Image Transfer Methods

Satya Prakash Mallick

Jan 28th, 2003

Objective

- Given two or more images of the same scene, the objective is to synthesize a novel view of the scene from a view point where there is no camera.

Left Image **Synthesized Images** **Right Image**

Approaches

- **1. Traditional Approach:**
  - **Generate a 3D representation of the scene**
  - **Render the scene, given a view point**
  - ***Problems:***
    - ***Generating 3D from 2D images is generally very noisy.***
    - ***Generating 3D from 2D can be computationally prohibitive for real-time applications.***

Approaches

- **Light Field Rendering/ Lumigraph**
  - **It's an elegant method in which one doesn't need to compute 3D information**
  - ***Problems:***
    - ***Calculation of the light field requires additional machinery.***
    - ***Calculation of light field can be very memory intensive.***

Approaches

- **Image Transfer Methods:**
  - Only 2D image operations are used to generate the novel view. 3D is never calculated explicitly.
  - Its extremely fast. (The OpenCV implementation of Stiez's algorithm works in real time!).
  - *Problems:*
    - *Usually these methods cannot generate views from any arbitrary view point.*
    - *There is a limitation on the quality of rendering that can be achieved. Ex. Its difficult to render using a different lighting condition.*

Papers (In Chronological Order)

- **E. Chen, L. Williams,View Interpolation for Image Synthesis , *SIGGRAPH 1993***
- **S. Seitz, C. Dyer, View Morphing *SIGGRAPH, pp. 21--30, 1996*.**
- **S Avidan, A. Shashua Novel View Synthesis by Cascading Trilinear Tensors IEEE Transactions on visualization and computer graphics, Vol 4, No. 4, October-November 1998.**
- **Y. Genc, J. Ponce. Image-Based Rendering Using Parameterized Image Varieties. International Journal of Computer Vision, Vol. 41, No. 3, pp. 143-170, 2001.**

What do these paper represent?

- **1993:** Image based rendering was still at it's infancy. The first paper DOESNOT represent an image transfer method
- **1996:** One of the first image transfer methods which used two images. However, only in-between views could be generated.
- **1998:** An image transfer method using 2 cameras was introduced which use trilinear tensor to overcome the limitation of view generation along a line only.
- **2001:** A new image transfer method is introduced. It relies on minimum parameterization of image space.

Theoretical background

- *Image Morphing:* **(Term coined by T. Beier, 1992)**

$$I_0, I_1 : \text{Input images}$$

$$\left.\begin{aligned} C_0 &: I_0 \Rightarrow I_1 \\ C_1 &: I_1 \Rightarrow I_0 \end{aligned}\right\} \text{Correspondence Maps}$$

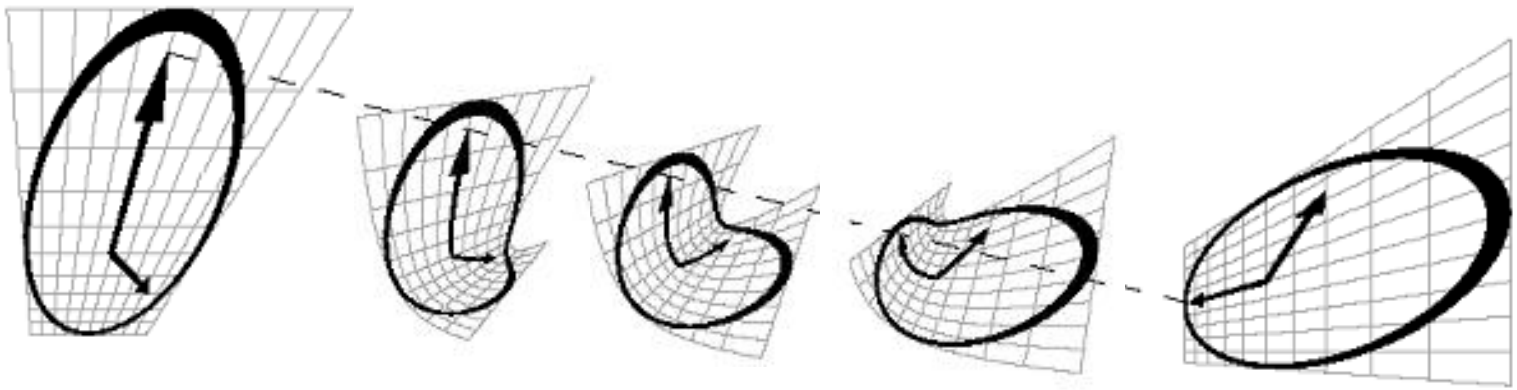
$$\left.\begin{aligned} W_0(p_0, s) &= (1-s)p_0 + sC_0(p_0) \\ W_1(p_1, s) &= (1-s)C_1(p_1) + sp_1 \end{aligned}\right\} \exists p_0 \in I_0, \exists p_1 \in I_1$$

$$s \in [0,1]$$

The warped image is an average of the pixel intensities obtained using forward warping and backward warping.

Physical Validity of Morphing

- If two images of the same object are morphed, the resulting image **may not** represent the same image geometrically.



Z - Buffering

- 1. Clear the color buffer to the background color
- 2. Initialize all *xy* coordinates in the Z buffer to one
- 3. For each fragment of each surface, compare depth values to those already stored in the *Z* buffer
  - - Calculate the distance from the projection plane for each xy position on the surface
  - - If the distance is less than the value currently stored in the *Z* buffer:
    - Set the corresponding position in the color buffer to the color of the fragment
    - Set the value in the *Z* buffer to the distance to that object
  - - Otherwise:
    - Leave the color and *Z* buffers unchanged

View Interpolation

- Given:
  - Many images of the same scene.
  - Their range data (3D is already given !).
  - Camera Transformation
- Objective:
  - To have a smooth navigation system through the scene (Virtual Walkthrough).
- Note: This is NOT an image transfer method.

View Interpolation

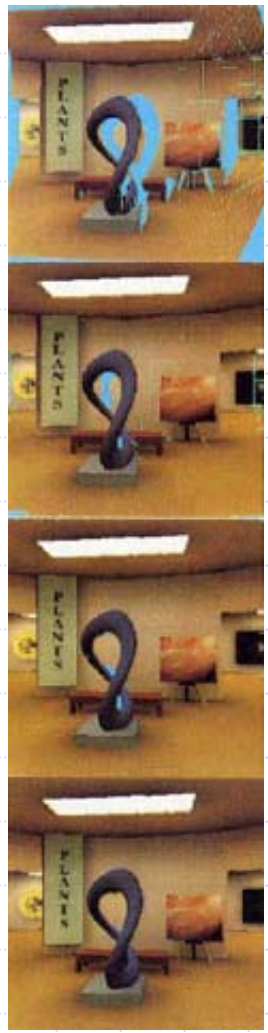
The algorithm:

1. **Pixel Correspondence**
   - **Since Pixel's screen coordinates are known (x,y,z) and the relative orientation of the cameras is also known, a 4x4 transformation relates the points in one image to other.**
   - **The above transformation can be stored as a offset vector for each pixel. This is called "morph map".**
2. **Interpolating Correspondences:**
   - **Offset vectors are interpolated linearly and the pixels in the source image are moved by the interpolated vector to the destination image.**

View Interpolation

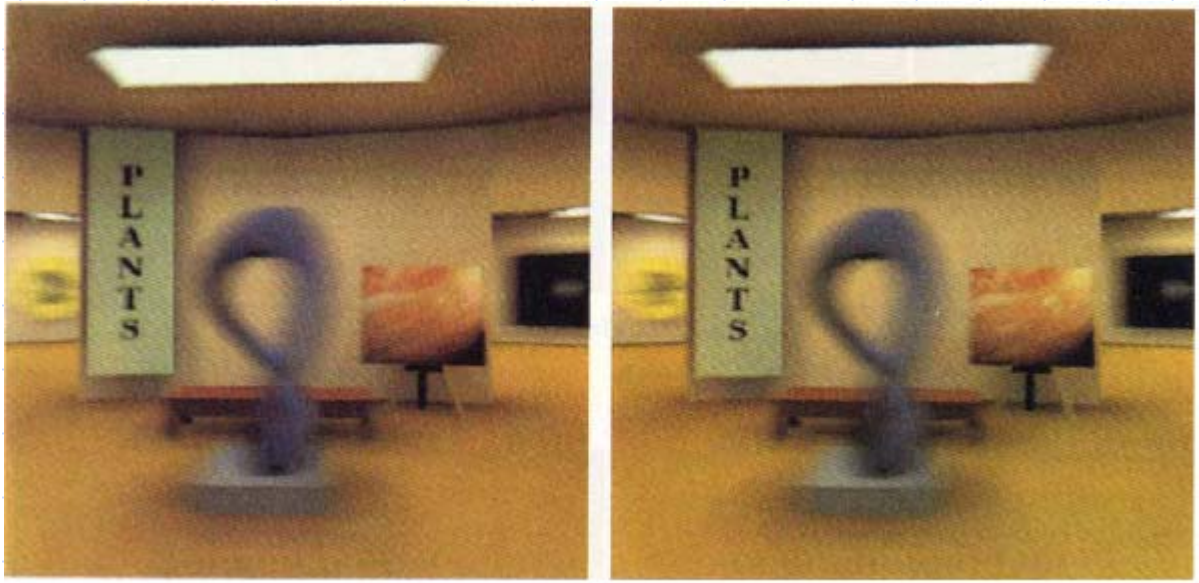
The algorithm (Contd.)

2. **Interpolating Correspondences:**
   - **The interpolation is an approximation of pixel coordinate transformation by a perspective viewing matrix.**
3. **Composting Images:**
   - **Visibility Resolution:**
     - **Z-buffering can be used.**
     - **A view independent visible priority can be used.**
   - **Holes:**
     - **Can be filled using interpolation of color from neighboring pixels.**
4. **Block Compression:**
   - **Neighboring pixel move together. Hence morphing map can be defined for blocks of images rather than pixels.**



Results

Results



Drawbacks

For real scenes having dense disparity or/and perfect correspondence between pixels is extremely difficult. The underlying assumptions make it suitable only for synthetic data. No wonder, only synthetic results were shown.

The Geometry of Multiple Views

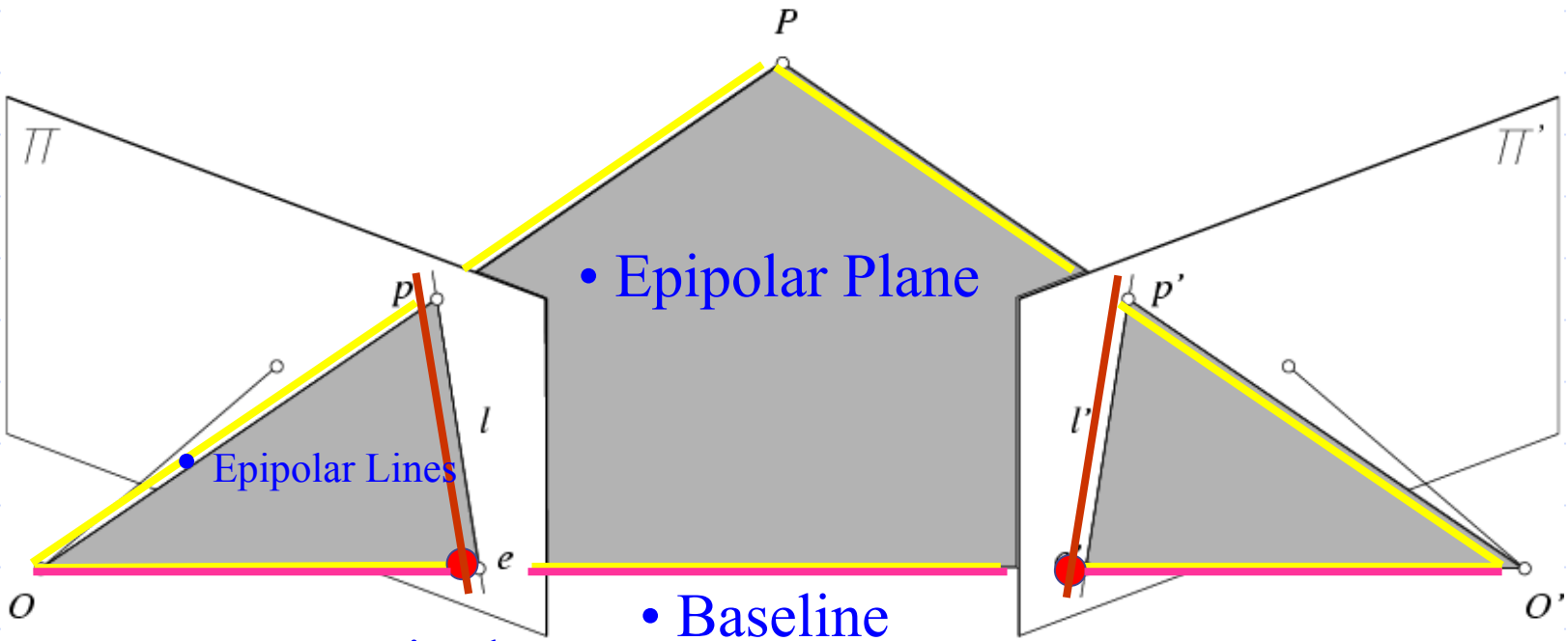
Epipolar Geometry

The Essential Matrix

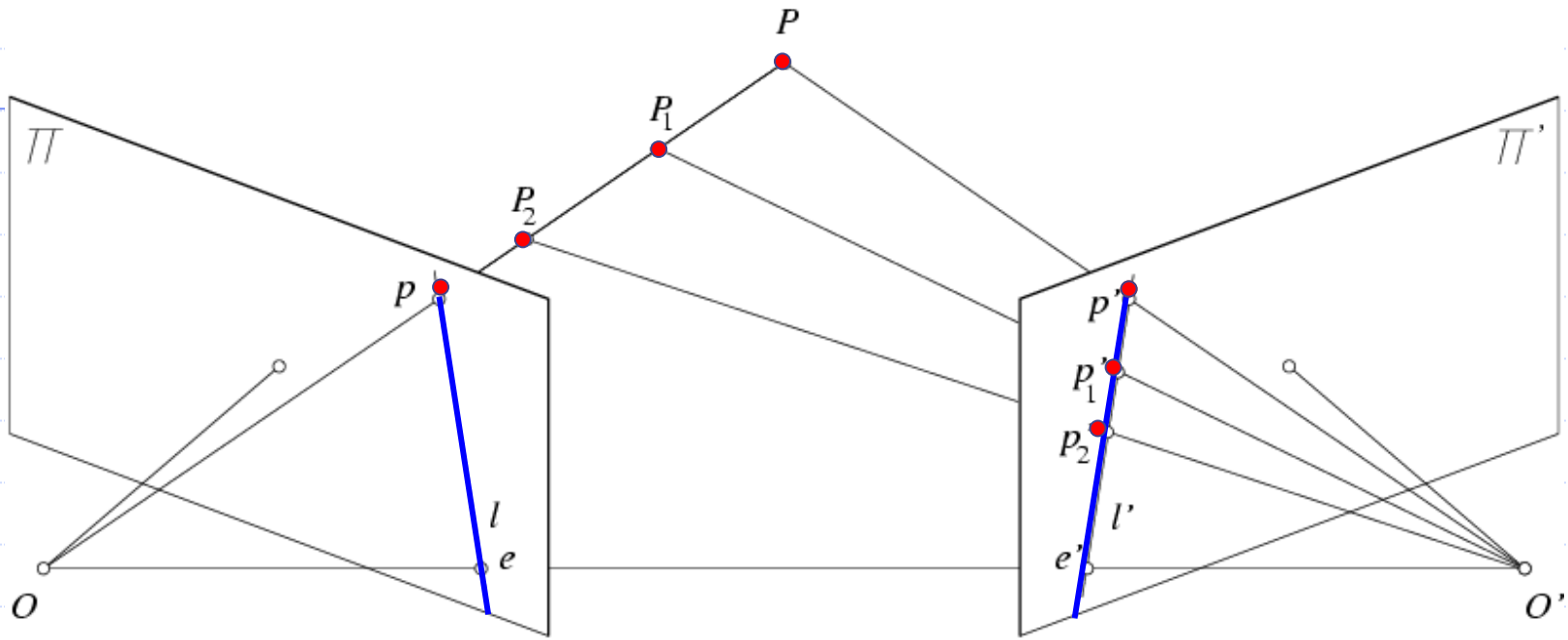
The Fundamental Matrix

The Trifocal Tensor

Epipolar Geometry



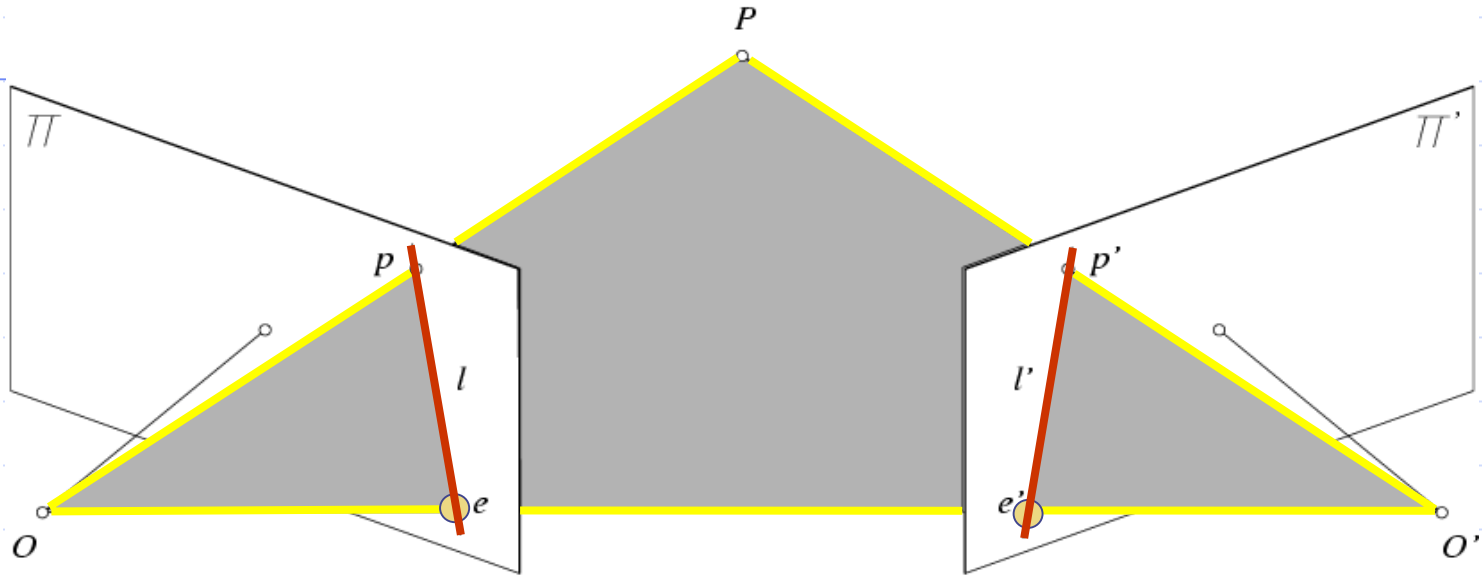
Epipolar Constraint



- Potential matches for *p* have to lie on the corresponding epipolar line *l'*.

- Potential matches for *p'* have to lie on the epipolar line *l*.

Epipolar Constraint: Ideal Case



$$\overrightarrow{Op} \cdot [\overrightarrow{OO'} \times \overrightarrow{O'p'}] = 0 \Rightarrow \boldsymbol{p} \cdot [\boldsymbol{t} \times (\mathcal{R}\boldsymbol{p}')] = 0 \quad \text{with} \begin{cases} \boldsymbol{p} = (u, v, 1)^T \\ \boldsymbol{p}' = (u', v', 1)^T \\ \mathcal{M} = (\text{Id} \quad \mathbf{0}) \\ \mathcal{M}' = (\mathcal{R}^T, -\mathcal{R}^T \boldsymbol{t}) \end{cases}$$

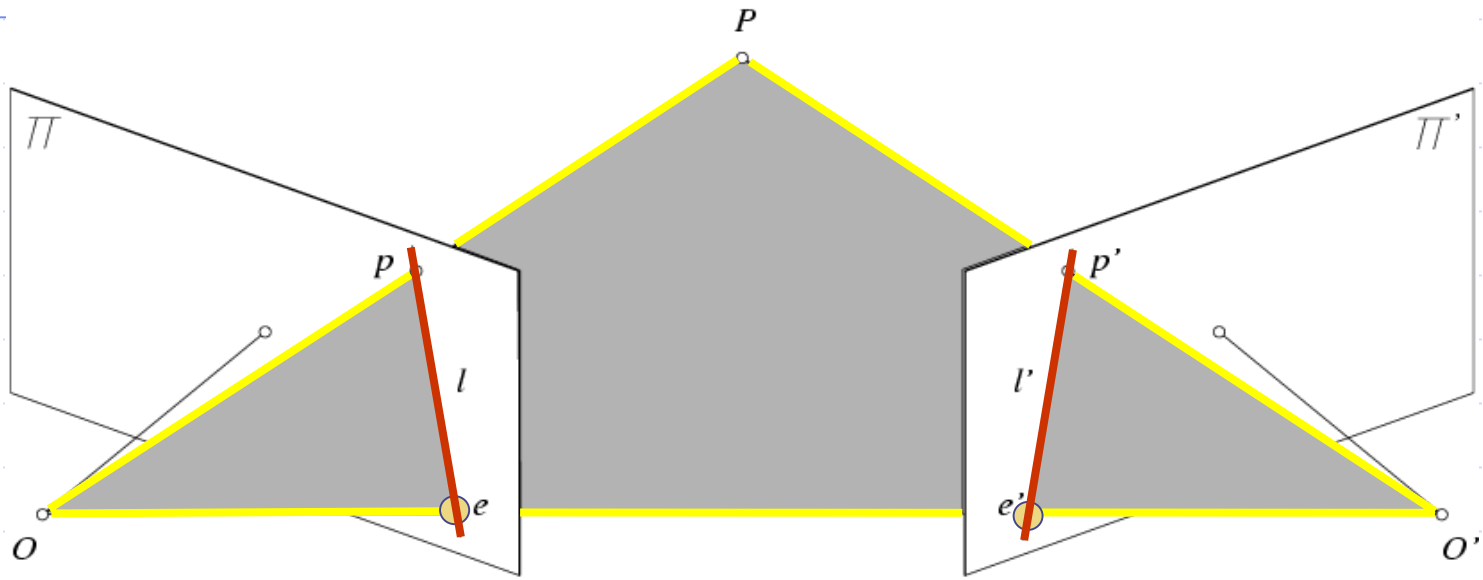
$$\boldsymbol{p}^T \mathcal{E} \boldsymbol{p}' = 0 \quad \text{with} \quad \mathcal{E} = [\boldsymbol{t}_\times]\mathcal{R}$$

Essential Matrix
(Longuet-Higgins, 1981

Properties of the Essential Matrix

- $E\,p'$ is the epipolar line associated with p'.
- $E^T p$ is the epipolar line associated with p.
- $E\,e'=0$ and $E^T e=0$.
- E is singular.
- E has two equal non-zero singular values (Huang and Faugeras, 1989).

Epipolar Constraint: Real Case



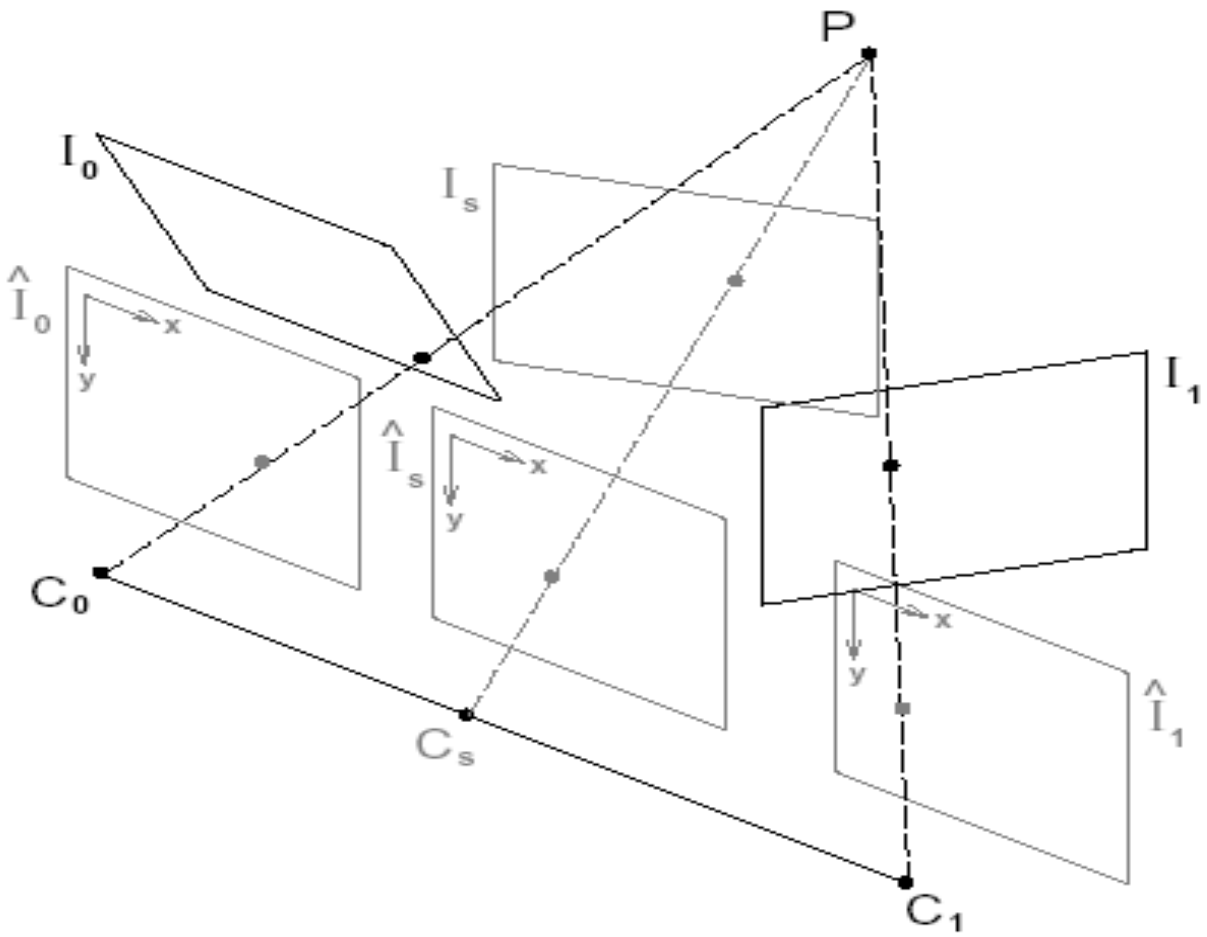
$$\hat{\boldsymbol{p}}^T \mathcal{E} \hat{\boldsymbol{p}}' = 0 \quad \Rightarrow \quad \boldsymbol{p}^T \mathcal{F} \boldsymbol{p}' = 0 \quad \text{with} \quad \mathcal{F} = \mathcal{K}^{-T} \mathcal{E} \mathcal{K}'^{-1}$$

$$\boldsymbol{p} = \mathcal{K} \hat{\boldsymbol{p}}$$

$$\boldsymbol{p}' = \mathcal{K}' \hat{\boldsymbol{p}}'$$

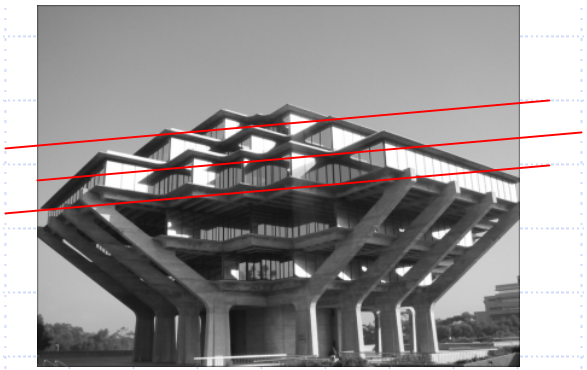
Fundamental Matrix
(Faugeras and Luong, 1992

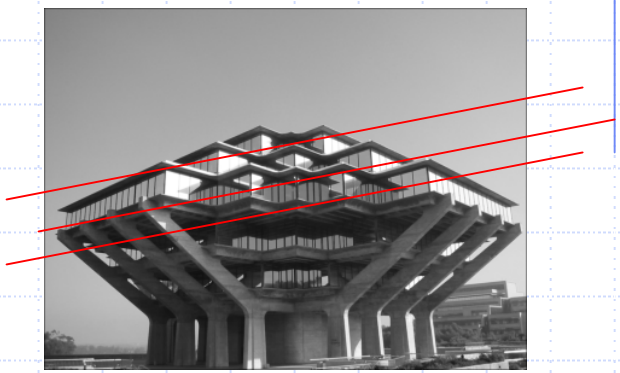
Image Rectification

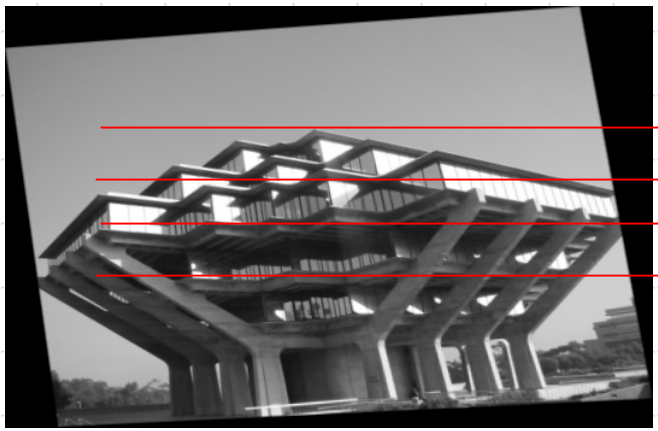


An Example

Due to Josh Wills







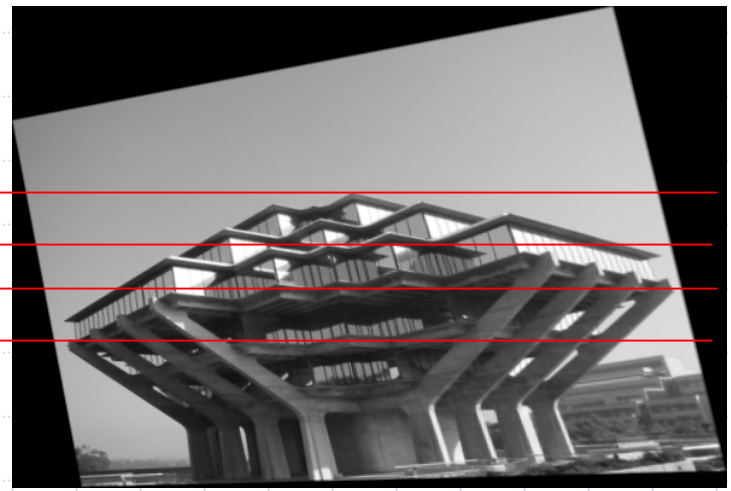
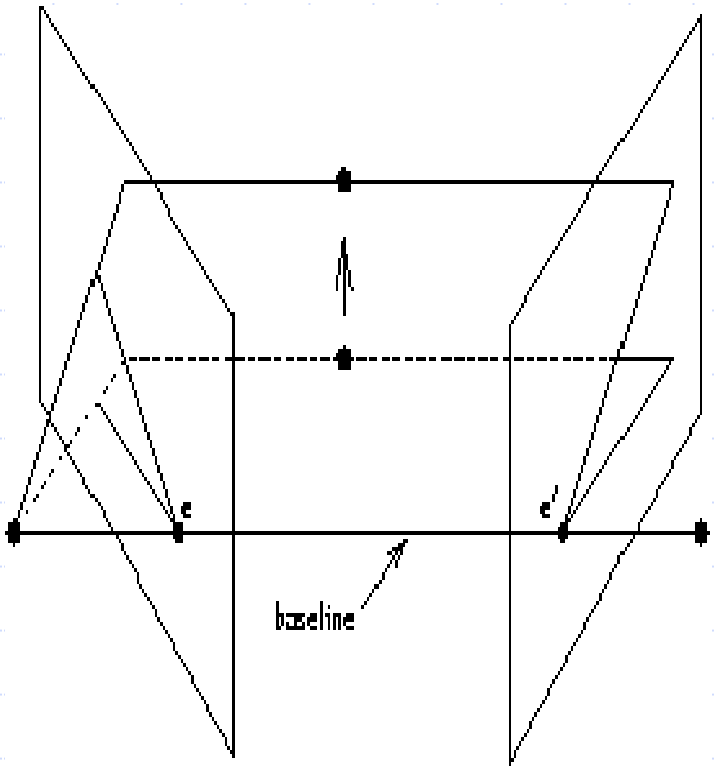


Image Rectification



- To make the epipolar lines parallel to each other and aligned to one of the scanlines, we should send the epipoles to infinity.
- Rectified image is obtained by a rotation of the original image keeping the optical center fixed. Hence, they are related by a homography.

Image Rectification

Such a Homography is given by:

H = GR, R = a rotation which takes epipole(e) to $(f,0,1)^T$

$$G=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/f & 0 & 1 \end{bmatrix}$$

As is easily seen,

$G^*(f,0,1)^T = (f,0,0)^T = \text{point at infinity}$

If the images are related by a fundamental matrix,

$F = [e]_\times M$, Then the homography which ensures that both images are aligned is given by:

$$H' = (I + Hea^T)HM$$

View Morphing

- Image morphing is shape preserving only in the special case when the images are taken using cameras having parallel optical axes.
- An additional step called image rectification is introduced. Image interpolation on rectified images produces physically valid in-between images.
- Rectification is a process of aligning corresponding epipolar lines in the two images along their scanlines.

The algorithm

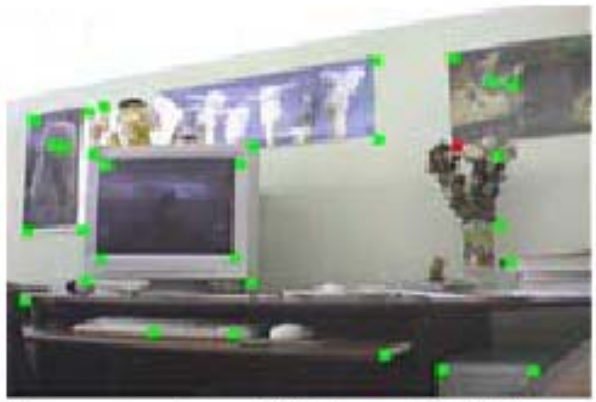
- Find a few point correspondences in both images.
- Estimate the fundamental matrix using 8-point algorithm along with RANSAC.
- Rectify both the images.
- Find dense correspondence and morph the rectified images
- Post warp the morphed image.
- Remove holes (moirs).

Step By Step

Original Image From Left Camera

Original Image From Right Camera

Correspondence Points on Left Image



Correspondence Points on Right Image

Step By Step

Some Scanlines on Left Iimage

Some Scanlines on Right Image

Surprise!

Some Results

Novel View Synthesis using Trilinear Tensors

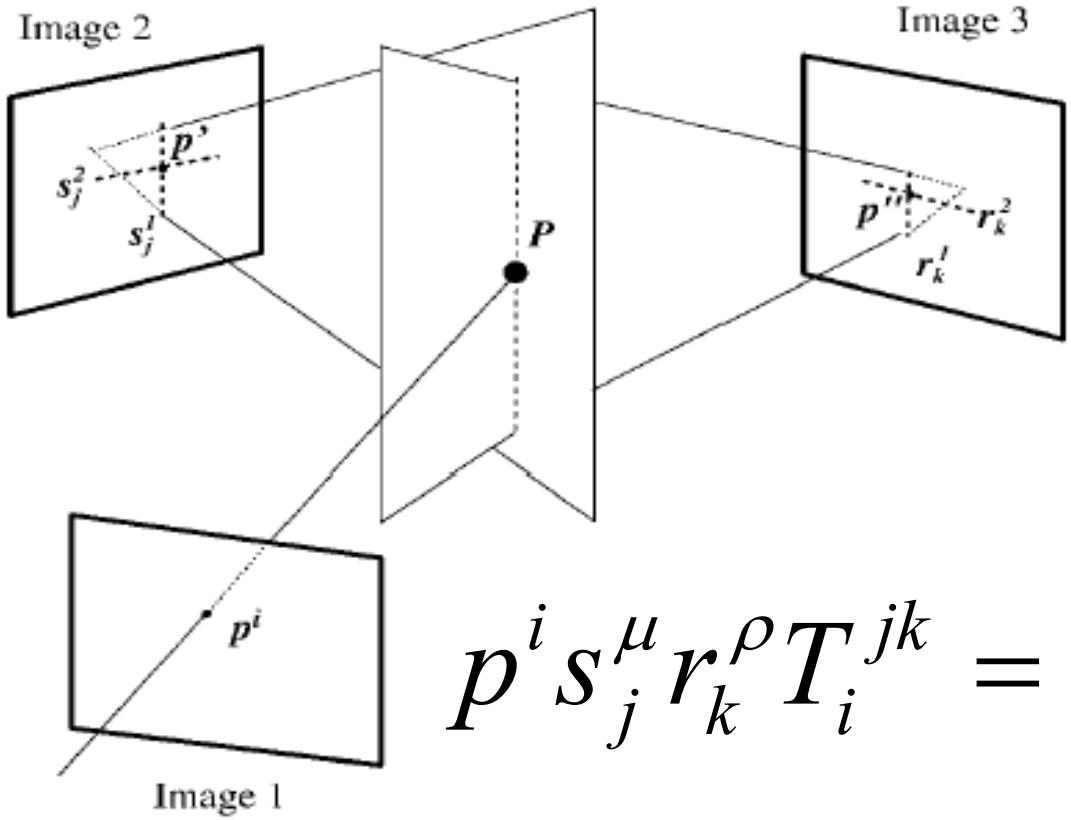
What is a Tensor?

Any array of numbers: A scalar is a zero dimensional tensor, a vector is a one dimensional tensor, a matrix is a two dimensional tensor and so on.

A Trilinear Tensor is a 3x3x3 array.

- Three views satisfy certain matching constraints represented by a tensor.
- Given two views in correspondence and a tensor, the corresponding third view can be generated by means of a warping function.

Constraint Equation

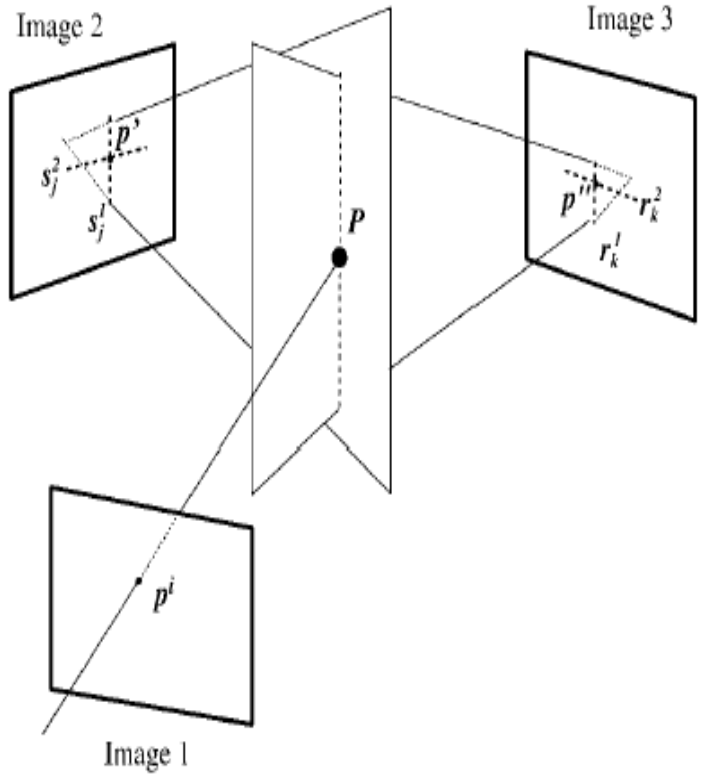


$$p^i s_j^{\mu} r_k^{\rho} T_i^{jk} = 0$$

More Hand waiving!

Let p, p’ are known. Then $p^i s_j^\mu T_i^{jk}$ is a point that coincides with all lines passing through p”. Hence, given images of a point in two views p, p' and a tensor we can find the image of the point in the third view by the reprojection equation given by:

$$p^i s_j^\mu T_i^{jk} \cong p''^k$$



Basic Tensor Operator

- Basic Tensor Operator describes how to modify a tensor so as to represent a new set of cameras. Let, [I;0], [A,V],[B,V′] and [C,V″] be the camera matrices associated with the views in question. Let T_i^{jk} be the tensor between views 1,2,3 and G_i^{jk} be the tensor between views 1,2,4. If the motion parameters between views 3 and 4 are represented by D, then we have a relationship:

$$G_i^{jk} = d_i^k T_i^{jk} + t^k a_i^j$$

Hence, corresponding to different D's, we can have different viewpoints.

The Algorithm

- Pre-processing steps:
  - Compute dense correspondence (optical-flow) between the pair of reference images.
  - Construct the trilinear tensor of the two images
  - Extract the rotation matrix of the two images from the trilinear tensor .
- View Synthesis steps:
  - Accept the rotation and translation of the novel image.
  - Construct the new tensor
  - Reproject the novel image.

Results

Rendering using PIV

- Punch Line:

" The set of all images of a rigid set of m points and n lines observed using a weak perspective camera can be represented by a six dimensional variety embedded in $R^{2(m+n)}$ and can be parameterized by the image coordinates of three reference points"

Note: Loosely speaking, a variety can be thought to be a subspace

Basic Ideas

Given:

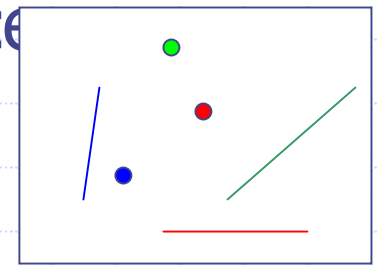
a) Images of a rigid scene taken from different view points. These images are called training images.

b) A set of point(m in no.) and line (n in no.) correspondences in all training images.

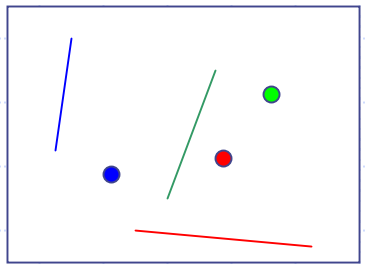
c) Coordinates of 3 reference points in all images.

The Algorithm

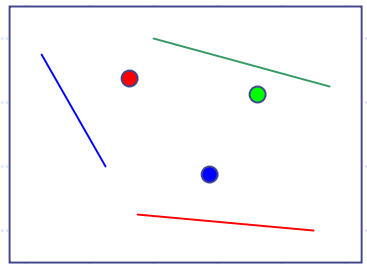
- The position of m points and the position and orientation of n lines can be used to calculate some parameters which define the sce



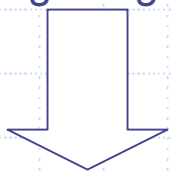
Training Image1



Training Image2

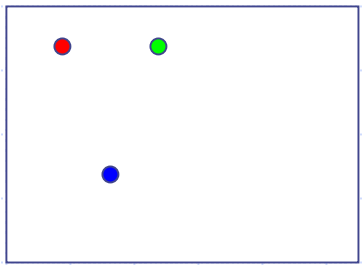


Training Image3

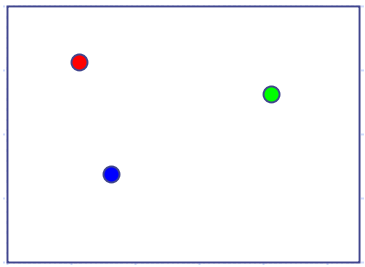


Parameters p1, p2,p3 ……..

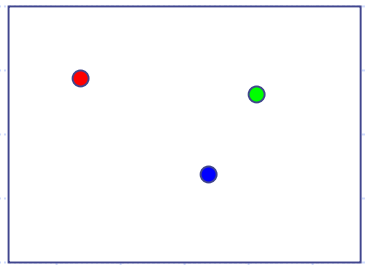
- The view point of the image to be synthesized is expressed by the location of the three points in the synthesized image.



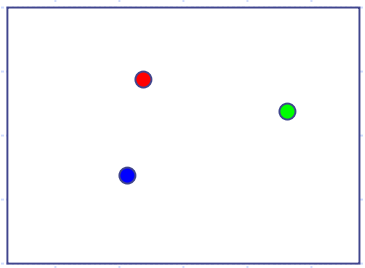
Training Image1



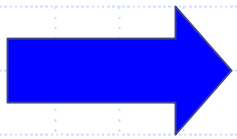
Training Image2



Training Image3



Synthesized Image

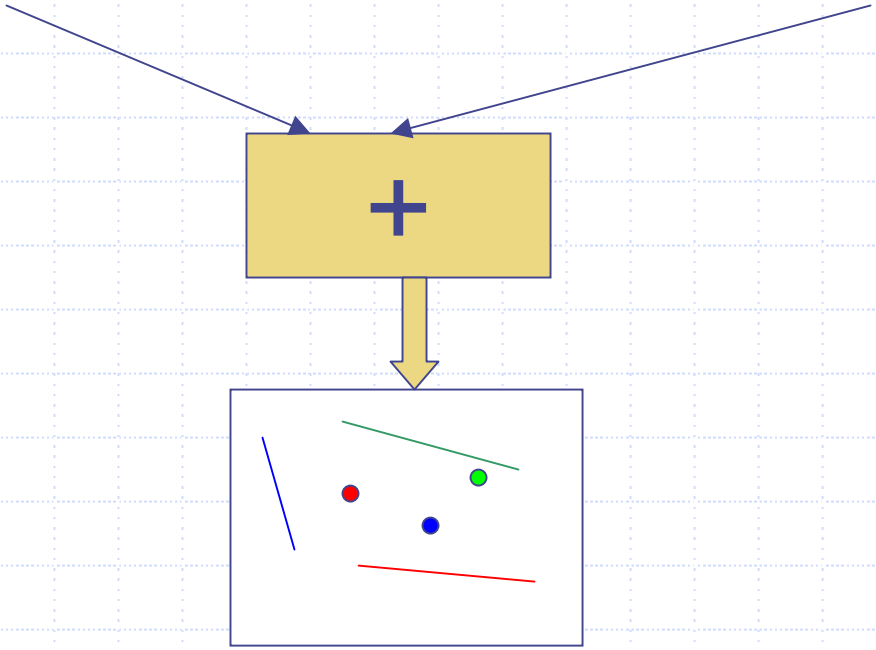


The positions of three points in the Synthesized image specifies the viewpoint (Instead of the normal viewing matrix or R and T)

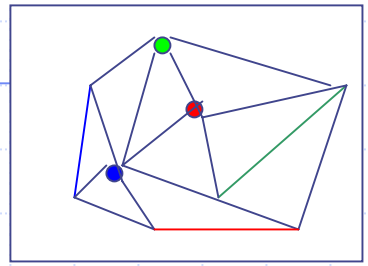
- The parameters can then be used to calculate the position and orientation of all m points and n lines in the synthesized image.

Parameters p1, p2, p3….

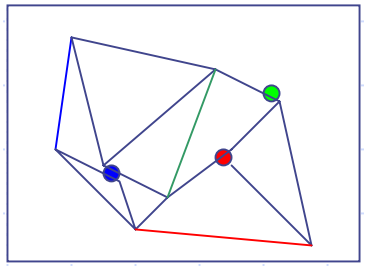
position of points which specify the viewpoint



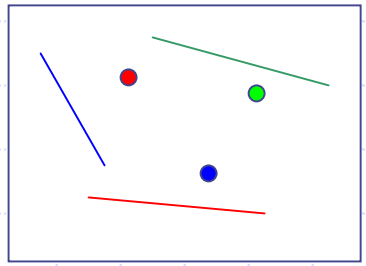
Synthesized Image: We get the position and orientation of lines and points in the synthesized image.



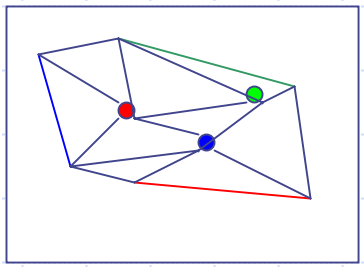
Training Image1



Training Image2



Training Image3



Synthesized Image

Triangulation based Rendering

Complete Synthesized Image

Results