



基于单视图的三维重建

章国锋

浙江大学CAD&CG国家重点实验室



Single View Modeling

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



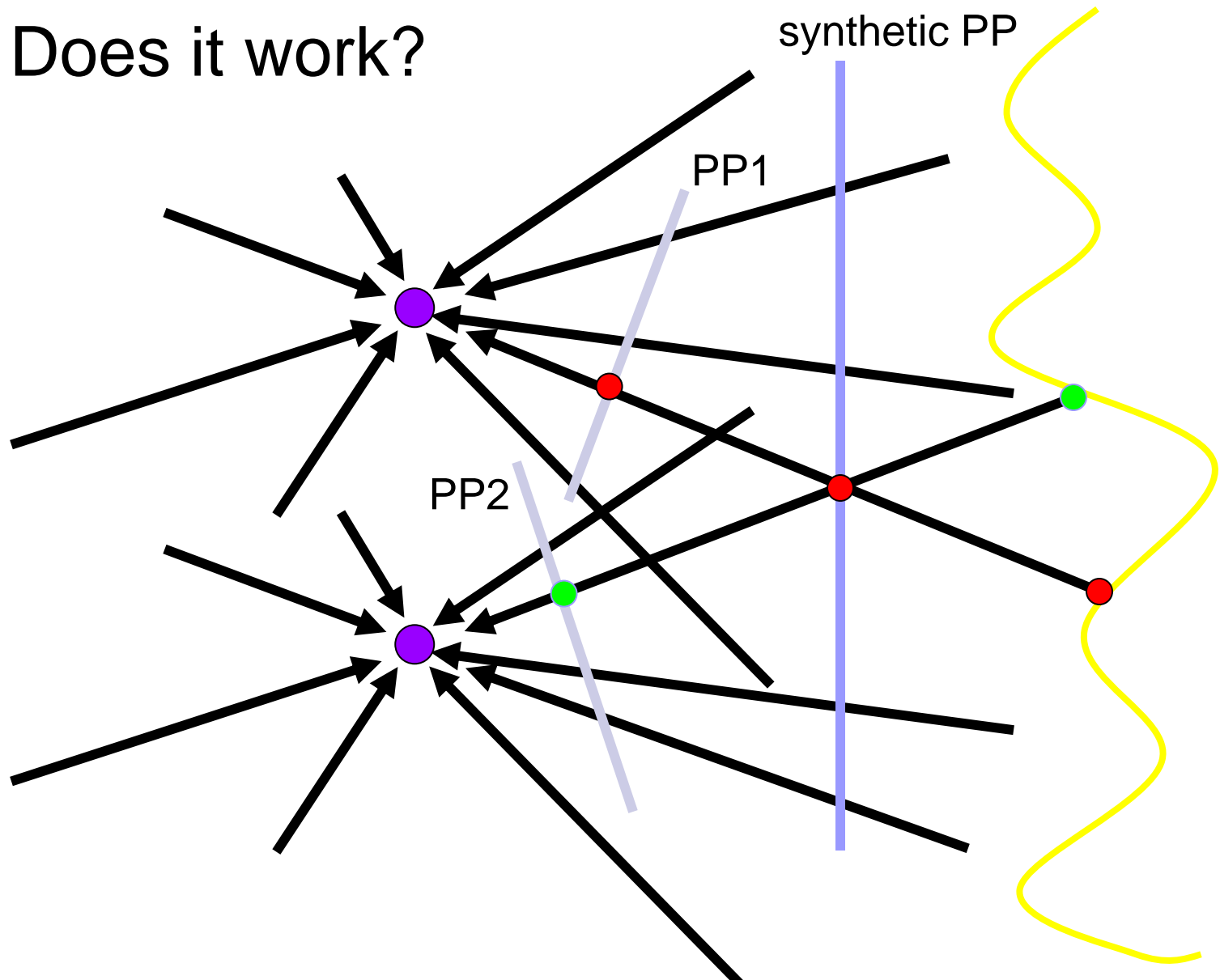
St.Petersburg
photo by A. Tikhonov

Virtual camera rotations

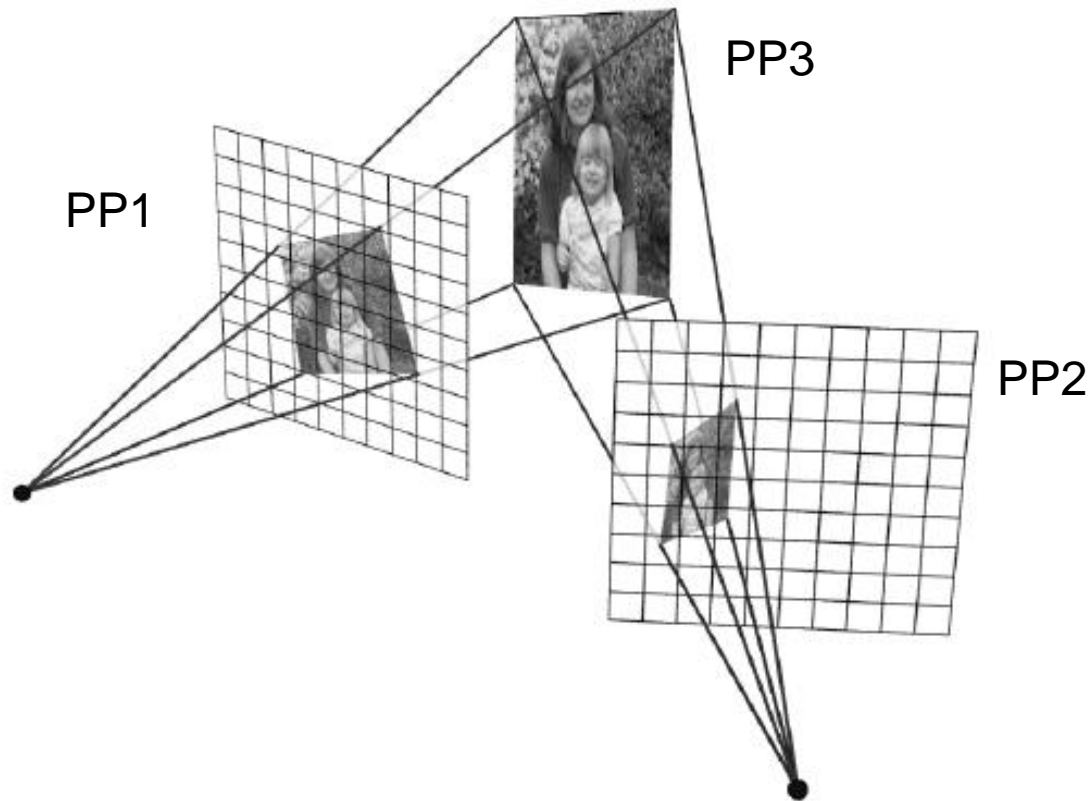


Camera translation

- Does it work?



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

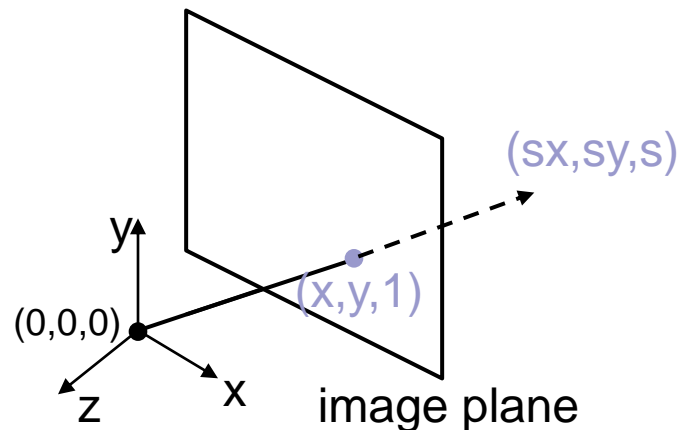
Silly Euclid



Parallel lines???

The projective plane

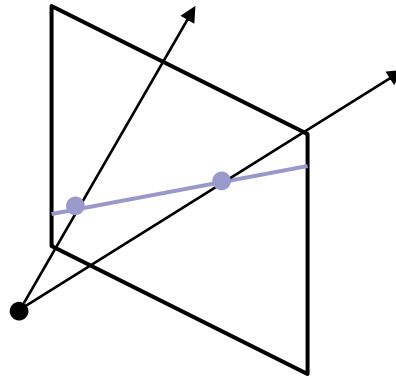
- Why do we need homogeneous coordinates?
 - represent points at infinity, homographies, perspective projection, multi-view relationships
- What is the geometric intuition?
 - a point in the image is a *ray* in projective space



- 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?



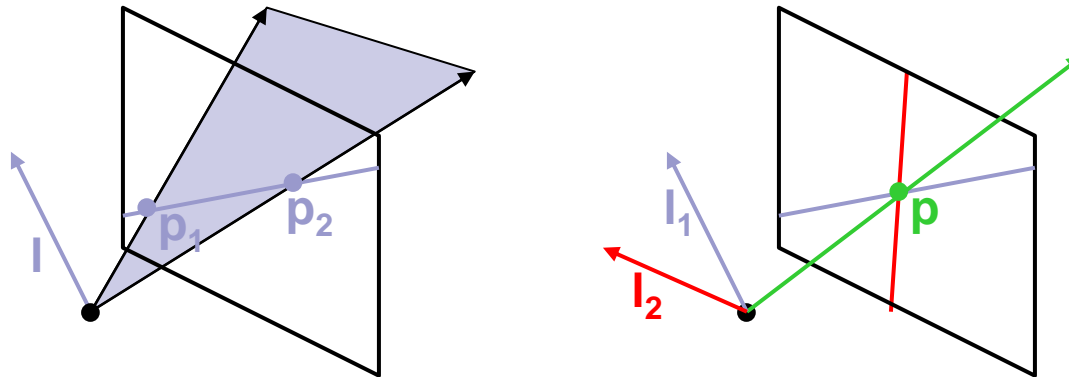
- A line is a *plane* of rays through origin
 - all rays (x,y,z) satisfying: $ax + by + cz = 0$

in vector notation $\mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$ $\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$

- A line is also represented as a homogeneous 3-vector \mathbf{l}

Point and line duality

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



What is the line \mathbf{l} spanned by rays \mathbf{p}_1 and \mathbf{p}_2 ?

- \mathbf{l} is \perp to \mathbf{p}_1 and $\mathbf{p}_2 \Rightarrow \mathbf{l} = \mathbf{p}_1 \times \mathbf{p}_2$
- \mathbf{l} is the plane normal

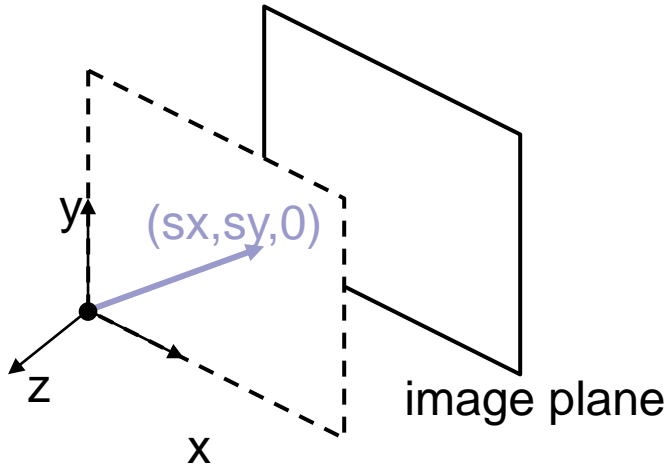
What is the intersection of two lines \mathbf{l}_1 and \mathbf{l}_2 ?

- \mathbf{p} is \perp to \mathbf{l}_1 and $\mathbf{l}_2 \Rightarrow \mathbf{p} = \mathbf{l}_1 \times \mathbf{l}_2$

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”)

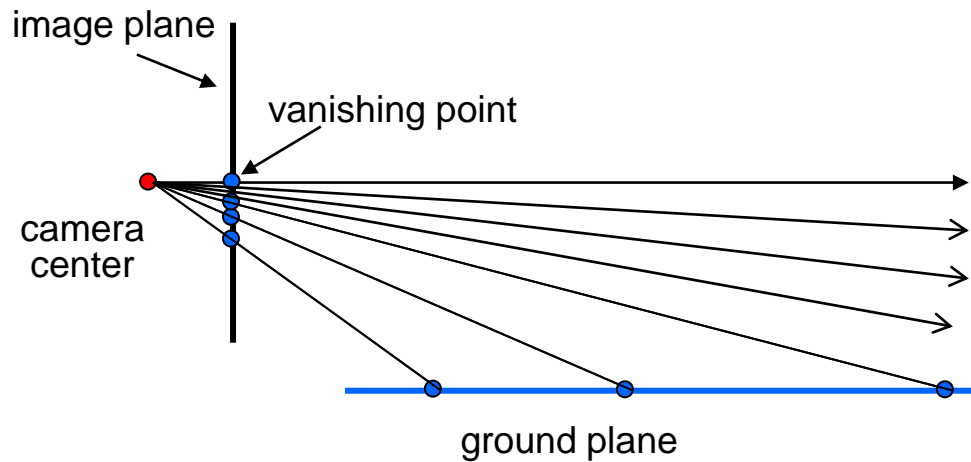
□ $p \cong (x, y, 0)$ – parallel to image plane

□ It has infinite image coordinates

Ideal line

• $l \cong (0, 0, 1)$ – parallel to image plane

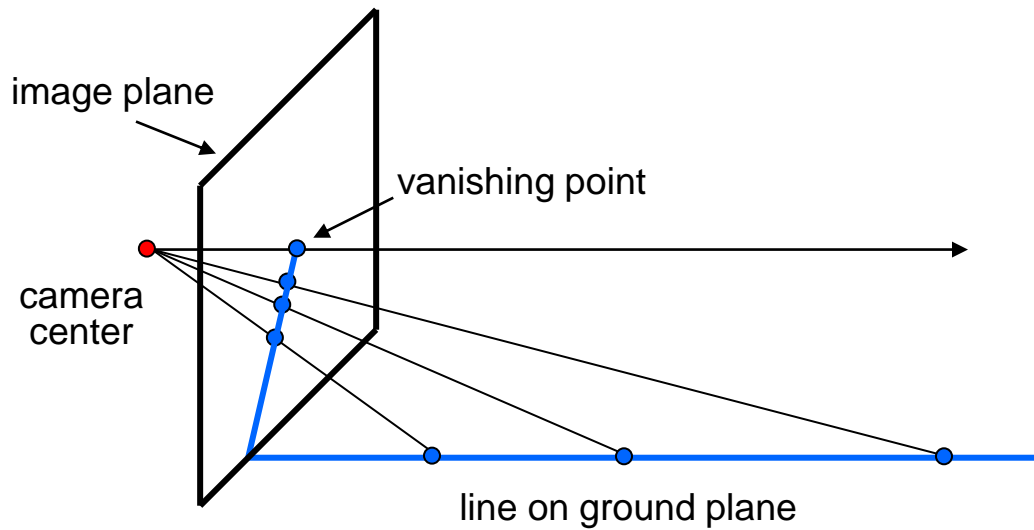
Vanishing points



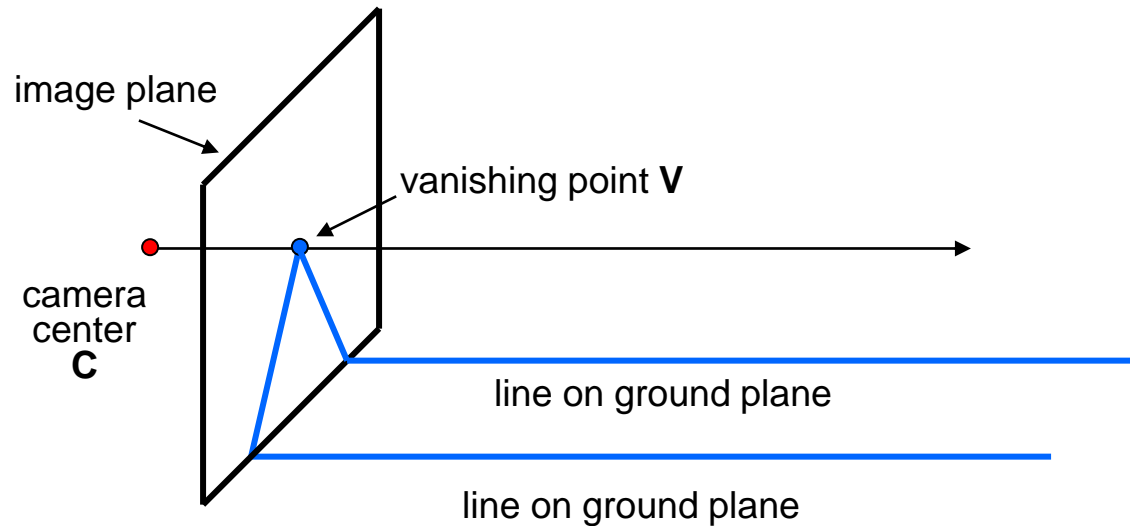
■ Vanishing point

- projection of a point at infinity
- Caused by ideal line

Vanishing points (2D)



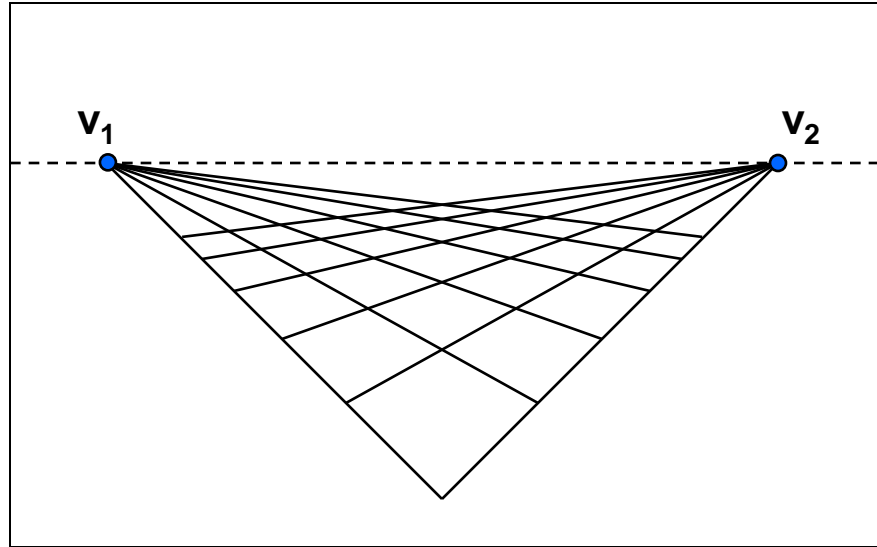
Vanishing points



■ Properties

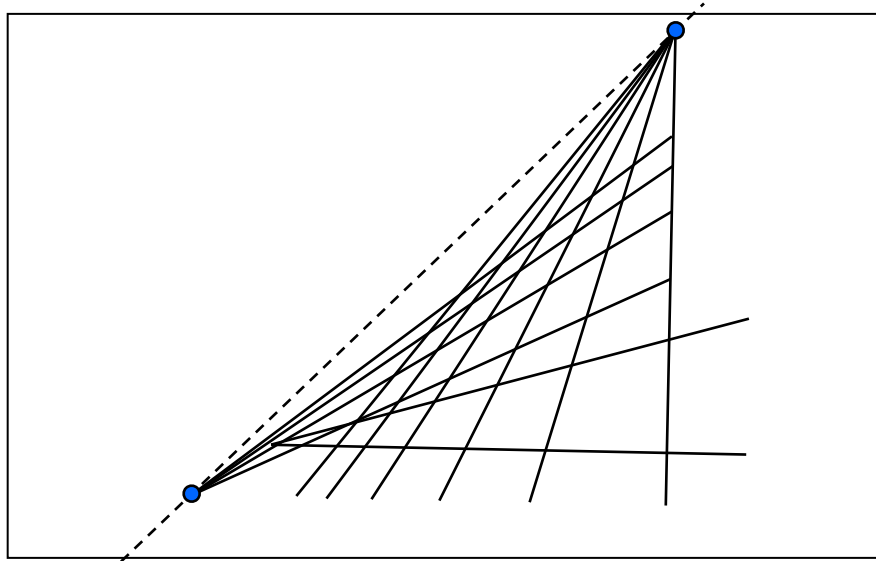
- Any two parallel lines have the same vanishing point v
- The ray from C through v is parallel to the lines
- An image may have more than one vanishing point

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 *horizon line*
 - also called *vanishing line*
 - Note that different planes define different vanishing lines

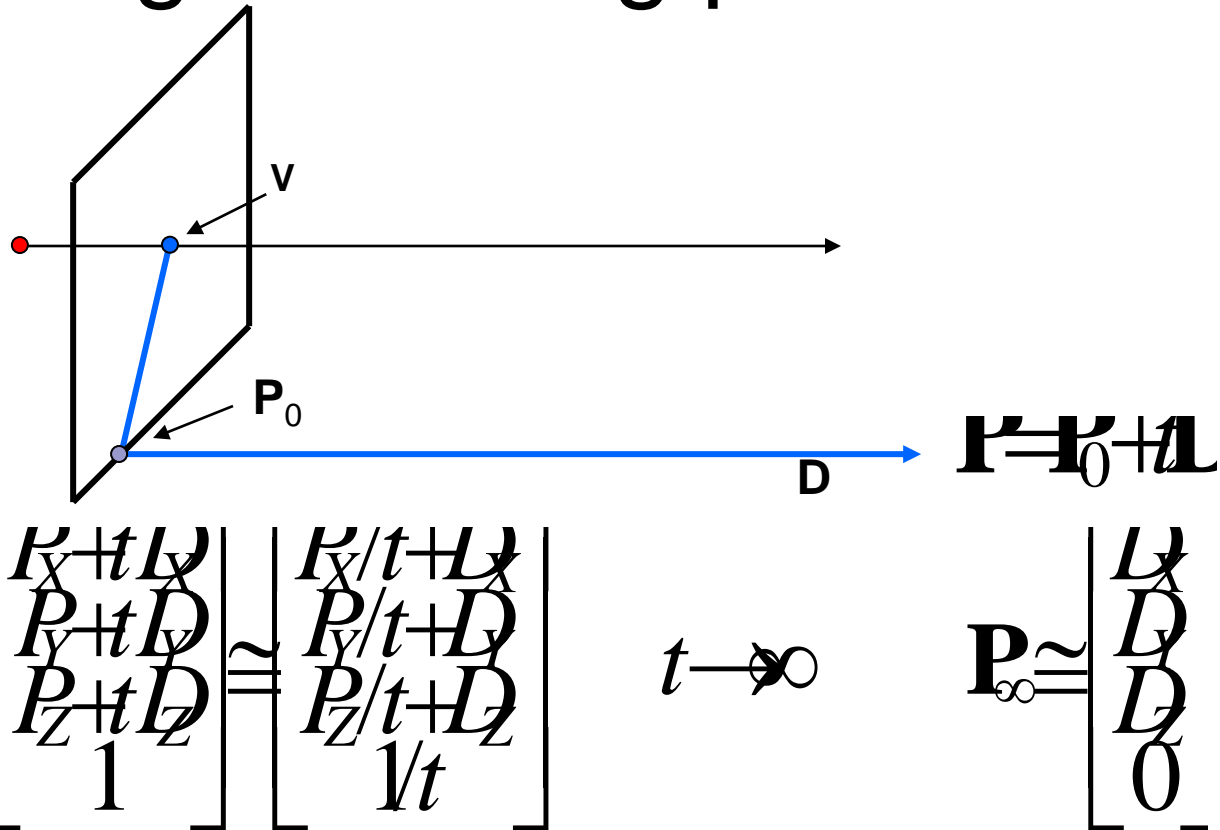
Vanishing lines



- 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

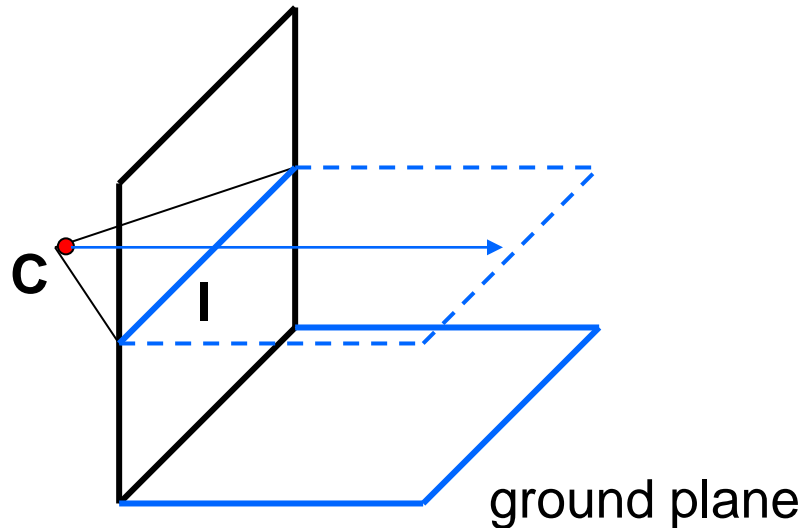
Computing vanishing points



■ Properties $\mathbf{v} = \mathbf{D}$

- \mathbf{P}_∞ is a point at *infinity*, \mathbf{v} is its projection
- They depend only on line *direction*
- Parallel lines $\mathbf{P}_0 + t\mathbf{D}$, $\mathbf{P}_1 + t\mathbf{D}$ intersect at \mathbf{P}_∞

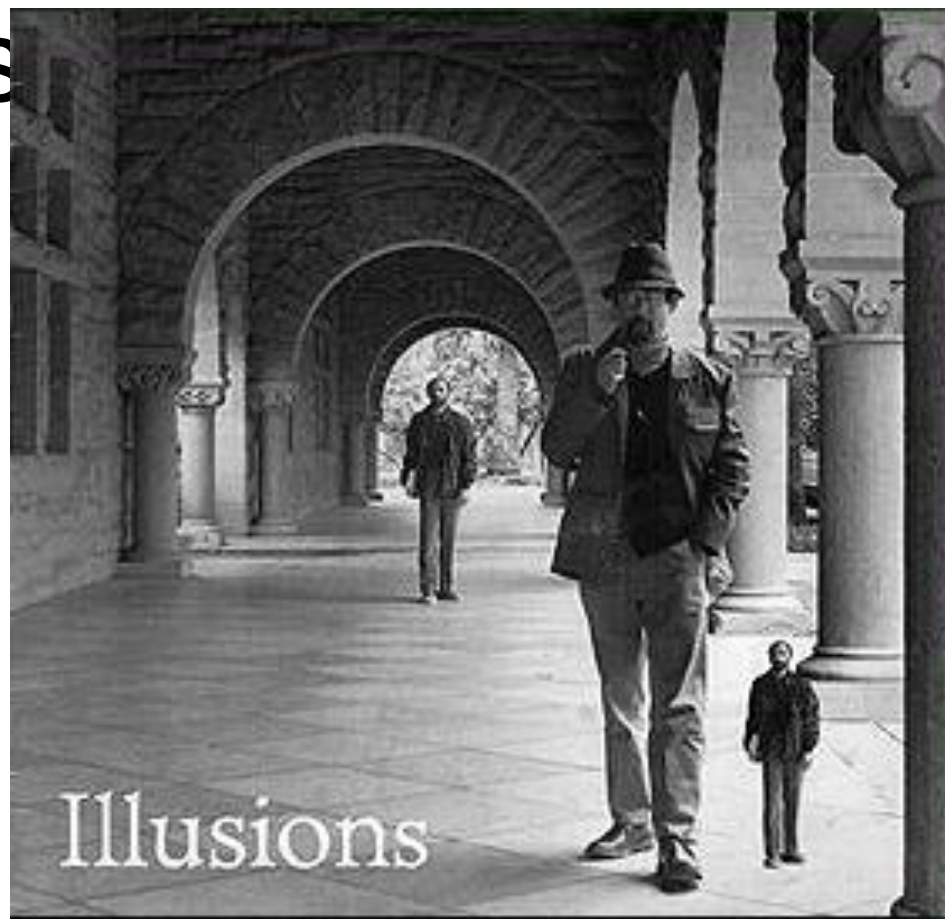
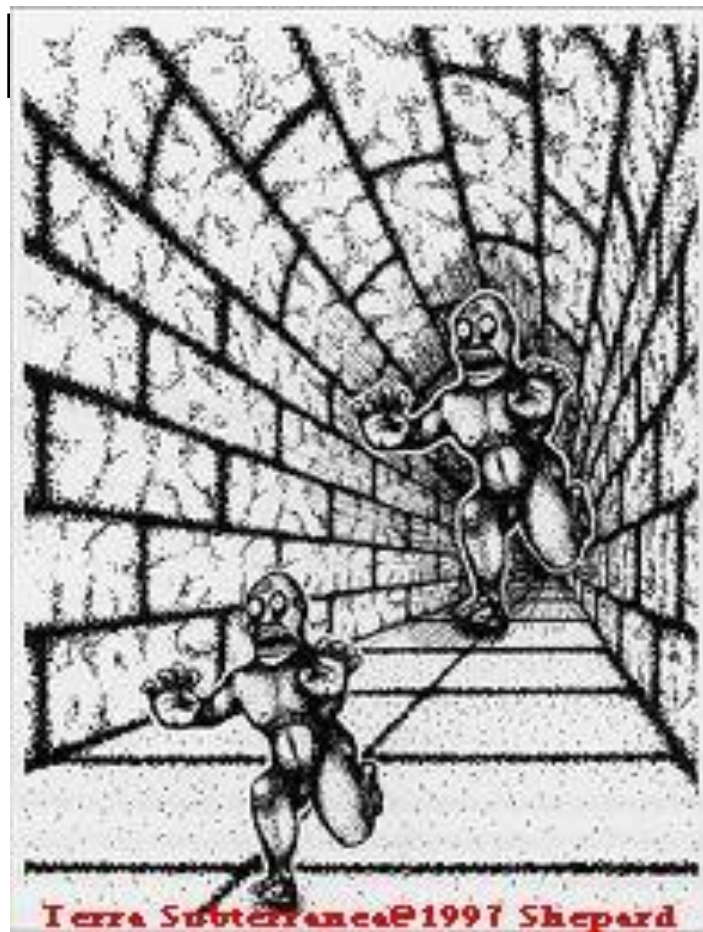
Computing vanishing lines



■ 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





“Tour into the Picture” (SIGGRAPH '97)

- Create a 3D “theatre stage” of five billboards



- Specify foreground objects through bounding polygons

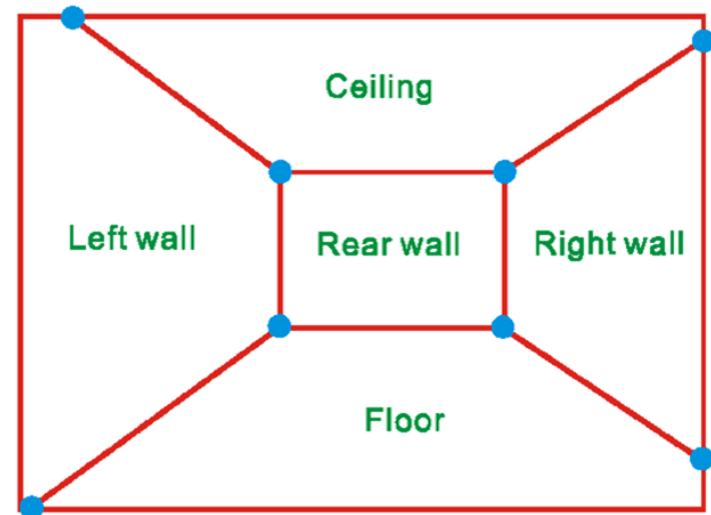


- Use camera transformations to navigate through the scene

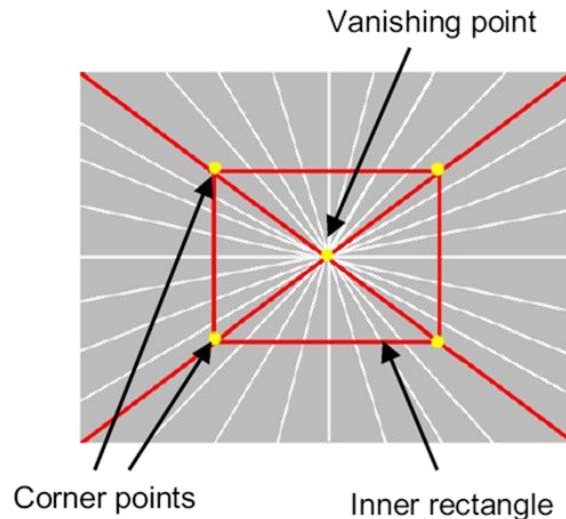


The idea

- 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!

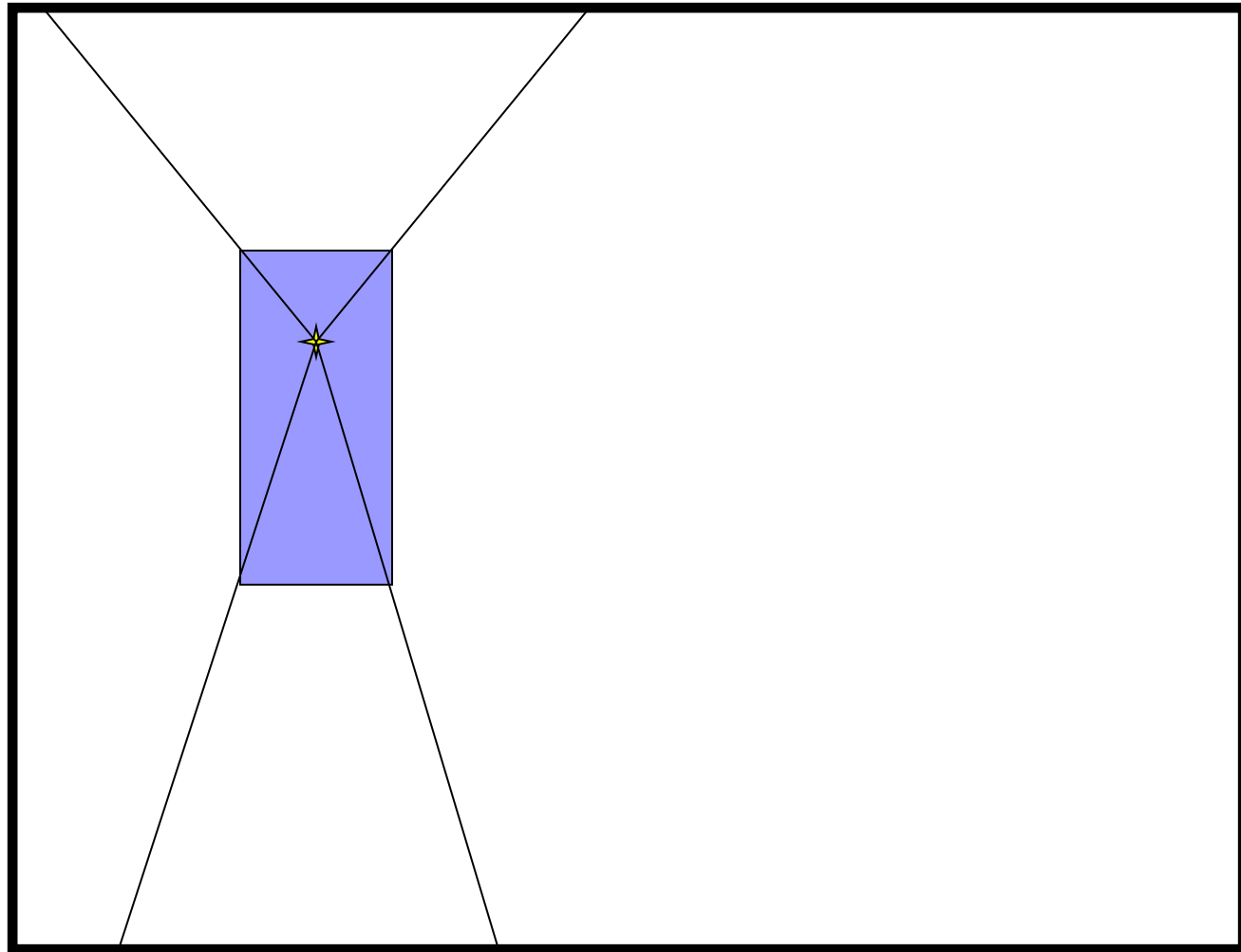


Fitting the box volume



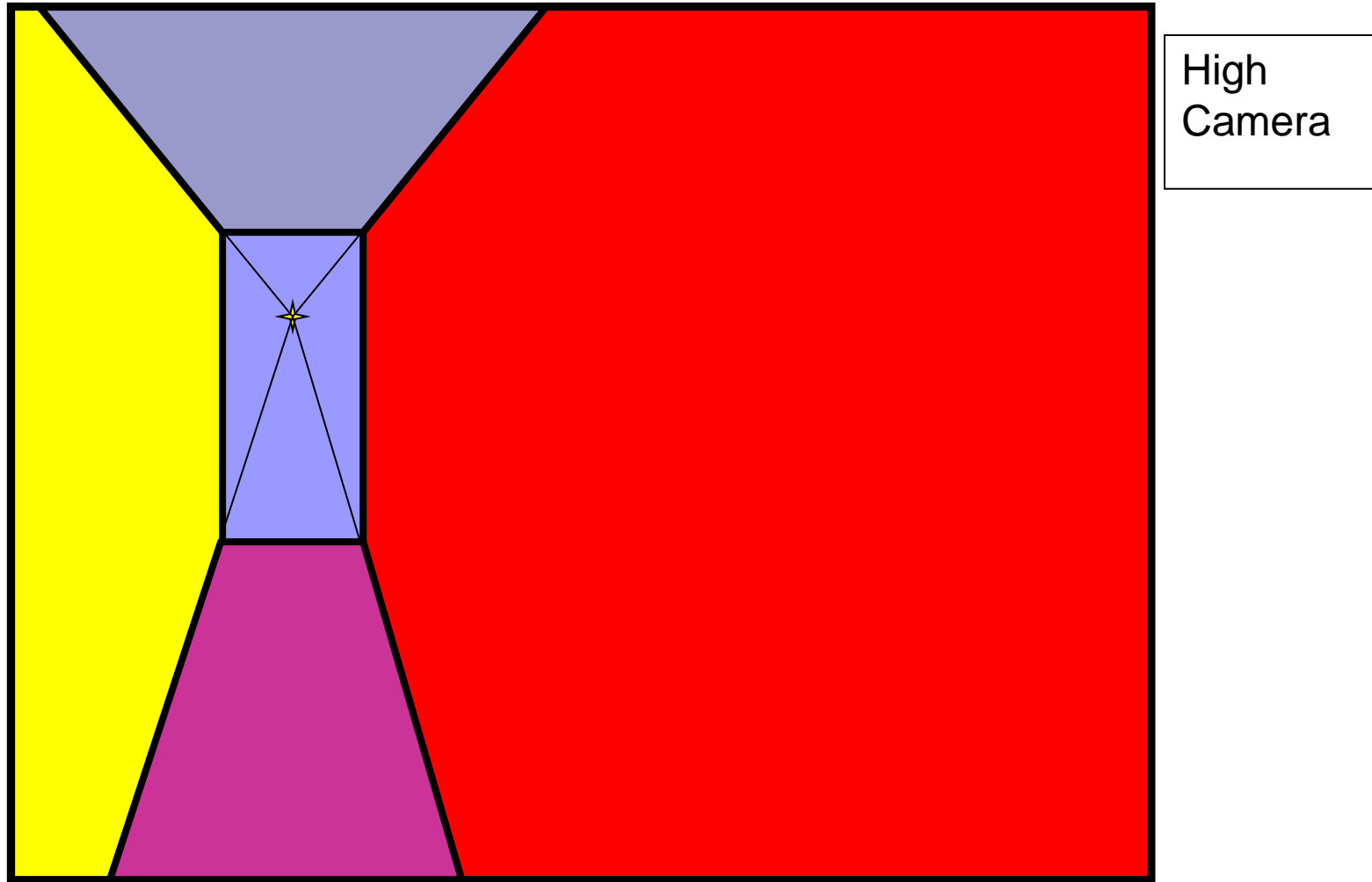
- User controls the inner box and the vanishing point placement (# of DOF???)
- Q: What's the significance of the vanishing point location?
- A: It's at eye level: ray from COP to VP is perpendicular to image plane.

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

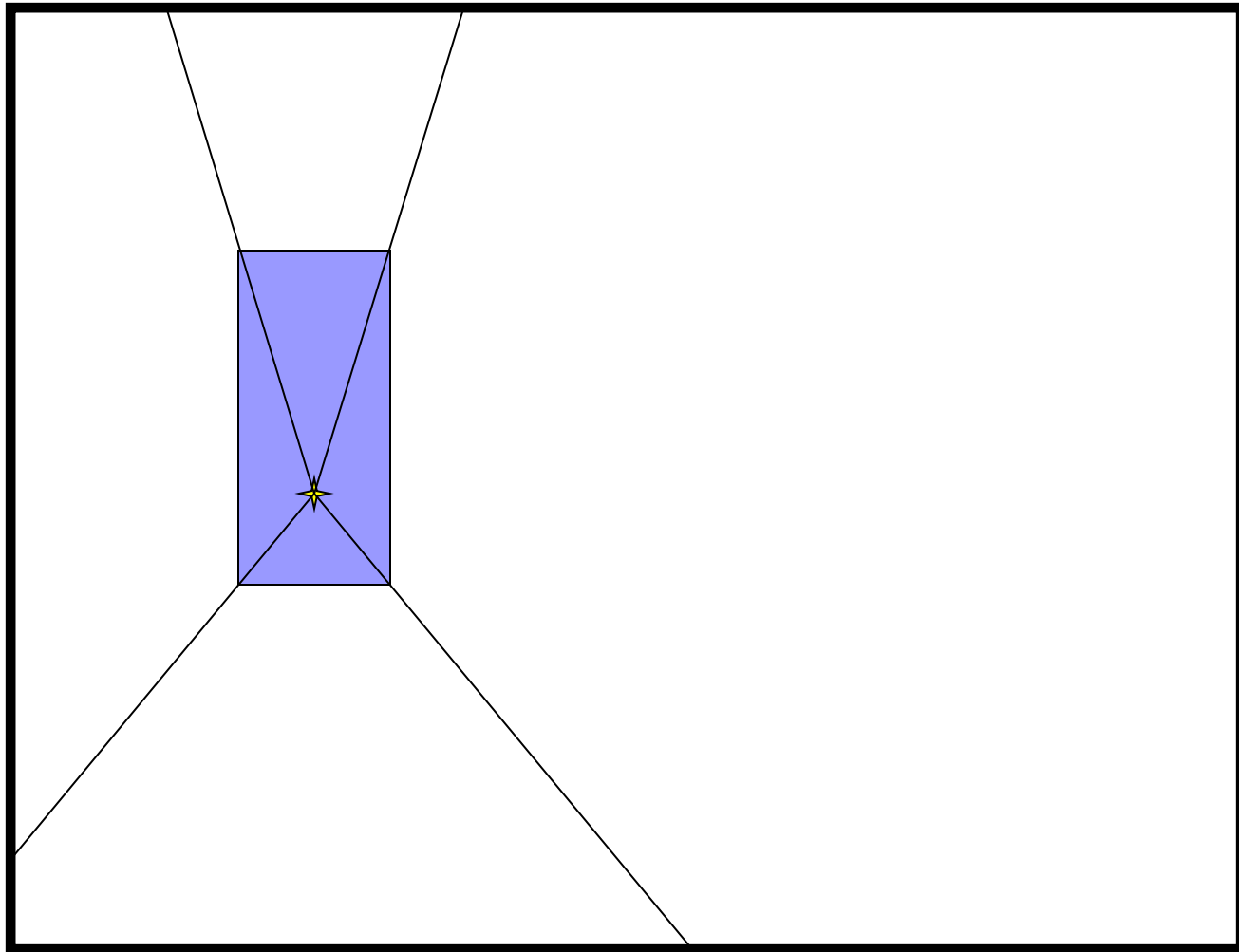


High
Camera

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

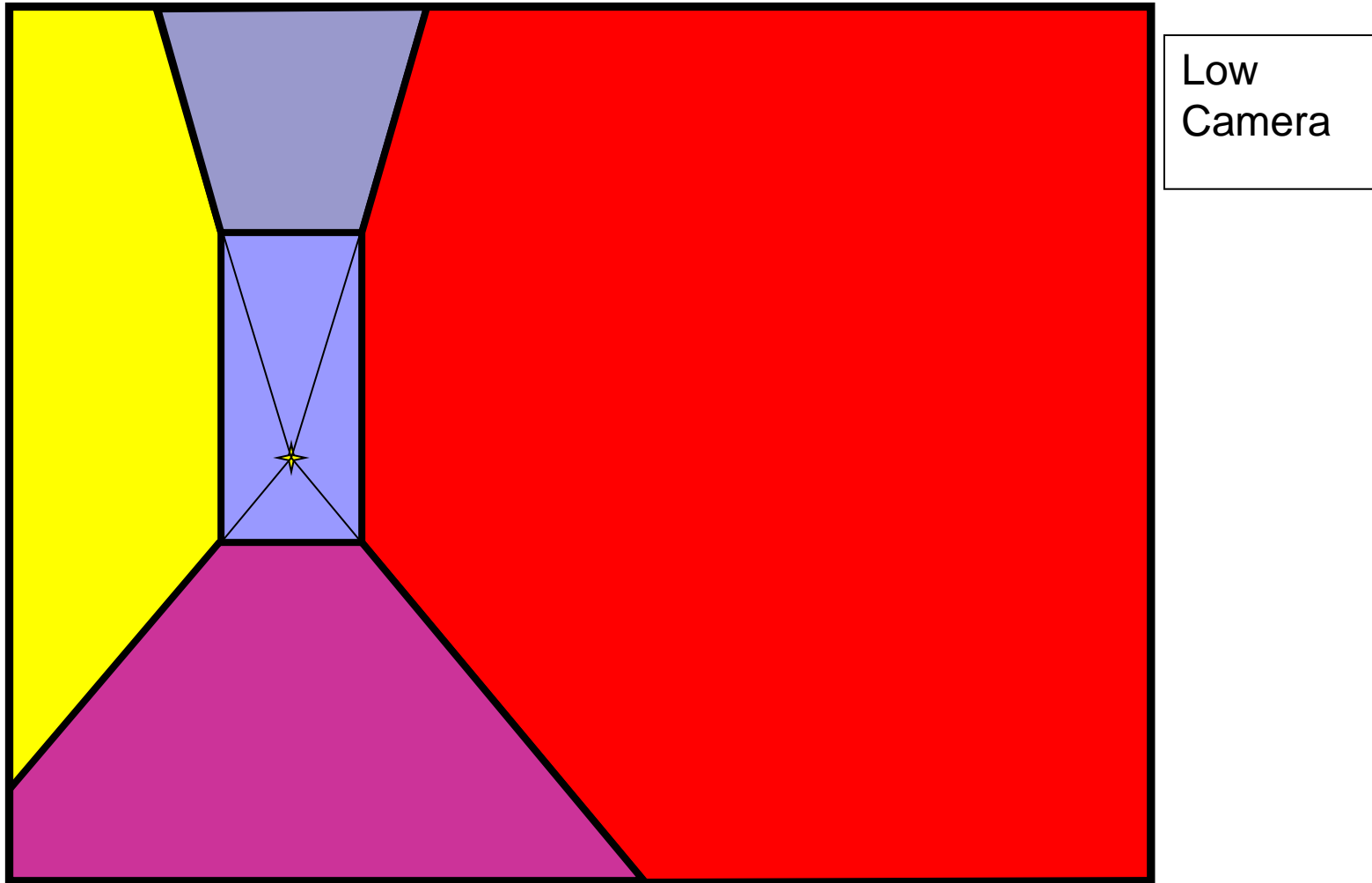


Example of user input: vanishing point and back face of view volume are defined

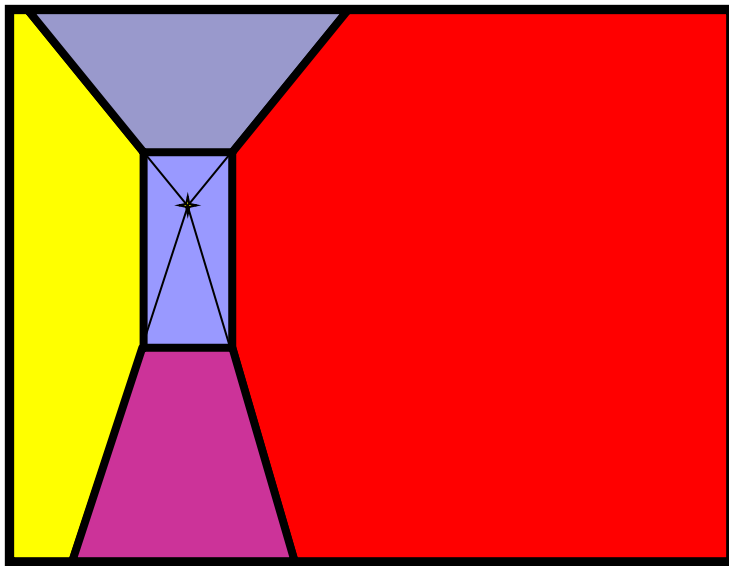


Low
Camera

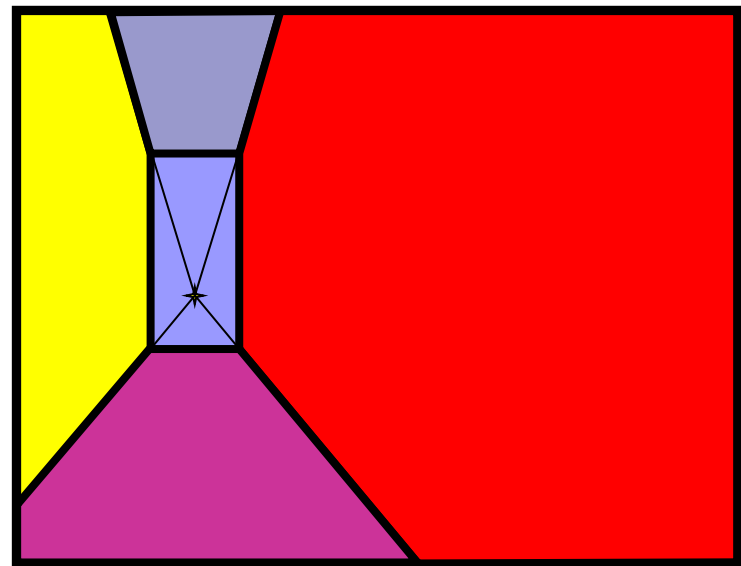
Example of user input: vanishing point and back face of view volume are defined



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.

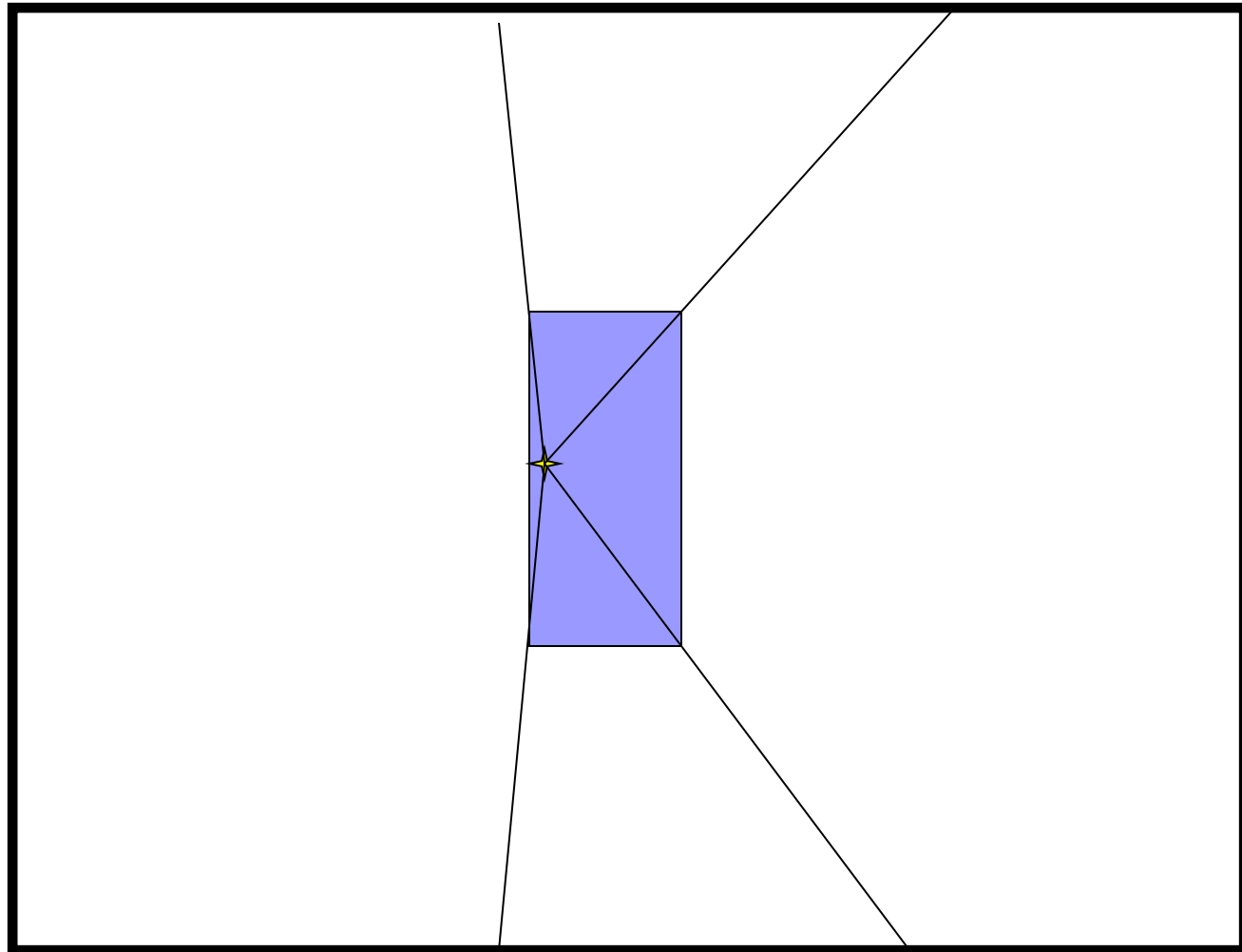


High Camera



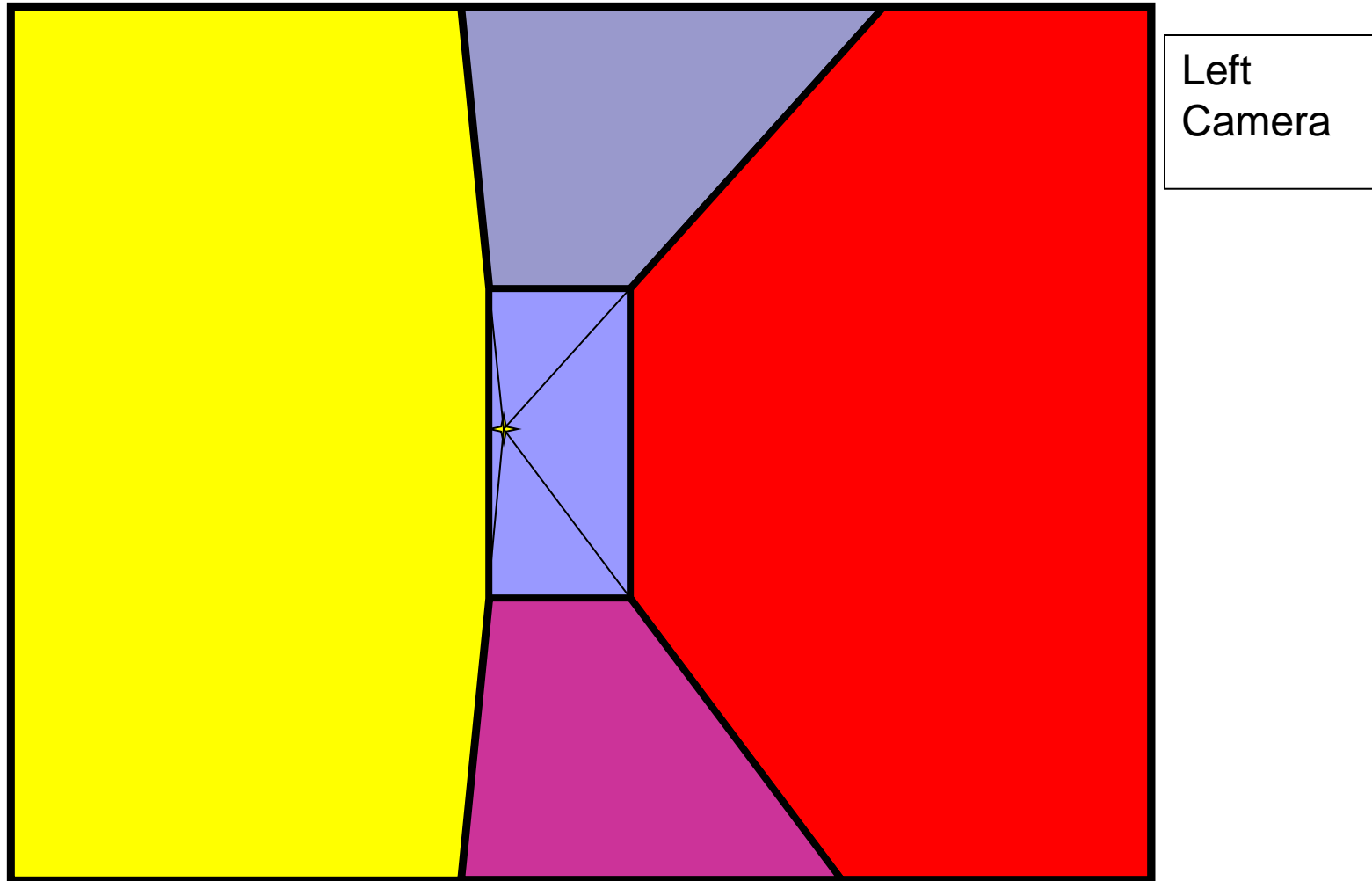
Low Camera

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

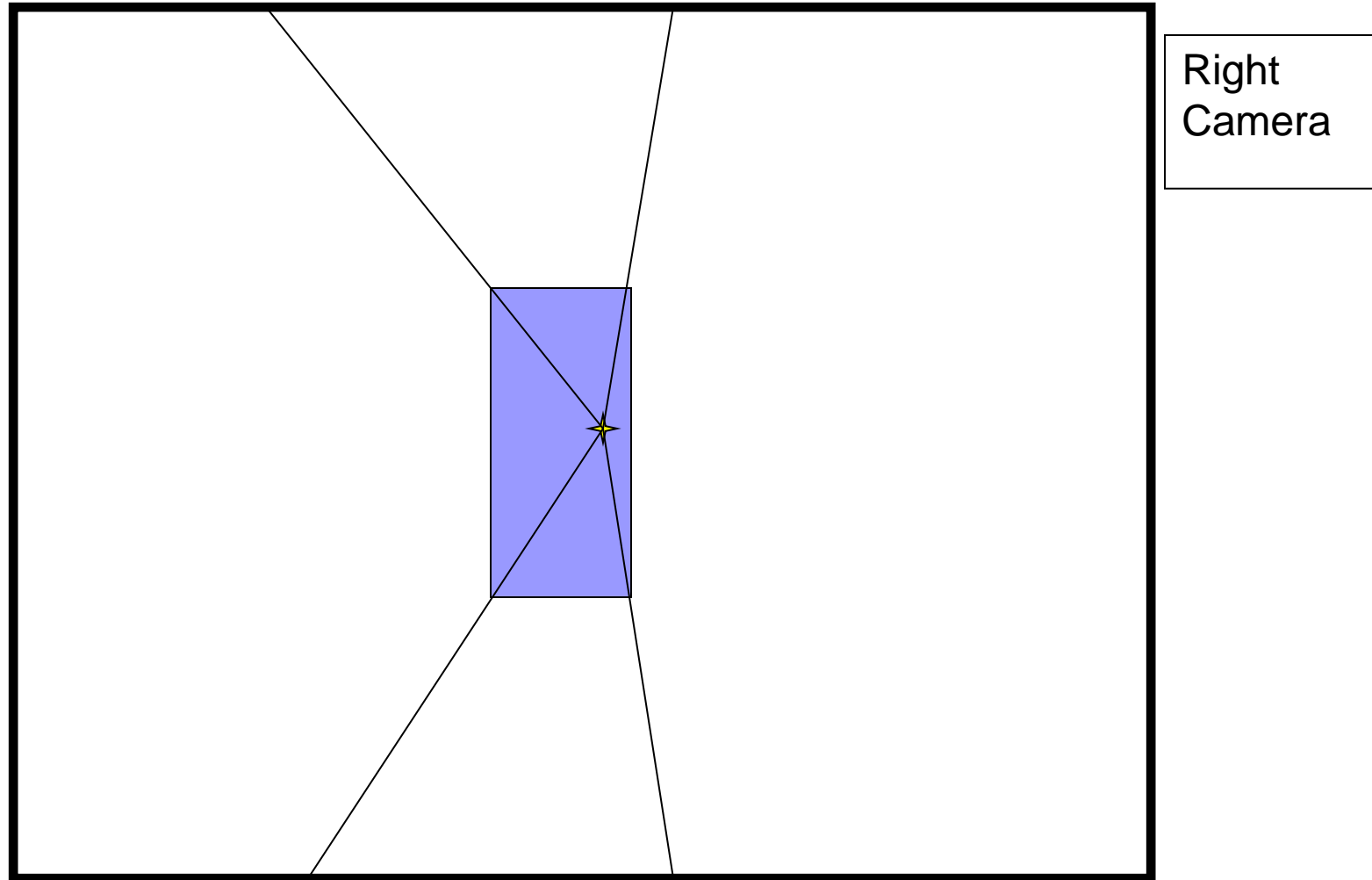


Left
Camera

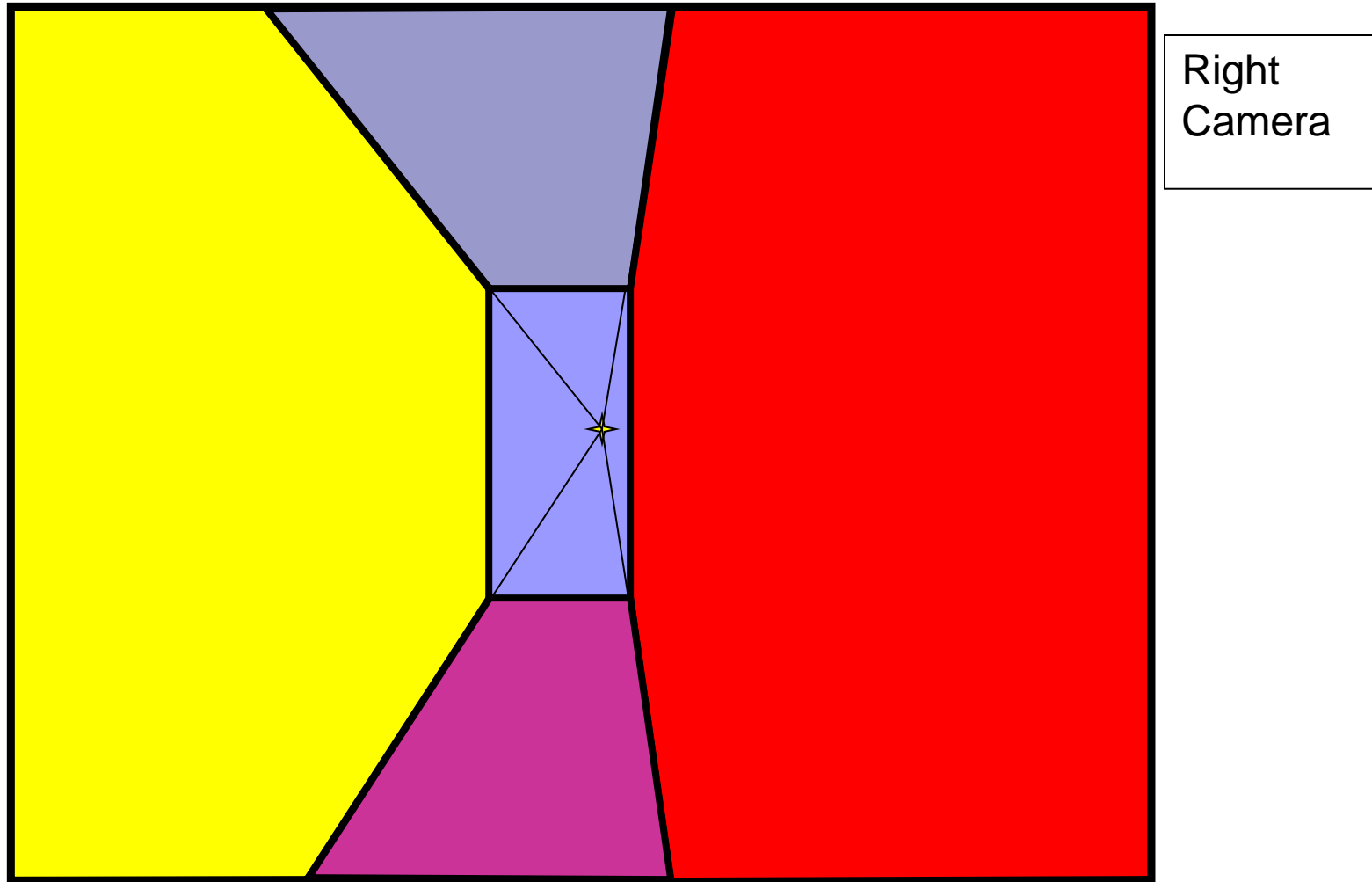
Another example of user input: vanishing point and back face of view volume are defined



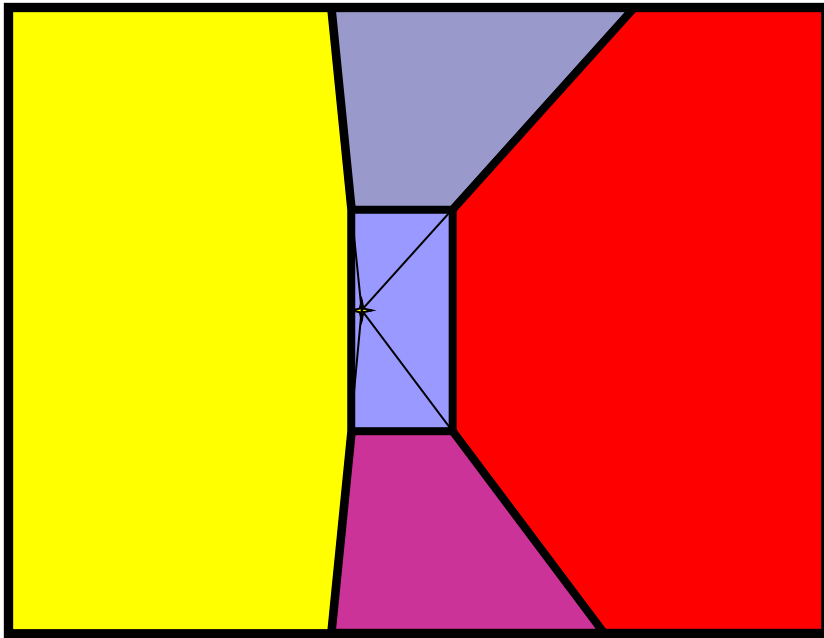
Another example of user input: vanishing point and back face of view volume are defined



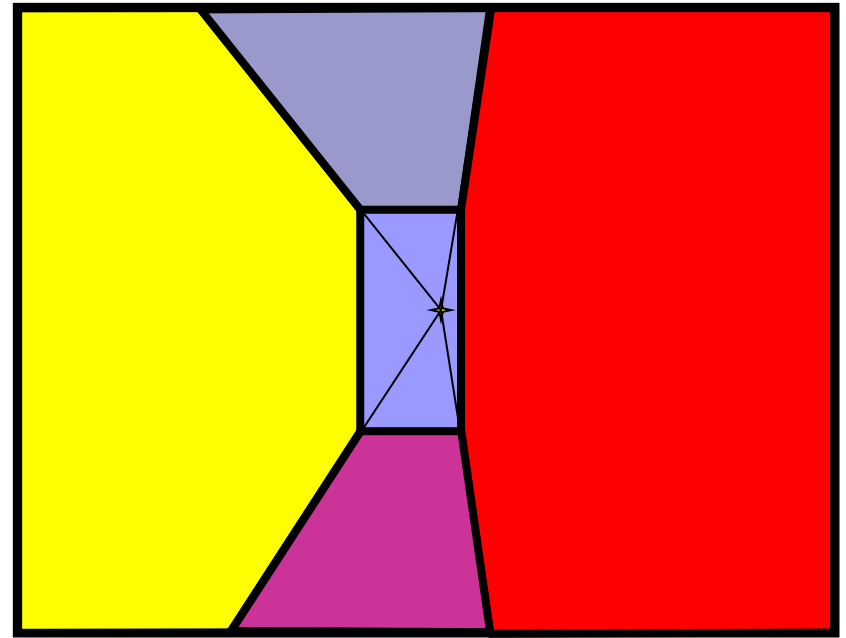
Another example of user input: vanishing point and back face of view volume are defined



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



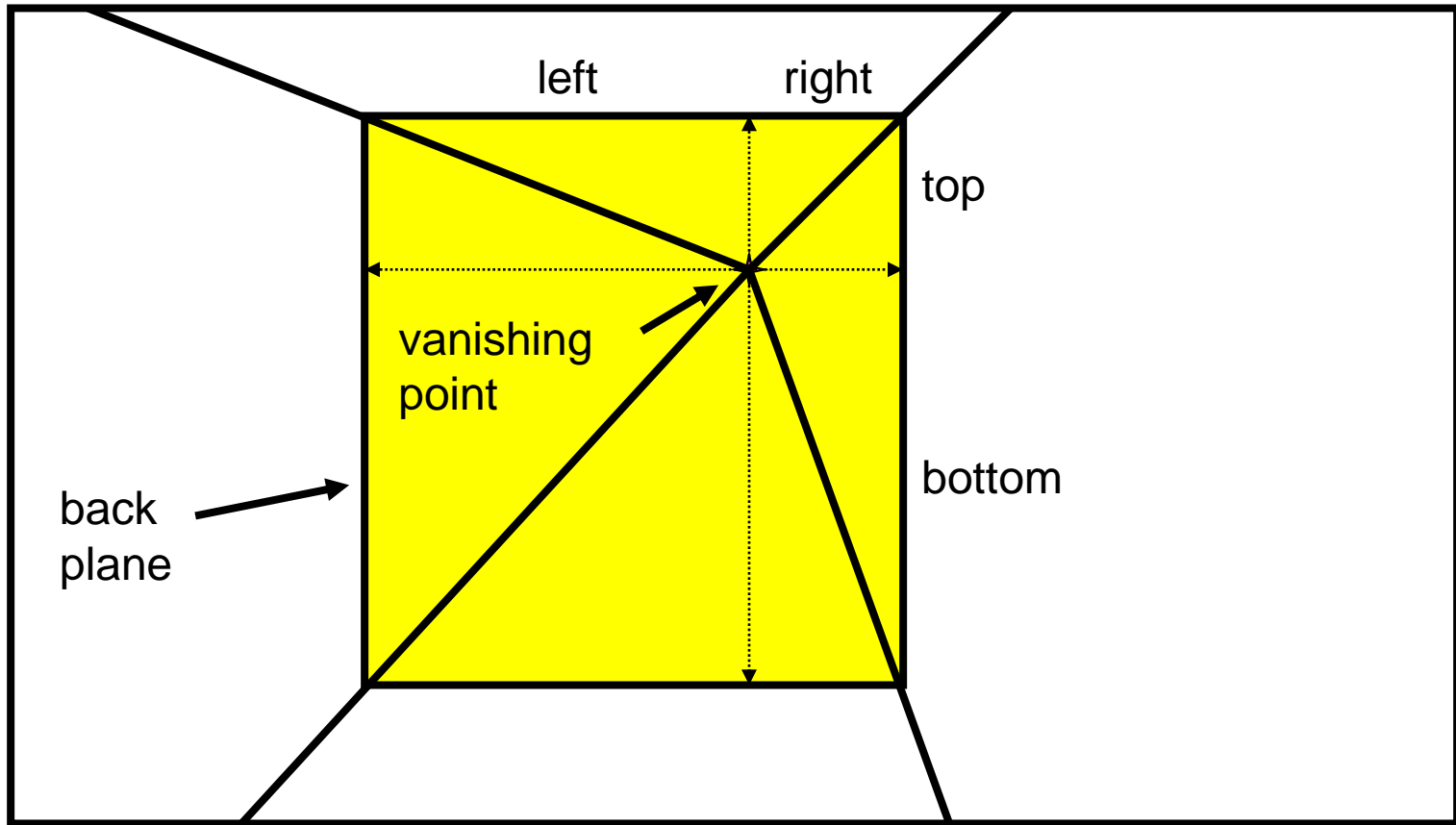
Left Camera



Right Camera

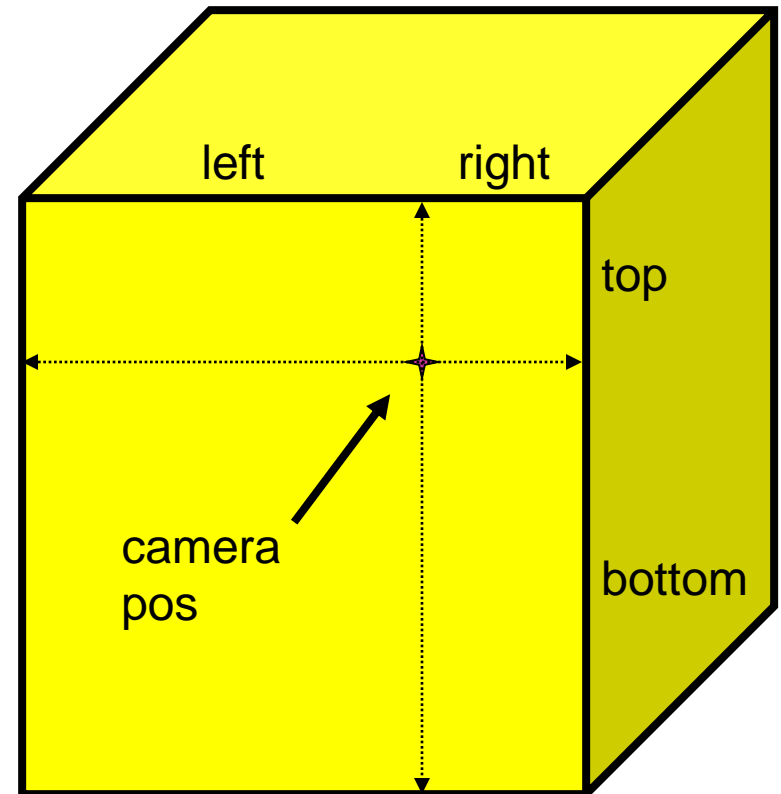
2D to 3D conversion

- First, we can get ratios

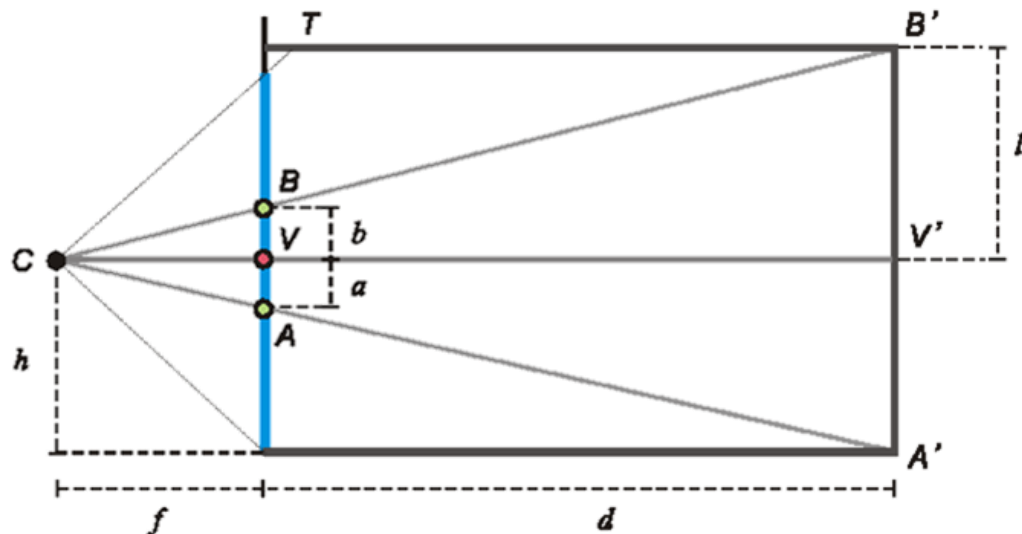
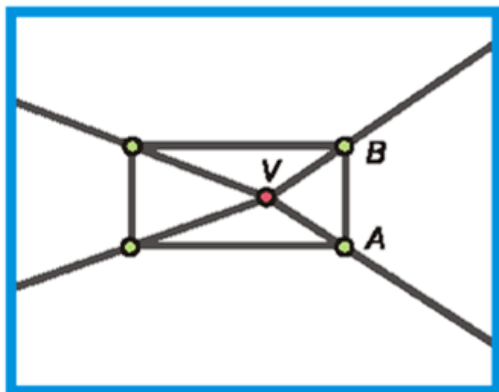


2D to 3D conversion

- 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



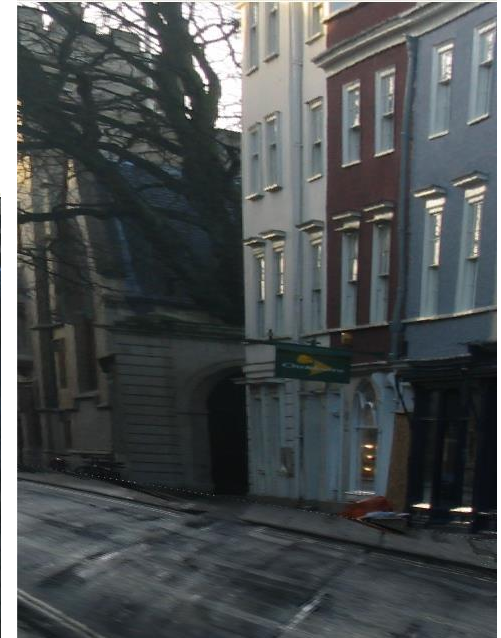
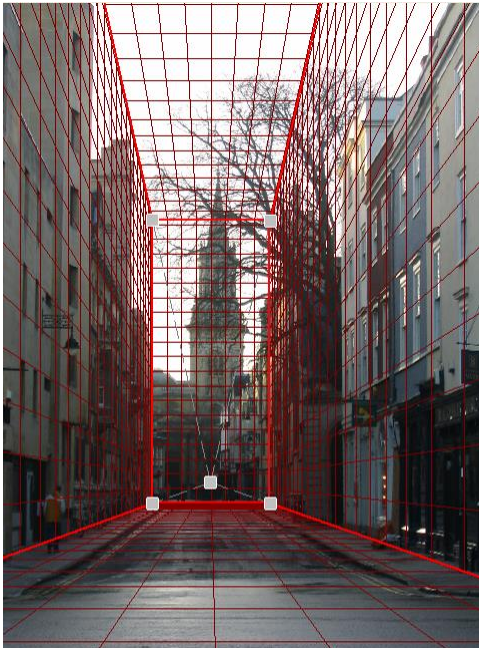
Depth of the box



- 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?

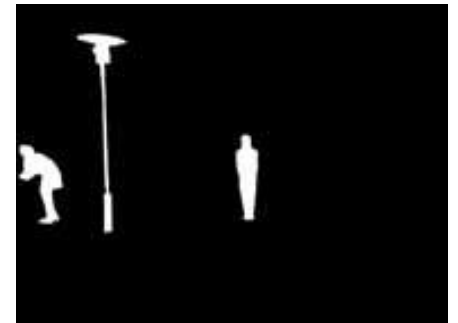
DEMO

- 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

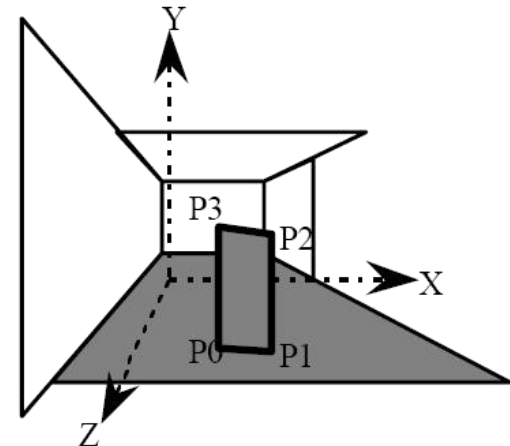


Foreground Objects

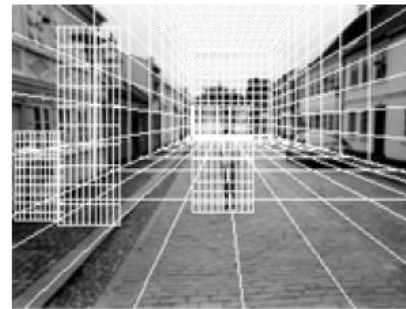
- Add vertical rectangles for each foreground object
- Can compute 3D coordinates P_0 , P_1 since they are on known plane.
- P_2 , P_3 can be computed as before (similar triangles)



(a) Specifying of a foreground object

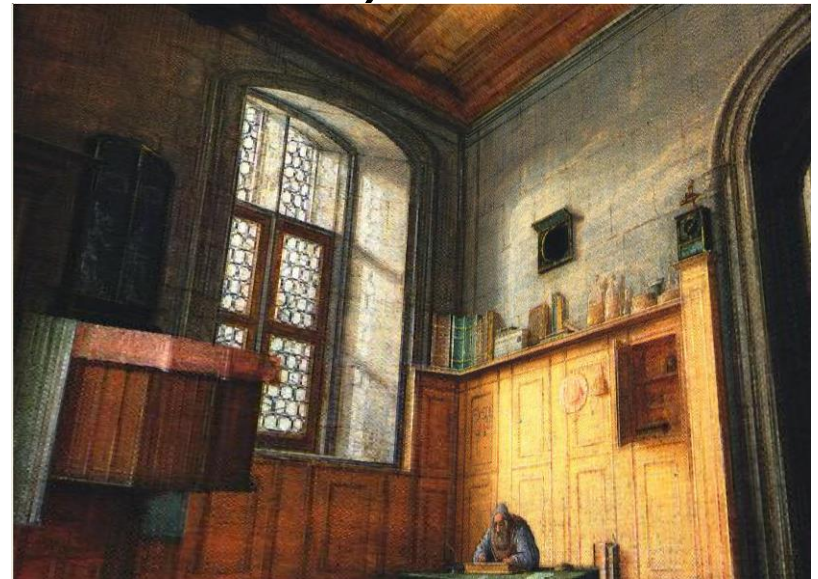


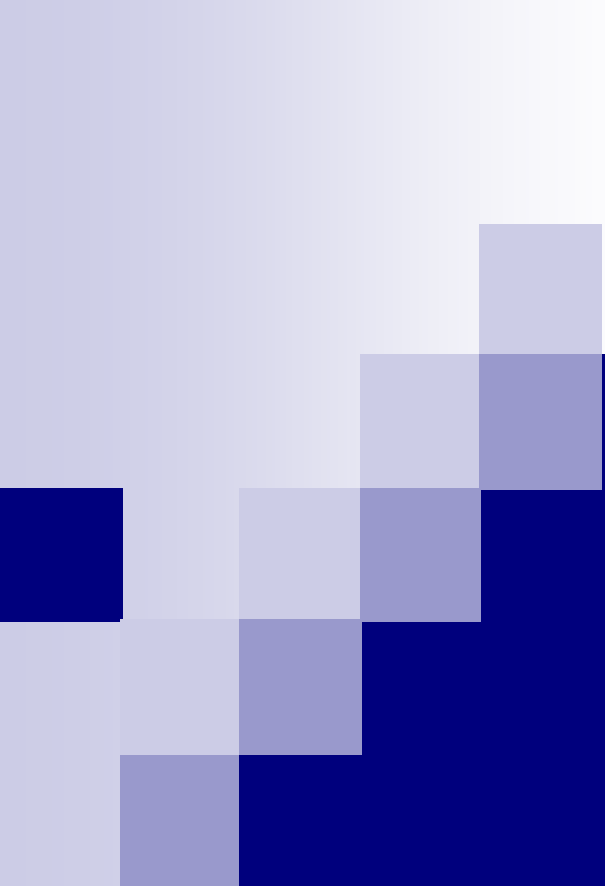
(b) Estimating the vertices of the foreground object model



(c) Three foreground object models

Foreground DEMO (and video)





Single View Modeling using Learning

Depth Map Prediction from a Single Image using a Multi-Scale Deep Network

主要思想:

在原始输入上建立两个网络，一个粗糙尺度网络线在全局上预测场景的深度。接着用一个精细尺度网络来精细化局部区域。这样局部网络可以编辑全局预测来合并精细尺度细节。并且提出了一个尺度不变的损失函数。

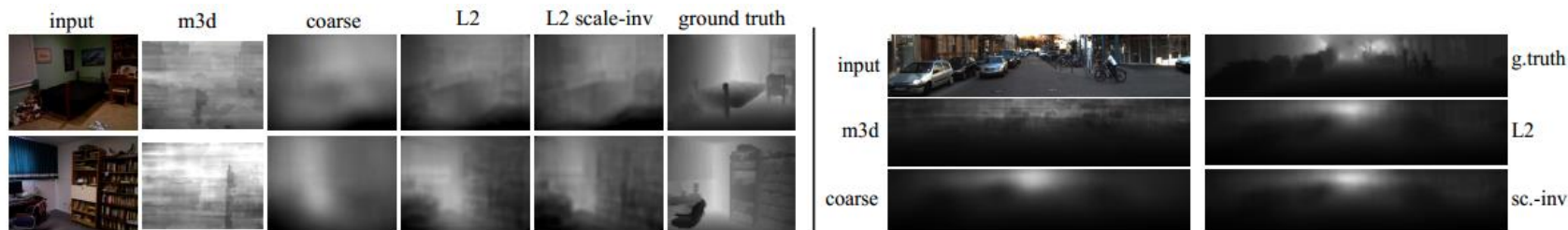
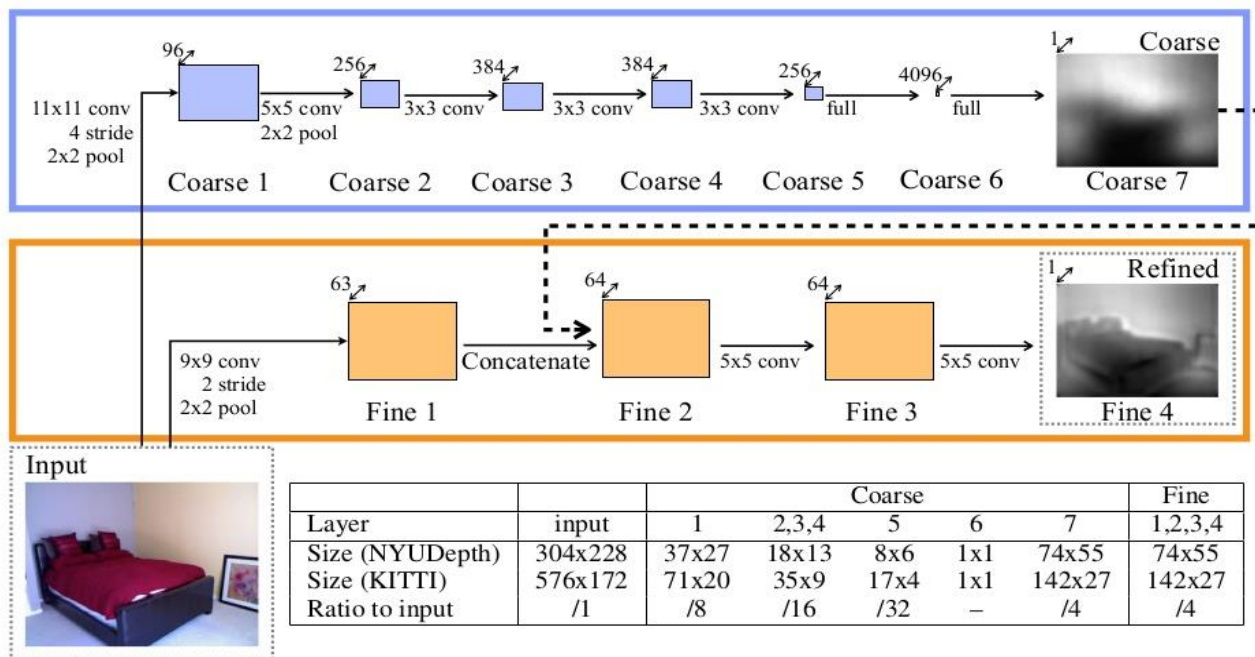
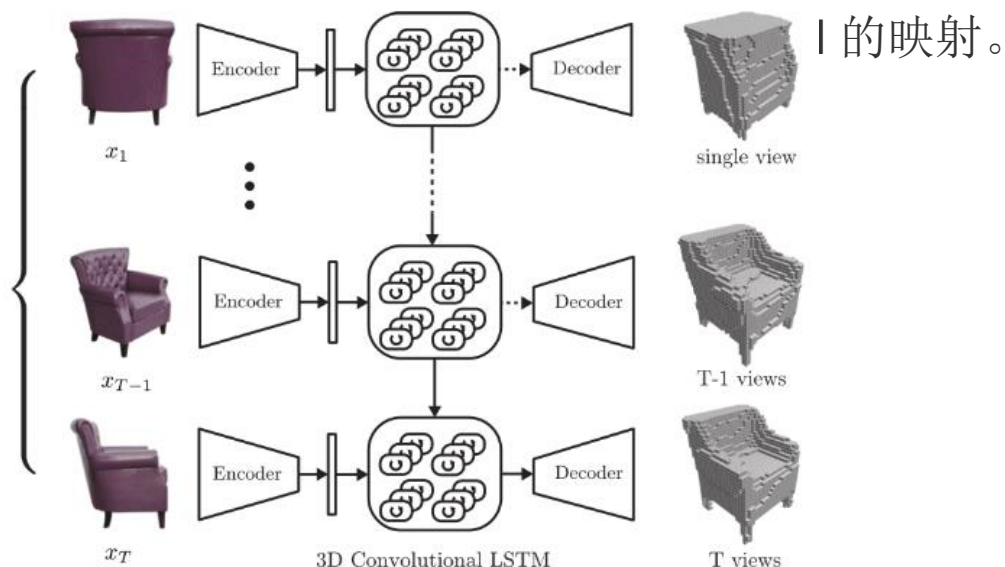


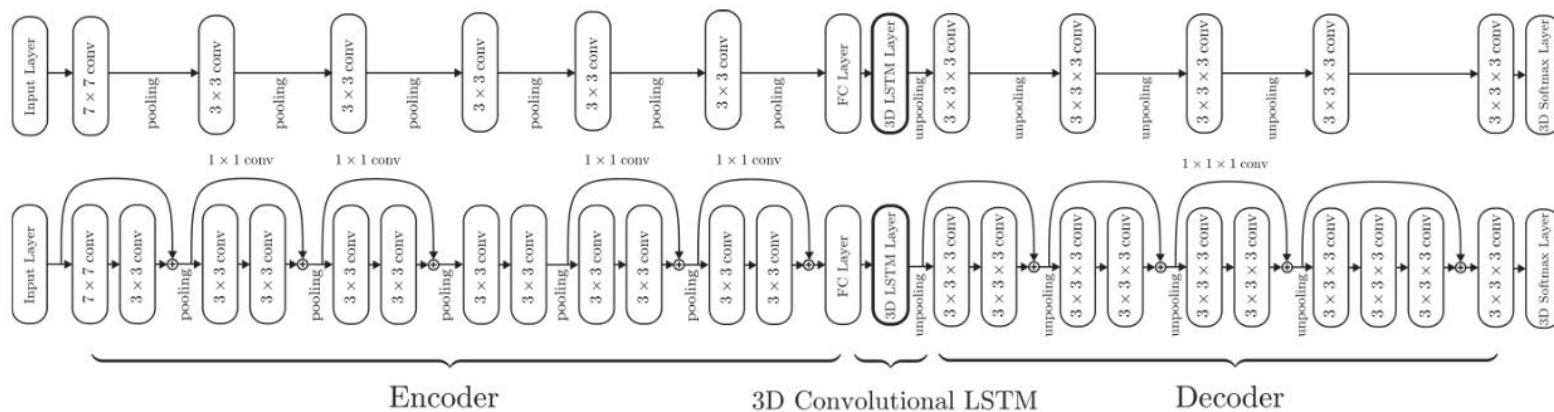
Figure 3: Qualitative comparison of Make3D, our method trained with l_2 loss ($\lambda = 0$), and our method trained with both l_2 and scale-invariant loss ($\lambda = 0.5$).

3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction

通



(a) Images of objects we wish to reconstruct (b) Overview of the network



3D-R2N2: A Unified Approach for Single and Multi-view 3D Object Reconstruction

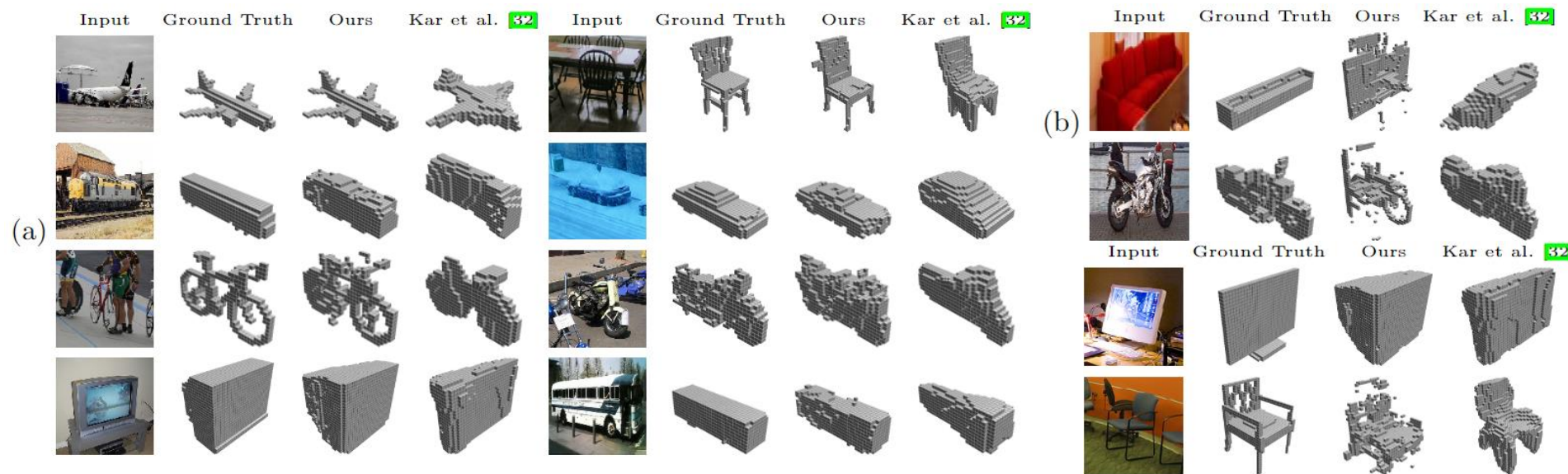
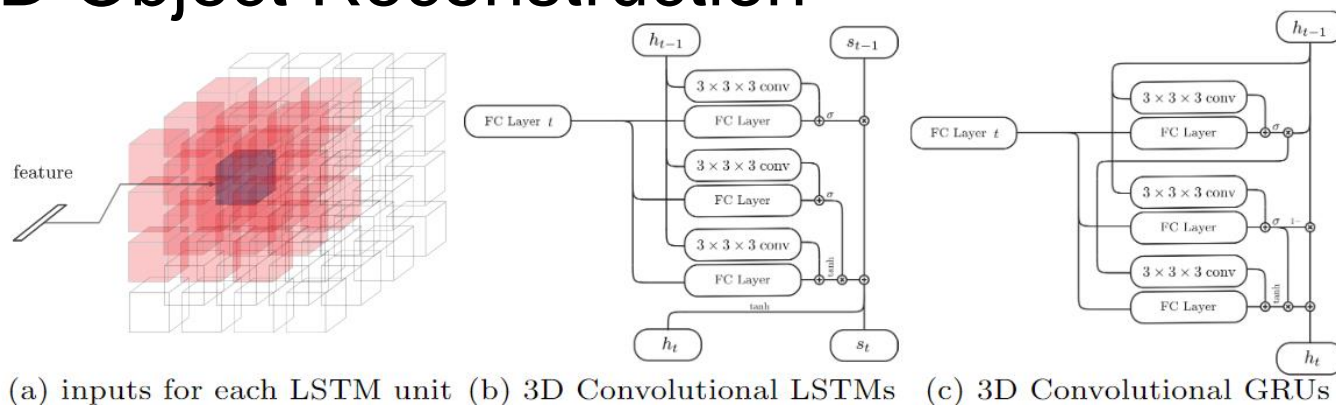


Fig. 4. Single Real-World Image Reconstruction. (a) Reconstruction samples of PASCAL VOC dataset. (b) Failed reconstructions on the PASCAL VOC dataset. Note that Kar et al. [32] is trained/tested percategory and takes ground-truth object segmentation masks and keypoint labels as additional input.

A Point Set Generation Network for 3D Object Reconstruction from a Single Image

主要思想:

利用深度网络通过单张图像直接生成点云，解决了基于单个图片对象生成3D几何的问题。

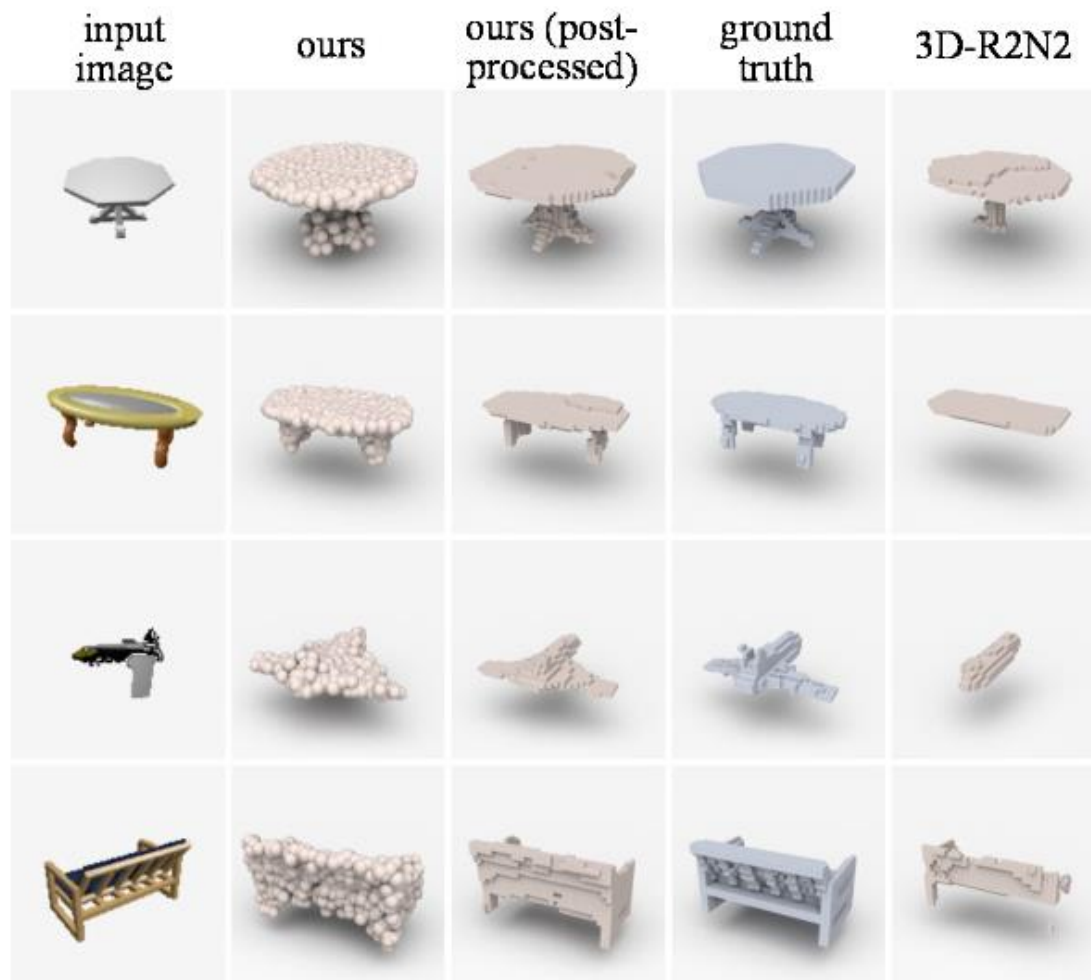
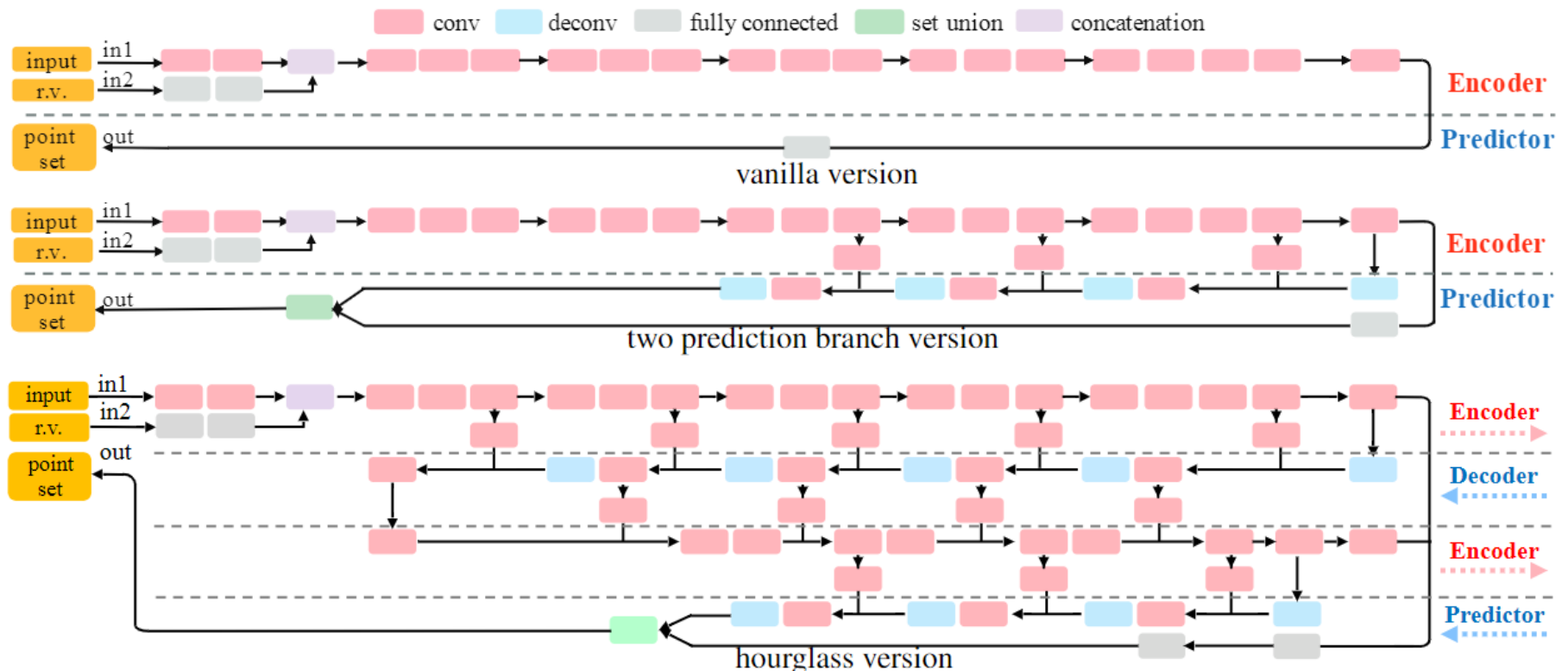


Figure 6. Visual comparison to 3D-R2N2. Our method better preserves thin structures of the objects.

A Point Set Generation Network for 3D Object Reconstruction from a Single Image



PointOutNet structure

Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images

主要思想：用一个椭球作为任意物体的初始形状，然后逐渐将这个形状变成目标物体。不借助点云、深度或者其他更加信息丰富的数据，而是直接从单张彩色图片直接得到 3D mesh

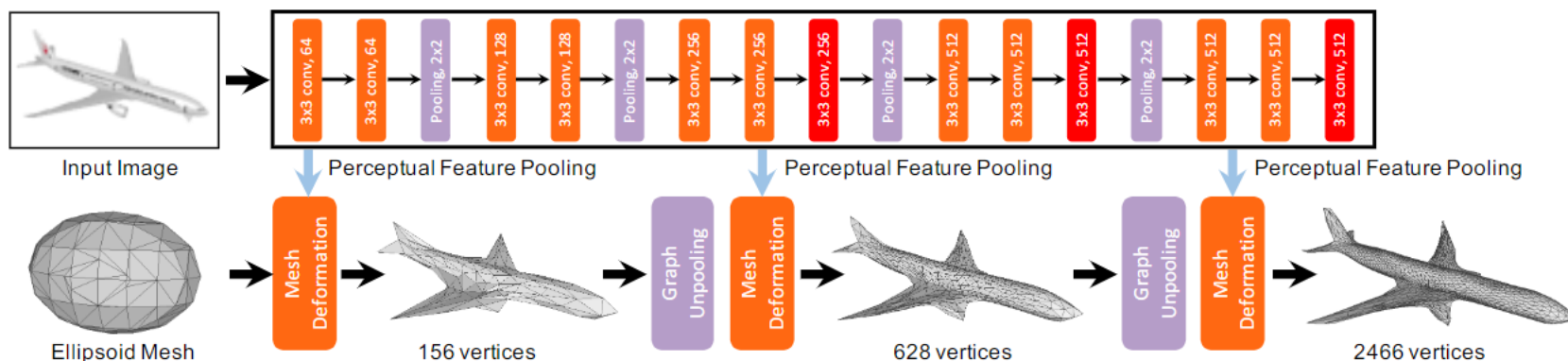
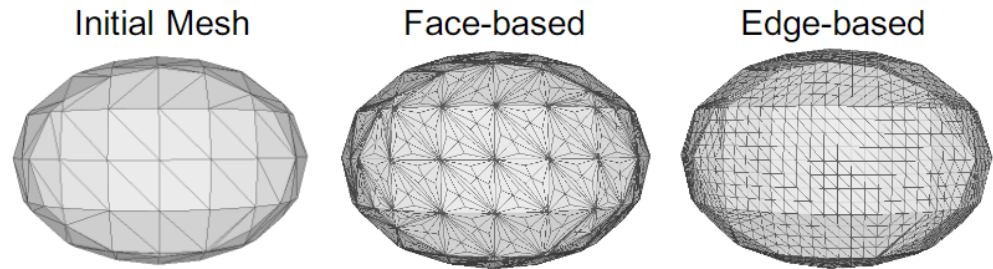
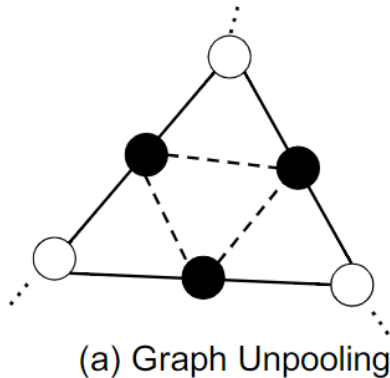


Fig. 2. The cascaded mesh deformation network. Our full model contains three mesh deformation blocks in a row. Each block increases mesh resolution and estimates vertex locations, which are then used to extract perceptual image features from the 2D CNN for the next block.

Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images



Chamfer Loss

$$l_c = \sum_p \min_q \|p - q\|_2^2 + \sum_q \min_p \|p - q\|_2^2.$$

Normal Loss

$$l_n = \sum_p \sum_{q=\arg \min_q (\|p-q\|_2^2)} \|\langle p - k, \mathbf{n}_q \rangle\|_2^2, \text{ s.t. } k \in \mathcal{N}(p)$$

Laplacian Regularization

$$l_{lap} = \sum_p \|\delta'_p - \delta_p\|_2^2.$$

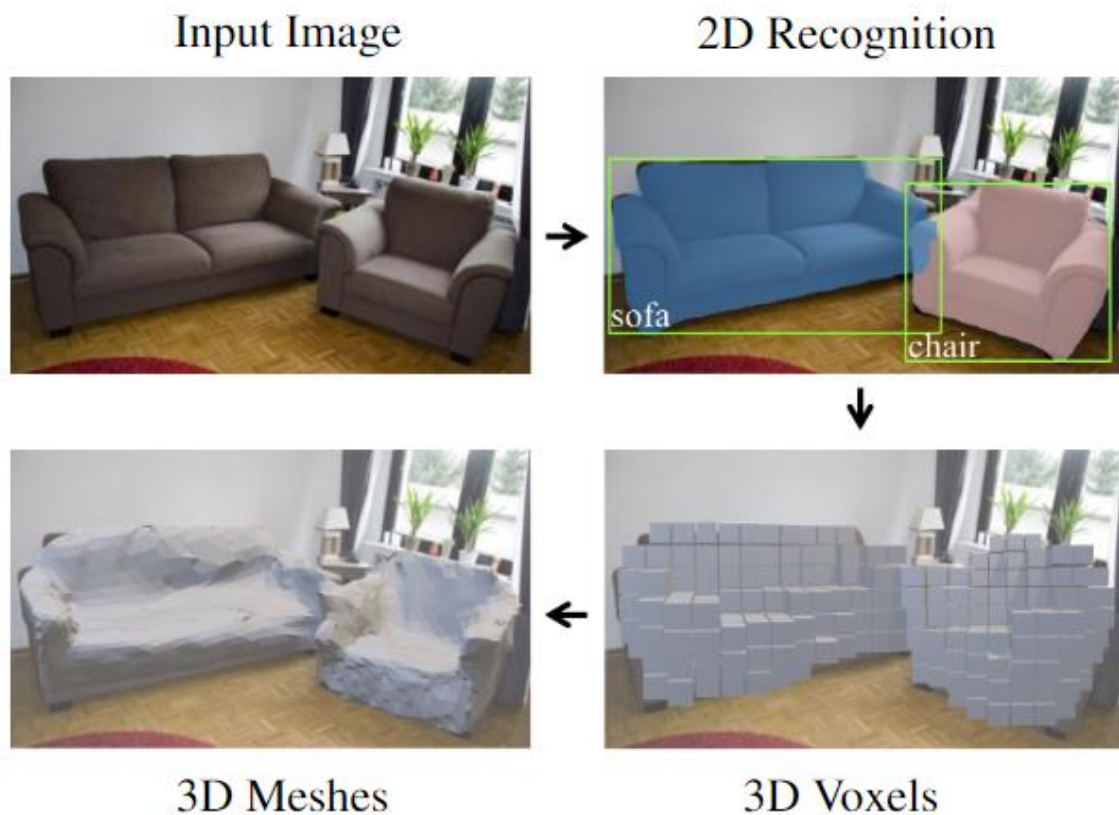
Edge length regularization

$$l_{loc} = \sum_p \sum_{k \in \mathcal{N}(p)} \|p - k\|_2^2.$$

Learning with Convolutional Networks(CNNs)

- Online courses: <http://cs231n.stanford.edu/>,
<https://zh.coursera.org/learn/neural-networks>
- Books: <http://www.deeplearningbook.org/>
- Papers: <https://github.com/kjw0612/awesome-deep-vision/tree/master>
- Platforms: [Pytorch](#), [Tensorflow](#), [MXNet](#), [Caffe](#)

Supervised Learning with Convolutional Networks(CNNs)



Mesh R-CNN是基于Mask R-CNN的增强网络，输入一个图像，检测图像中的所有对象，并输出所有对象的类别标签，边界框、分割掩码以及三维三角形网格。

Figure 1. Mesh R-CNN takes an input image, predicts object instances in that image and infers their 3D shape. To capture diversity in geometries and topologies, it first predicts coarse voxels which are refined for accurate mesh predictions.

Supervised Learning with Convolutional Networks(CNNs)

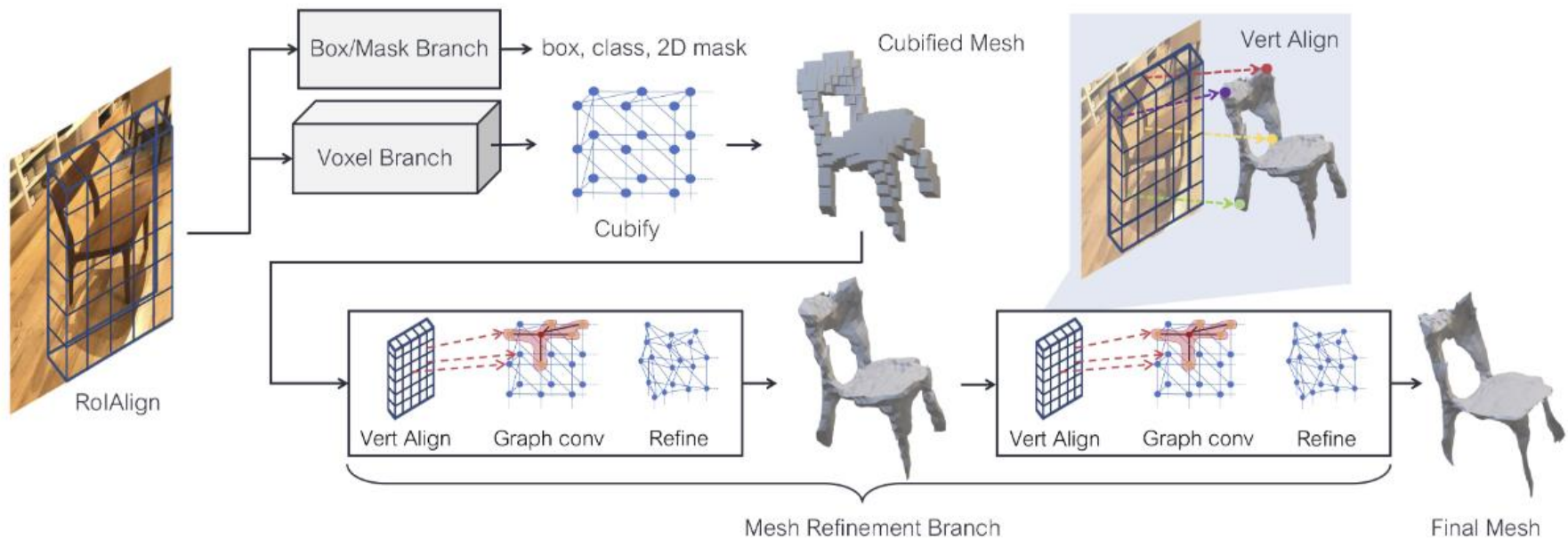
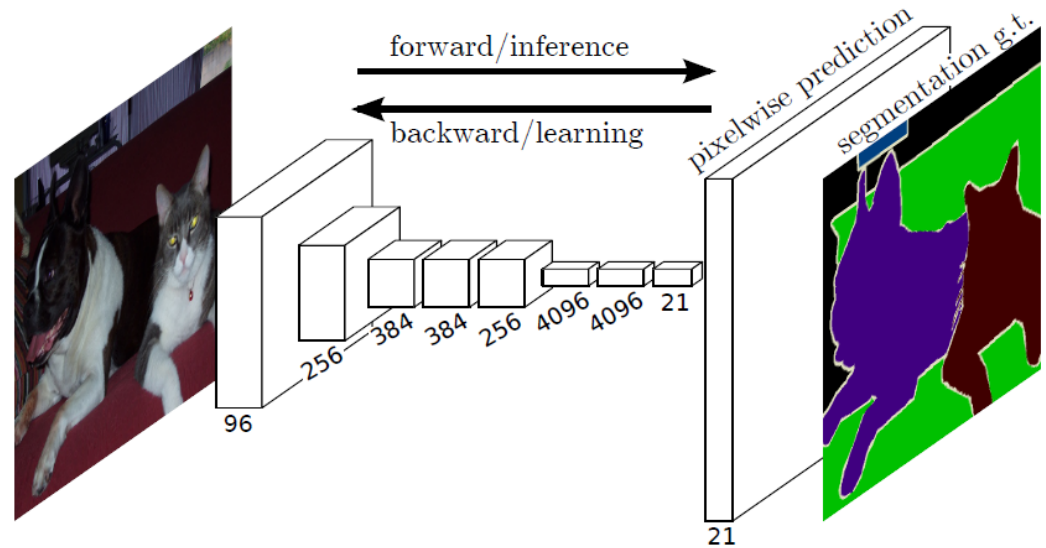


Figure 3. System overview of Mesh R-CNN. We augment Mask R-CNN with 3D shape inference. The *voxel branch* predicts a coarse shape for each detected object which is further deformed with a sequence of refinement stages in the *mesh refinement branch*.

模型主框架基于mask-rcnn，使用一个额外的网格预测器来获得三维形状。

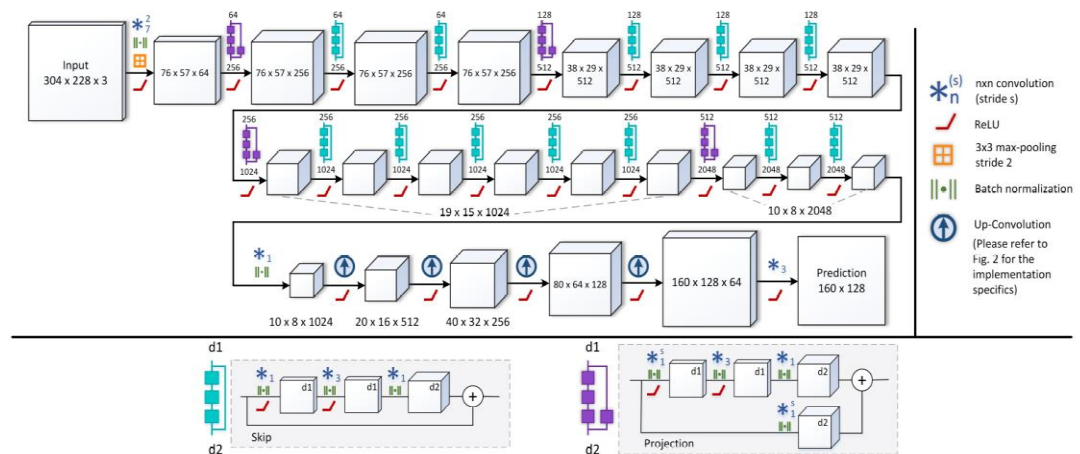
Supervised Learning with Convolutional Networks(CNNs)

- Shared same network structure with other pixel level tasks(e.g. semantic segmentation)
- An end-to-end learning way



Supervised Learning with Convolutional Networks(CNNs)

- Works focus on designing network structures
- Learn depth with more powerful features, consistent with other advances in CNNs



Supervised Learning with Convolutional Networks(CNNs)

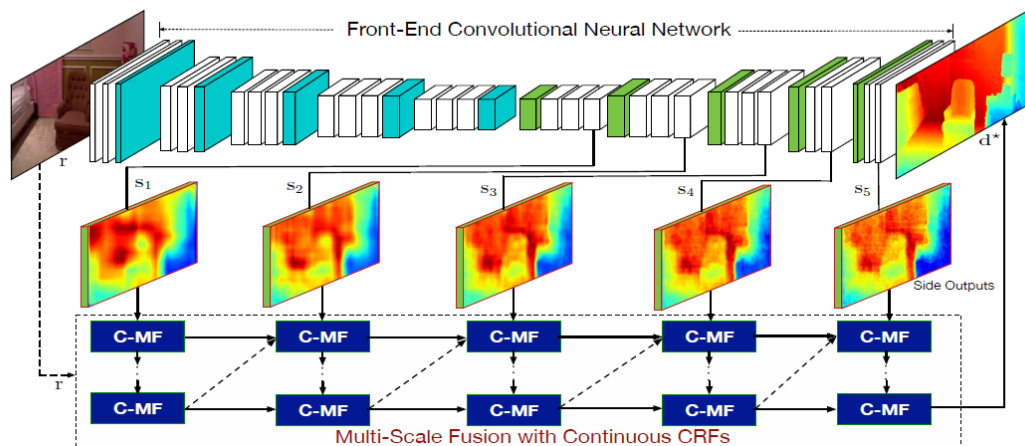
- Works focus on designing loss functions
- Constrain desired property into the loss functions

$$L_{depth}(D, D^*) = \frac{1}{n} \sum_i d_i^2 - \frac{1}{2n^2} \left(\sum_i d_i \right)^2 + \frac{1}{n} \sum_i [(\nabla_x d_i)^2 + (\nabla_y d_i)^2] \quad (1)$$

Not only focus on depth information but also on corresponding gradients

Supervised Learning with Convolutional Networks(CNNs)

- Works focus on post processing methods
- Mainly relied on Conditional Random Fields (CRFs) to recover more scene details



Supervised Learning with Convolutional Networks(CNNs)

- Works focus on combining highly related works to boost each other
- The highly related works have some shared properties that can be used

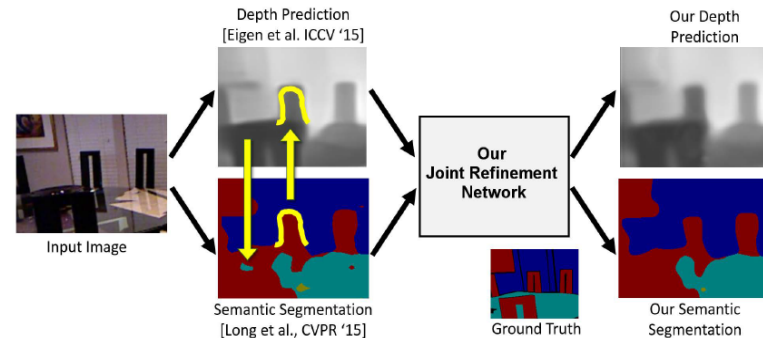
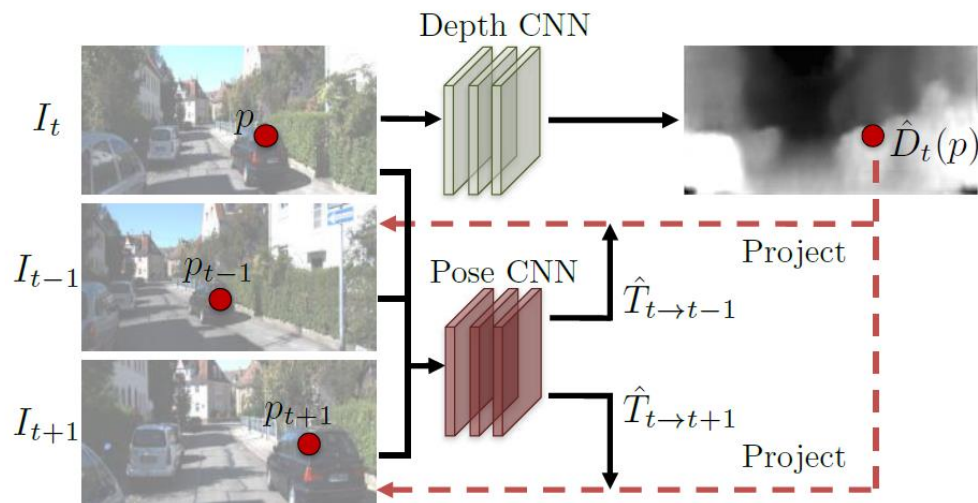


Fig. 1. Example processing flow of our joint refinement network. A single RGB image is first processed separately by two state-of-the-art neural networks for depth estimation and semantic segmentation. The two resulting predictions contain information which can mutually improve each other: (1) yellow arrow from depth to semantic segmentation means that a smooth depth map does not support an isolated region (cyan means furniture); (2) yellow arrow from semantic segmentation to depth map means that the exact shape of the chair can improve the depth outline of the chair. (3) In most areas the two modalities positively enforce each other (e.g. the vertical wall (dark blue) supports a smooth depth map. The *cross-modality influences* between the two modalities are exploited by our joint refinement network, which fuses the features from the two input prediction maps and jointly processes both modalities for an overall prediction improvement. (Best viewed in color.)

Unsupervised Learning with Convolutional Networks(CNNs)

- Learning without ground-truth depth information
- Modeling the learning target with video sequences



Unsupervised Learning with Convolutional Networks(CNNs)

- The key supervision signal is the view synthesis
- synthesize a target view given a per-pixel depth in that image, plus the pose and visibility in a nearby view

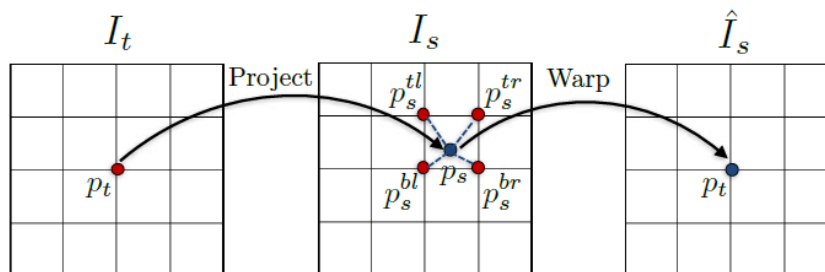


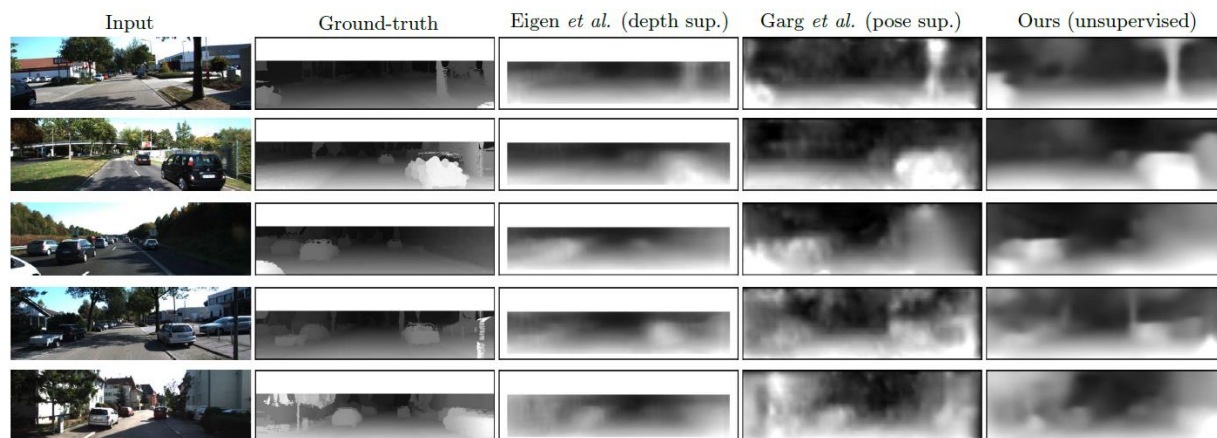
Figure 3. Illustration of the differentiable image warping process. For each point p_t in the target view, we first project it onto the source view based on the predicted depth and camera pose, and then use bilinear interpolation to obtain the value of the warped image \hat{I}_s at location p_t .

$$\mathcal{L}_{vs} = \sum_s \sum_p |I_t(p) - \hat{I}_s(p)|,$$

Unsupervised Learning with Convolutional Networks(CNNs)

- Those pixels at moving objects should not be taken into consideration
- So a cross-entropy loss with constant label 1 at each pixel location is optimized to obtain the mask

Unsupervised Learning with Convolutional Networks(CNNs)



Promising results can be achieved with carefully design

Method	Seq. 09	Seq. 10
ORB-SLAM (full)	0.014 ± 0.008	0.012 ± 0.011
ORB-SLAM (short)	0.064 ± 0.141	0.064 ± 0.130
Mean Odom.	0.032 ± 0.026	0.028 ± 0.023
Ours	0.021 ± 0.017	0.020 ± 0.015

Table 3. Absolute Trajectory Error (ATE) on the KITTI odometry split averaged over all 5-frame snippets (lower is better). Our method outperforms baselines with the same input setting, but falls short of ORB-SLAM (full) that uses strictly more data.

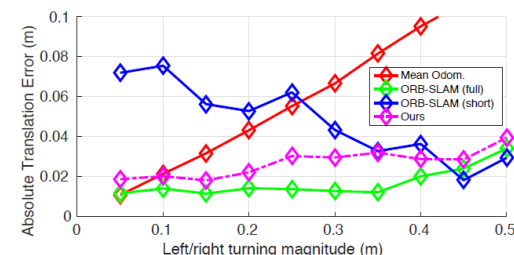


Figure 9. Absolute Trajectory Error (ATE) at different left/right turning magnitude (coordinate difference in the side-direction between the start and ending frame of a testing sequence). Our method performs significantly better than ORB-SLAM (short) when side rotation is small, and is comparable with ORB-SLAM (full) across the entire spectrum.



Semi-supervised Learning with Convolutional Networks(CNNs)

- Have been explored in a stereo setting, remains a topic for single view modeling
- It may further boost the performance of unsupervised depth learning, sparse depth can be obtained with LiDAR sensors



Thank you!