Scene Modeling for a Single View

...with a lot of slides stolen from Steve Seitz and David Brogan,

15-463: Computational Photography
Alexei Efros, CMU, Fall 2006
Breaking out of 2D

…now we are ready to break out of 2D

And enter the real world!
on to 3D…

Enough of images!

We want more of the plenoptic function

We want real 3D scene walk-throughs:
   Camera rotation
   Camera translation

Can we do it from a single photograph?
Camera rotations with homographies

Original image

St. Petersburg photo by A. Tikhonov

Virtual camera rotations
Camera translation

Does it work?

PP1

PP2

synthetic PP
Yes, with planar scene (or far away)

PP3 is a projection plane of both centers of projection, so we are OK!
So, what can we do here?

Model the scene as a set of planes!

Now, just need to find the orientations of these planes.
Some preliminaries: projective geometry
Silly Euclid: Trix are for kids!

Parallel lines???
The projective plane

Why do we need homogeneous coordinates?

- represent points at infinity, homographies, perspective projection, multi-view relationships

What is the geometric intuition?

- a point in the image is a *ray* in projective space
  - Each point \((x, y)\) on the plane is represented by a ray \((sx, sy, s)\)
    - all points on the ray are equivalent: \((x, y, 1) \cong (sx, sy, s)\)
Projective lines

What does a line in the image correspond to in projective space?

- A line is a *plane* of rays through origin
  - all rays $(x,y,z)$ satisfying: $ax + by + cz = 0$

  \[
  \begin{bmatrix}
  x \\
  y \\
  z
  \end{bmatrix}
  =
  \begin{bmatrix}
  a & b & c
  \end{bmatrix}
  \begin{bmatrix}
  l \\
  p
  \end{bmatrix}
  \]

- A line is also represented as a homogeneous 3-vector $l$
Point and line duality

- A line \( l \) is a homogeneous 3-vector
- It is \( \perp \) to every point (ray) \( p \) on the line: \( l \cdot p = 0 \)

What is the intersection of two lines \( l_1 \) and \( l_2 \)?
- \( p \) is \( \perp \) to \( l_1 \) and \( l_2 \) \( \Rightarrow \) \( p = l_1 \times l_2 \)

What is the line \( l \) spanned by rays \( p_1 \) and \( p_2 \)?
- \( l \) is \( \perp \) to \( p_1 \) and \( p_2 \) \( \Rightarrow \) \( l = p_1 \times p_2 \)
- \( l \) is the plane normal

Points and lines are *dual* in projective space
- given any formula, can switch the meanings of points and lines to get another formula
Ideal points and lines

Ideal point ("point at infinity")
- $p \cong (x, y, 0)$ – parallel to image plane
- It has infinite image coordinates

Ideal line
- $l \cong (0, 0, 1)$ – parallel to image plane
Vanishing points

Vanishing point
• projection of a point at infinity
• Caused by ideal line
Vanishing points (2D)
Vanishing points

Properties

• Any two parallel lines have the same vanishing point \( v \)
• The ray from \( C \) through \( v \) is parallel to the lines
• An image may have more than one vanishing point
Vanishing lines

Multiple Vanishing Points

• Any set of parallel lines on the plane define a vanishing point
• The union of all of these vanishing points is the \textit{horizon line}
  – also called \textit{vanishing line}
• Note that different planes define different vanishing lines
Vanishing lines

Multiple Vanishing Points

- Any set of parallel lines on the plane define a vanishing point
- The union of all of these vanishing points is the horizon line – also called vanishing line
- Note that different planes define different vanishing lines
Computing vanishing points

Properties

- \( P_\infty \) is a point at *infinity*, \( v \) is its projection
- They depend only on line *direction*
- Parallel lines \( P_0 + tD, P_1 + tD \) intersect at \( P_\infty \)

\[
P_t = \begin{bmatrix} P_x + tD_x \\ P_y + tD_y \\ P_z + tD_z \\ 1 \end{bmatrix} \approx \begin{bmatrix} P_x/t + D_x \\ P_y/t + D_y \\ P_z/t + D_z \\ 1/t \end{bmatrix} \quad t \rightarrow \infty \quad P_\infty \approx \begin{bmatrix} D_x \\ D_y \\ D_z \\ 0 \end{bmatrix}
\]

\( v = \Pi P_\infty \)
Computing vanishing lines

Properties

• \( I \) is intersection of horizontal plane through \( C \) with image plane
• Compute \( I \) from two sets of parallel lines on ground plane
• All points at same height as \( C \) project to \( I \)
  – points higher than \( C \) project above \( I \)
• Provides way of comparing height of objects in the scene
Fun with vanishing points
“Tour into the Picture” (SIGGRAPH ’97)

Create a 3D “theatre stage” of five billboards

Specify foreground objects through bounding polygons

Use camera transformations to navigate through the scene
The idea

Many scenes (especially paintings), can be represented as an axis-aligned box volume (i.e. a stage)

Key assumptions:
  • All walls of volume are orthogonal
  • Camera view plane is parallel to back of volume
  • Camera up is normal to volume bottom

How many vanishing points does the box have?
  • Three, but two at infinity
  • Single-point perspective

Can use the vanishing point to fit the box to the particular Scene!
Fitting the box volume

User controls the inner box and the vanishing point placement (# of DOF???)

Q: What’s the significance of the vanishing point location?
A: It’s at eye level: ray from COP to VP is perpendicular to image plane. Why?
Example of user input: vanishing point and back face of view volume are defined.
Example of user input: vanishing point and back face of view volume are defined
Example of user input: vanishing point and back face of view volume are defined
Example of user input: vanishing point and back face of view volume are defined
Comparison of how image is subdivided based on two different camera positions. You should see how moving the vanishing point corresponds to moving the eyepoint in the 3D world.

High Camera

Low Camera
Another example of user input: vanishing point and back face of view volume are defined.
Another example of user input: vanishing point and back face of view volume are defined.
Another example of user input: vanishing point and back face of view volume are defined.
Another example of user input: vanishing point and back face of view volume are defined.
Comparison of two camera placements – left and right. Corresponding subdivisions match view you would see if you looked down a hallway.
2D to 3D conversion

First, we can get ratios

back plane

left

right

top

vanishing point

bottom
2D to 3D conversion

- Size of user-defined back plane must equal size of camera plane (orthogonal sides)
- Use top versus side ratio to determine relative height and width dimensions of box
- Left/right and top/bot ratios determine part of 3D camera placement
Depth of the box

Can compute by similar triangles (CVA vs. CV’A’)
Need to know focal length f (or FOV)

Note: can compute position on any object on the ground
  • Simple unprojection
  • What about things off the ground?
DEMO

Now, we know the 3D geometry of the box
We can texture-map the box walls with texture from the image
Foreground Objects

Use separate billboard for each

For this to work, three separate images used:

- Original image.
- Mask to isolate desired foreground images.
- Background with objects removed
Foreground Objects

Add vertical rectangles for each foreground object.

Can compute 3D coordinates $P_0$, $P_1$ since they are on known plane.

$P_2$, $P_3$ can be computed as before (similar triangles)
Foreground DEMO (and video)