

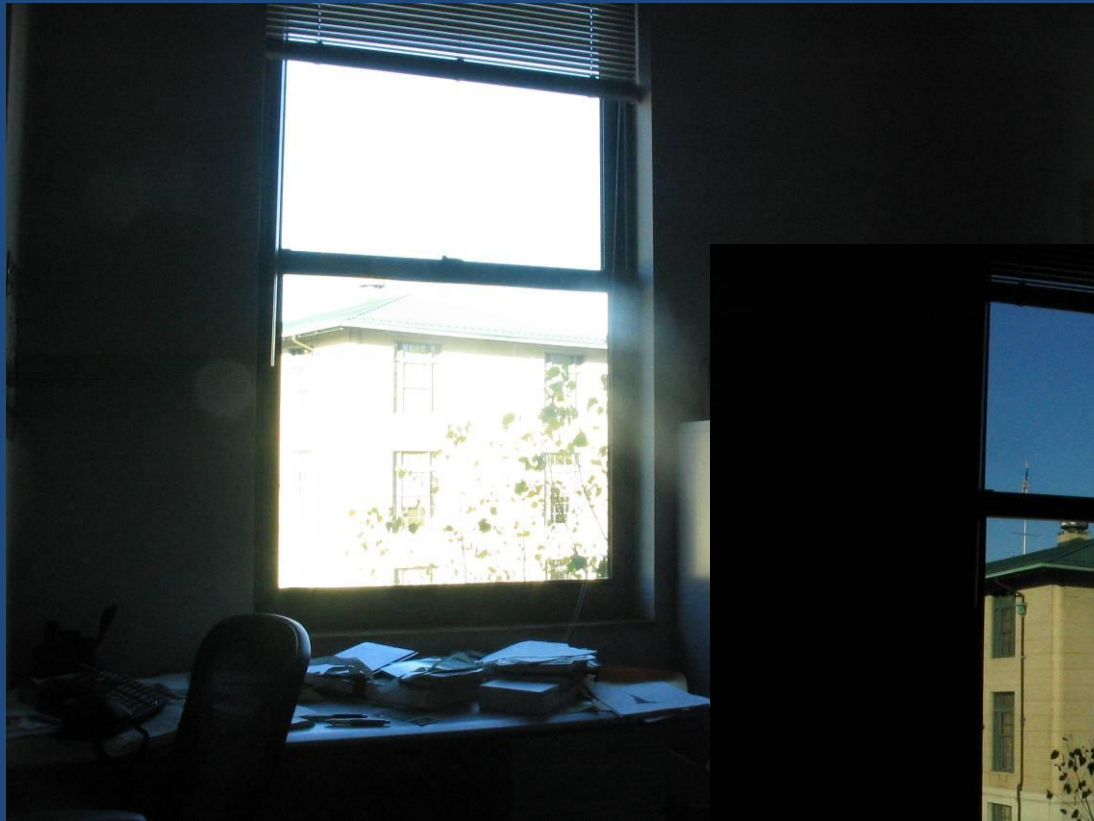


Computational Photography and Video: Video Synthesis I

Prof. Marc Pollefeys

Dr. Gabriel Brostow

Last Week: HDR



Schedule	Computational Photography and Video	
20 Feb	Introduction to Computational Photography	
27 Feb	More on Cameras, Sensors and Color	Assignment 1: Color
5 Mar	Warping, morphing and mosaics	Assignment 2: Alignment
12 Mar	Image pyramids, Graphcuts	Assignment 3: Blending
19 Mar	Dynamic Range, HDR imaging, tone mapping	Assignment 4: HDR
26 Mar	<i>Easter holiday – no classes</i>	
2 Apr	Video Synthesis I	Project proposals
9 Apr	TBD	Papers
16 Apr	TBD	Papers
23 Apr	TBD	Papers
30 Apr	TBD	Project update
7 May	TBD	Papers
14 May	TBD	Papers
21 May	TBD	Papers
28 May	TBD	Final project presentation

Breaking out of 2D

- ...now we are ready to break out of 2D



But must we go to full 3D? 4D?



Today's schedule

- Tour Into the Picture¹
- Video Textures²

¹Slides borrowed from Alexei Efros, who built on Steve Seitz's and David Brogan's

²Slides from Arno Schoedl

on to 3D...

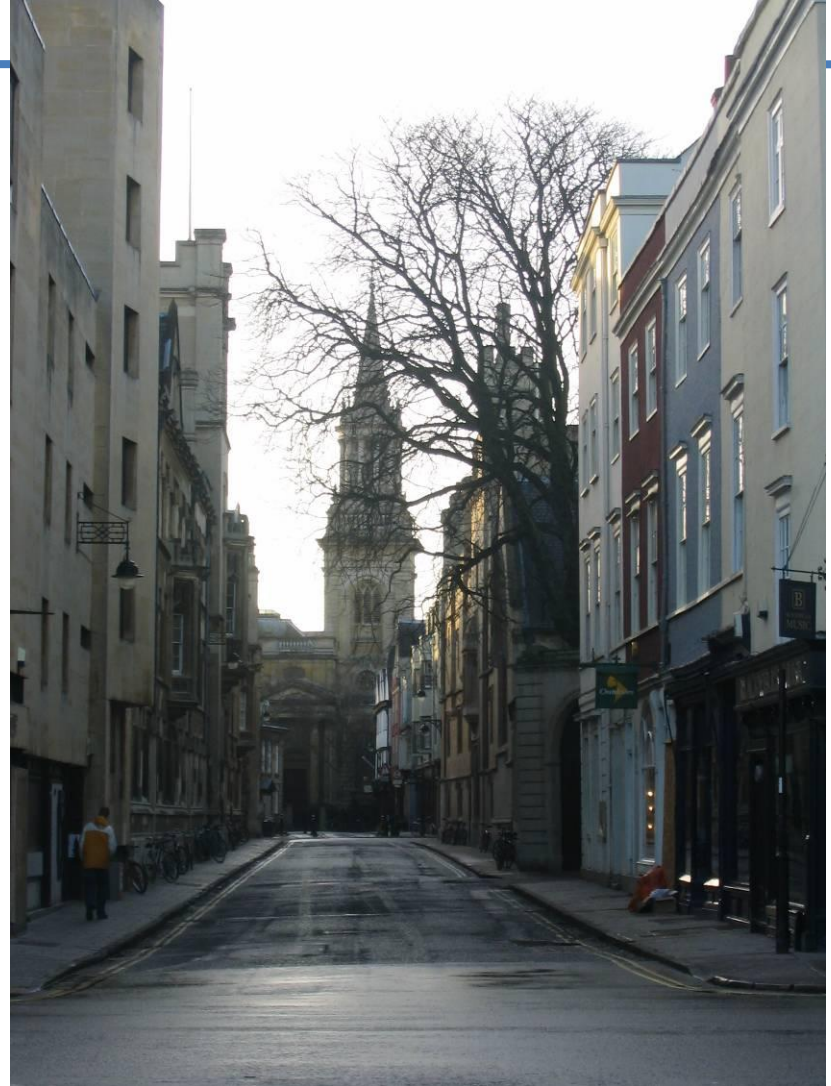
We want more of the
plenoptic function

We want real 3D scene
walk-throughs:

- Camera rotation

- Camera translation

Can we do it from a single
photograph?



Camera rotations with

Original image



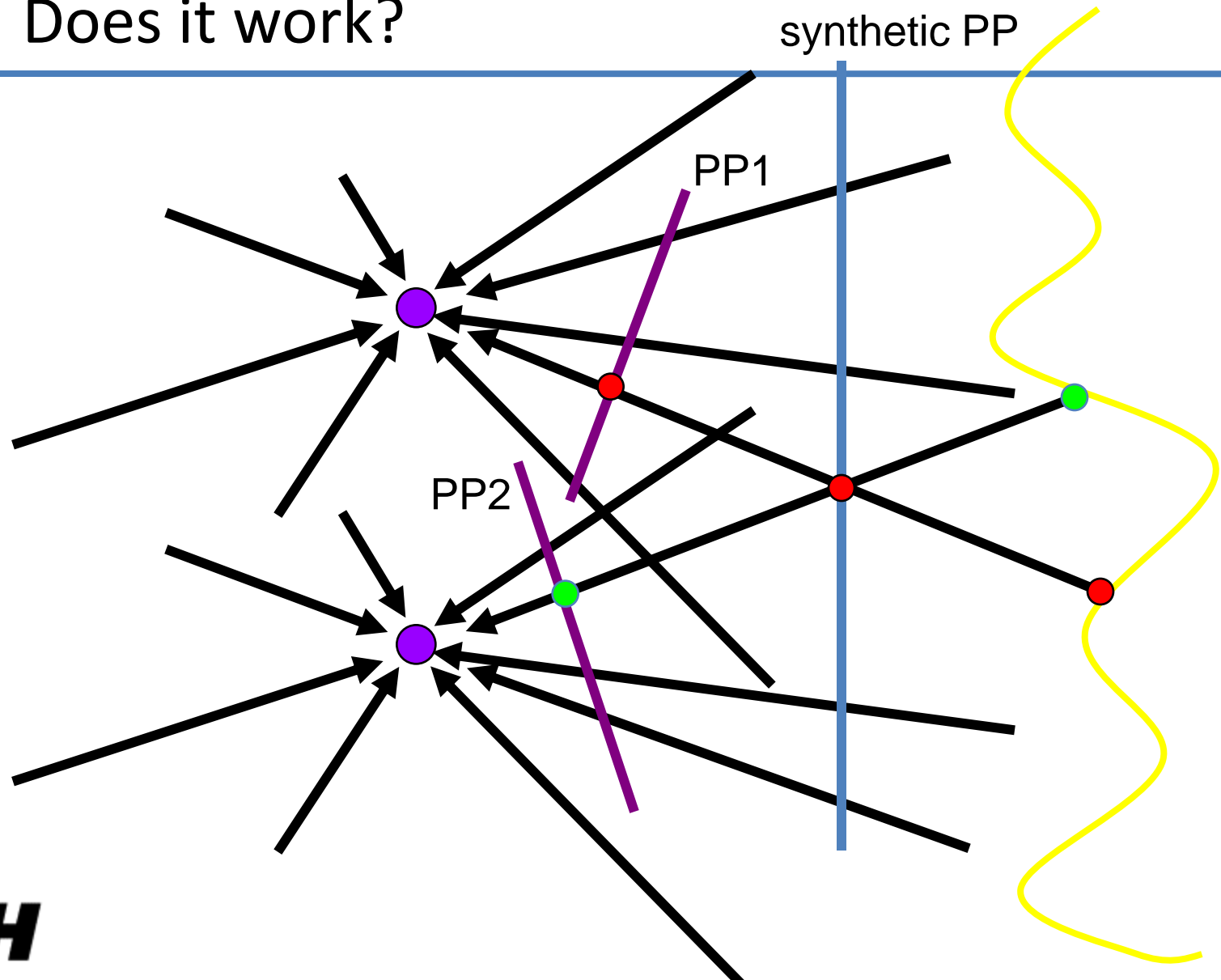
St.Petersburg
photo by A. Tikhonov

Virtual camera rotations

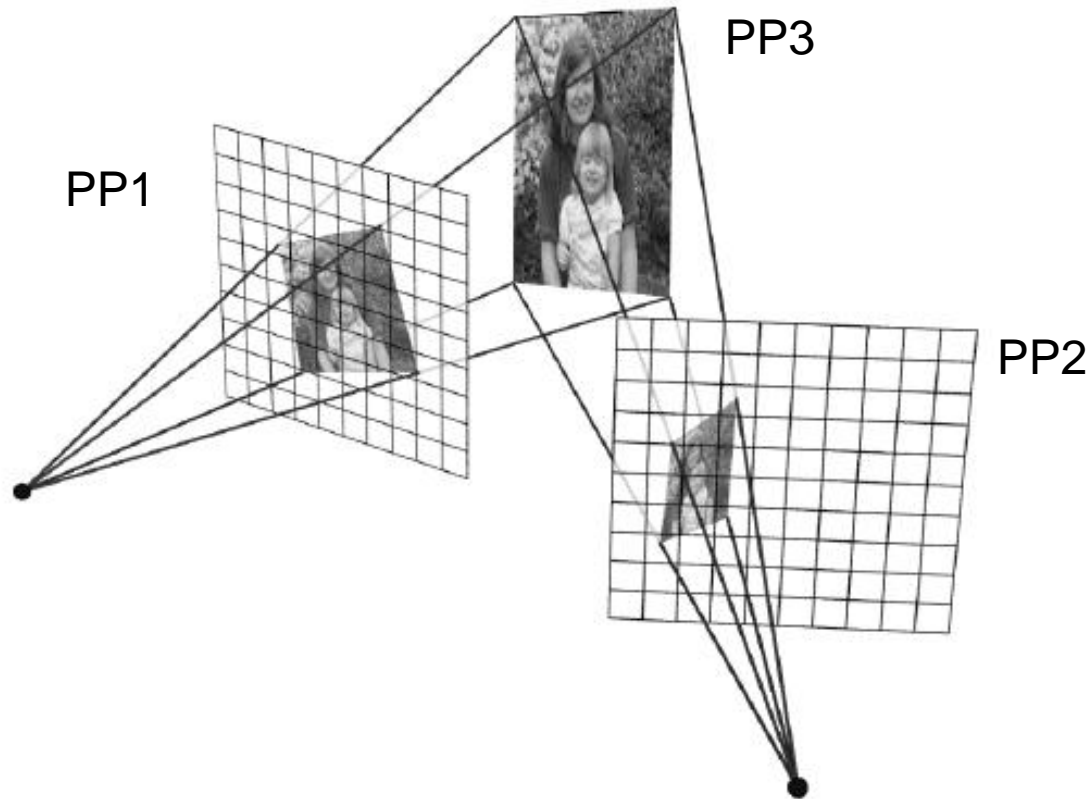


Camera translation

- Does it work?



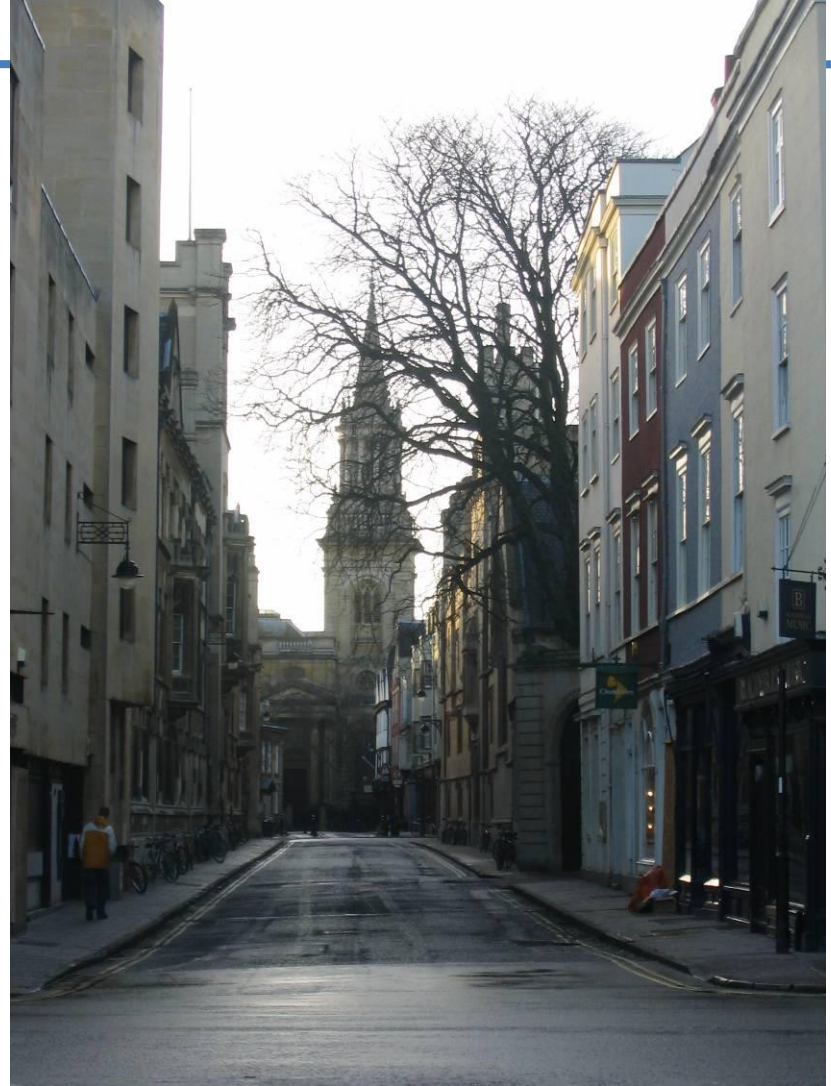
Yes, with planar scene (or far away)



- 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



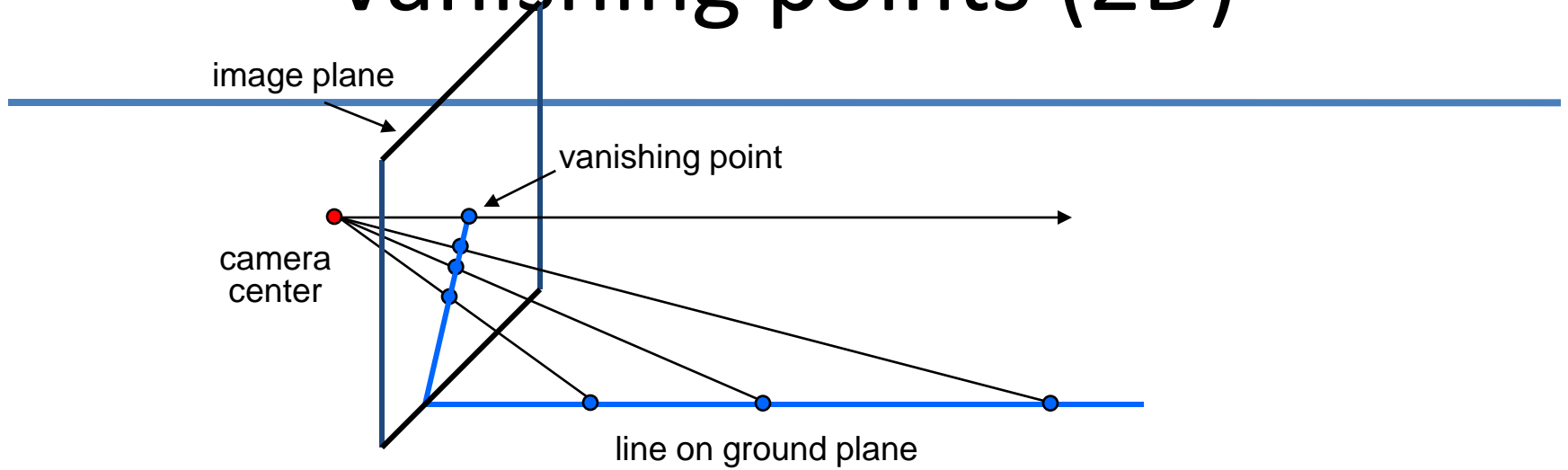
[Ames Room](#)

Silly Euclid: Trix are for kids!

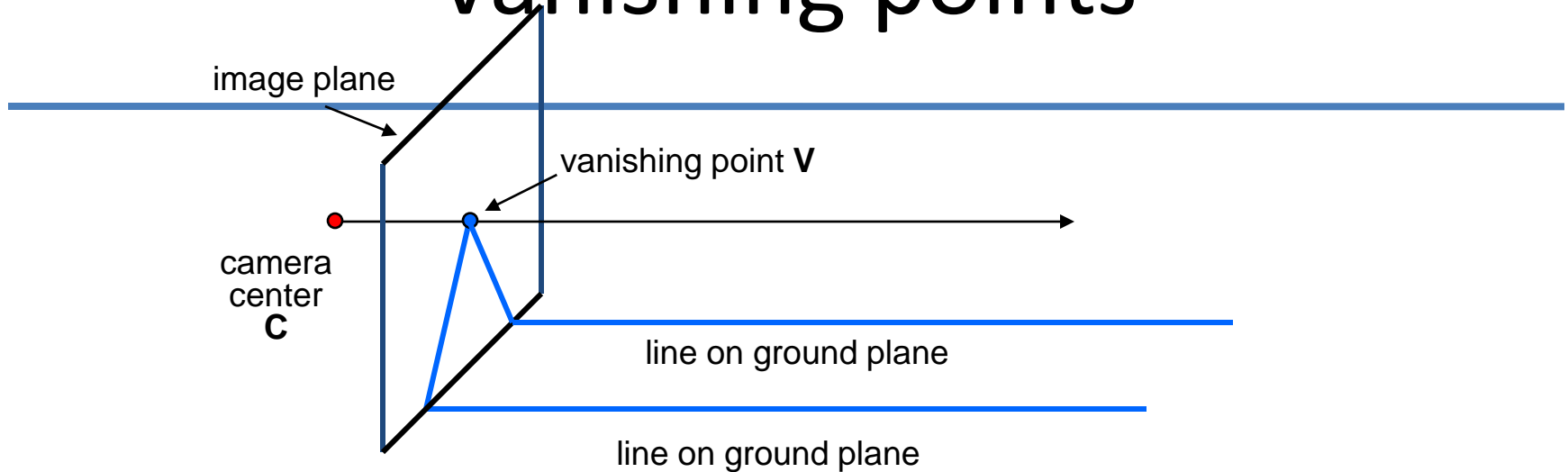


Parallel lines???

Vanishing points (2D)

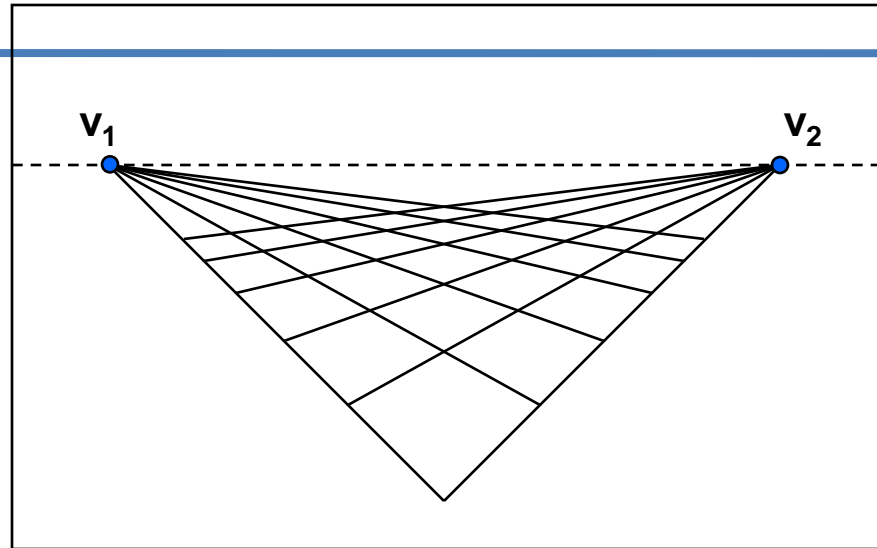


Vanishing points



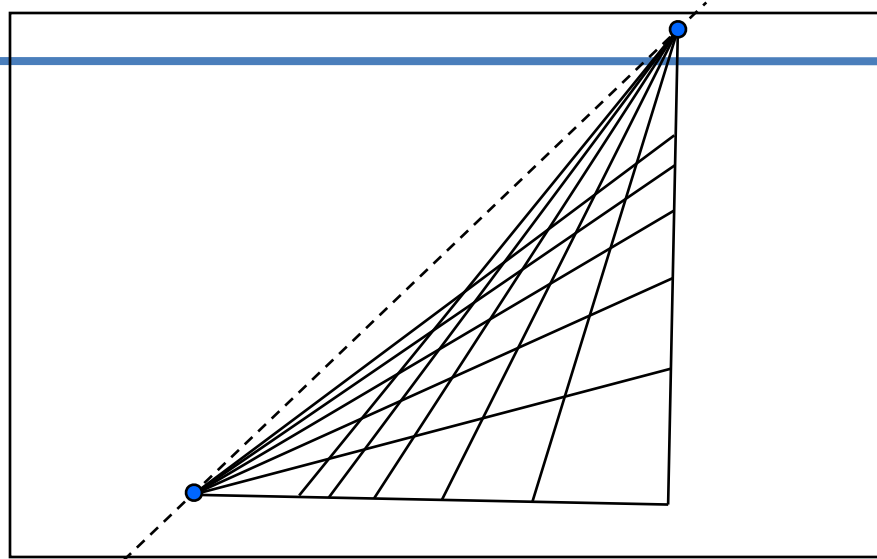
- Properties
 - Any two parallel lines have the same vanishing point v
 - The ray from C through v is parallel to the lines
 - An image may have more than one vanishing point

Vanishing lines



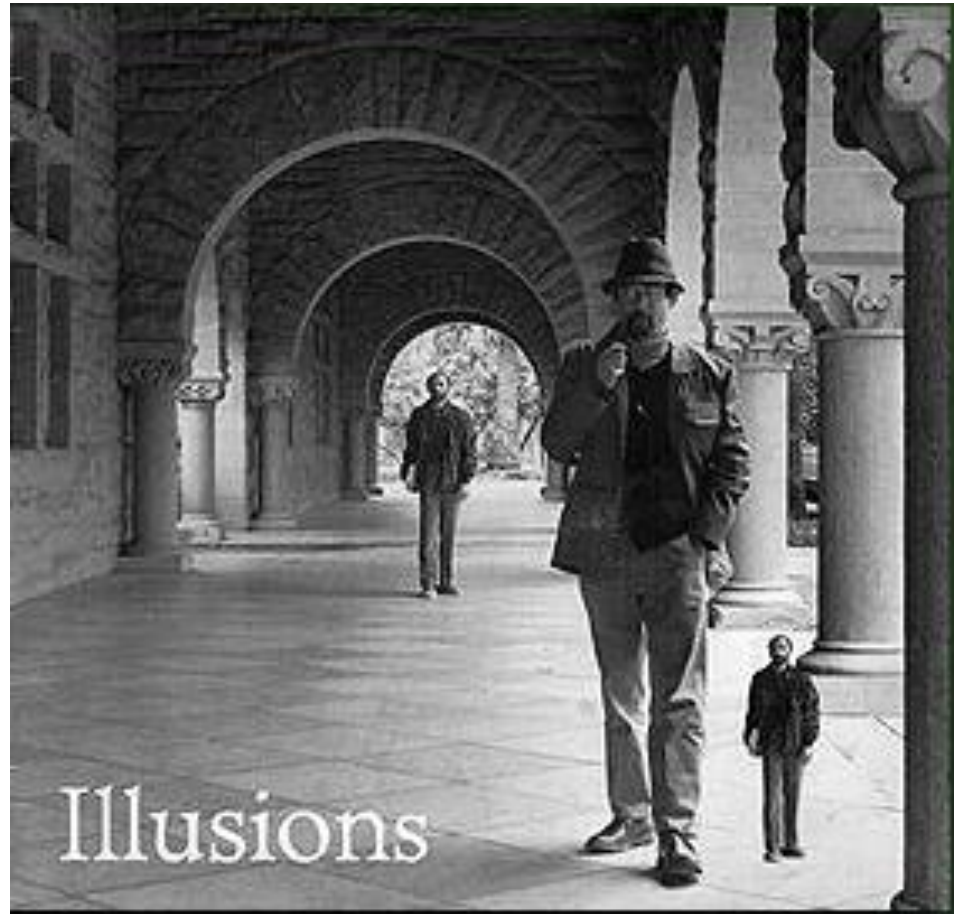
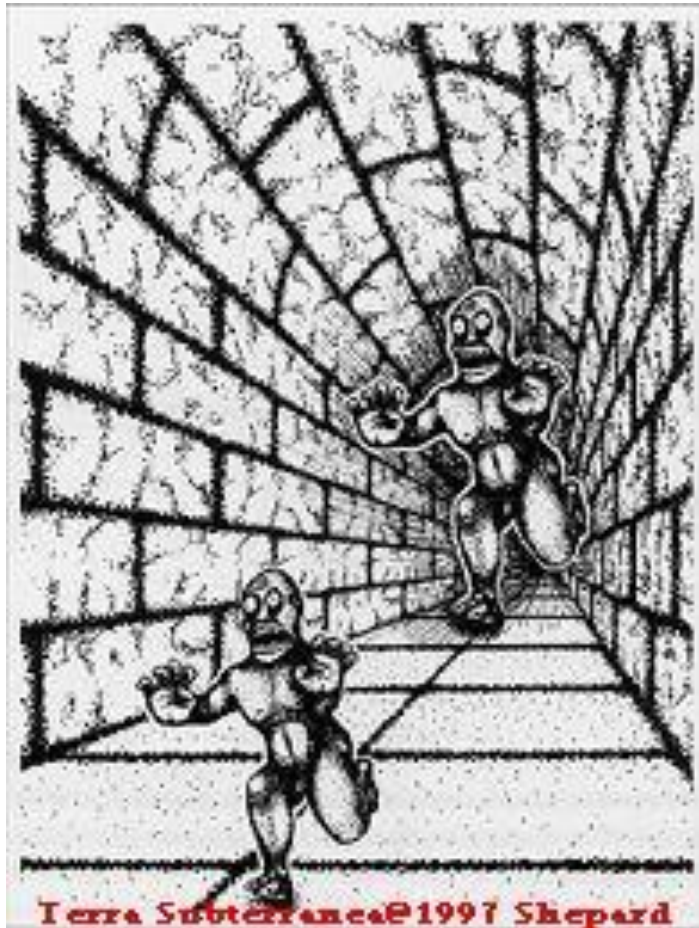
- Multiple Vanishing Points
 - Any set of parallel lines on the plane define a vanishing point
 - The union of all of these vanishing points is the *horizon line*
 - also called *vanishing line*
 - Note that different planes define different vanishing lines

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

Fun with vanishing points



“Tour into the Picture” (SIGGRAPH '97)

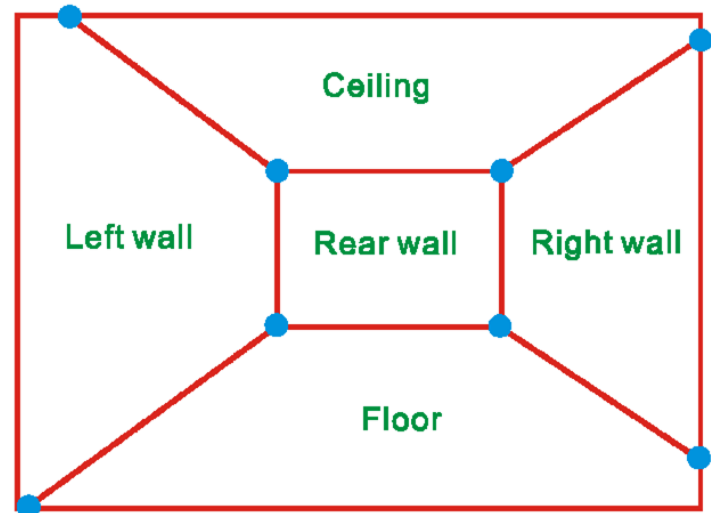
Horry, Anjyo, Arai

- Create a 3D “theatre stage” of five billboards
- Specify foreground objects through bounding polygons
- Use camera transformations to navigate through the scene

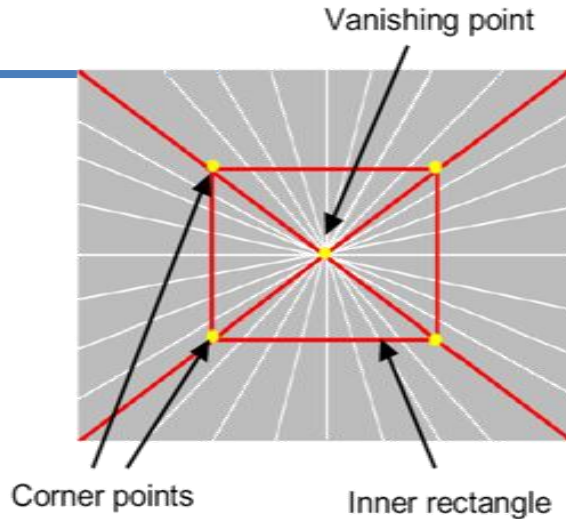


The idea

- 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!

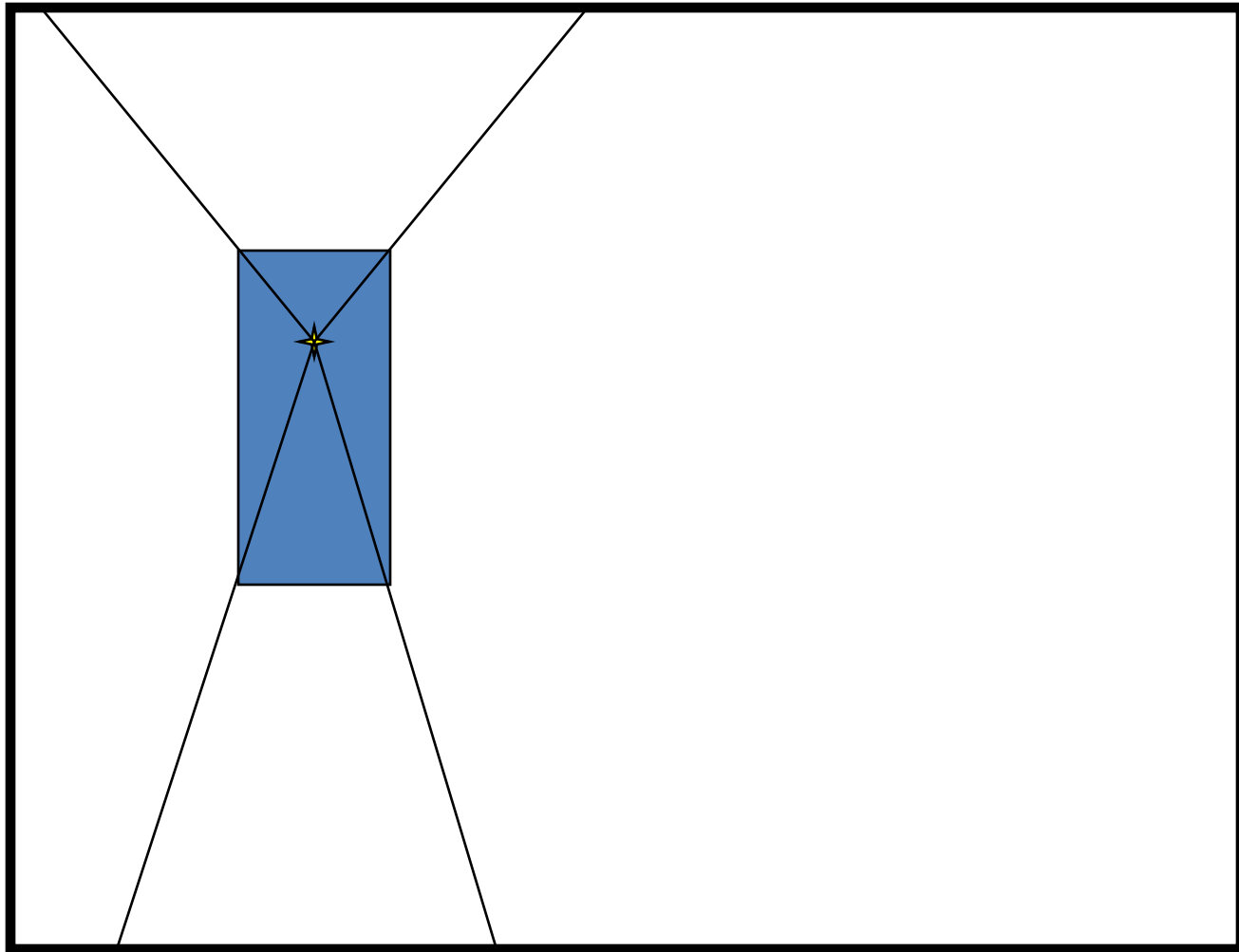


Fitting the box volume



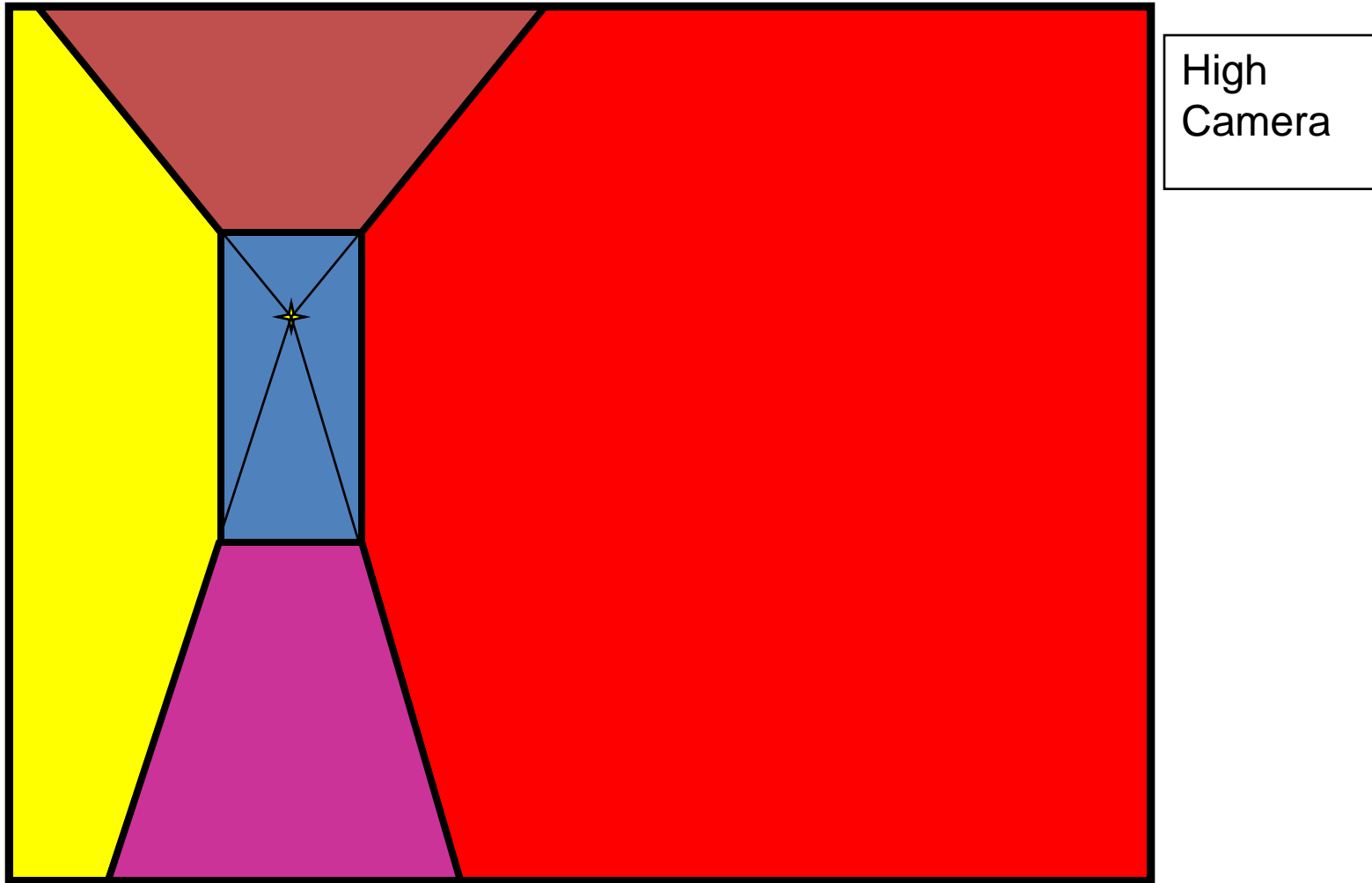
- User controls the inner box and the vanishing point placement (# of DOF???)
- Q: What's the significance of the vanishing point location?
- A: It's at eye level: ray from COP to VP is perpendicular to image plane.

Example of user input: vanishing point and back face of view volume are defined

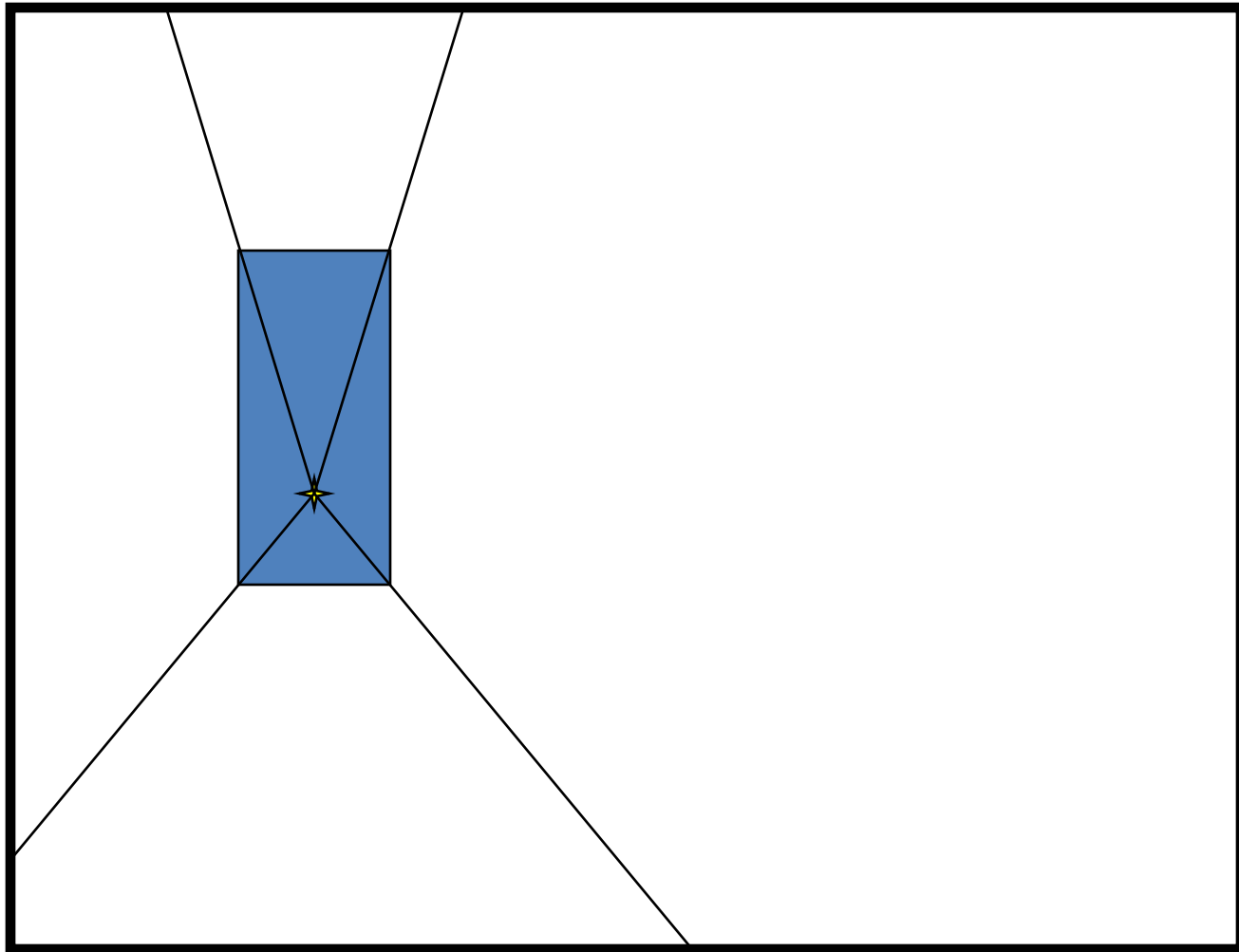


High
Camera

Example of user input: vanishing point and back face of view volume are defined

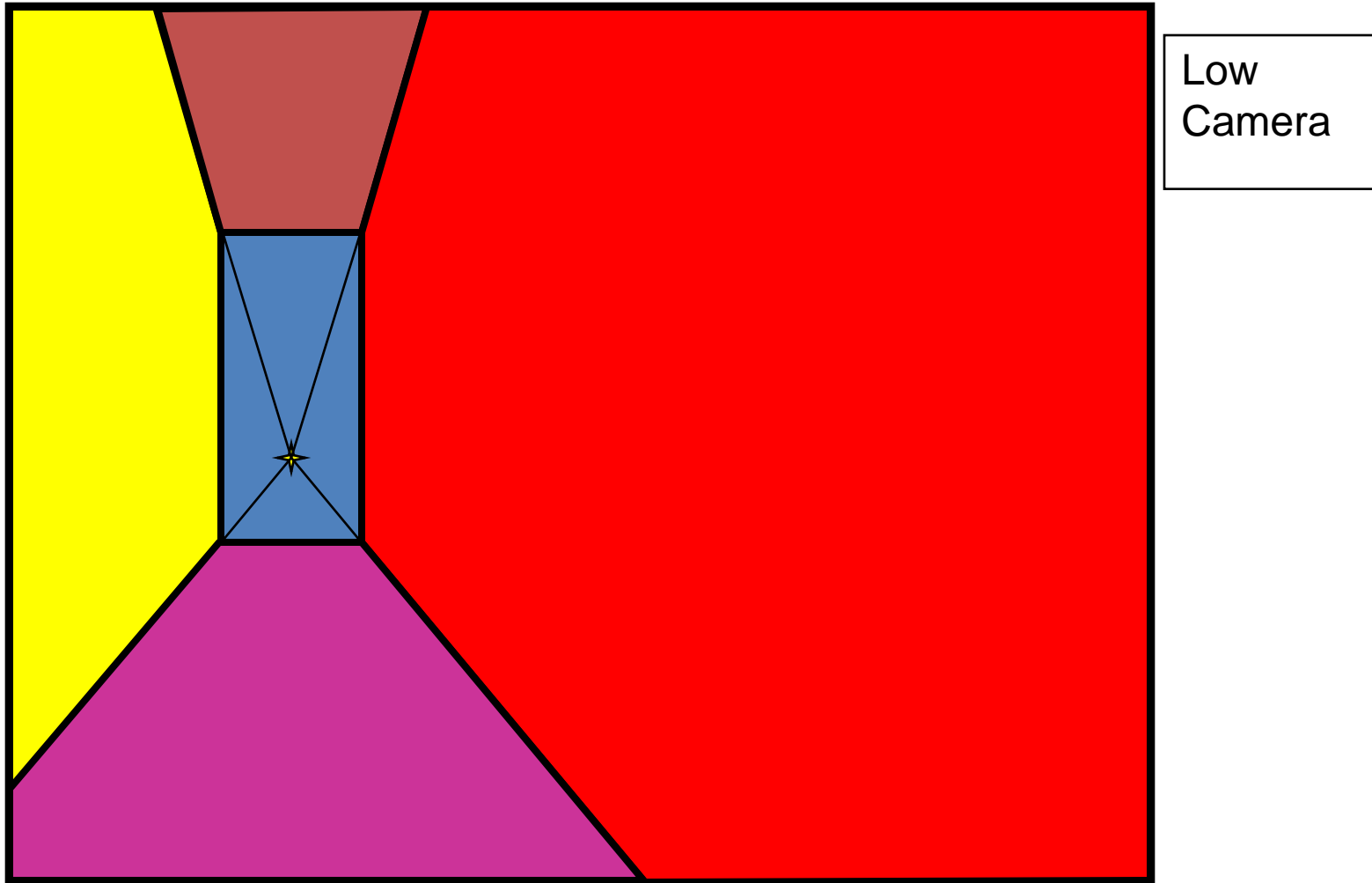


Example of user input: vanishing point and back face of view volume are defined

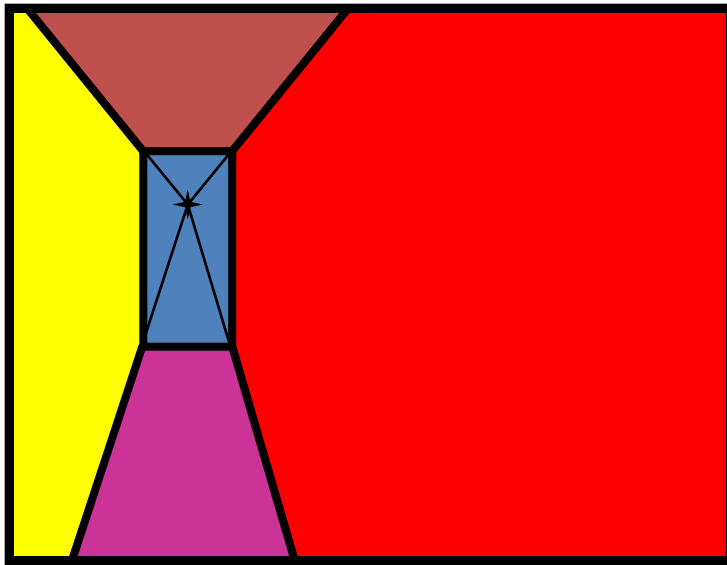


Low
Camera

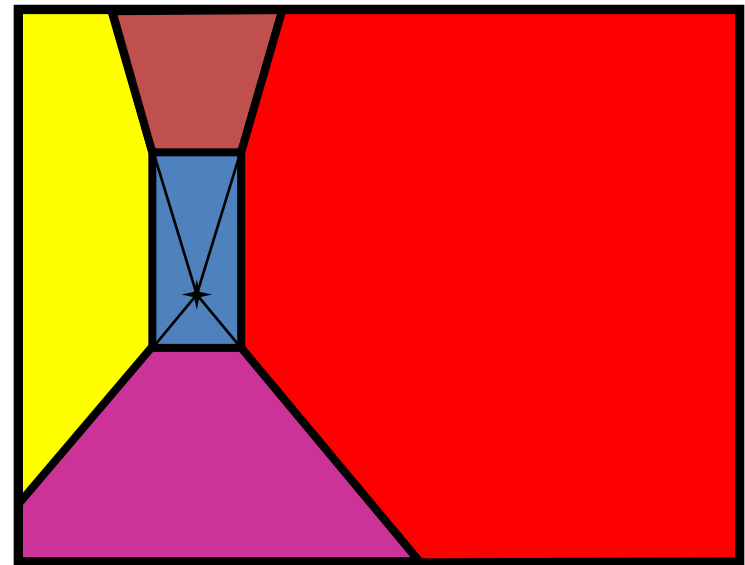
Example of user input: vanishing point and back face of view volume are defined



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.

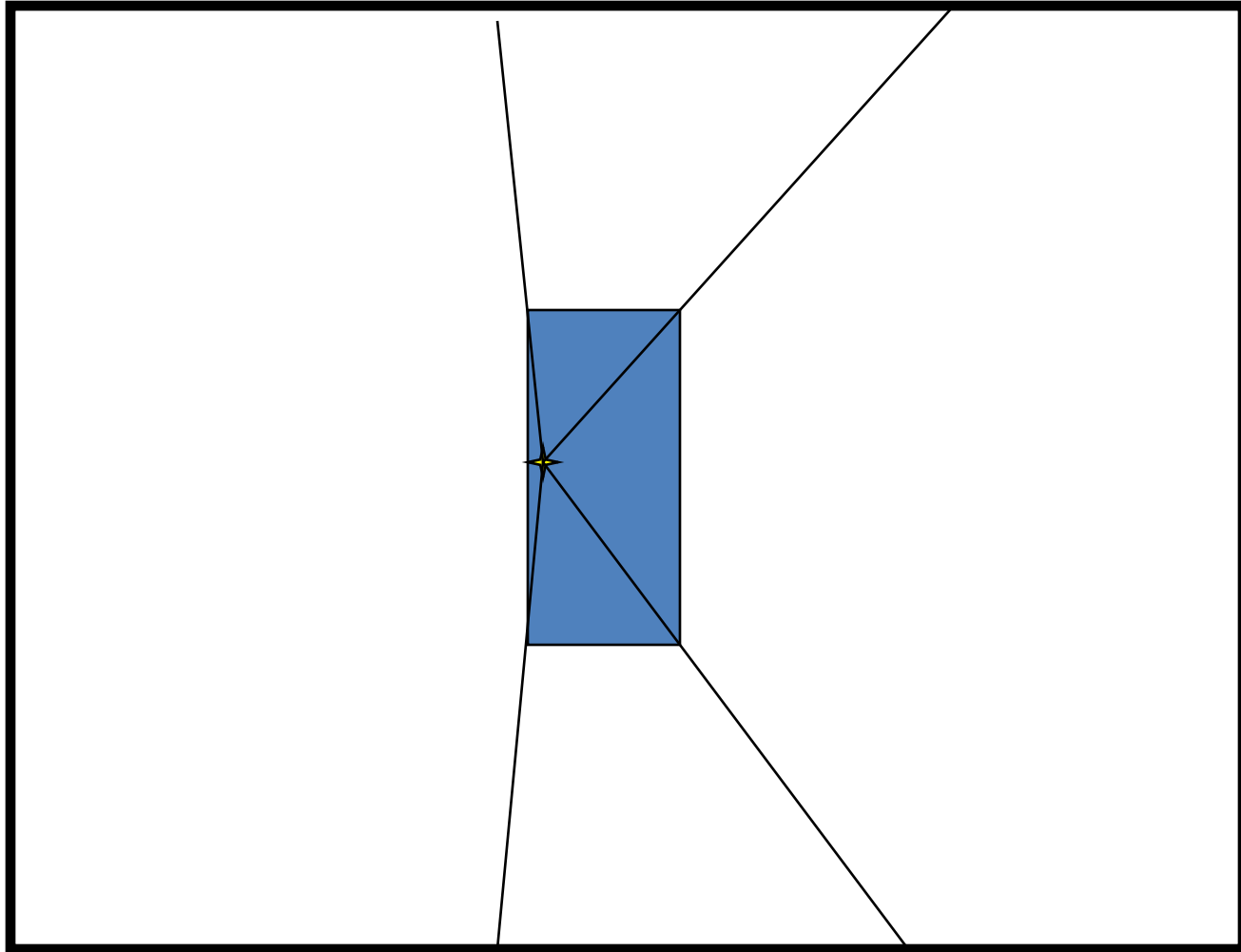


High Camera



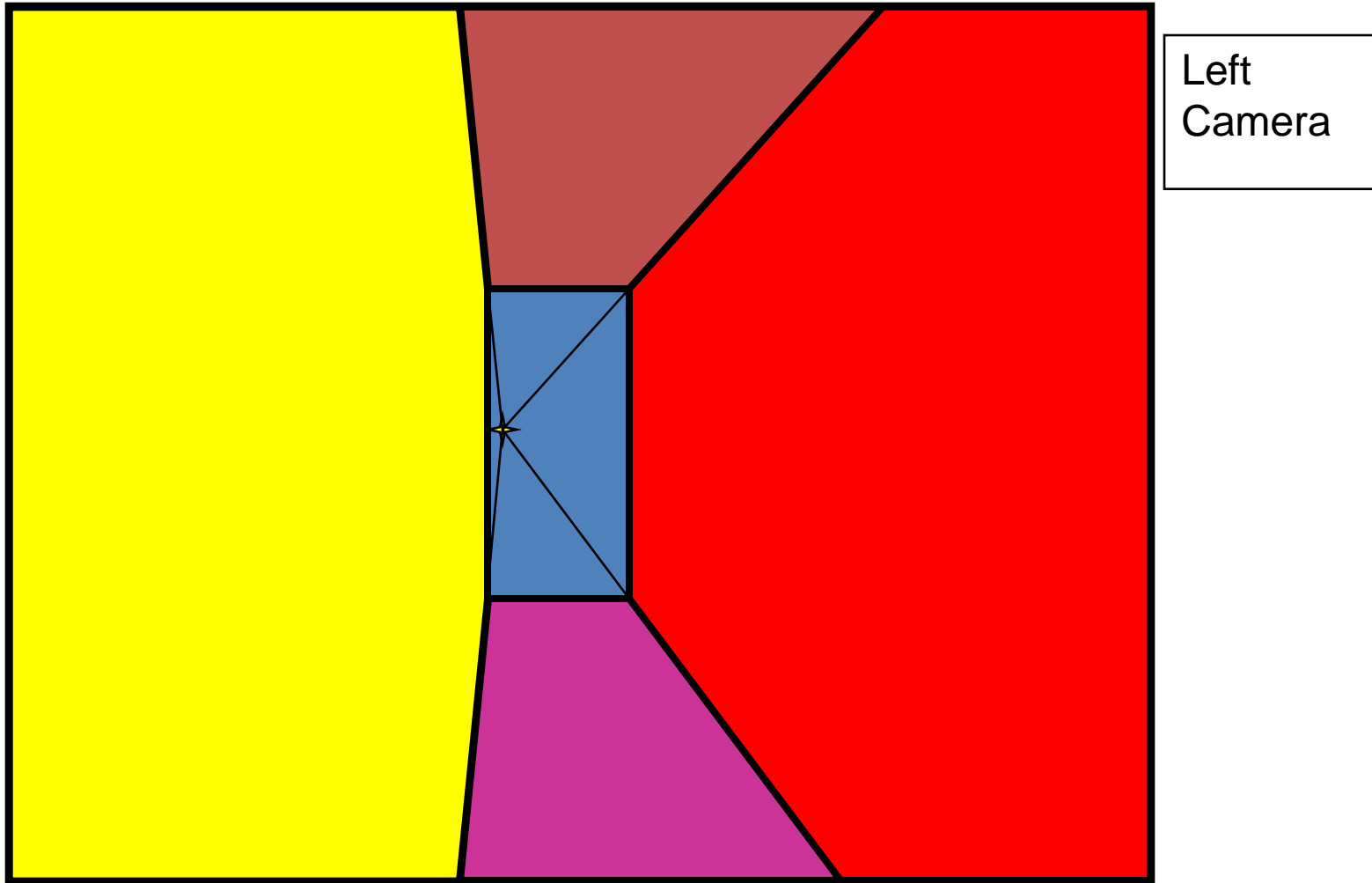
Low Camera

Another example of user input: vanishing point and back face of view volume are defined

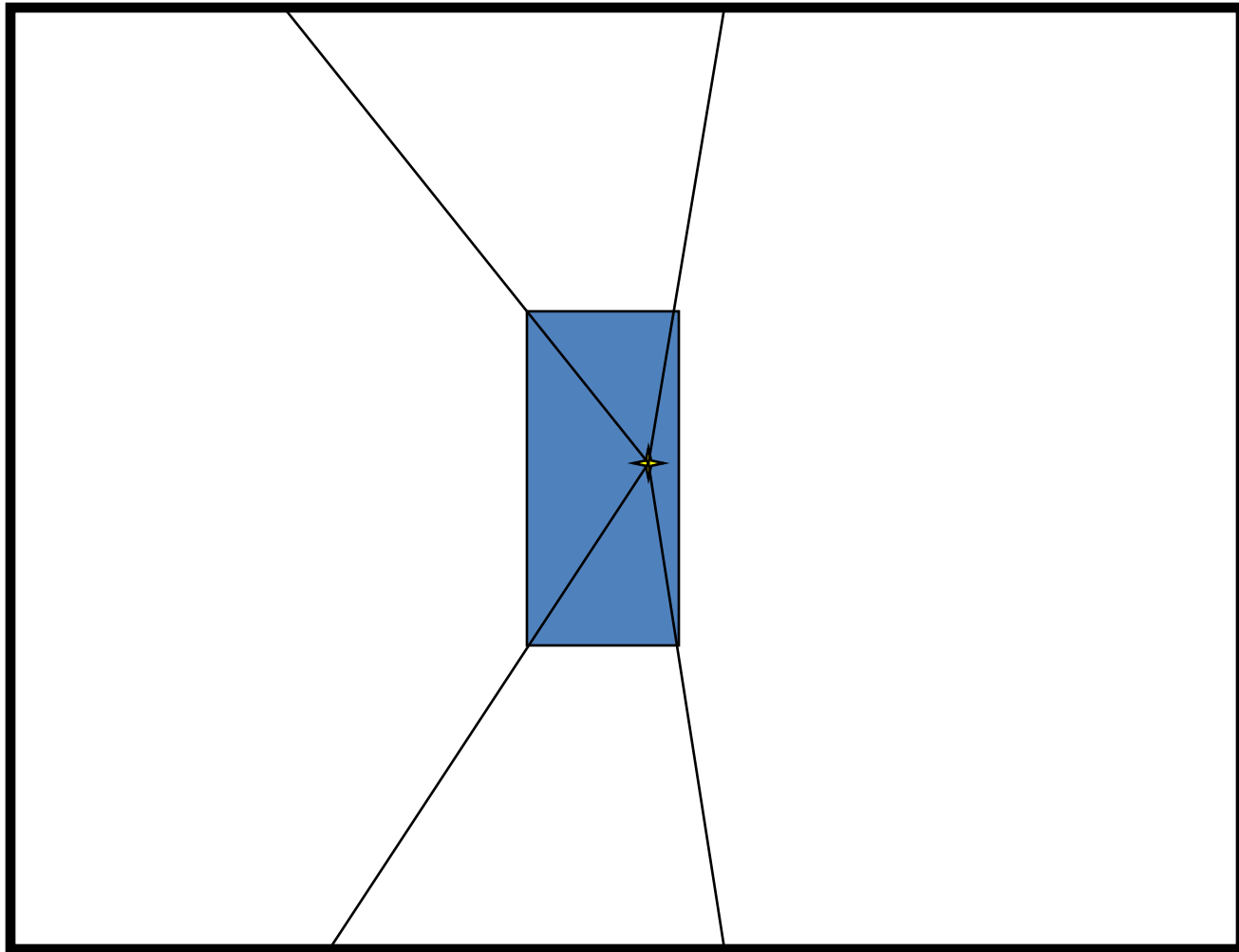


Left
Camera

Another example of user input: vanishing point and back face of view volume are defined

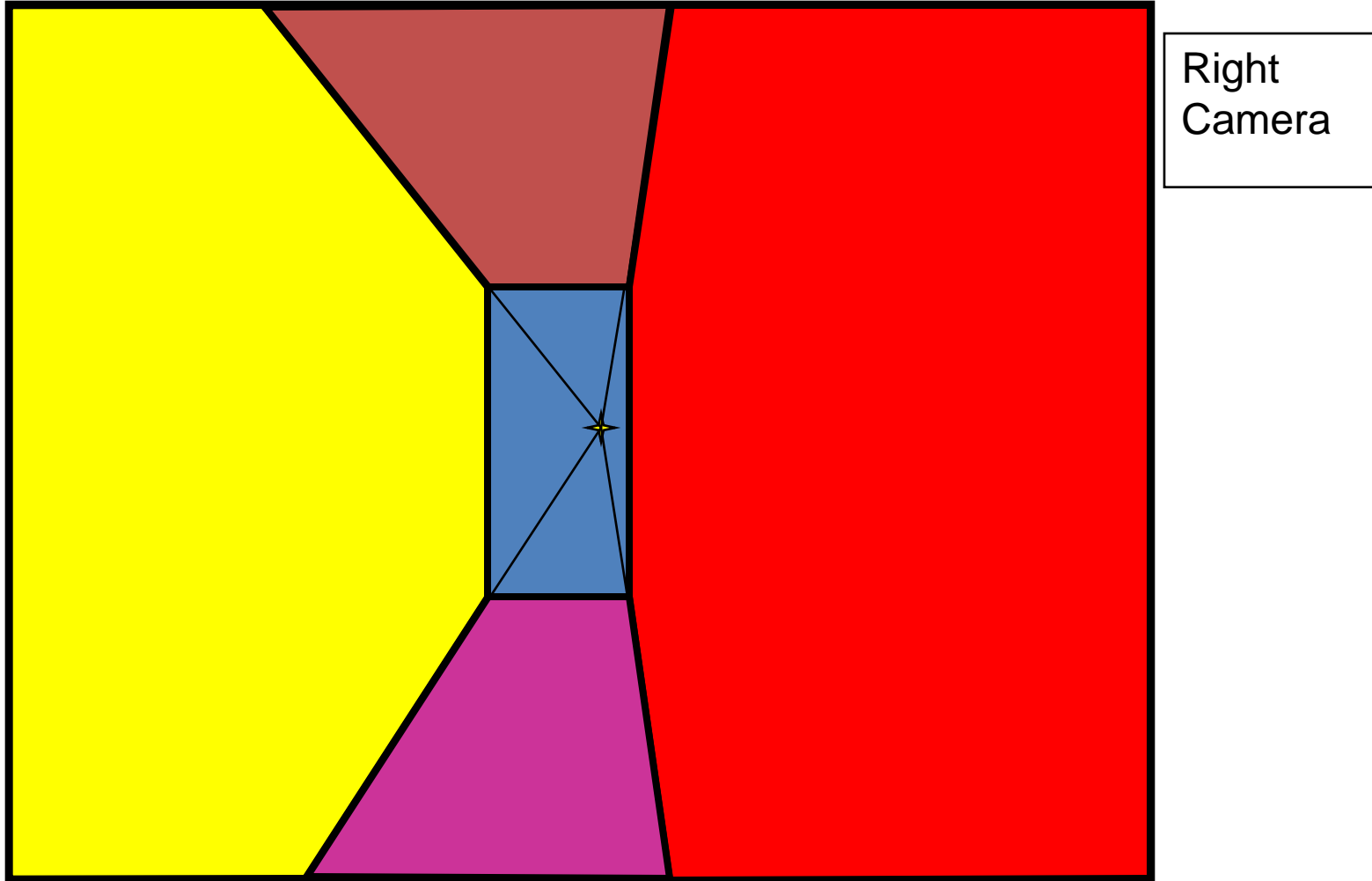


Another example of user input: vanishing point and back face of view volume are defined

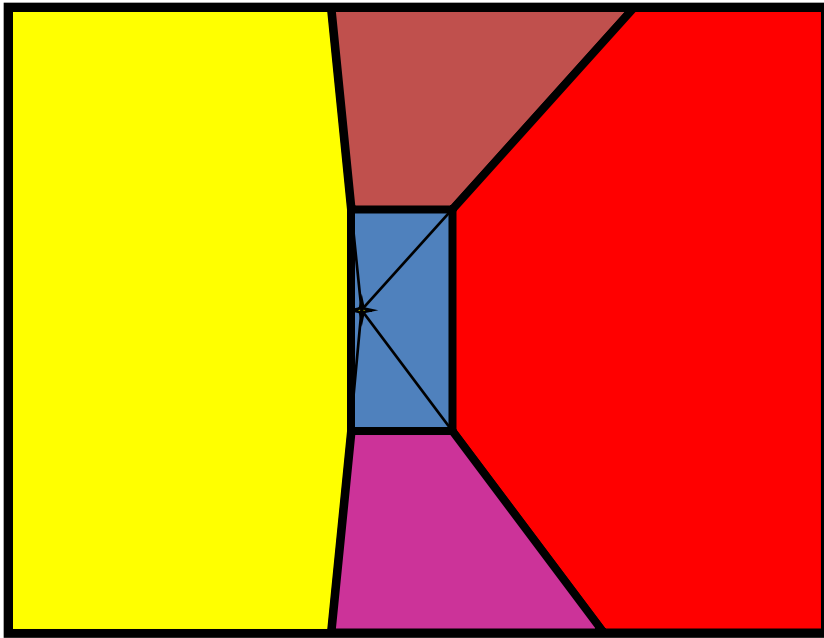


Right
Camera

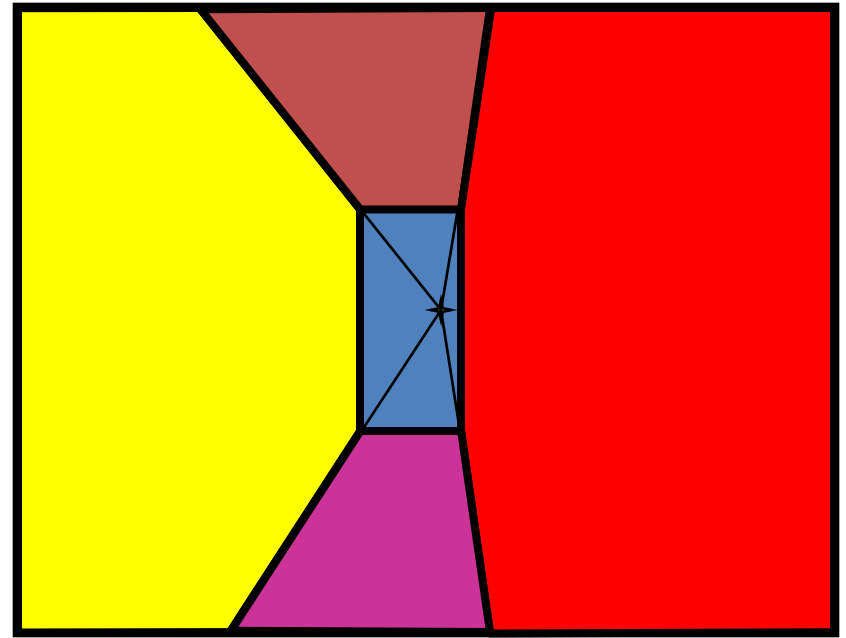
Another example of user input: vanishing point and back face of view volume are defined



Comparison of two camera placements – left and right.
Corresponding subdivisions match view you would see if
you looked down a hallway.



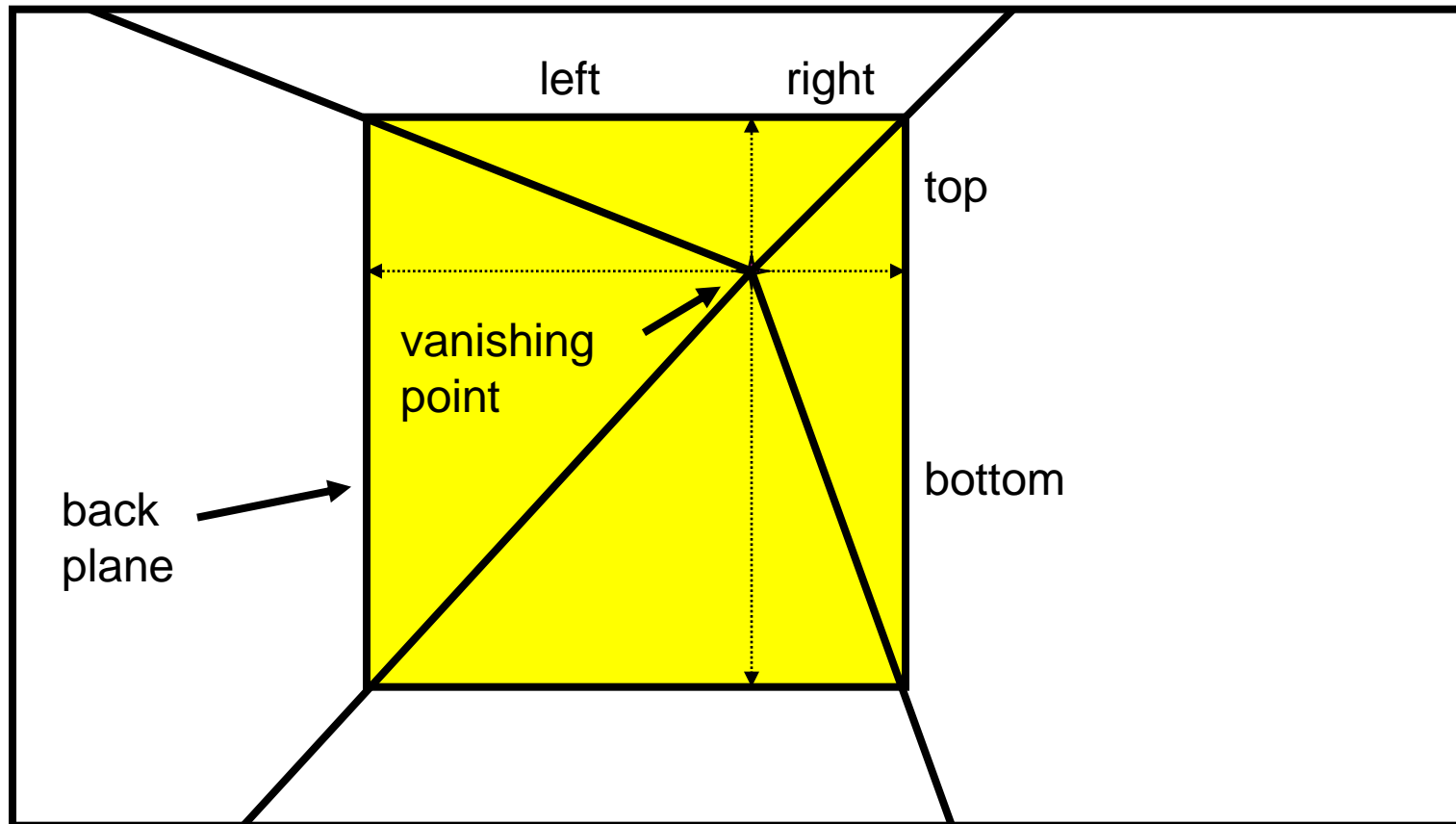
Left Camera



Right Camera

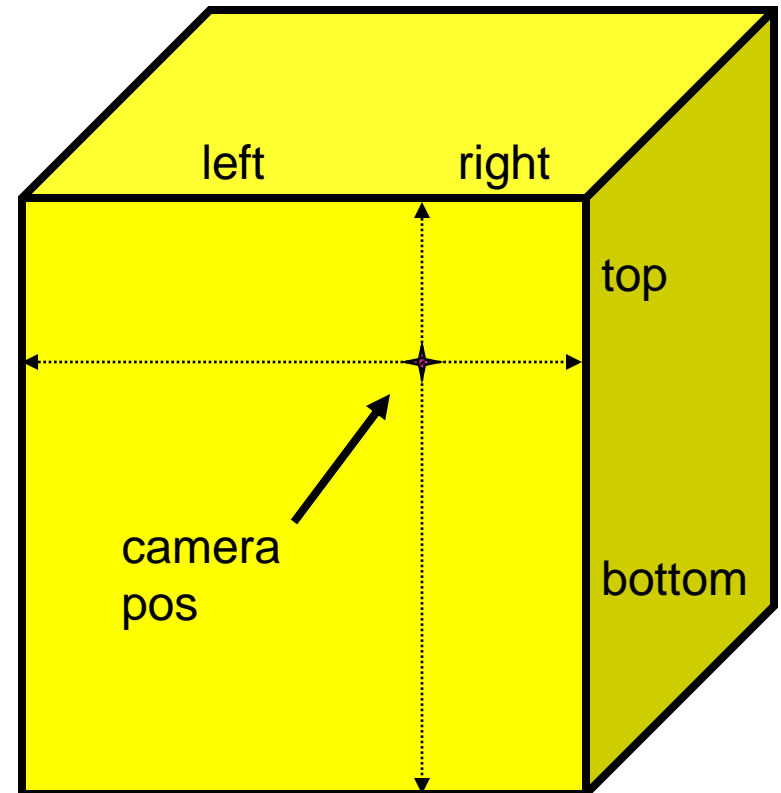
2D to 3D conversion

- First, we can get ratios



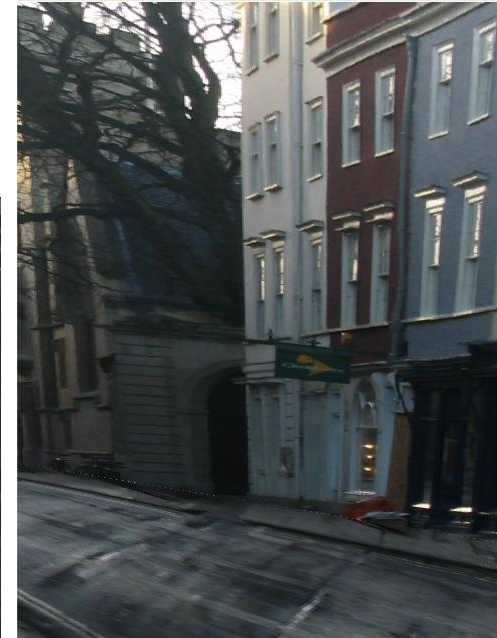
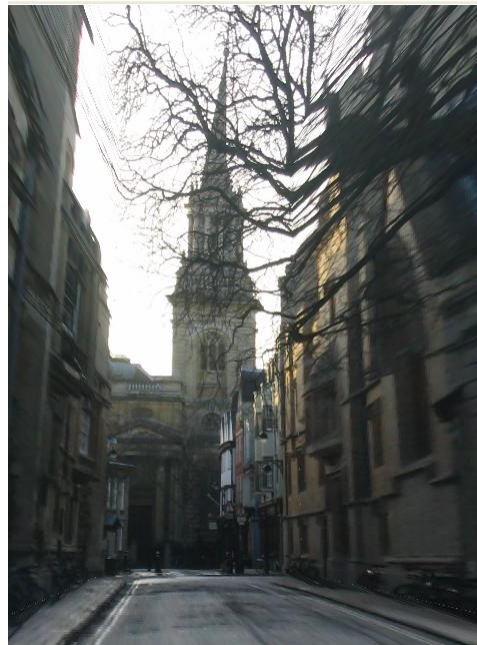
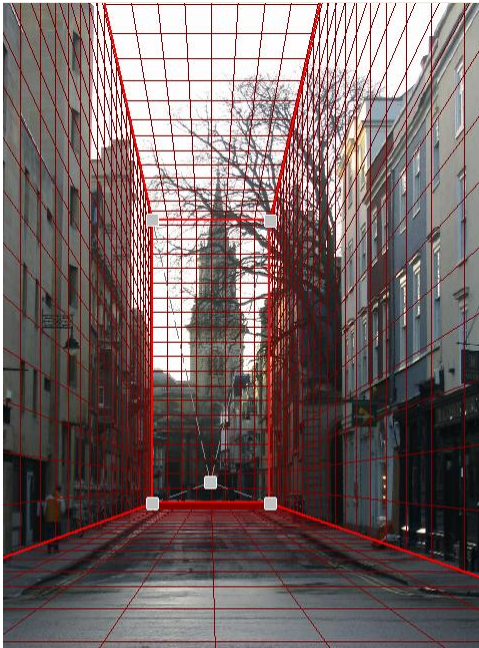
2D to 3D conversion

- 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



DEMO

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



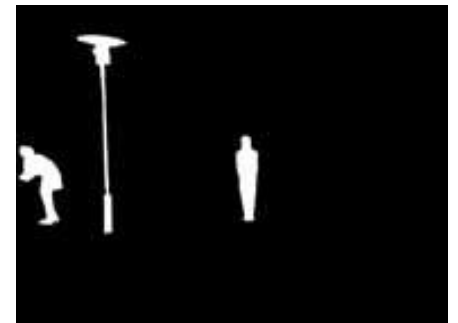
Foreground Objects

- Use separate billboard for each



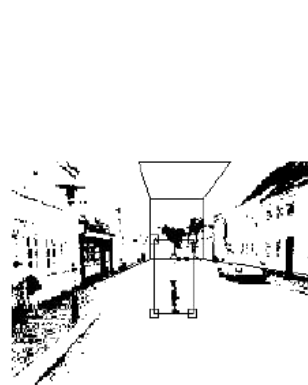
- For this to work, three separate images used:

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

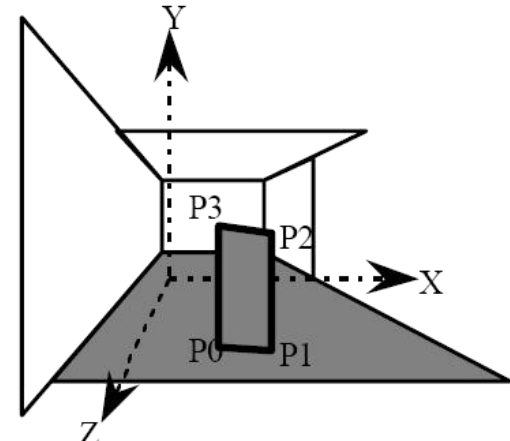


Foreground Objects

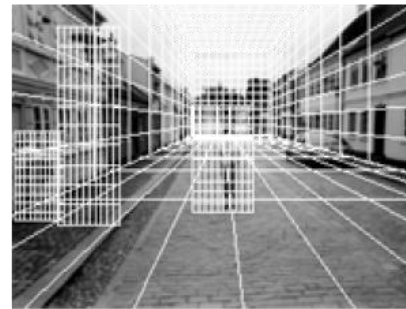
- Add vertical rectangles for each foreground object
- Can compute 3D coordinates P_0 , P_1 since they are on known plane.
- P_2 , P_3 can be computed as before (similar triangles)



(a) Specifying of a foreground object

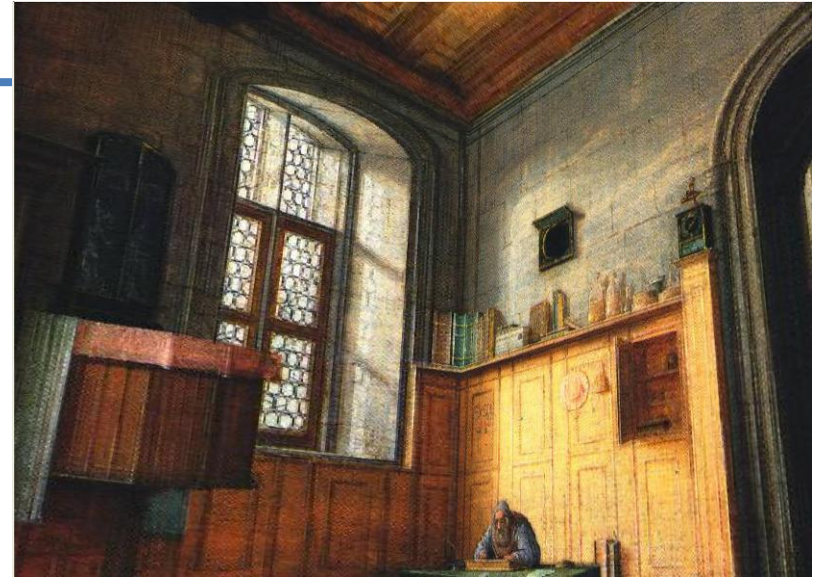
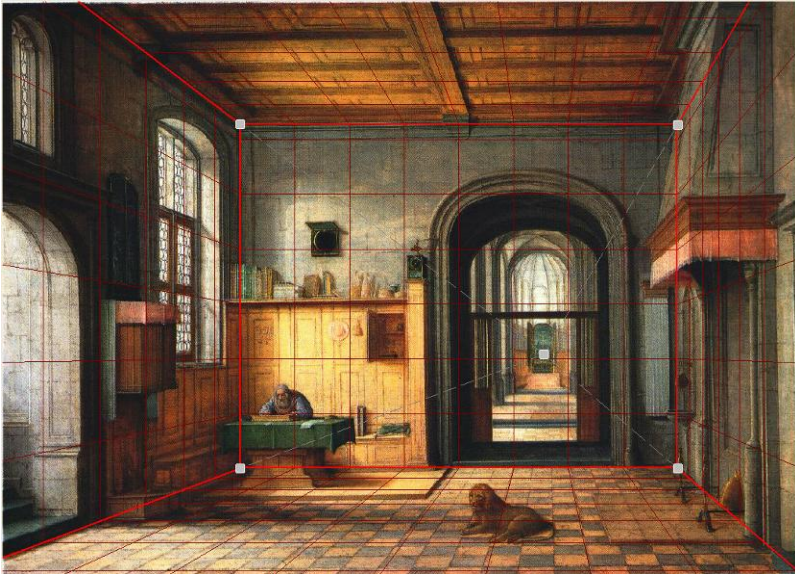


(b) Estimating the vertices of the foreground object model



(c) Three foreground object models

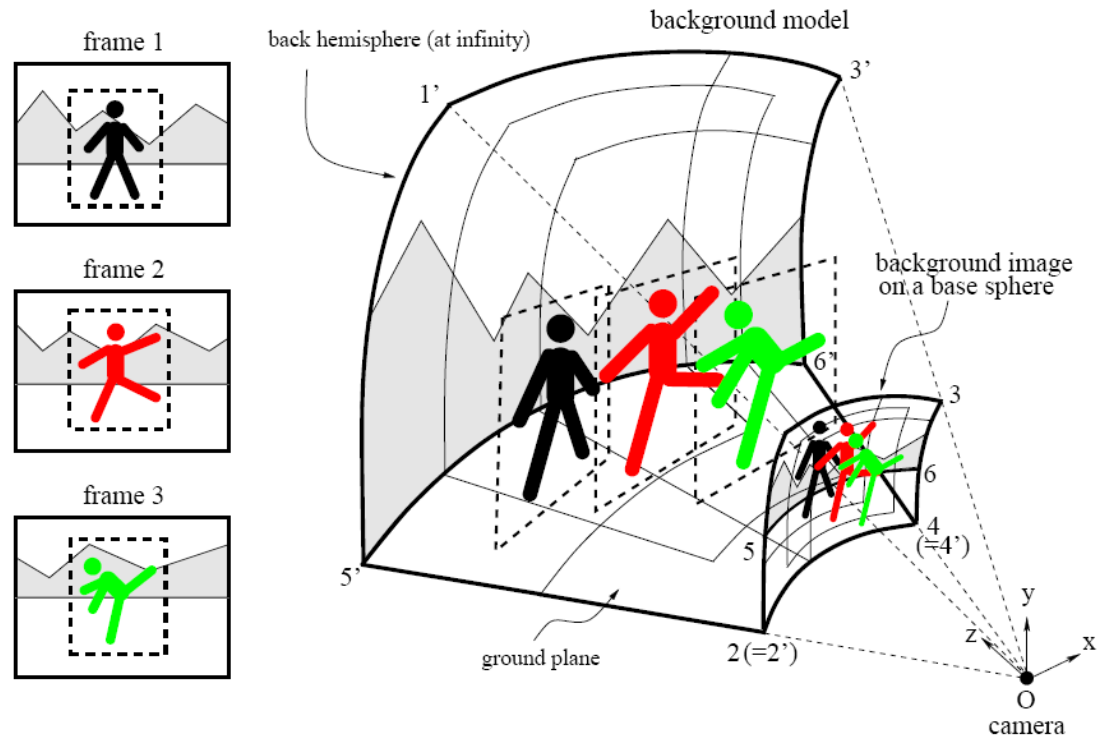
Foreground



[TIP movie](#)
[UVA example](#)

See also...

- Tour into the picture with water surface reflection
- Tour into the
Video:
– by Kang + Shin



Today's schedule

- Tour Into the Picture¹
- Video Textures²

¹Slides borrowed from Alexei Efros, who built on Steve Seitz's and David Brogan's

²Slides from Arno Schoedl

Markov Chains

- probability of going from state i to state j in n time steps:

$$p_{ij}^{(n)} = \Pr(X_n = j \mid X_0 = i)$$

and the single-step transition as:

$$p_{ij} = \Pr(X_1 = j \mid X_0 = i)$$

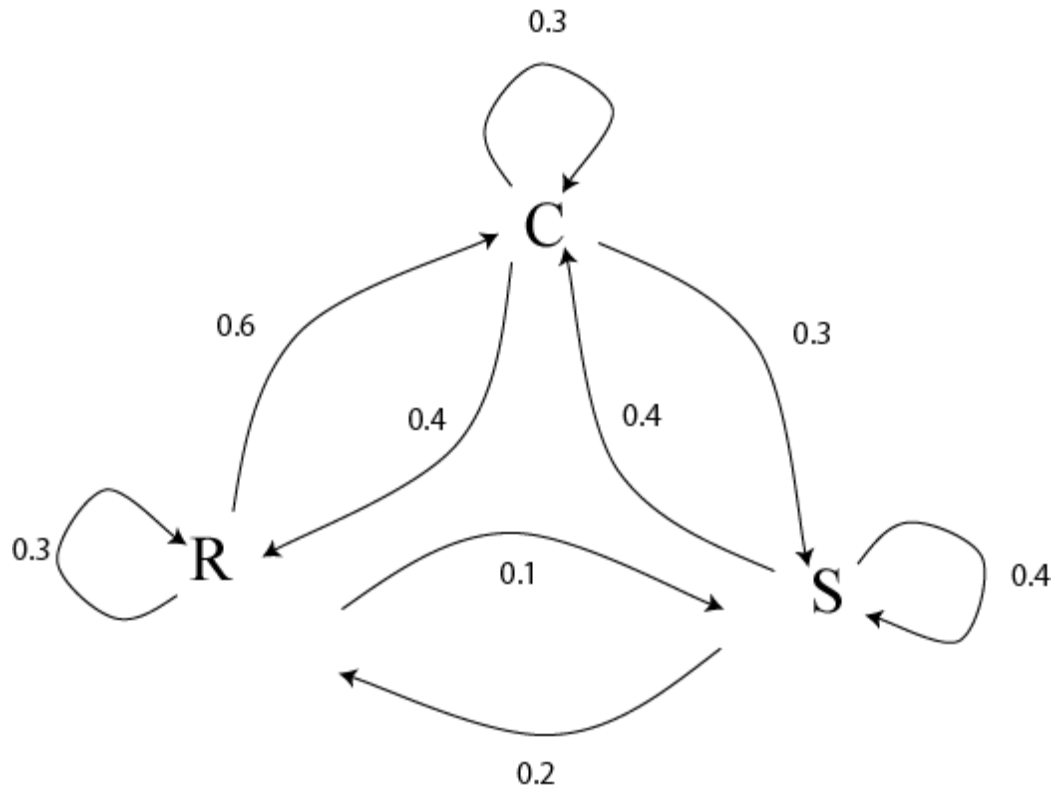
The n -step transition satisfies the [Chapman-Kolmogorov equation](#), that for any $0 < k < n$:

$$p_{ij}^{(n)} = \sum_{r \in S} p_{ir}^{(k)} p_{rj}^{(n-k)}$$

Markov Chains

- Regular Markov chain: class of Markov chains where the starting state of the chain has little or no impact on the $p(X)$ after many steps.

Markov Chain



$$\begin{pmatrix} 0.3 & 0.6 & 0.1 \\ 0.4 & 0.3 & 0.3 \\ 0.2 & 0.4 & 0.4 \end{pmatrix}$$

What if we know today and yestarday's weather?

Text Synthesis

- [Shannon,'48] proposed a way to generate English-looking text using N-grams:
 - Assume a generalized Markov model
 - Use a large text to compute prob. distributions of each letter given N-1 previous letters
 - Starting from a seed repeatedly sample this Markov chain to generate new letters
 - Also works for whole words

WE NEED TO EAT CAKE

Mark V. Shaney (Bell Labs)

- Results (using `alt.singles` corpus):
 - *“As I've commented before, really relating to someone involves standing next to impossible.”*
 - *“One morning I shot an elephant in my arms and kissed him.”*
 - *“I spent an interesting evening recently with a grain of salt”*

Video Textures

Arno Schödl
Richard Szeliski
David Salesin
Irfan Essa

Microsoft Research, Georgia Tech



[Link to local version](#)

[Gondry Example](#)

Still photos



Video clips



Video textures



Problem statement



video clip

video texture

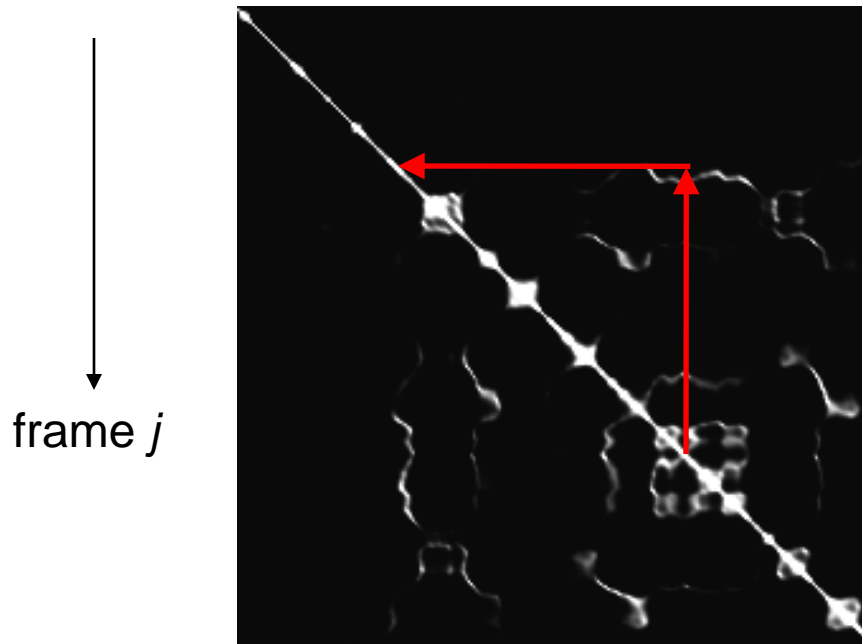
Our approach



- How do we find good transitions?

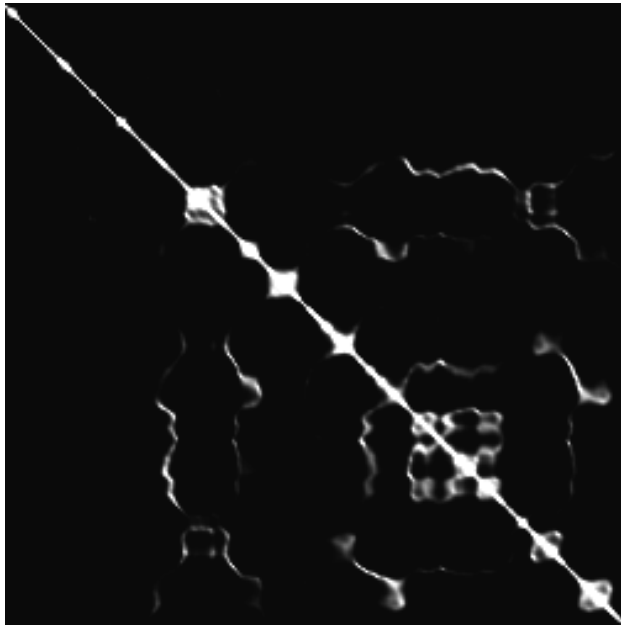
Finding good transitions

- Compute L_2 distance $D_{i,j}$ between all frames
vs. $\xrightarrow{\text{frame } i}$



Similar frames make good transitions

Markov chain representation



1

2

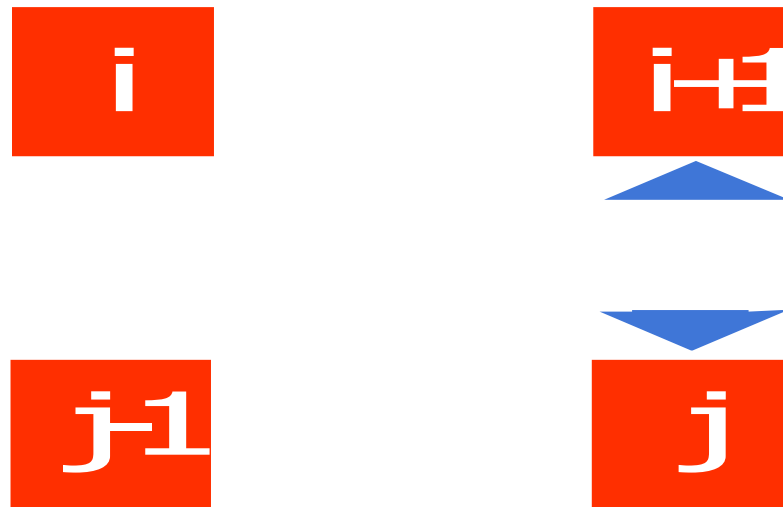
3

4

Similar frames make good transitions

Transition costs

- Transition from i to j if successor of i is similar to j
 - Cost function: $C_{i \rightarrow j} = D_{i+1, j}$



Transition probabilities

- Probability for transition $P_{i \rightarrow j}$ inversely related to cost:

$$P_{i \rightarrow j} \sim \exp (- C_{i \rightarrow j} / \sigma^2)$$



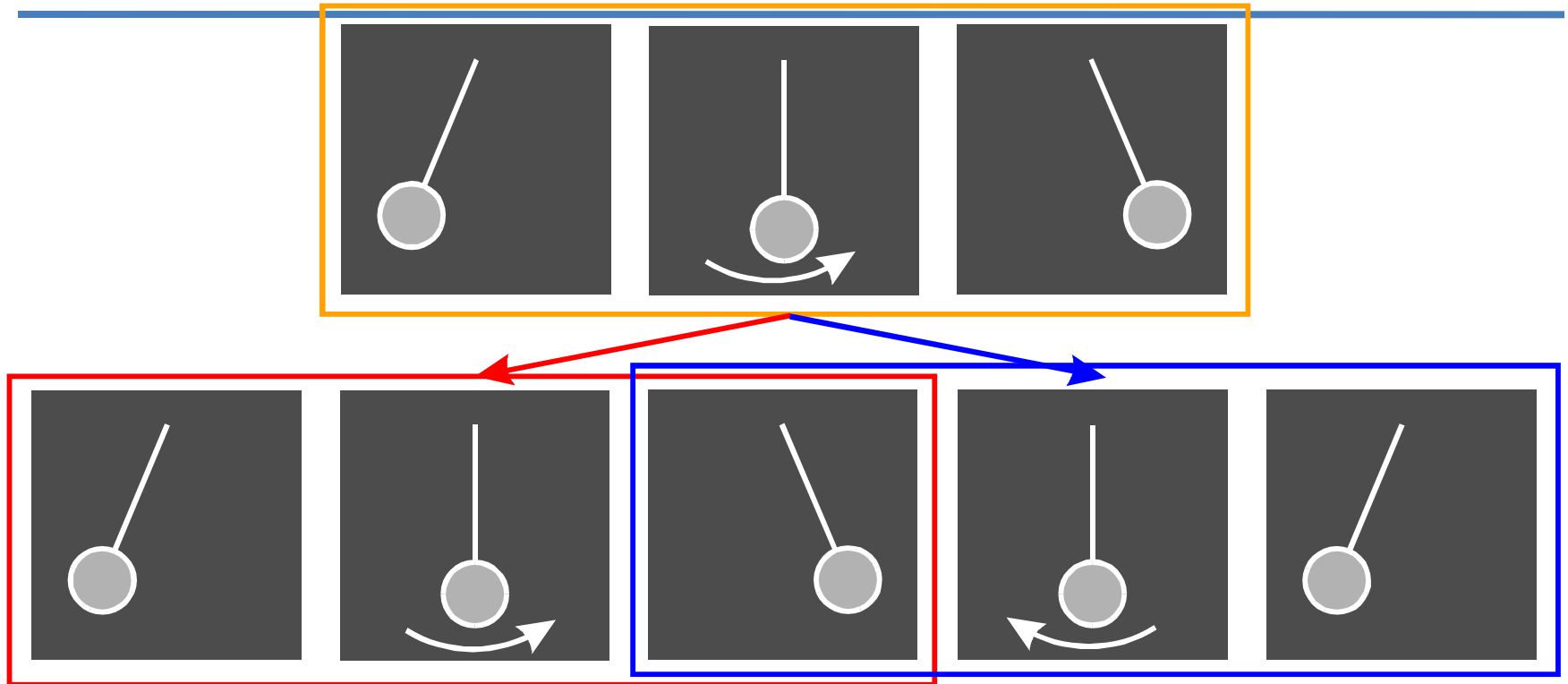
high σ

low σ

Preserving dynamics

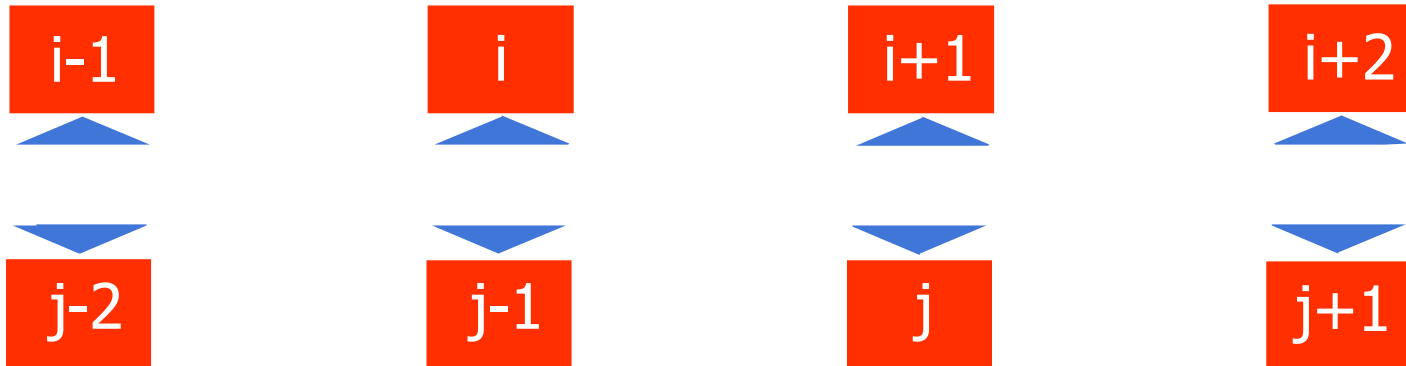


Preserving dynamics



Preserving dynamics

- Cost for transition $i \rightarrow j$
 - $C_{i \rightarrow j} = \sum_k w_k D_{i+k+1, j+k}$



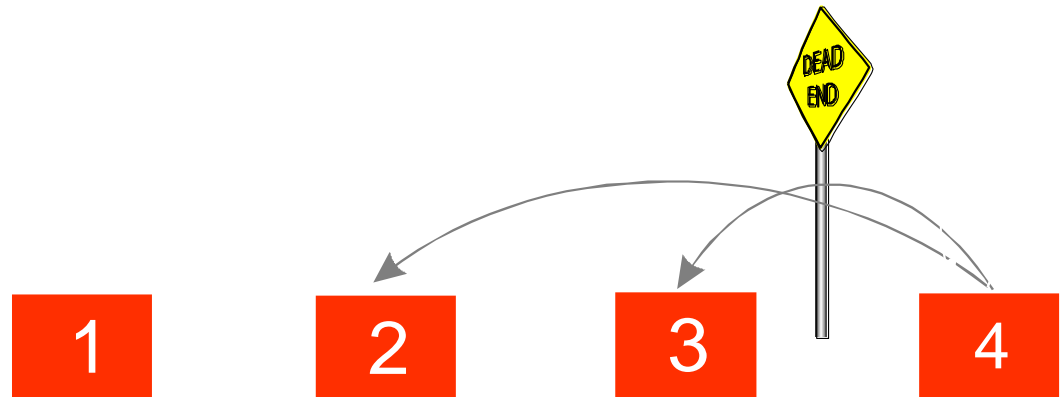
Preserving dynamics – effect

- Cost for transition $i \rightarrow j$
 - $C_{i \rightarrow j} = \sum_k w_k D_{i+k+1, j+k}$



Dead ends

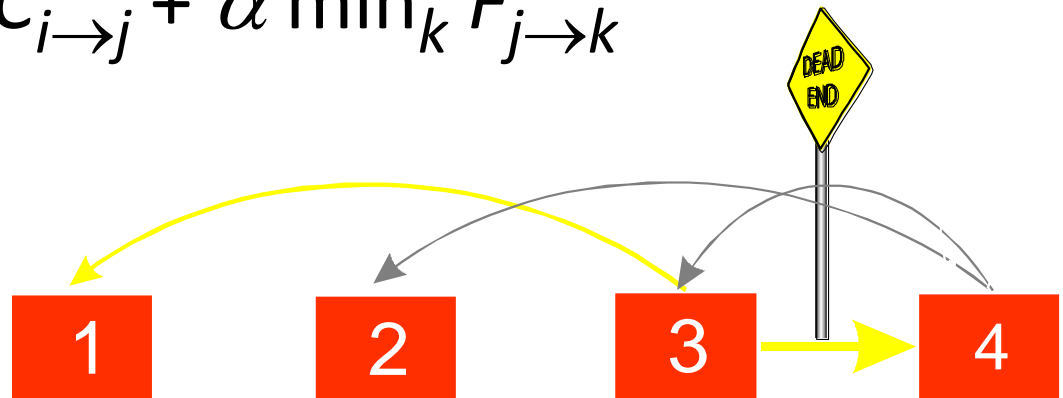
- No good transition at the end of sequence



Future cost

- Propagate future transition costs backward
- Iteratively compute new cost

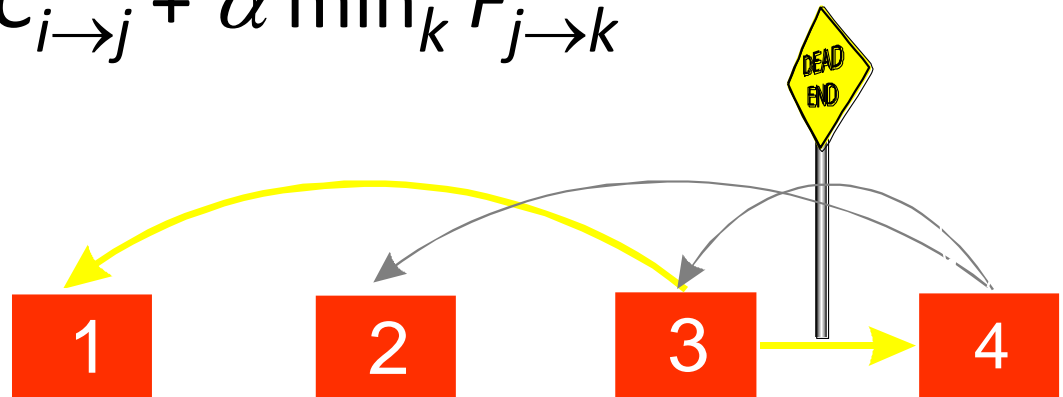
- $$F_{i \rightarrow j} = C_{i \rightarrow j} + \alpha \min_k F_{j \rightarrow k}$$



Future cost

- Propagate future transition costs backward
- Iteratively compute new cost

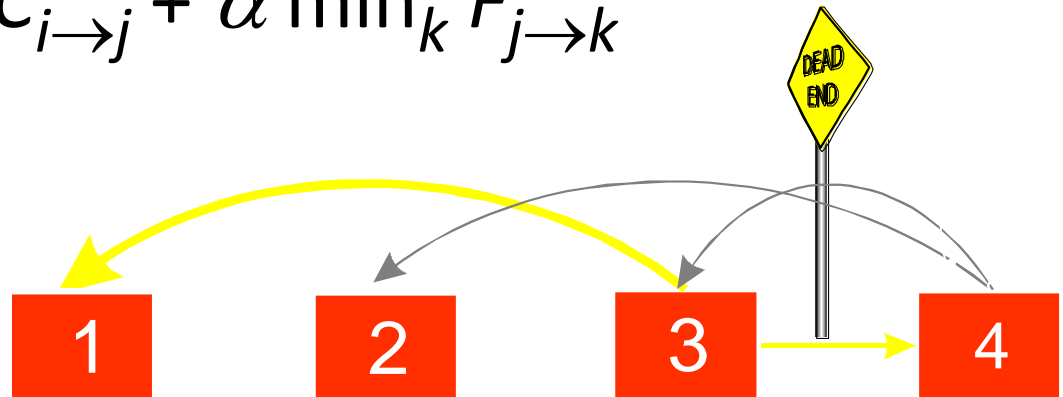
- $$F_{i \rightarrow j} = C_{i \rightarrow j} + \alpha \min_k F_{j \rightarrow k}$$



Future cost

- Propagate future transition costs backward
- Iteratively compute new cost

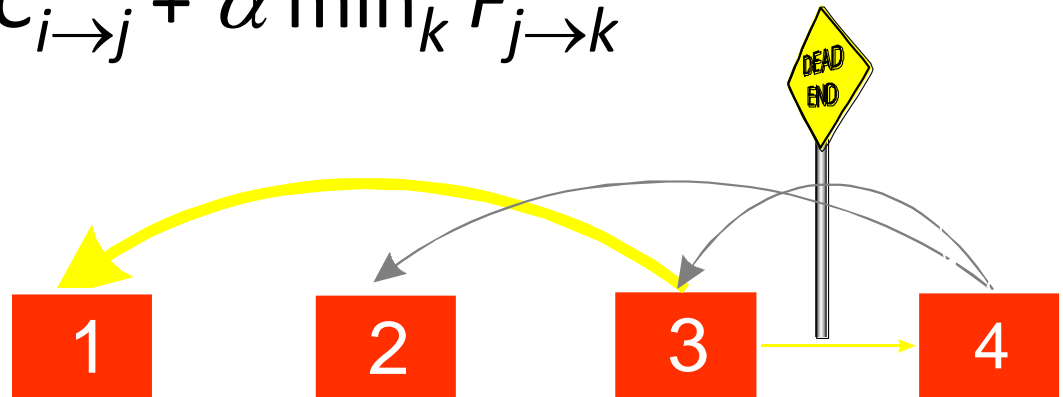
- $$F_{i \rightarrow j} = C_{i \rightarrow j} + \alpha \min_k F_{j \rightarrow k}$$



Future cost

- Propagate future transition costs backward
- Iteratively compute new cost

- $$F_{i \rightarrow j} = C_{i \rightarrow j} + \alpha \min_k F_{j \rightarrow k}$$

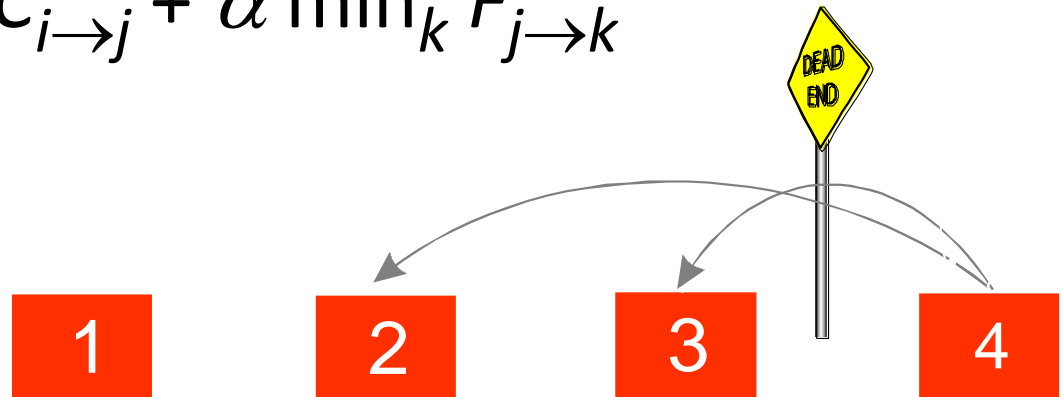


Future cost

- Propagate future transition costs backward
- Iteratively compute new cost

- $F_{i \rightarrow j} = C_{i \rightarrow j} + \alpha \min_k F_{j \rightarrow k}$

- Q-learning



Future cost – effect



Finding good loops

- Alternative to random transitions
- Precompute set of loops up front



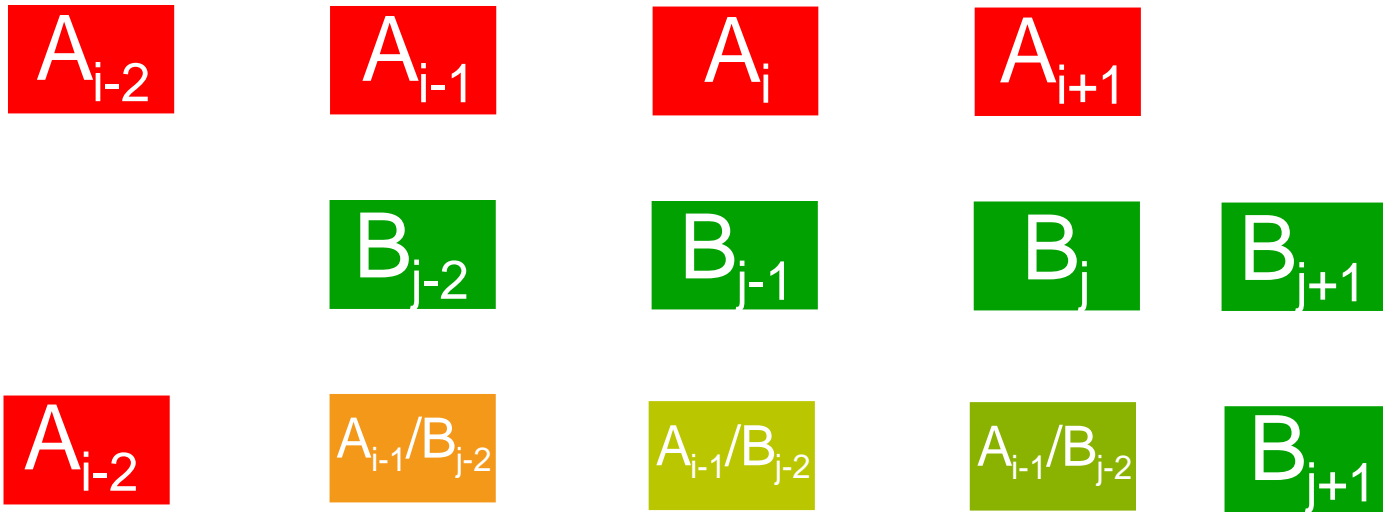
Visual discontinuities

- Problem: Visible “Jumps”



Crossfading

- Solution: Crossfade from one sequence to the other.



Morphing

- Interpolation task:

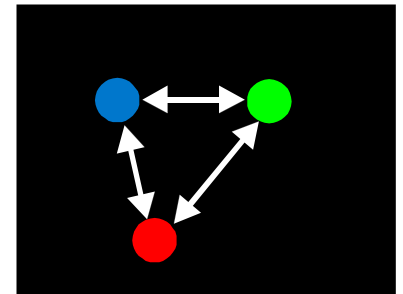


Morphing

- Interpolation task:



- Compute correspondence between pixels of all frames

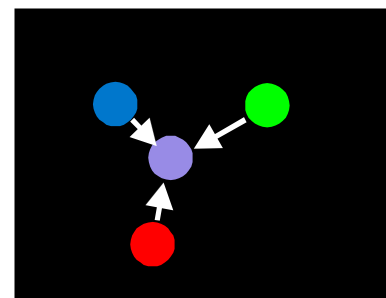


Morphing

- Interpolation task:



- Compute correspondence between pixels of all frames
- Interpolate pixel position and color in morphed frame
- based on [Shum 2000]



Results – crossfading/morphing



Results – crossfading/morphing



Jump Cut

Crossfade

Morph

Crossfading



Frequent jump & crossfading



Video portrait



- Useful for web pages

Video portrait – 3D



- Combine with IBR techniques

Region-based analysis

- Divide video up into regions



- Gene

Automatic region analysis



User-controlled video textures



slow

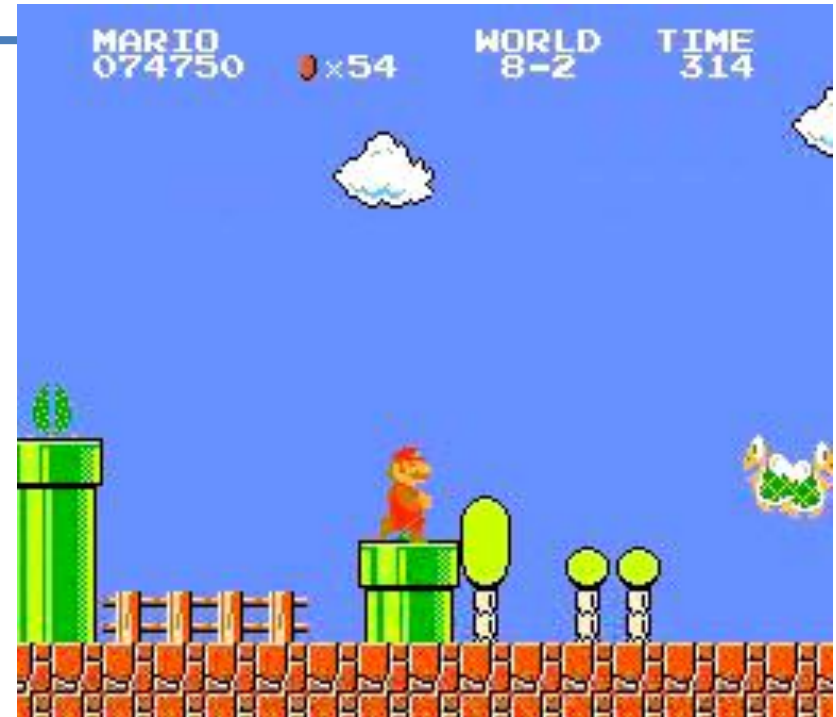
variable

fast

User selects target frame range

Video-based animation

- Like sprites
computer games
- Extract sprites
from real video
- Interactively control
desired motion



©1985 Nintendo of America Inc.

Video sprite extraction

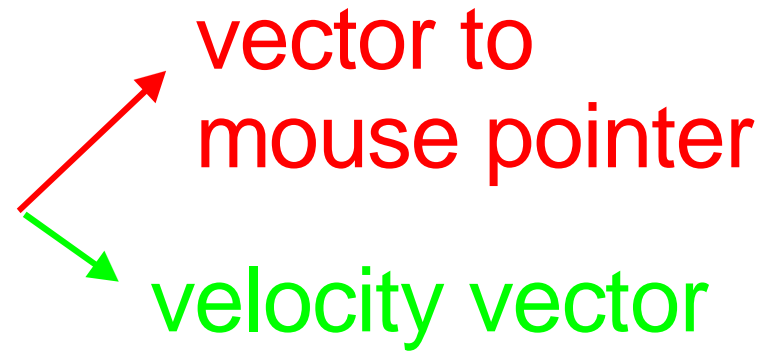


Blue screen
matting and
velocity
estimation



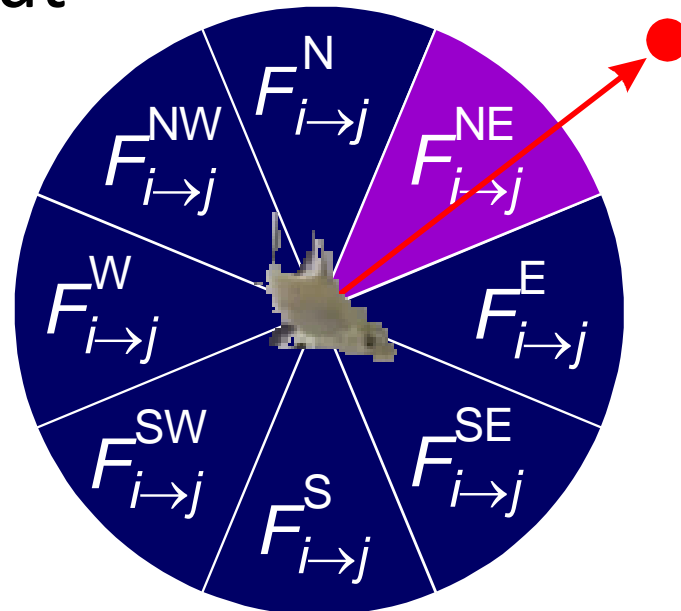
Video sprite control

- Augmented transition cost:



Video sprite control

- Need future cost computation
- Precompute future costs for a few angles.
- Switch between precomputed angles according to user input
- [GIT-GVU-00-11]



Interactive fish



Summary

- Video clips → video textures
 - define Markov process
 - preserve dynamics
 - avoid dead-ends
 - disguise visual discontinuities



Discussion

- Some things are relatively easy



Discussion

- Some are hard



A final example



Michel Gondry train video

<http://youtube.com/watch?v=qUEs1BwVXGA>