma V}^T = \sum_{i=1}^{\min(m,n)} \sigma_i U_i V_i^T$$

where $\mathbf{\Sigma}$ is a $m \times n$ diagonal matrix with $\sigma_i = \mathbf{\Sigma}_{ii} \geq 0$, and $\mathbf{U}$ $(m \times m)$ and $\mathbf{V}$ $(n \times n)$ are composed of orthornormal columns

- The rank of $\mathbf{A}$ is the number of $\sigma_i > 0$
- An orthonormal basis for the null space of $\mathbf{A}$ is composed of V_i for indices i such that $\sigma_i = 0$

Singular value decomposition

Theorem (SVD) :

Let $\mathbf{A}$ be an $m \times n$ matrix. $\mathbf{A}$ may be expressed as :

$$\mathbf{A} = \mathbf{U\Sigma V}^T = \sum_{i=1}^{\min(m,n)} \sigma_i U_i V_i^T$$

where $\mathbf{\Sigma}$ is a $m \times n$ diagonal matrix with $\sigma_i = \mathbf{\Sigma}_{ii} \geq 0$, and $\mathbf{U}$ $(m \times m)$ and $\mathbf{V}$ $(n \times n)$ are composed of orthornormal columns

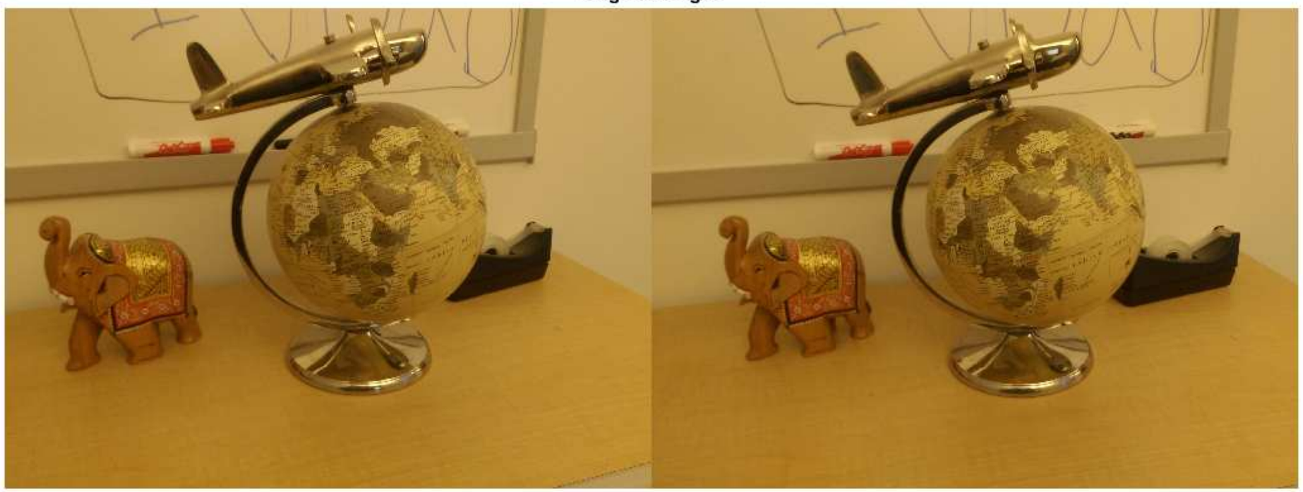
- The rank of $\mathbf{A}$ is the number of $\sigma_i > 0$
- An orthonormal basis for the null space of $\mathbf{A}$ is composed of V_i for indices i such that $\sigma_i = 0$
- By convention, the σ_i are aligned in descending order by the decomposition algorithms.

Outline

- The 3D representation of points
- The pinhole camera model
- Applying a coordinate transformation
- Homogeneous representations and algebraic operations
- The fundamental matrix
- The essential matrix
- Rectification

Why is this part "fundamental" ? (cheap joke)

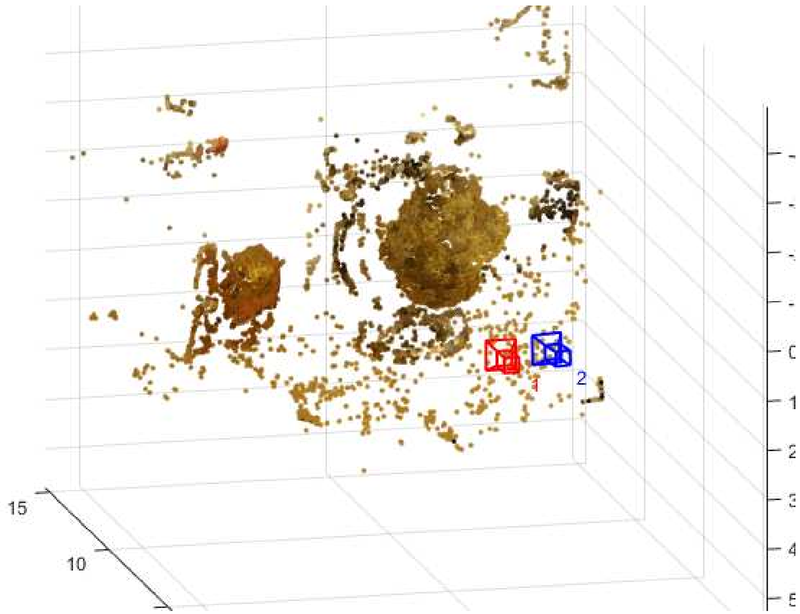
What we can get from two views :



Why is this part “fundamental” ? (cheap joke)

What we can get from two views :

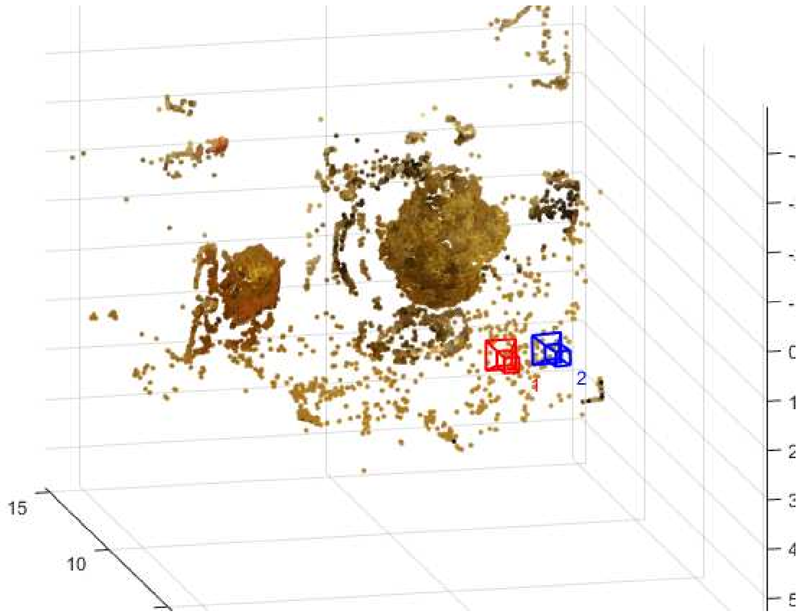
- Sparse 3D reconstruction



Why is this part "fundamental" ? (cheap joke)

What we can get from two views :

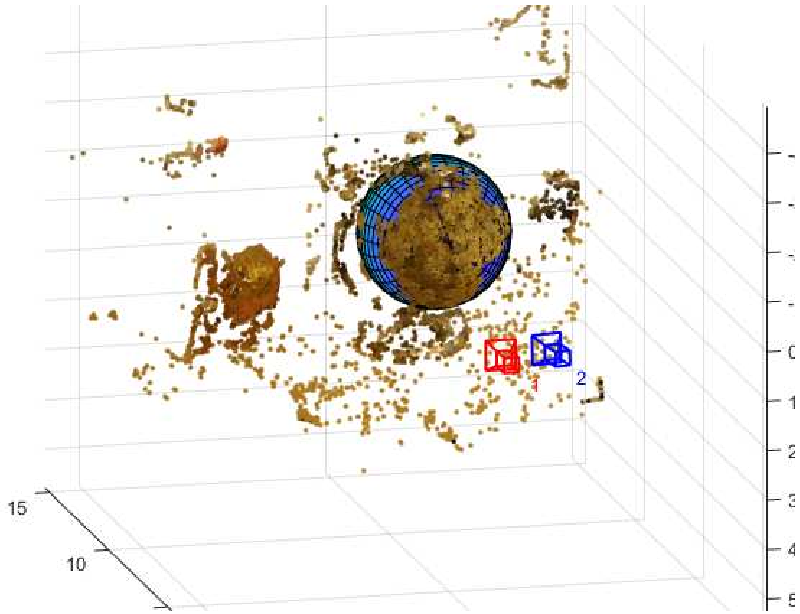
- Sparse 3D reconstruction
- Relative camera pose estimation



Why is this part "fundamental" ? (cheap joke)

What we can get from two views :

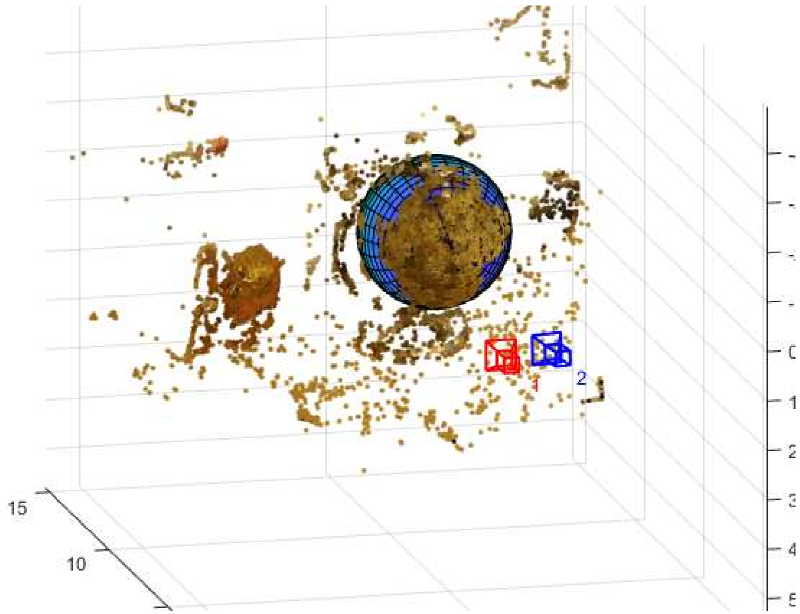
- Sparse 3D reconstruction
- Relative camera pose estimation
- Parametric surface fitting



Why is this part "fundamental" ? (cheap joke)

What we can get from two views :

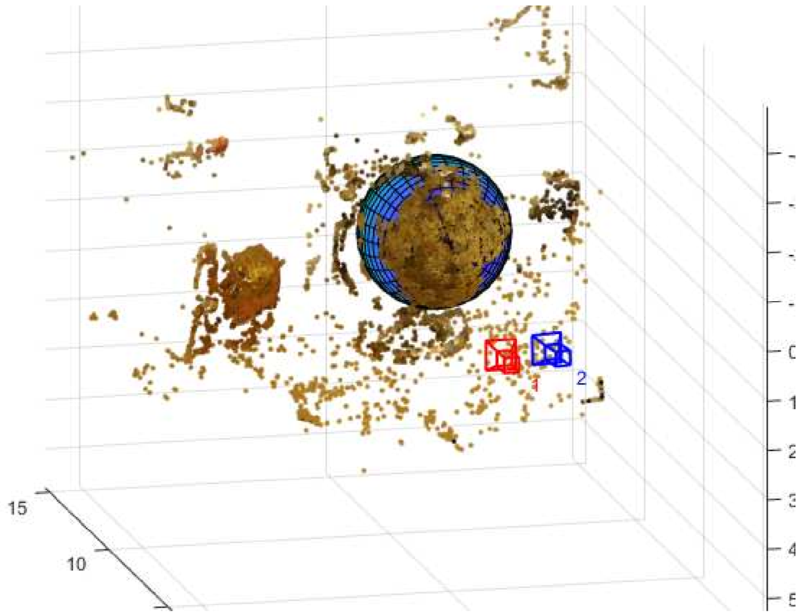
- Sparse 3D reconstruction
- Relative camera pose estimation
- Parametric surface fitting
- Dense 3D reconstruction (more complex work required for this)



Why is this part "fundamental" ? (cheap joke)

What we can get from two views :

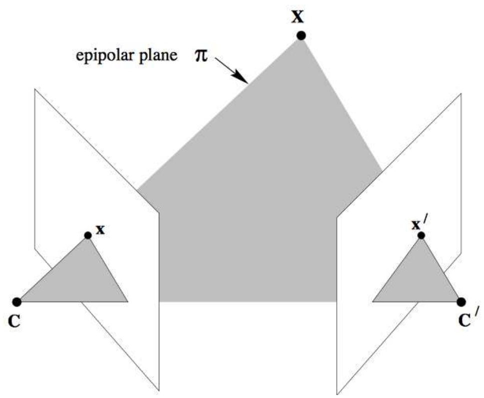
- Sparse 3D reconstruction
- Relative camera pose estimation
- Parametric surface fitting
- Dense 3D reconstruction (more complex work required for this)
- ... but also many multi-view algorithms extend nicely from two-view analysis



The anatomy of two views

Some important observations :

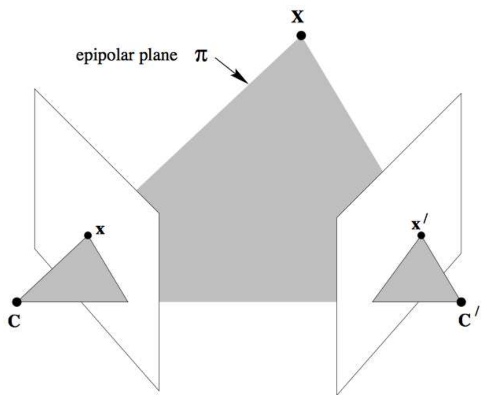
- the pixel projection is along the ray defined by the 3D point and the camera center (i.e. as for **x**, **X** and **C**)



The anatomy of two views

Some important observations :

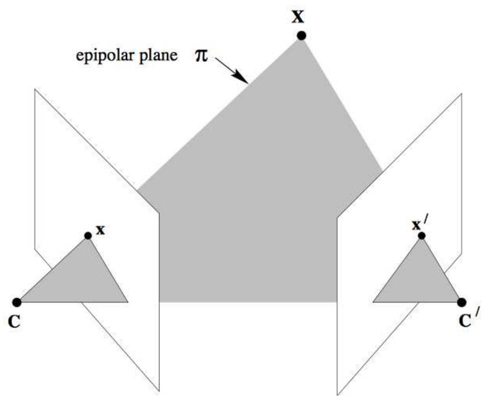
- the pixel projection is along the ray defined by the 3D point and the camera center (i.e. as for **x**, **X** and **C**)
- conversely, if **x** and **x**′ do correspond to the same 3D point, the two rays intersect



The anatomy of two views

Some important observations :

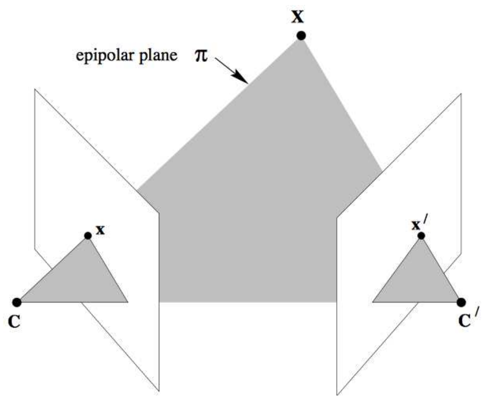
- the pixel projection is along the ray defined by the 3D point and the camera center (i.e. as for $\mathbf{x}$, $\mathbf{X}$ and $\mathbf{C}$)
- conversely, if $\mathbf{x}$ and $\mathbf{x}'$ do correspond to the same 3D point, the two rays intersect
- the two rays define a plane π denoted as *epipolar plane*



The anatomy of two views

Some important observations :

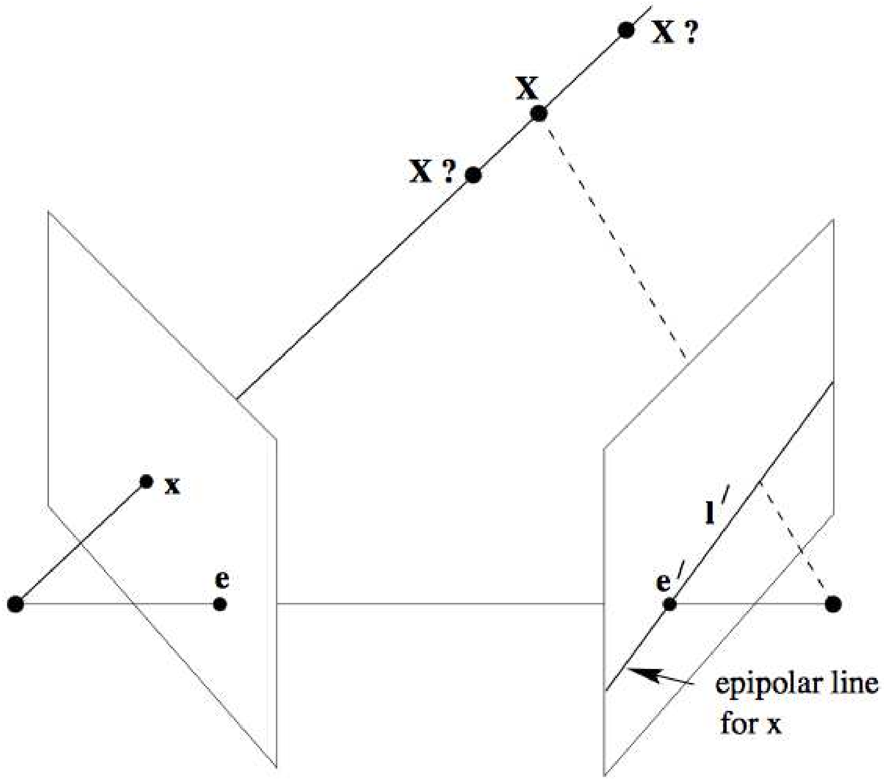
- the pixel projection is along the ray defined by the 3D point and the camera center (i.e. as for $\mathbf{x}$, $\mathbf{X}$ and $\mathbf{C}$)
- conversely, if $\mathbf{x}$ and $\mathbf{x}'$ do correspond to the same 3D point, the two rays intersect
- the two rays define a plane π denoted as *epipolar plane*
- the epipolar plane also contains the ray defined by the camera centers



The anatomy of two views

From the projection in the two views we have :

$$\lambda \mathbf{x} = \mathbf{K}\mathbf{X} \quad \lambda' \mathbf{x}' = \mathbf{K}'(\mathbf{R}\mathbf{X} + \mathbf{t})$$



The anatomy of two views

From the projection in the two views we have :

$$\lambda \mathbf{x} = \mathbf{K}\mathbf{X} \quad \lambda' \mathbf{x}' = \mathbf{K}'(\mathbf{R}\mathbf{X} + \mathbf{t})$$

By eliminating $\mathbf{X}$ we get :

$$\mathbf{X} = \lambda \mathbf{K}^{-1}\mathbf{x} \quad \lambda' \mathbf{x}' = \mathbf{K}'(\lambda \mathbf{R}\mathbf{K}^{-1}\mathbf{x} + \mathbf{t})$$

The anatomy of two views

From the projection in the two views we have :

$$\lambda \mathbf{x} = \mathbf{K}\mathbf{X} \quad \lambda' \mathbf{x}' = \mathbf{K}'(\mathbf{R}\mathbf{X} + \mathbf{t})$$

By eliminating $\mathbf{X}$ we get :

$$\mathbf{X} = \lambda \mathbf{K}^{-1}\mathbf{x} \quad \lambda' \mathbf{x}' = \mathbf{K}'(\lambda \mathbf{R}\mathbf{K}^{-1}\mathbf{x} + \mathbf{t})$$

$$\lambda' \mathbf{K}'^{-1}\mathbf{x}' = \lambda \mathbf{R}\mathbf{K}^{-1}\mathbf{x} + \mathbf{t}$$

The anatomy of two views

From the projection in the two views we have :

$$\lambda \mathbf{x} = \mathbf{K}\mathbf{X} \quad \lambda' \mathbf{x}' = \mathbf{K}'(\mathbf{R}\mathbf{X} + \mathbf{t})$$

By eliminating $\mathbf{X}$ we get :

$$\mathbf{X} = \lambda \mathbf{K}^{-1}\mathbf{x} \quad \lambda' \mathbf{x}' = \mathbf{K}'(\lambda \mathbf{R}\mathbf{K}^{-1}\mathbf{x} + \mathbf{t})$$

$$\lambda' \mathbf{K}'^{-1}\mathbf{x}' = \lambda \mathbf{R}\mathbf{K}^{-1}\mathbf{x} + \mathbf{t}$$

We eliminate the sum by applying a cross product with $\mathbf{t}$:

$$\lambda' \mathbf{t}_\times \mathbf{K}'^{-1}\mathbf{x}' = \lambda \mathbf{t}_\times \mathbf{R}\mathbf{K}^{-1}\mathbf{x}$$

The anatomy of two views

From the projection in the two views we have :

$$\lambda \mathbf{x} = \mathbf{K}\mathbf{X} \quad \lambda' \mathbf{x}' = \mathbf{K}'(\mathbf{R}\mathbf{X} + \mathbf{t})$$

By eliminating $\mathbf{X}$ we get :

$$\mathbf{X} = \lambda \mathbf{K}^{-1}\mathbf{x} \quad \lambda' \mathbf{x}' = \mathbf{K}'(\lambda \mathbf{R}\mathbf{K}^{-1}\mathbf{x} + \mathbf{t})$$

$$\lambda' \mathbf{K}'^{-1}\mathbf{x}' = \lambda \mathbf{R}\mathbf{K}^{-1}\mathbf{x} + \mathbf{t}$$

We eliminate the sum by applying a cross product with $\mathbf{t}$:

$$\lambda' \mathbf{t}_{\times} \mathbf{K}'^{-1}\mathbf{x}' = \lambda \mathbf{t}_{\times} \mathbf{R}\mathbf{K}^{-1}\mathbf{x}$$

We multiply by $\mathbf{K}'^{-1}\mathbf{x}'$ in order to get a null mixed product :

$$0 = \lambda \mathbf{K}'^{-1}\mathbf{x}' \mathbf{t}_{\times} \mathbf{R}\mathbf{K}^{-1}\mathbf{x}$$

The anatomy of two views

From the projection in the two views we have :

$$\lambda \mathbf{x} = \mathbf{K}\mathbf{X} \quad \lambda' \mathbf{x}' = \mathbf{K}'(\mathbf{R}\mathbf{X} + \mathbf{t})$$

By eliminating $\mathbf{X}$ we get :

$$\mathbf{X} = \lambda \mathbf{K}^{-1}\mathbf{x} \quad \lambda' \mathbf{x}' = \mathbf{K}'(\lambda \mathbf{R}\mathbf{K}^{-1}\mathbf{x} + \mathbf{t})$$

$$\lambda' \mathbf{K}'^{-1}\mathbf{x}' = \lambda \mathbf{R}\mathbf{K}^{-1}\mathbf{x} + \mathbf{t}$$

We eliminate the sum by applying a cross product with $\mathbf{t}$:

$$\lambda' \mathbf{t}_\times \mathbf{K}'^{-1}\mathbf{x}' = \lambda \mathbf{t}_\times \mathbf{R}\mathbf{K}^{-1}\mathbf{x}$$

We multiply by $\mathbf{K}'^{-1}\mathbf{x}'$ in order to get a null mixed product :

$$0 = \lambda \mathbf{K}'^{-1}\mathbf{x}' \mathbf{t}_\times \mathbf{R}\mathbf{K}^{-1}\mathbf{x}$$

Finally, by transposing $\mathbf{K}'^{-1}\mathbf{x}'$ and ignoring the scalar λ we get :

$$\mathbf{x}'^{T} \underbrace{\mathbf{K}'^{-T}\mathbf{t}_\times \mathbf{R}\mathbf{K}^{-1}}_{\mathbf{F}} \mathbf{x} = 0$$

The fundamental matrix F

$$\boxed{\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0}$$

- applying the **F** constraint does not require information about the scene 3D structure

The fundamental matrix F

$$\boxed{\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0}$$

- applying the **F** constraint does not require information about the scene 3D structure
- **F** is valid for the whole image

The fundamental matrix F

$$\boxed{\mathbf{x'}^T \mathbf{F} \mathbf{x} = 0}$$

- applying the **F** constraint does not require information about the scene 3D structure
- **F** is valid for the whole image
- we may apply the constraint without performing/knowing the camera calibration

The fundamental matrix F

$$\boxed{\mathbf{x'}^T\mathbf{F}\mathbf{x} = 0}$$

- applying the **F** constraint does not require information about the scene 3D structure
- **F** is valid for the whole image
- we may apply the constraint without performing/knowing the camera calibration
- For a given point $\mathbf{x}'$,we denote by $\mathbf{l}'$ its corresponding *epipolar line*. It follows from $\mathbf{x'}^T\mathbf{F}\mathbf{x} = 0$ that

$$\mathbf{l}' = \mathbf{F}\mathbf{x}$$

The fundamental matrix F

$$\boxed{\mathbf{x'}^T\mathbf{F}\mathbf{x} = 0}$$

- applying the **F** constraint does not require information about the scene 3D structure
- **F** is valid for the whole image
- we may apply the constraint without performing/knowing the camera calibration
- For a given point $\mathbf{x}'$,we denote by $\mathbf{l}'$ its corresponding *epipolar line*. It follows from $\mathbf{x'}^T\mathbf{F}\mathbf{x} = 0$ that
$$\mathbf{l}' = \mathbf{F}\mathbf{x}$$
- Similarly, $\mathbf{l} = \mathbf{F}^T\mathbf{x}'$

The fundamental matrix F

$$\boxed{\mathbf{x'}^T \mathbf{F} \mathbf{x} = 0}$$

- applying the **F** constraint does not require information about the scene 3D structure
- **F** is valid for the whole image
- we may apply the constraint without performing/knowing the camera calibration
- For a given point $\mathbf{x}'$,we denote by $\mathbf{l}'$ its corresponding *epipolar line*. It follows from $\mathbf{x'}^T \mathbf{F} \mathbf{x} = 0$ that
$$\mathbf{l}' = \mathbf{F}\mathbf{x}$$
- Similarly, $\mathbf{l} = \mathbf{F}^T \mathbf{x}'$
- The fundamental matrix constraint translates to a search along the epipolar line ...

The fundamental matrix F

$$\boxed{\mathbf{x'}^T \mathbf{F} \mathbf{x} = 0}$$

- applying the **F** constraint does not require information about the scene 3D structure
- **F** is valid for the whole image
- we may apply the constraint without performing/knowing the camera calibration
- For a given point $\mathbf{x}'$,we denote by $\mathbf{l}'$ its corresponding *epipolar line*. It follows from $\mathbf{x'}^T \mathbf{F} \mathbf{x} = 0$ that
$$\mathbf{l}' = \mathbf{F}\mathbf{x}$$
- Similarly, $\mathbf{l} = \mathbf{F}^T \mathbf{x}'$
- The fundamental matrix constraint translates to a search along the epipolar line ...
- ... but also $\mathbf{F} = \mathbf{K'}^{-T} \mathbf{t}_\times \mathbf{R} \mathbf{K}^{-1}$ encodes, along with the calibration matrices, *the rotation and translation* between views

The fundamental matrix F

Theorem

The condition which is necessary and sufficient for a matrix **F** to be a fundamental matrix is that

$$\det(\mathbf{F}) = 0$$

Multiple ways to notice that **F** is rank deficient :

- it follows from the fact that $\det(\mathbf{t}_\times) = 0$

The fundamental matrix F

Theorem

The condition which is necessary and sufficient for a matrix $\mathbf{F}$ to be a fundamental matrix is that

$$\det(\mathbf{F}) = 0$$

Multiple ways to notice that $\mathbf{F}$ is rank deficient :

- it follows from the fact that $\det(\mathbf{t}_\times) = 0$
- it follows from the fact that $\mathbf{Fe} = 0$

Computing F - the 8 point algorithm

Straightforward approach :

- each observation (match) provides a constraint on F as $\mathbf{x_i'}^T \mathbf{F} \mathbf{x_i} = 0$

Computing F - the 8 point algorithm

Straightforward approach :

- each observation (match) provides a constraint on F as $\mathbf{x_i'}^T \mathbf{F} \mathbf{x_i} = 0$
- if we group the unknowns as the column vector $\mathbf{f} = [f_{11}\ f_{12} \ldots f_{33}]$, the constraint may be expressed as $\mathbf{a_i f} = 0$, with $\mathbf{a_i}$ a row vector

Computing F - the 8 point algorithm

Straightforward approach :

- each observation (match) provides a constraint on F as $\mathbf{x_i'}^T \mathbf{F} \mathbf{x_i} = 0$
- if we group the unknowns as the column vector $\mathbf{f} = [f_{11}\ f_{12} \dots f_{33}]$, the constraint may be expressed as $\mathbf{a_i} \mathbf{f} = 0$, with $\mathbf{a_i}$ a row vector
- only 8 parameters are independent, since the scale is not determined

Computing F - the 8 point algorithm

Straightforward approach :

- each observation (match) provides a constraint on F as ${\mathbf{x}'_\mathbf{i}}^T \mathbf{F}\mathbf{x}_\mathbf{i} = 0$
- if we group the unknowns as the column vector $\mathbf{f} = [f_{11}\ f_{12} \dots f_{33}]$, the constraint may be expressed as $\mathbf{a}_\mathbf{i}\mathbf{f} = 0$, with $\mathbf{a}_\mathbf{i}$ a row vector
- only 8 parameters are independent, since the scale is not determined
- the search for $\mathbf{f}$ may be expressed as :

$$\min_{\mathbf{f}} \|\mathbf{A}\mathbf{f}\|, \text{subject to} \|\mathbf{f}\| = 1$$

where $\mathbf{A} = [\mathbf{a_1}\ \mathbf{a_2} \dots \mathbf{a_8}]$

Computing F - the 8 point algorithm

Straightforward approach :

- each observation (match) provides a constraint on F as $\mathbf{x_i'}^T \mathbf{F} \mathbf{x_i} = 0$
- if we group the unknowns as the column vector $\mathbf{f} = [f_{11}\ f_{12} \ldots f_{33}]$, the constraint may be expressed as $\mathbf{a_i} \mathbf{f} = 0$, with $\mathbf{a_i}$ a row vector
- only 8 parameters are independent, since the scale is not determined
- the search for $\mathbf{f}$ may be expressed as :

$$\min_{\mathbf{f}} \|\mathbf{A}\mathbf{f}\|, \text{subject to } \|\mathbf{f}\| = 1$$

  where $\mathbf{A} = [\mathbf{a_1}\ \mathbf{a_2} \ldots \mathbf{a_8}]$
- Solution : $\mathbf{f}$ is the last column of $\mathbf{V}$, where $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ is the SVD of $\mathbf{A}$

Computing F - the 8 point algorithm

Straightforward approach :

- each observation (match) provides a constraint on F as $\mathbf{x_i'}^T \mathbf{F}\mathbf{x_i} = 0$
- if we group the unknowns as the column vector $\mathbf{f} = [f_{11}\ f_{12} \dots f_{33}]$, the constraint may be expressed as $\mathbf{a_i}\mathbf{f} = 0$, with $\mathbf{a_i}$ a row vector
- only 8 parameters are independent, since the scale is not determined
- the search for $\mathbf{f}$ may be expressed as :

$$\min_{\mathbf{f}} \|\mathbf{A}\mathbf{f}\|, \text{subject to } \|\mathbf{f}\| = 1$$

  where $\mathbf{A} = [\mathbf{a_1}\ \mathbf{a_2} \dots \mathbf{a_8}]$
- Solution : $\mathbf{f}$ is the last column of $\mathbf{V}$, where $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T$ is the SVD of $\mathbf{A}$
- Proof :
  $\left\|\mathbf{U}\mathbf{D}\mathbf{V}^T\mathbf{f}\right\| = \left\|\mathbf{D}\mathbf{V}^T\mathbf{f}\right\|$, and $\|\mathbf{f}\| = \left\|\mathbf{V}^T\mathbf{f}\right\|$. We have to minimize $\left\|\mathbf{D}\mathbf{V}^T\mathbf{f}\right\|$ subject to $\left\|\mathbf{V}^T\mathbf{f}\right\| = 1$. If $\mathbf{y} = \mathbf{V}^T\mathbf{f}$, then we minimize $\|\mathbf{D}\mathbf{y}\|$ subject to $\|\mathbf{y}\| = 1$. Since $\mathbf{D}$ is diagonal with values in descending order, it means that $\mathbf{y} = (0, 0 \dots, 1)$, and $\mathbf{f} = \mathbf{V}\mathbf{y}$ is the last column of $\mathbf{V}$. (*A5.3, Hartley and Zisserman*)

Considerations - the 8 point algorithm

Straightforward approach :

- major issue : the solution **F** may violate the rank constraint !

Considerations - the 8 point algorithm

Straightforward approach :

- major issue : the solution **F** may violate the rank constraint !
- Hack : decompose **F** using SVD, set $\sigma_3 = 0$ and recompose.

Considerations - the 8 point algorithm

Straightforward approach :

- major issue : the solution **F** may violate the rank constraint !
- Hack : decompose **F** using SVD, set $\sigma_3 = 0$ and recompose.
- What about searching directly for a rank 2 solution for **F** ?

Considerations - the 8 point algorithm

Straightforward approach :

- major issue : the solution $\mathbf{F}$ may violate the rank constraint !
- Hack : decompose $\mathbf{F}$ using SVD, set $\sigma_3 = 0$ and recompose.
- What about searching directly for a rank 2 solution for $\mathbf{F}$?

The 7 point algorithm :

- Use 7 constraints for $\mathbf{Af} = \mathbf{0}$

Considerations - the 8 point algorithm

Straightforward approach :

- major issue : the solution $\mathbf{F}$ may violate the rank constraint !
- Hack : decompose $\mathbf{F}$ using SVD, set $\sigma_3 = 0$ and recompose.
- What about searching directly for a rank 2 solution for $\mathbf{F}$?

The 7 point algorithm :

- Use 7 constraints for $\mathbf{Af} = \mathbf{0}$
- Use SVD on $\mathbf{A}$ in order to find the vectors $\mathbf{f_1}$ and $\mathbf{f_2}$ that span the null space (the kernel) of $\mathbf{A}$

Considerations - the 8 point algorithm

Straightforward approach :

- major issue : the solution $\mathbf{F}$ may violate the rank constraint !
- Hack : decompose $\mathbf{F}$ using SVD, set $\sigma_3 = 0$ and recompose.
- What about searching directly for a rank 2 solution for $\mathbf{F}$?

The 7 point algorithm :

- Use 7 constraints for $\mathbf{Af} = \mathbf{0}$
- Use SVD on $\mathbf{A}$ in order to find the vectors $\mathbf{f_1}$ and $\mathbf{f_2}$ that span the null space (the kernel) of $\mathbf{A}$
- Find an element in the kernel expressed by the linear combination $\mathbf{f} = \mathbf{f_1} + \alpha\mathbf{f_2}$ which also satisfies $\det(\mathbf{F}) = 0$

Considerations - the 8 point algorithm

Straightforward approach :

- major issue : the solution $\mathbf{F}$ may violate the rank constraint !
- Hack : decompose $\mathbf{F}$ using SVD, set $\sigma_3 = 0$ and recompose.
- What about searching directly for a rank 2 solution for $\mathbf{F}$?

The 7 point algorithm :

- Use 7 constraints for $\mathbf{Af} = \mathbf{0}$
- Use SVD on $\mathbf{A}$ in order to find the vectors $\mathbf{f_1}$ and $\mathbf{f_2}$ that span the null space (the kernel) of $\mathbf{A}$
- Find an element in the kernel expressed by the linear combination $\mathbf{f} = \mathbf{f_1} + \alpha \mathbf{f_2}$ which also satisfies $\det(\mathbf{F}) = 0$
- $\det(\mathbf{F_1} + \alpha \mathbf{F_2})$ is a third degree polynomial, so up to three potential solutions may be recovered

Considerations - the 8 point algorithm

Straightforward approach :

- major issue : the solution $\mathbf{F}$ may violate the rank constraint !
- Hack : decompose $\mathbf{F}$ using SVD, set $\sigma_3 = 0$ and recompose.
- What about searching directly for a rank 2 solution for $\mathbf{F}$?

The 7 point algorithm :

- Use 7 constraints for $\mathbf{Af} = \mathbf{0}$
- Use SVD on $\mathbf{A}$ in order to find the vectors $\mathbf{f_1}$ and $\mathbf{f_2}$ that span the null space (the kernel) of $\mathbf{A}$
- Find an element in the kernel expressed by the linear combination $\mathbf{f} = \mathbf{f_1} + \alpha \mathbf{f_2}$ which also satisfies $\det(\mathbf{F}) = 0$
- $\det(\mathbf{F_1} + \alpha \mathbf{F_2})$ is a third degree polynomial, so up to three potential solutions may be recovered
- This algorithm is also preferred as fewer observations are needed

Outline

- The 3D representation of points
- The pinhole camera model
- Applying a coordinate transformation
- Homogeneous representations and algebraic operations
- The fundamental matrix
- The essential matrix
- Rectification

Using the camera calibration and the essential matrix

If the calibration matrices $\mathbf{K}$ and $\mathbf{K}'$ are known :

- we may recover the pose information from $\mathbf{F} = \mathbf{K}'^{-T}\mathbf{t}_\times\mathbf{R}\mathbf{K}^{-1}$:

$$\mathbf{E} = \mathbf{t}_\times\mathbf{R} = \mathbf{K}'^{T}\mathbf{F}\mathbf{K}$$

Using the camera calibration and the essential matrix

If the calibration matrices $\mathbf{K}$ and $\mathbf{K}'$ are known :

- we may recover the pose information from $\mathbf{F} = \mathbf{K}'^{-T}\mathbf{t}_\times\mathbf{R}\mathbf{K}^{-1}$:

$$\mathbf{E} = \mathbf{t}_\times\mathbf{R} = \mathbf{K}'^{T}\mathbf{F}\mathbf{K}$$

- $\mathbf{E}$ has five degrees of freedom (and not six) because the relative translation $\mathbf{t}$ has a scale ambiguity (just as $\mathbf{F}$).

Using the camera calibration and the essential matrix

If the calibration matrices $\mathbf{K}$ and $\mathbf{K}'$ are known :

- we may recover the pose information from $\mathbf{F} = \mathbf{K}'^{-T}\mathbf{t}_\times \mathbf{R}\mathbf{K}^{-1}$:

$$\mathbf{E} = \mathbf{t}_\times \mathbf{R} = \mathbf{K}'^{T}\mathbf{F}\mathbf{K}$$

- $\mathbf{E}$ has five degrees of freedom (and not six) because the relative translation $\mathbf{t}$ has a scale ambiguity (just as $\mathbf{F}$).
- Beside $\det(\mathbf{E}) = 0$, there is an additional constraint with respect to $\mathbf{F}$, which results from the structure of $\mathbf{E}$:

  Theorem : The condition which is necessary and sufficient for a matrix $\mathbf{E}$ to be an essential matrix is that two of its singular values be equal, and the third one be 0.

Using the camera calibration and the essential matrix

If the calibration matrices $\mathbf{K}$ and $\mathbf{K}'$ are known :

- we may recover the pose information from $\mathbf{F} = \mathbf{K}'^{-T}\mathbf{t}_\times \mathbf{R}\mathbf{K}^{-1}$:

$$\mathbf{E} = \mathbf{t}_\times \mathbf{R} = \mathbf{K}'^{T}\mathbf{F}\mathbf{K}$$

- $\mathbf{E}$ has five degrees of freedom (and not six) because the relative translation $\mathbf{t}$ has a scale ambiguity (just as $\mathbf{F}$).
- Beside $\det(\mathbf{E}) = 0$, there is an additional constraint with respect to $\mathbf{F}$, which results from the structure of $\mathbf{E}$:

  Theorem : The condition which is necessary and sufficient for a matrix $\mathbf{E}$ to be an essential matrix is that two of its singular values be equal, and the third one be 0.
- There are thus at least five points needed for recovering directly $\mathbf{E}$ from an image pair, assuming that the calibration matrices are known, and there is an algorithm which solves this minimal problem(Nistér, David. "An efficient solution to the five-point relative pose problem." IEEE Transactions on Pattern Analysis and Machine Intelligence (2004).)

Using the camera calibration and the essential matrix

If the calibration matrices $\mathbf{K}$ and $\mathbf{K}'$ are known :

- we may recover the pose information from $\mathbf{F} = \mathbf{K}'^{-T}\mathbf{t}_\times\mathbf{R}\mathbf{K}^{-1}$:

$$\mathbf{E} = \mathbf{t}_\times\mathbf{R} = \mathbf{K}'^{T}\mathbf{F}\mathbf{K}$$

- $\mathbf{E}$ has five degrees of freedom (and not six) because the relative translation $\mathbf{t}$ has a scale ambiguity (just as $\mathbf{F}$).
- Beside $\det(\mathbf{E}) = 0$, there is an additional constraint with respect to $\mathbf{F}$, which results from the structure of $\mathbf{E}$:

  Theorem : The condition which is necessary and sufficient for a matrix $\mathbf{E}$ to be an essential matrix is that two of its singular values be equal, and the third one be 0.
- There are thus at least five points needed for recovering directly $\mathbf{E}$ from an image pair, assuming that the calibration matrices are known, and there is an algorithm which solves this minimal problem(Nistér, David. "An efficient solution to the five-point relative pose problem." IEEE Transactions on Pattern Analysis and Machine Intelligence (2004).)
- Knowing $\mathbf{E}$: interesting for relative pose estimation

Using the camera calibration and the essential matrix

If the calibration matrices $\mathbf{K}$ and $\mathbf{K}'$ are known :

- we may recover the pose information from $\mathbf{F} = \mathbf{K}'^{-T}\mathbf{t}_\times\mathbf{R}\mathbf{K}^{-1}$:

$$\mathbf{E} = \mathbf{t}_\times\mathbf{R} = \mathbf{K}'^{T}\mathbf{F}\mathbf{K}$$

- $\mathbf{E}$ has five degrees of freedom (and not six) because the relative translation $\mathbf{t}$ has a scale ambiguity (just as $\mathbf{F}$).
- Beside $\det(\mathbf{E}) = 0$, there is an additional constraint with respect to $\mathbf{F}$, which results from the structure of $\mathbf{E}$:

  Theorem : The condition which is necessary and sufficient for a matrix $\mathbf{E}$ to be an essential matrix is that two of its singular values be equal, and the third one be 0.
- There are thus at least five points needed for recovering directly $\mathbf{E}$ from an image pair, assuming that the calibration matrices are known, and there is an algorithm which solves this minimal problem(Nistér, David. "An efficient solution to the five-point relative pose problem." IEEE Transactions on Pattern Analysis and Machine Intelligence (2004).)
- Knowing $\mathbf{E}$: interesting for relative pose estimation
- Main disadvantage : $\mathbf{K}$ and $\mathbf{K}'$ are required to get to $\mathbf{E}$

Recovering R and t from E

It has been shown that the decomposition of **E** is possible and there are actually four valid solutions (*9.6.2, Hartley and Zisserman*) :

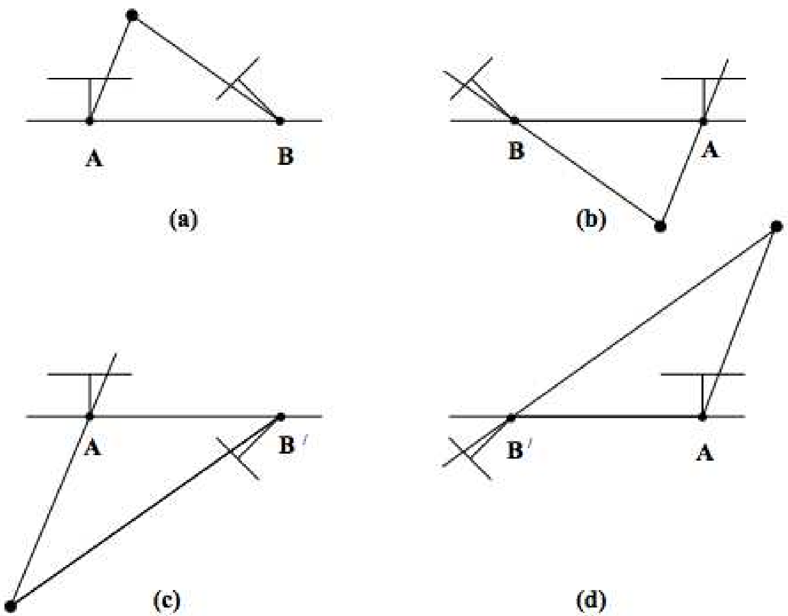


Fig. 9.12. **The four possible solutions for calibrated reconstruction from E.** *Between the left and right sides there is a baseline reversal. Between the top and bottom rows camera B rotates* 180° *about the baseline. Note, only in (a) is the reconstructed point in front of both cameras.*

- Identify the correct solution : cheirality check (the 3D points have to be in front of the camera) with an additional match from the two views

Outline

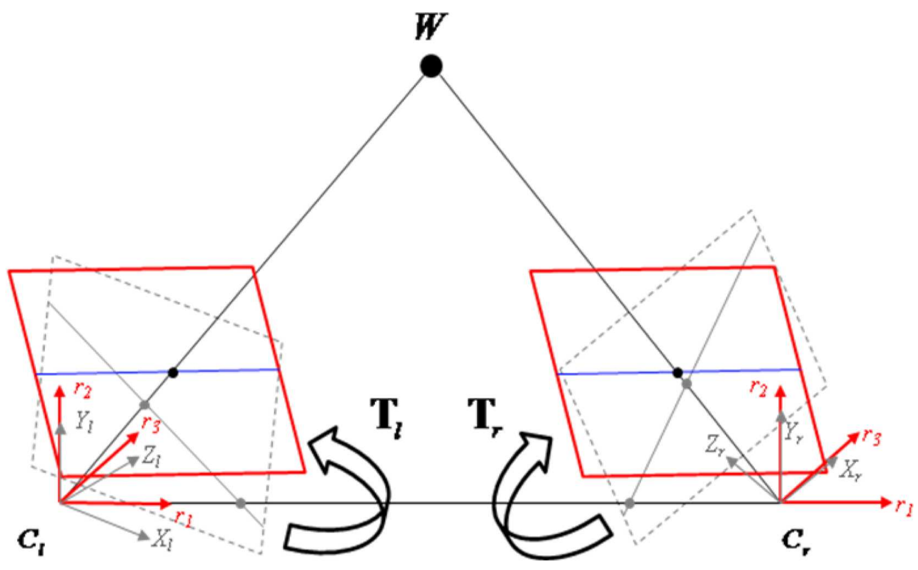
- The 3D representation of points
- The pinhole camera model
- Applying a coordinate transformation
- Homogeneous representations and algebraic operations
- The fundamental matrix
- The essential matrix
- Rectification

Rectification

Using $\mathbf{F}$, we restrict the search for the corresponding projection $\mathbf{x}'$ of a point $\mathbf{x}$ to a line (the epipolar line $\mathbf{l}' = \mathbf{F}\mathbf{x}$).

Stereo rectification

- Apply an adjustment to the images in order to get horizontal epipolar lines in both views

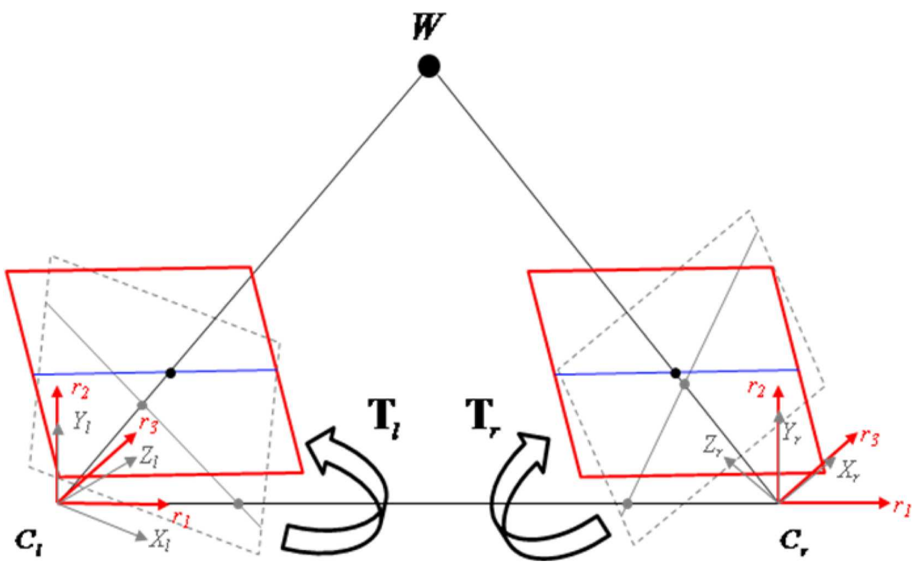


Rectification

Using $\mathbf{F}$, we restrict the search for the corresponding projection $\mathbf{x}'$ of a point $\mathbf{x}$ to a line (the epipolar line $\mathbf{l}' = \mathbf{F}\mathbf{x}$).

Stereo rectification

- Apply an adjustment to the images in order to get horizontal epipolar lines in both views
- The search for $\mathbf{x}'$ takes place simply along the same corresponding row in the second image : interesting for dense correspondence

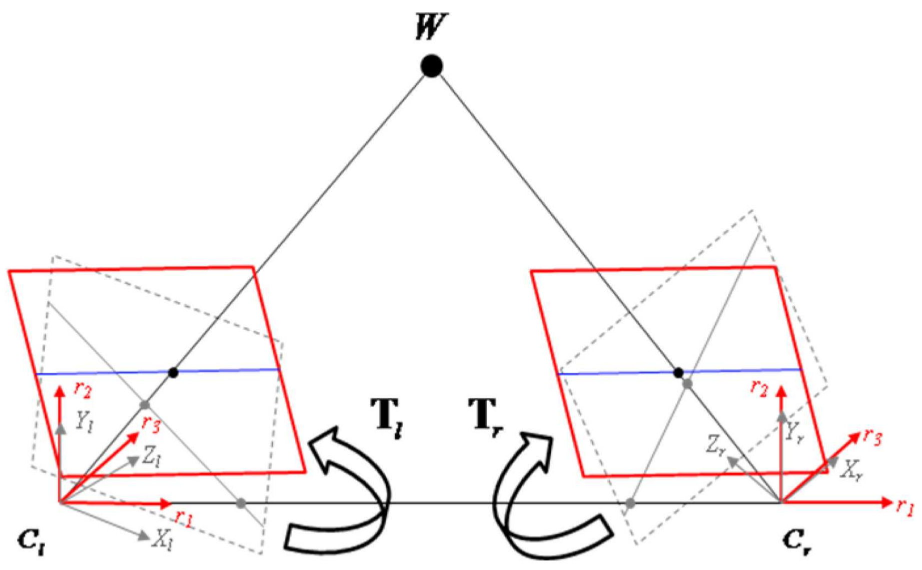


Rectification

Using **F**, we restrict the search for the corresponding projection $\mathbf{x}'$ of a point $\mathbf{x}$ to a line (the epipolar line $\mathbf{l}' = \mathbf{F}\mathbf{x}$).

Stereo rectification

- Apply an adjustment to the images in order to get horizontal epipolar lines in both views
- The search for $\mathbf{x}'$ takes place simply along the same corresponding row in the second image : interesting for dense correspondence
- This implies that epipoles are at horizontal infinity : $\mathbf{e} = \mathbf{e}' = [1\ 0\ 0]^T$

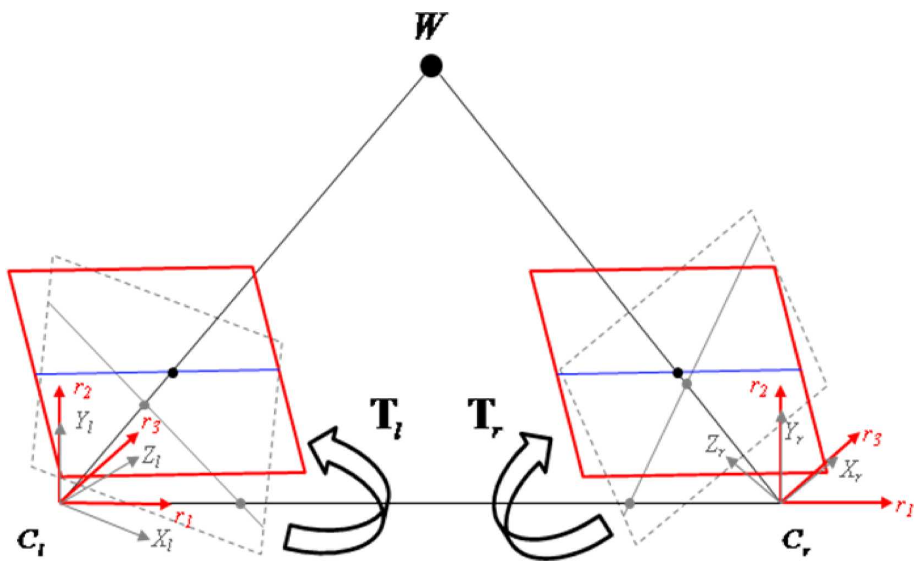


Rectification

Using $\mathbf{F}$, we restrict the search for the corresponding projection $\mathbf{x}'$ of a point $\mathbf{x}$ to a line (the epipolar line $\mathbf{l}' = \mathbf{F}\mathbf{x}$).

Stereo rectification

- Apply an adjustment to the images in order to get horizontal epipolar lines in both views
- The search for $\mathbf{x}'$ takes place simply along the same corresponding row in the second image : interesting for dense correspondence
- This implies that epipoles are at horizontal infinity : $\mathbf{e} = \mathbf{e}' = [1\ 0\ 0]^T$
- Apply a virtual rotation of cameras (Fusiello, A. ; Trucco, E. ; Verri, A. A compact algorithm for rectification of stereo pairs. Mach. Vision Appl 2000)

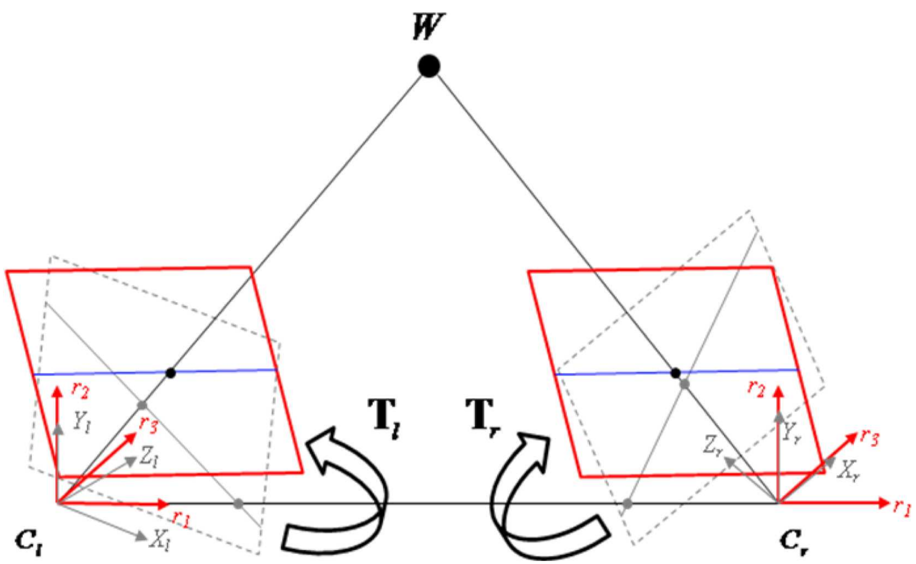


Rectification

Using $\mathbf{F}$, we restrict the search for the corresponding projection $\mathbf{x}'$ of a point $\mathbf{x}$ to a line (the epipolar line $\mathbf{l}' = \mathbf{F}\mathbf{x}$).

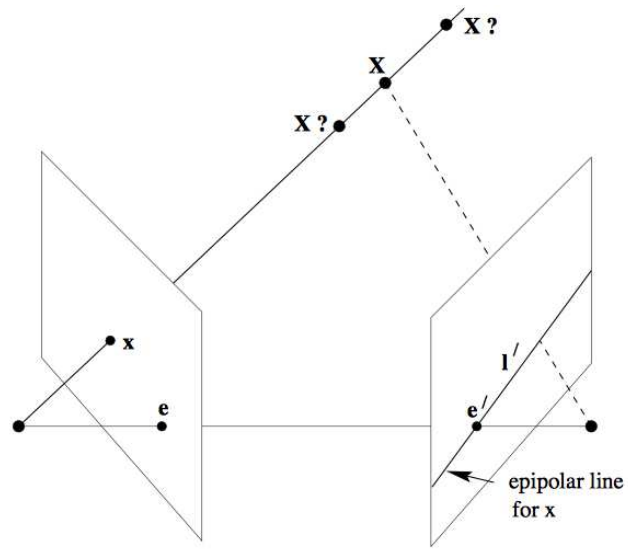
Stereo rectification

- Apply an adjustment to the images in order to get horizontal epipolar lines in both views
- The search for $\mathbf{x}'$ takes place simply along the same corresponding row in the second image : interesting for dense correspondence
- This implies that epipoles are at horizontal infinity : $\mathbf{e} = \mathbf{e}' = [1\ 0\ 0]^T$
- Apply a virtual rotation of cameras (Fusiello, A. ; Trucco, E. ; Verri, A. A compact algorithm for rectification of stereo pairs. Mach. Vision Appl 2000)
- An interpolation is required for creating the new images, but high computation gain overall



Triangulation - the building block of 3D reprojections

We have the pose $\mathbf{R}, \mathbf{t}'$ between cameras and the projection locations $\mathbf{x}, \mathbf{x}'$. What now ?



Get $\mathbf{X}$: triangulate the point in 3D

Triangulation - the building block of 3D reprojections

We have the pose $\mathbf{R}, \mathbf{t}'$ between cameras and the projection locations $\mathbf{x}, \mathbf{x}'$. What now?

Get $\mathbf{X}$: triangulate the point in 3D

- Back to our stereo projection equations :

$$\lambda \mathbf{x} = \mathbf{K}\mathbf{X} \quad \lambda' \mathbf{x}' = \mathbf{K}'(\mathbf{R}\mathbf{X} + \mathbf{t})$$

Triangulation - the building block of 3D reprojections

We have the pose $\mathbf{R}, \mathbf{t}'$ between cameras and the projection locations $\mathbf{x}, \mathbf{x}'$. What now ?

Get $\mathbf{X}$: triangulate the point in 3D

- Back to our stereo projection equations :

$$\lambda \mathbf{x} = \mathbf{K}\mathbf{X} \quad \lambda' \mathbf{x}' = \mathbf{K}'(\mathbf{R}\mathbf{X} + \mathbf{t})$$

- We have five scalar unknowns and six equations - a direct approach is possible by solving an overdetermined linear system

Triangulation - the building block of 3D reprojections

We have the pose $\mathbf{R}, \mathbf{t}'$ between cameras and the projection locations $\mathbf{x}, \mathbf{x}'$. What now ?

Get $\mathbf{X}$: triangulate the point in 3D

- Back to our stereo projection equations :

$$\lambda \mathbf{x} = \mathbf{K}\mathbf{X} \quad \lambda' \mathbf{x}' = \mathbf{K}'(\mathbf{R}\mathbf{X} + \mathbf{t})$$

- We have five scalar unknowns and six equations - a direct approach is possible by solving an overdetermined linear system
- There are other algorithms which are more accurate, but costlier
  Hartley, R. I., Sturm, P. (1997). Triangulation. Computer vision and image understanding, 68(2), 146-157

  Lindstrom, Peter. "Triangulation made easy." In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pp. 1554-1561

Triangulation - the building block of 3D reprojections

We have the pose $\mathbf{R}, \mathbf{t}'$ between cameras and the projection locations $\mathbf{x}, \mathbf{x}'$. What now ?

Get $\mathbf{X}$: triangulate the point in 3D

- Back to our stereo projection equations :

$$\lambda \mathbf{x} = \mathbf{K}\mathbf{X} \quad \lambda' \mathbf{x}' = \mathbf{K}'(\mathbf{R}\mathbf{X} + \mathbf{t})$$

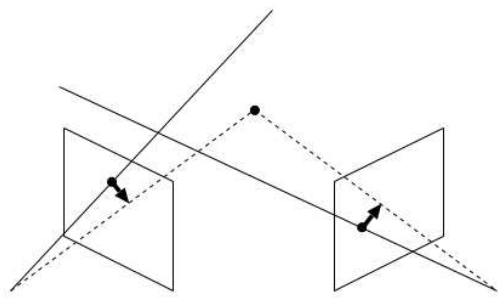
- We have five scalar unknowns and six equations - a direct approach is possible by solving an overdetermined linear system
- There are other algorithms which are more accurate, but costlier
  Hartley, R. I., Sturm, P. (1997). Triangulation. Computer vision and image understanding, 68(2), 146-157

  Lindstrom, Peter. "Triangulation made easy." In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, pp. 1554-1561
- The linear approach is reasonably good, and it is effective especially if used as an initialization for a nonlinear refinement (as we will see in the following slides)

Triangulation - how to use multiple views

If we have multiple views, the unknown $\mathbf{X}_j$ may be constrained by multiple observations $\mathbf{z}_{j,\tau}$ from cameras C_τ characterized by some pose parametrization $\mathbf{s}_\tau$. How to use them effectively together ?

Nonlinear optimization

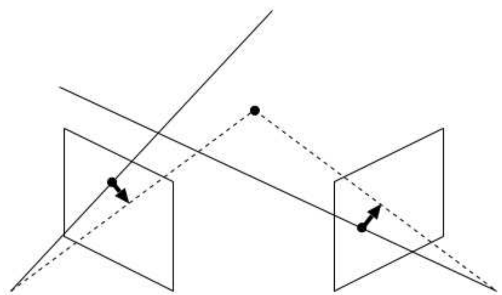


Triangulation - how to use multiple views

If we have multiple views, the unknown $\mathbf{X}_j$ may be constrained by multiple observations $\mathbf{z}_{j,\tau}$ from cameras C_τ characterized by some pose parametrization $\mathbf{s}_\tau$. How to use them effectively together ?

Nonlinear optimization

- Analytical solutions are not practical, in most cases we solve the optimization iteratively



Triangulation - how to use multiple views

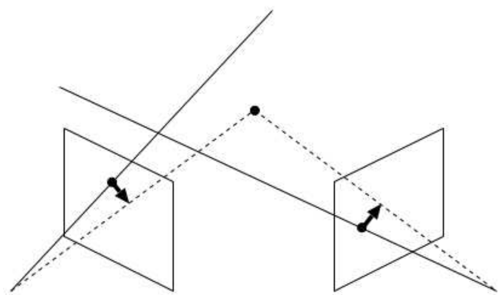
If we have multiple views, the unknown $\mathbf{X}_j$ may be constrained by multiple observations $\mathbf{z}_{j,\tau}$ from cameras C_τ characterized by some pose parametrization $\mathbf{s}_\tau$. How to use them effectively together ?

Nonlinear optimization

- Analytical solutions are not practical, in most cases we solve the optimization iteratively
- We define an error related to each of the observation, i.e. the distance between the observation and the projection of $\mathbf{X}_j$: $e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_j) = \mathbf{z}_j - g(\mathbf{s}_\tau, \mathbf{X}_j)$, where g is the camera projection function. Then, we have :

$$\hat{\mathbf{X}}_j = \arg\min_{\mathbf{X}_j} \sum_\tau e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_j)^T e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_j)$$

- Use Gauss-Newton or LM (usually the optimum is not far from a reasonable initialization)



Triangulation - how to use multiple views

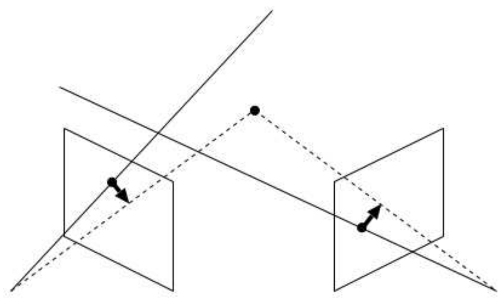
If we have multiple views, the unknown $\mathbf{X}_j$ may be constrained by multiple observations $\mathbf{z}_{j,\tau}$ from cameras C_τ characterized by some pose parametrization $\mathbf{s}_\tau$. How to use them effectively together ?

Nonlinear optimization

- Analytical solutions are not practical, in most cases we solve the optimization iteratively
- We define an error related to each of the observation, i.e. the distance between the observation and the projection of $\mathbf{X}_j$: $e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_j) = \mathbf{z}_j - g(\mathbf{s}_\tau, \mathbf{X}_j)$, where g is the camera projection function. Then, we have :

$$\hat{\mathbf{X}}_j = \arg\min_{\mathbf{X}_j} \sum_\tau e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_j)^T e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_j)$$

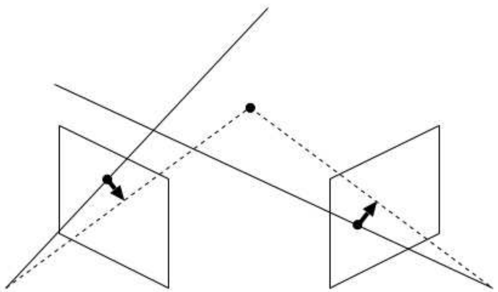
- Use Gauss-Newton or LM (usually the optimum is not far from a reasonable initialization)
- More than one 3D point may be refined, but in this way the optimizations are decoupled



Pose estimation - how to use multiple views

Opposite problem : we have a set of 3D points $\mathbf{X_j}$ (computed previously) which are visible from camera C_τ. Based on current observations $\mathbf{z}_{j,\tau}$ from C_τ we would like to estimate its pose $\mathbf{s}_\tau$.

Nonlinear optimization



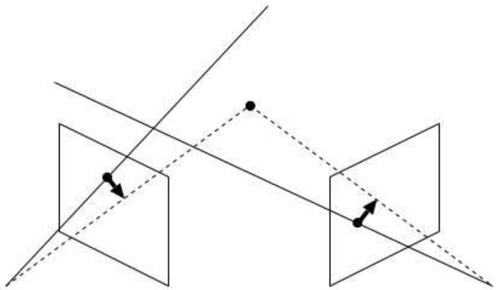
Pose estimation - how to use multiple views

Opposite problem : we have a set of 3D points $\mathbf{X_j}$ (computed previously) which are visible from camera C_τ. Based on current observations $\mathbf{z}_{j,\tau}$ from C_τ we would like to estimate its pose $\mathbf{s}_\tau$.

Nonlinear optimization

- We define an error related to each of the observations, i.e. the distance between the observation and the projection of $\mathbf{X}_j$: $e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau}) = \mathbf{z}_{j,\tau} - g(\mathbf{s}_\tau, \mathbf{X}_j)$, where g is the camera projection function. Then, we have :

$$\hat{\mathbf{s}}_\tau = \arg\min_{\mathbf{s}_\tau} \sum_j e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau})^T e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau})$$



Pose estimation - how to use multiple views

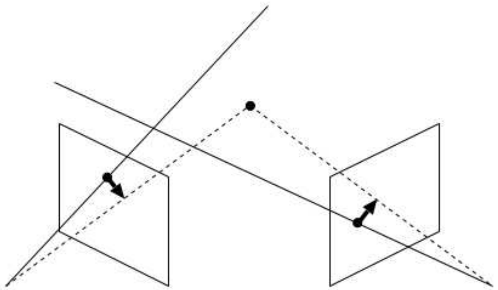
Opposite problem : we have a set of 3D points $\mathbf{X_j}$ (computed previously) which are visible from camera C_τ. Based on current observations $\mathbf{z}_{j,\tau}$ from C_τ we would like to estimate its pose $\mathbf{s}_\tau$.

Nonlinear optimization

- We define an error related to each of the observations, i.e. the distance between the observation and the projection of $\mathbf{X}_j$: $e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau}) = \mathbf{z}_{j,\tau} - g(\mathbf{s}_\tau, \mathbf{X}_j)$, where g is the camera projection function. Then, we have :

$$\hat{\mathbf{s}}_\tau = \arg\min_{\mathbf{s}_\tau} \sum_j e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau})^T e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau})$$

- Use Gauss-Newton or LM, but the initialization is very important. Two strategies help :



Pose estimation - how to use multiple views

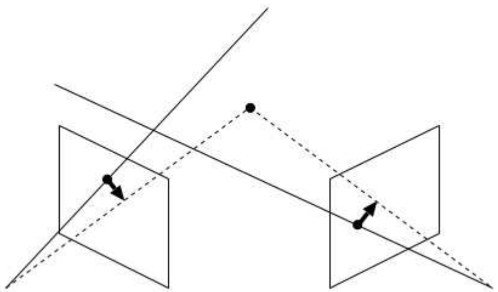
Opposite problem : we have a set of 3D points $\mathbf{X_j}$ (computed previously) which are visible from camera C_τ. Based on current observations $\mathbf{z}_{j,\tau}$ from C_τ we would like to estimate its pose $\mathbf{s}_\tau$.

Nonlinear optimization

- We define an error related to each of the observations, i.e. the distance between the observation and the projection of $\mathbf{X}_j$: $e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau}) = \mathbf{z}_{j,\tau} - g(\mathbf{s}_\tau, \mathbf{X}_j)$, where g is the camera projection function. Then, we have :

$$\hat{\mathbf{s}}_\tau = \arg\min_{\mathbf{s}_\tau} \sum_j e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau})^T e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau})$$

- Use Gauss-Newton or LM, but the initialization is very important. Two strategies help :
  - if the camera is moving, predict the current location based on its previous trajectory



Pose estimation - how to use multiple views

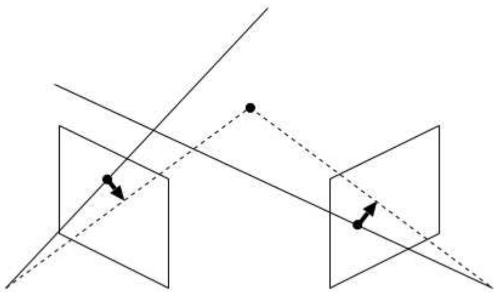
Opposite problem : we have a set of 3D points $\mathbf{X_j}$ (computed previously) which are visible from camera C_τ. Based on current observations $\mathbf{z}_{j,\tau}$ from C_τ we would like to estimate its pose $\mathbf{s}_\tau$.

Nonlinear optimization

- We define an error related to each of the observations, i.e. the distance between the observation and the projection of $\mathbf{X}_j$: $e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau}) = \mathbf{z}_{j,\tau} - g(\mathbf{s}_\tau, \mathbf{X}_j)$, where g is the camera projection function. Then, we have :

$$\hat{\mathbf{s}}_\tau = \arg\min_{\mathbf{s}_\tau} \sum_j e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau})^T e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau})$$

- Use Gauss-Newton or LM, but the initialization is very important. Two strategies help :
  - if the camera is moving, predict the current location based on its previous trajectory
  - from the projection of three 3D points in space and their projections, one may compute the camera pose in a closed form (the P3P problem)



Limitations of previous approaches

Assumptions :

Limitations of previous approaches

Assumptions :

- for triangulation : we assume that the pose is correctly estimated

Limitations of previous approaches

Assumptions :

- for triangulation : we assume that the pose is correctly estimated
- for pose estimation : we assume that the 3D locations are accurate

Limitations of previous approaches

Assumptions :

- for triangulation : we assume that the pose is correctly estimated
- for pose estimation : we assume that the 3D locations are accurate
- in reality all estimations we perform are noisy

Limitations of previous approaches

Assumptions :

- for triangulation : we assume that the pose is correctly estimated
- for pose estimation : we assume that the 3D locations are accurate
- in reality all estimations we perform are noisy
- if we also apply the process iteratively (triangulation, pose estimation and repeat) the errors will be amplified (drift)

Global optimization - initial step

Since computational power is widely available for autonomous systems, we favour a solution which minimizes jointly with respect to the point locations and to the poses.

Initial step :

Global optimization - initial step

Since computational power is widely available for autonomous systems, we favour a solution which minimizes jointly with respect to the point locations and to the poses.

Initial step :

- we will just add a new unknown pose to the previous set of variables and refine it :

$$\hat{\mathbf{s}}_\tau = \arg\min_{\mathbf{s}_\tau} \sum_j e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau})^T e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau})$$

Global optimization - initial step

Since computational power is widely available for autonomous systems, we favour a solution which minimizes jointly with respect to the point locations and to the poses.

Initial step :

- we will just add a new unknown pose to the previous set of variables and refine it :
$$\hat{\mathbf{s}}_\tau = \arg\min_{\mathbf{s}_\tau} \sum_j e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau})^T e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau})$$
- observation : this step does not modify $\mathbf{X}$

Global optimization - initial step

Since computational power is widely available for autonomous systems, we favour a solution which minimizes jointly with respect to the point locations and to the poses.

Initial step :

- we will just add a new unknown pose to the previous set of variables and refine it :

$$\hat{\mathbf{s}}_\tau = \arg\min_{\mathbf{s}_\tau} \sum_j e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau})^T e(\mathbf{s}_\tau, \mathbf{X}_j, \mathbf{z}_{j,\tau})$$

- observation : this step does not modify $\mathbf{X}$
- the interest of the initial step is just to provide a quality initialization for $\mathbf{s}_\tau$ as $\hat{\mathbf{s}}_t$

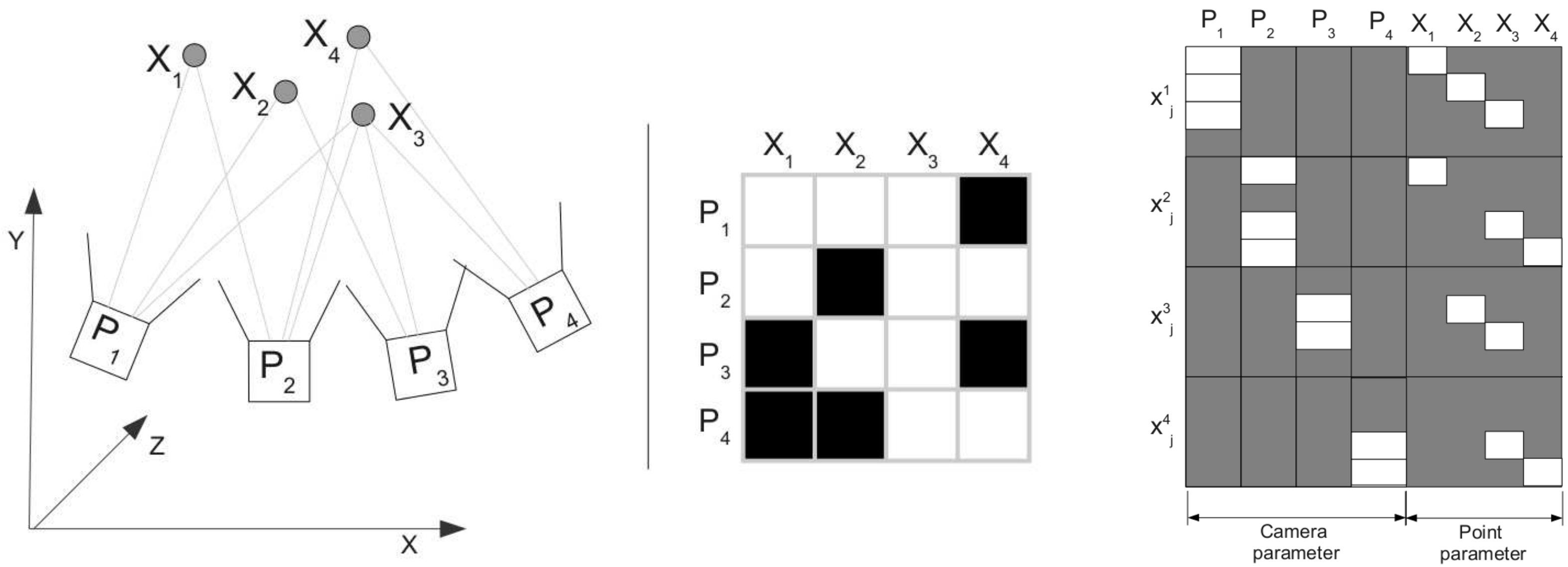
Global optimization - final step

We compute the MAP (Maximum A Posteriori) for the maximum amount of preliminary estimations and observations that we have at that moment (brutal, massive optimization). The solution we search this time is provided by :

$$\tilde{\mathbf{S}}_{0:t}, \tilde{\mathbf{X}} = \arg\min_{\mathbf{S}_{0:t}, \mathbf{X}} \sum_{\tau=0}^{T} \sum_{j=1}^{M} e(\mathbf{s}_\tau, \mathbf{X}_{j,\tau}, z_{j,\tau})^T \, e(\mathbf{s}_\tau, \mathbf{X}_{j,\tau}, z_{j,\tau})$$

The complexity of this algorithm, once we exploit the sparseness of its Jacobian : $O(T^3 + MT^2)$, which is very interesting since $M \gg T$.

Towards real time reconstruction



An example of configuration : 5207 3D points, 54 poses, 24609 projections, 15945 variables, 21 it., 7.99 sec.
Not fast enough !

- Selection of key-frames
- Parallel execution of tracking et BA (initial and final steps)
- Limit the number of iterations (when needed)
- Local Bundle Adjustment

Typical architecture for RT optimization