“An idea which can be used once is a trick.
If it can be used more than once it becomes a method”

- George Pólya and Gabor Szegő
Rigid Body Transformations &
Generalized Camera Models

CSE 4392-5369 “Vision-based Robot Sensing, Localization and Control”

Dr. Gian Luca Mariottini, Ph.D.
Department of Computer Science and Engineering
University of Texas at Arlington

WEB: http://ranger.uta.edu/~gianluca/
Looking for a General Pinhole Model

So far we examined a simple pinhole camera model .... why simple?

**GOAL:** Extend the simple model to the case of a 3-D point expressed in the **world** reference frame. \( \{W\} \)

Image Coordinates

\[
x \triangleq \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z_c} \begin{bmatrix} X_c \\ Y_c \end{bmatrix}
\]

Pixel Coordinates

\[
s\hat{u} = K \Pi_0 C \tilde{X}
\]
Our goal is to find a representation for a rigid transformation between \{C\} and \{W\}.
Why is the $\{W\}$ frame important?

The 3D CAD model of an object will match the real image of that object when $\{W\} = \{C\}$.
We can immediately see that:

\[ \mathbf{C} \mathbf{X} = f(\mathbf{W} \mathbf{X}) \]

\[ \mathbf{C} \mathbf{X} = \mathbf{C}_W \mathbf{t} + \mathbf{W} \mathbf{X} \quad \text{(i)} \]

that is:

\[
\begin{bmatrix}
X_C \\
Y_C
\end{bmatrix} =
\begin{bmatrix}
t_x \\
t_y
\end{bmatrix} +
\begin{bmatrix}
X_W \\
Y_W
\end{bmatrix}
\]

The above equation (i) is valid also in the 3-D case.

Note that we assumed here that there is no rotation between the two frames.
2-D Rotation

Consider this simple 2-D case:

\[
\begin{align*}
\{W\} & \quad \{C\} \\
Y_W & \quad Y_C \\
X_W & \quad X_C \\
\theta & \quad \theta
\end{align*}
\]

GOAL: \( ^C X = f( ^W X ) \)

From easy geometric considerations:

\[
X_C = X_W \cos \theta - Y_W \sin \theta
\]

And analogously (try it!)

\[
Y_C = X_W \sin \theta + Y_W \cos \theta
\]

Writing it in vector form we obtain:

\[
^C X = \begin{bmatrix} X_C \\ Y_C \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \end{bmatrix} \triangleq ^W X
\]

\[
^W R \quad = \text{2-D Rotation Matrix}
\]

The expression of a pure 2-D rotation moving the frame \( \{C\} \) to \( \{W\} \) (right-hand) transform points as:

\[
^C X = ^C R \; ^W X
\]

(Rotates points from \( \{W\} \) to \( \{C\} \))
Analogously to the previous case it can be shown that:

\[
\begin{align*}
X_W &= X_C \cos \theta + Y_C \sin \theta \\
Y_W &= -Y_C \sin \theta + X_C \cos \theta \\
\begin{bmatrix} X_W \\ Y_W \end{bmatrix} &= \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} X_C \\ Y_C \end{bmatrix}
\end{align*}
\]

Comparing with the previous rotation matrix we clearly see that

\[
W X = C R C X
\]

In a geometric sense, the orientation of \{C\} wrt \{W\} is the inverse of the orientation of \{W\} wrt \{C\}

\[
C_W R^T = C_W R^{-1}
\]

See `Ex_3_2DRotations.m`
2-D Rotation: orthonormality

- The **columns** (and thus the **rows**) of a rotation matrix are **orthogonal**:

  \[
  \mathbf{c}_1 = \begin{bmatrix}
  \cos \theta \\
  \sin \theta 
  \end{bmatrix} \quad \mathbf{c}_2 = \begin{bmatrix}
  -\sin \theta \\
  \cos \theta 
  \end{bmatrix}
  \]

- Each **column vector** (and thus the **rows**) of a rotation matrix have unit length (**norm**)

  \[
  \|\mathbf{c}_1\| = \sqrt{(\cos \theta)^2 + (\sin \theta)^2} = 1
  \]

- The **determinant** of a rotation matrix is equal to:

  \[
  \det \mathbf{R} = 1
  \]

- See **Ex_3_2DRotations.m**
In the case that the points are specified in a 3-D space, then we need to use 3x3 rotation matrices.

The same orthonormality property valid for 2-D rotations are still valid in the 3-D case.
Composition of Successive Rotations (1)

Suppose a camera rotating from \( \{W\} \) through \( \{C1\} \) to \( \{C2\} \).

We assume given \( \frac{C_1}{W} R \) and \( \frac{C_2}{C_1} R \)

The vector \( \frac{W}{C_2} X \) can be expressed in \( \frac{C_2}{W} X \) by considering that:

\[
C_2 X = \frac{C_2}{W} R \frac{W}{C_1} X
\]

GOAL: Find \( \frac{C_2}{W} R \), i.e. the rotation that brings frame \( \{C2\} \) onto \( \{W\} \)

We know that:

\[
C_2 X = \frac{C_2}{C_1} R \frac{C_1}{W} X \quad (i)
\]

\[
C_1 X = \frac{C_1}{W} R \frac{W}{C_1} X \quad (ii)
\]

and using (ii) into (i) we obtain

\[
C_2 X = \frac{C_2}{C_1} R \frac{C_1}{W} R \frac{W}{C_1} R \frac{W}{C_1} X
\]

that is:

\[
\frac{C_2}{W} R = \frac{C_2}{C_1} R \frac{C_1}{W} R
\]

Try with a 2-D case!

(Recall that \( \cos(A + B) = \cos A \cos B - \sin A \sin B \) and \( \sin(a + b) = \sin a \cos b + \cos a \sin b \))
Composition of Successive Rotations (2)

The rotation matrix $C_2^W R$ can be interpreted as the composition of successive rotations, i.e.,

$$C_2^W R = C_2^C R C_1^W R$$

It can be regarded as obtained from two steps:

1) First rotate the frame $\{C2\}$ (of $C_2^C R$) so as to align it with $\{C1\}$;
2) Then rotate the new frame $\{C1\}$ to the frame $\{W\}$, by using $C_1^W R$.

Note again that the overall rotation has been expressed as a sequence of partial rotations, each one defined with respect to the preceding ref. frame.

What happens if we invert the order of applied rotations?

Changing the order of successive rotations (wrt current frame) has different results.
Euler Angles: ZYZ (1)

A rotation matrix has only 3 **degrees-of-freedom**, that is only 3 **parameters** are sufficient to describe it.

A minimal representation of orientation can be obtained by using a set of **three angles**:

\[
[\begin{bmatrix} \varphi \\ \theta \\ \psi \end{bmatrix}]^T
\]

The rotation described by the **ZYZ angles** is the following:

+ Rotate the initial frame by the angle $\varphi$ about axis $Z$, i.e. $R_z(\varphi)$
+ Rotate the current frame by the angle $\theta$ about axis $Y'$, i.e. $R_{y'}(\theta)$
+ Rotate the current frame by the angle $\psi$ about axis $Z''$, i.e. $R_{z''}(\psi)$
Euler Angles: ZYZ (2)

Since the ZYZ rotation is obtained by composition of rotations with respect the current frame, then it can be written as:

$$R_{zyz} = R_z(\varphi)R_y'(\theta)R_z''(\psi)$$

$$= \begin{bmatrix}
  c_\varphi c_\theta c_\psi - s_\varphi s_\psi & -c_\varphi c_\theta s_\psi - s_\varphi c_\psi & c_\varphi s_\theta \\
  s_\varphi c_\theta c_\psi + c_\varphi s_\psi & -s_\varphi c_\theta s_\psi + c_\varphi c_\psi & s_\varphi s_\theta \\
  -s_\theta c_\psi & s_\theta s_\psi & c_\theta
\end{bmatrix}$$

The inverse problem is very useful (e.g. $\varphi = f(R_{zyz})$)

Any idea?
Euler Angles: RPY Angles

RPY stands for Roll, Pitch and Yaw angles

\[
\begin{bmatrix}
\varphi \\
\theta \\
\psi
\end{bmatrix}^T
\]

In Euler Angles, the rotations are made \textbf{wrt the fixed reference frame}.

The rotation described by the RPY angles is:

+ Rotate the reference frame by \( \psi \) about axis \( x \) (yaw), i.e., \( R_x(\psi) \)
+ Rotate the reference frame by \( \theta \) about axis \( y \) (pitch), i.e., \( R_y(\theta) \)
+ Rotate the reference frame by \( \varphi \) about axis \( z \) (roll), i.e., \( R_z(\varphi) \)

\[
R(\phi) = R_z(\varphi) R_y(\theta) R_x(\psi)
\]

(Note the \textbf{inverted order} for rotations about reference frame)
In the figure, a rotation and a translation separate \( \{C\} \) from \( \{W\} \).

**GOAL:**

\[
C_X = f(W_X)
\]

On the basis of simple geometry:

\[
C_X = C_W t + C_W R_W X
\]

If we consider the 2-D case we have:

\[
\begin{bmatrix}
X_C \\
Y_C \\
1
\end{bmatrix} = \begin{bmatrix}
t_x \\
t_y
\end{bmatrix} + \begin{bmatrix}
c_\theta & -s_\theta \\
s_\theta & c_\theta
\end{bmatrix} \begin{bmatrix}
X_W \\
Y_W
\end{bmatrix}
\]

Which, rewriting the points in **homogeneous coordinates**, it leads to an expression linear in point coordinates

\[
C \tilde{X} = C_W H W \tilde{X}
\]

Homogeneous coordinate transformation
Suppose given a mobile robot \{R\} with a mounted-on camera. The camera \{C\} is displaced at \( R_{C} X = [0, \ell]^T \) (wrt \{R\}).

The pose (position and orientation) of the robot with respect to \{W\} is \( W_{R} X = [X_R, Y_R]^T \) and \( \phi \)

**GOAL:** Compute \( W_{C} X \)

**Pair!**
The rigid transformation between two frames can be expressed in **homogeneous form** as:

\[
\begin{bmatrix}
C \tilde{X} = C \tilde{W} \tilde{X}
\end{bmatrix}
\]

where

\[
C \tilde{W} H = \begin{bmatrix}
C R & C t \\
0^T & 1
\end{bmatrix}
\]

Suppose we want to compute the **inverse homogeneous transform** from \{W\} to \{C\}, i.e.:

\[
\begin{bmatrix}
W \hat{H} = \begin{bmatrix}
W R & W t \\
0^T & 1
\end{bmatrix}
\end{bmatrix}
\]

Thus showing that

\[
C \hat{W} H^{-1} \neq C \hat{W} H^T
\]

\[
C \tilde{X} = C \tilde{W} t + C \tilde{W} R \tilde{X} \quad \Rightarrow \quad W \tilde{X} = \frac{C}{W} R^T C \tilde{X} - \hat{C} \hat{W} R^T \hat{C} \tilde{W} t
\]
Our Goal: Generalized Pinhole Model

Rewrite the pinhole camera model as:

\[ s \tilde{u} = K \Pi_0 C \tilde{X} \quad \text{such that} \quad C X = f(W X) \]

The generic rigid motion transformation is given by:

\[ s \tilde{u} = K \Pi_0 C W H W \tilde{X} \]

Intrinsic parameters

\[ s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f k_u & 0 & u_0 \\ 0 & f k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} \]
General Pinhole Model: EGT example

See Ex4_2DRotations.m
Full Stereo-Camera Model

Stereo cameras allow to perceive the **depth** of an object.

We need a **mathematical model** of this camera to compute the **3D** reconstruction from the **left and right images**.

**Assumptions:**
+ Correspondences are known;
+ Extrinsic parameters are known;

**Triangulate** the two correspondences (measurements) to obtain the 3D point.
Stereo-Camera Triangulation: noise-free case

The 3D point $\ell X$ can be written as:

$$\ell X = \lambda_\ell \ell x$$

and analogously for $r X$

Note that:

$$\ell x = K^{-1} \tilde u_\ell$$

(The last coordinate of $\ell x$ must be $= f$)

(i) $\lambda_\ell \ell x - \ell t = \ell R(\lambda_r r x) \Rightarrow \ell x \times (\lambda_\ell \ell x - \ell t) = \ell x \times (\ell R r x \lambda_r)$

Cross-Product

$$-\ell x \times \ell t = (\ell x \times \ell R r x) \lambda_r$$

From which we have: $\lambda_r = \frac{a^T b}{a^T a}$, and $\lambda_\ell$ is easily obtained from (i)

Even though pretty elegant, the above solution is not useful when the image points are affected by noise, or there is error in the extrinsic camera parameters. Why?

Gian Luca Mariottini, Univ. Texas at Arlington
Stereo-Camera Triangulation: noisy case

In case of **noise** in the image the two lines won't meet anymore at a point!

**GOAL:** compute the midpoint along the shortest line $n$ that connects $\lambda_\ell \ell x$ and $\lambda_r r x$

The following holds true: $\lambda_\ell \ell x + \lambda_n n = \lambda_r r R^r x + r t$ \(i\)

where $n = \ell x \times r R^r x$, i.e., the normal vector of the plane formed by $r x$ and $\ell x$

Analogously to before, we can obtain a closed form solution for $\lambda_n$ as:

$$\lambda_n = (a^T b)/(b^T b)$$

such that $b \triangleq (\ell x \times r R^r x)^T r t$ and $a \triangleq (\ell x \times r R^r x)^T (\ell x \times r R^r x)$

$\lambda_n$ can be substituted in (i) and, similarly to prev. slide, $\lambda_r$ and $\lambda_\ell$ can be computed.
Each camera (left and right) obeys to the pinhole model

\[
\begin{bmatrix}
    u \\
    v \\
    1
\end{bmatrix} =
\begin{bmatrix}
f k_u & 0 & u_0 \\
0 & f k_v & v_0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    C \cdot R \\
    C \cdot t \\
    1
\end{bmatrix}
\]

\[
s_{\ell} \tilde{u}_{\ell} = W^\ell P \tilde{X}_W \quad \text{(i)}
\]

\[
s_{r} \tilde{u}_{r} = W^r P \tilde{X}_W \quad \text{(ii)}
\]

**GOAL:** Compute the 3-D point \(X_W\) for a given pair \((u_r, u_\ell)\) and known camera parameters \(W^\ell P\) and \(W^r P\).

By taking the cross product of (i) and (ii) with \(\tilde{u}_{\ell}\) and \(\tilde{u}_{r}\), respectively

\[
0 = \tilde{u}_{\ell} \times s_{\ell} \tilde{u}_{\ell} = \tilde{u}_{\ell} \times W^\ell P \tilde{X}_W
\]

\[
0 = [\tilde{u}_{\ell}] \times W^\ell P \tilde{X}_W
\]

Remember that:

\[
a \times b = [a] \times b =
\begin{bmatrix}
    0 & -a_3 & a_2 \\
    a_3 & 0 & -a_1 \\
    -a_2 & a_1 & 0
\end{bmatrix}
\begin{bmatrix}
b_1 \\
b_2 \\
b_3
\end{bmatrix}
\]

"Skew-symmetric matrix"
... so that \( 0 = [\tilde{u}_\ell] \times W^\ell P \tilde{X}_W \) can be written as:

\[
0 = \begin{bmatrix}
-\mathbf{P}_\ell(2,:)^T + v_\ell \mathbf{P}_\ell(3,:)^T \\
\mathbf{P}_\ell(1,:)^T - u_\ell \mathbf{P}_\ell(3,:)^T \\
-v_\ell \mathbf{P}_\ell(1,:)^T + u_\ell \mathbf{P}_\ell(2,:)^T
\end{bmatrix} \tilde{X}_W
\]

3rd row of \( W^\ell P \)

linear comb. of row-1 and row-2

Analogously for the right view: \( 0 = [\tilde{u}_r] \times W^\ell P \tilde{X}_W \)

In each case we use only the first two equations (each point provides two independent equations!) thus leading to:

\[
\begin{bmatrix}
-\mathbf{P}_\ell(2,:)^T + v_\ell \mathbf{P}_\ell(3,:)^T \\
\mathbf{P}_\ell(1,:)^T - u_\ell \mathbf{P}_\ell(3,:)^T \\
-\mathbf{P}_r(2,:)^T + v_r \mathbf{P}_r(3,:)^T \\
\mathbf{P}_r(1,:)^T - u_r \mathbf{P}_r(3,:)^T
\end{bmatrix} \tilde{X}_W = 0
\]

unknown

which we rewrite as:

\[
A_{4 \times 4} \tilde{X}_W = 0
\]

\[
\text{argmin} \|A \tilde{X}_W\|^2
\]

such that:

Gian Luca Mariottini, Univ. Texas at Arlington
For ease of notation let us *rewrite the Least Squares cost function* to be minimized as:

\[ J = \| A x \|^2 = x^T A^T A x \] (i)

Let us note that is always possible to decompose a matrix \( A \) via *Singular Value Decomposition (SVD)*

\[ A = U D V^T \] (ii)

where:
- \( A \) is an \( m \times n \) matrix;
- \( U \) is \( m \times n \), while \( V \) is \( n \times n \);
- \( U \) and \( V \) are orthonormal, i.e., \( U^T U = I \);
- \( D \) is a diagonal matrix with nonnegative real numbers (*singular values*) on the diagonal, in *decreasing order*

Try in MATLAB (see example `Ex_SVD.m`)

Substituting (ii) into (i) yields:

\[
J = x^T V D U^T U D V^T x = x^T V D^2 V^T x
\]

\[
J = \| A x \|^2 = \| D V^T x \|^2
\]

and, if we call \( y = V^T x \), then the original minimization problem becomes:

\[
\text{argmin}_{x: \| x \|=1} \| A x \|^2 = \text{argmin}_{y: \| y \|=1} \| D y \|^2 \Rightarrow y^* = [0, 0, \ldots, 1] \Rightarrow x^* = \text{"Last column of } V\text{"}
\]