Spherical Warping

Image warping

We did this using homographies.

We can view homographies as linear transformations if we consider that the planes are in 3D.

Project one image onto another image's view plan.

However, we cannot handle cases like this.

So, let's project onto a sphere.

Nice and separate!
Going from \((x,y)\) to \((\phi, \theta)\) space.

**Question:** These methods hinge on a single viewpoint. What if we have multiple?

Same on camera one

different on camera two

This is bad for panorama stitching, but great for depth estimation!

576 cam Mk II:

Too many locations contribute to one point on the sensor

576 cam Mk II:

Good! Realistic

876 cam Mk III (M)

Mathematically useful to us...
How do we get $x_{\text{img}}$ from $x_{\text{world}}$?

$$x_{\text{img}} = \begin{bmatrix} \text{Intrinsics} \end{bmatrix} \begin{bmatrix} \text{Projection} \end{bmatrix} \begin{bmatrix} \text{Extrinsics} \end{bmatrix} x_{\text{world}}$$

- focal length, pixel coord. system
- 3D to 2D
- camera pose $X_{\text{cam}}$ (in world)

Activity: HM Problem 1

HM Problem 2

Find $(x', y', z')$!

$$(x', y', z') = \left( \frac{x_w}{z_w}, \frac{y_w}{z_w}, -1 \right)$$
Let's express this using linear algebra

\[
\begin{bmatrix}
\frac{x_w}{2w} \\
y_w \\
\frac{z_w}{2w} \\
1
\end{bmatrix} \sim
\begin{bmatrix}
x_w \\
y_w \\
z_w
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_w \\
y_w \\
z_w
\end{bmatrix}
\]

True! But not at all useful... Boring!

\[(x', y', z') = \left( -\frac{f \cdot y_w}{z_w}, -\frac{f \cdot x_w}{z_w}, f \right)\]

\[
\begin{bmatrix}
\frac{f x_w}{z_w} \\
f y_w \\
\frac{f y_w}{z_w} \\
1
\end{bmatrix} \sim
\begin{bmatrix}
\frac{f x_w}{z_w} \\
f y_w \\
\frac{f y_w}{z_w} \\
z_w
\end{bmatrix} =
\begin{bmatrix}
f & 0 & 0 \\
0 & f & 0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_w \\
y_w \\
z_w
\end{bmatrix}
\]

Better! Version all of our projection & extrinsic matrix
Assume (for now) \( x_w \) is known (we don't, but things will work out)

\[
\frac{Z_{W}}{f} = \frac{x_w}{x_l}, \quad \frac{b - x_w}{f} = \frac{Z_{W}}{f}
\]

\[
x_w = \frac{Z_{W}x_l}{f}, \quad x_w = \frac{b - Z_{W}x_r}{f}
\]

Set these equal! Goodbye \( x_w \! \).  

\[
\frac{Z_{W}x_l}{f} = b + \frac{Z_{W}x_r}{f}
\]

\[
\frac{Z_{W}x_l - Z_{W}x_r}{f} = b
\]

\[
Z_{W}\left(\frac{x_l - x_r}{f}\right) = b
\]

\[
Z_{W} = \frac{fb}{x_l - x_r}
\]
This was, of course, for the simple case:

But what if we have cameras with different view planes?

We could do some painful 3D math, or...

We can project CR's view plane onto CL.
The nice thing and the hard thing:

We only need to look at rows to find matches:

Less work (good!)

But more chances to make mistakes (not good!)

So we need some criterion to find good matches...

The correspondence problem:

We need a way to assign a cost to some matching between two rows:

- SSD - Sum. Squared Difference
- SAD - Sum. Abs. Difference
- CC - Cross correlation...

If we have some pattern we want to find, if we use it as a filter in cross correlation, it should create "bright spots" (large numbers) in areas that are similar to it.

However, we want these characteristics to be consistent and comparable from one another... so we normalized.
Camera matrix

From last time!

$\begin{bmatrix}
F & 0 & c_x & 0 \\
0 & F & c_y & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
? \\
X_{world}
\end{bmatrix}
= 
\begin{bmatrix}
x \\
y \\
z \\
1
\end{bmatrix}$

Why the extra column? 3D Homogeneous coordinates.

$\begin{bmatrix}
c_x \\
c_y
\end{bmatrix}$

The principal point: The location of the optical axis in pixel coordinates.

Most of the time, this is just the center of the image $(\frac{w}{2}, \frac{h}{2})$
Example:

Intrinsic:

\[
\begin{bmatrix}
100 & 0 & 50 \\
0 & 100 & 50 \\
0 & 0 & 1
\end{bmatrix}
\]

Extrinsics: Our camera isn't nicely placed:

We want to be able to work returns to the camera... so we want \( M \), s.t.

\[
M \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}
\]

So, \( M = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \)