Master in
Computer Vision
Barcelona

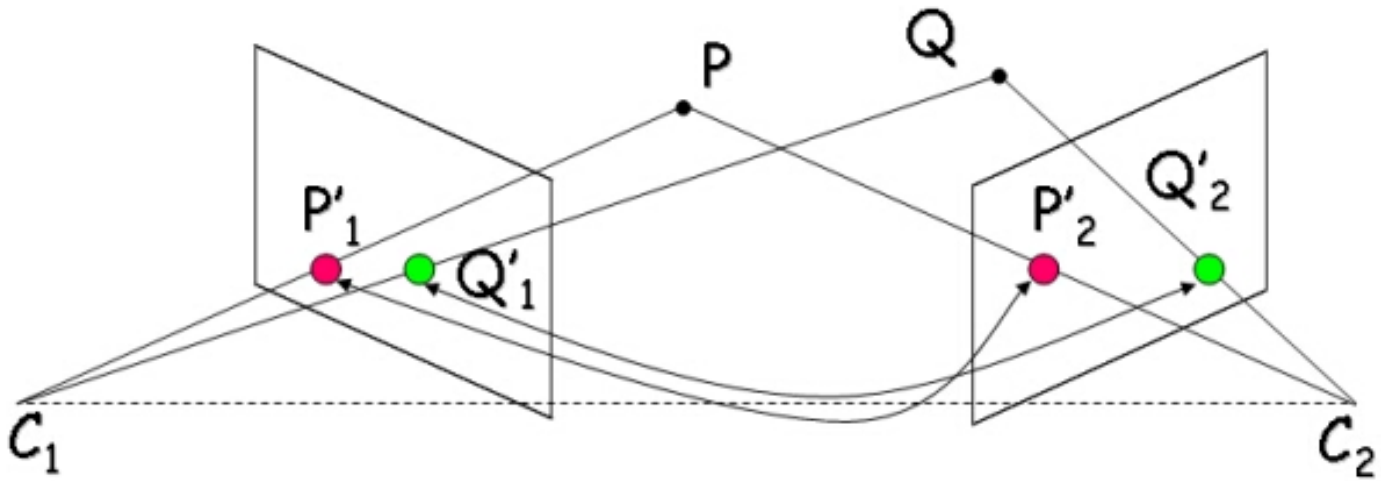
Module: 3D Vision
Lecture 7: Triangulation methods.
Depth computation.
New view synthesis.
Lecturer: Gloria Haro

Outline

- Triangulation methods
- Stereo and depth computation
- New view synthesis

Triangulation methods

Problem statement
Determining a point in 3D space given its projections onto two, or more, images. Called **structure computation**.



We know:

- the projection matrices,
- the point projections and their correspondences.

Image source: http://cvg.deis.unibo.it

Triangulation methods

Problem: Visual rays do not always intersect

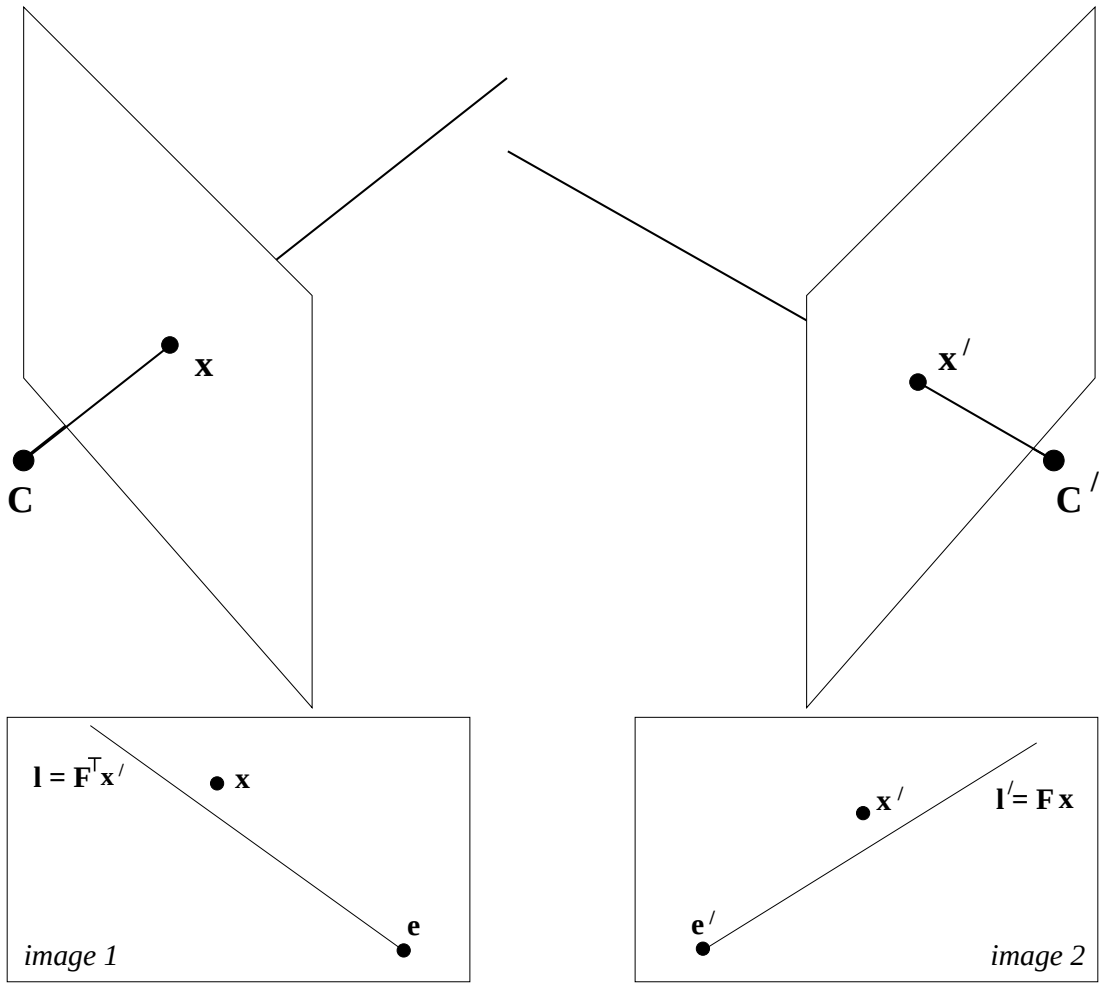
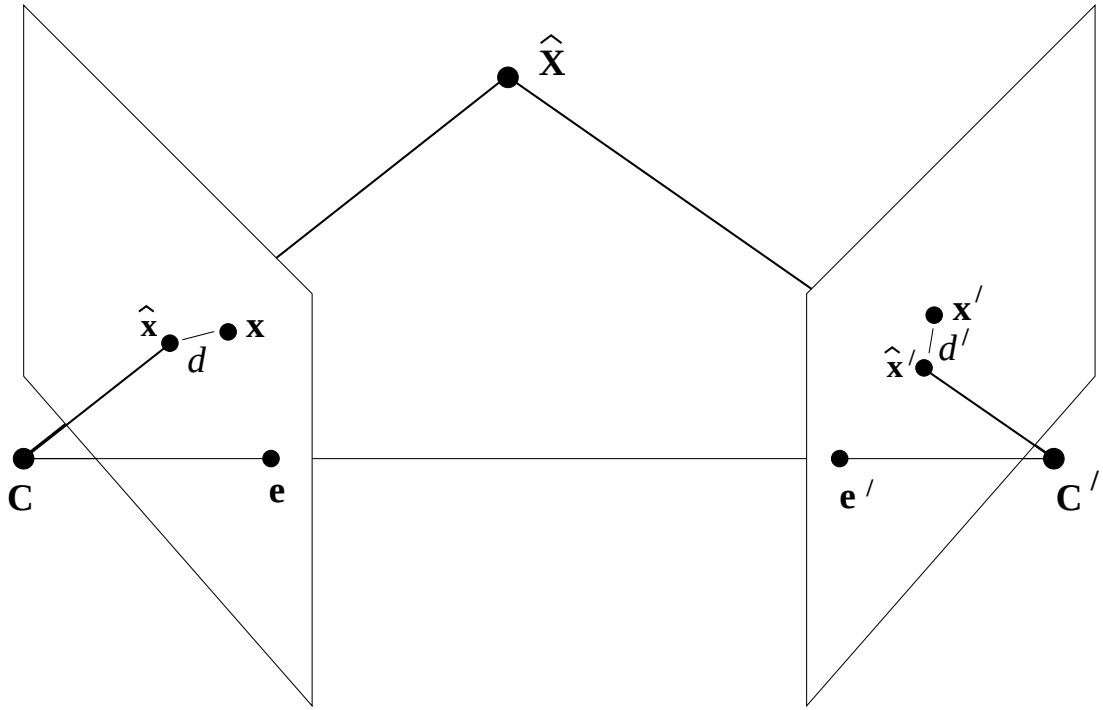


Image source: [Hartley and Zisserman 2004]

Triangulation methods

Problem formulation Two camera case



GOAL: Estimate a point $\widehat{\mathbf{X}} \in \mathbb{P}^3$ that satisfies $\widehat{\mathbf{x}} = P\widehat{\mathbf{X}}, \widehat{\mathbf{x}}' = P'\widehat{\mathbf{X}}$, for some points $\widehat{\mathbf{x}}$ and $\widehat{\mathbf{x}}'$ in the images near the corresponding points $\mathbf{x}$, $\mathbf{x}'$.

Image source: [Hartley and Zisserman 2004]

Triangulation methods

Two kind of methods:

- **Algebraic (linear) methods**
  - **Homogeneous method (DLT)**
  - **Inhomogeneous method**
- **Geometric methods**

Triangulation methods

Linear triangulation methods

We have a correspondence $\mathbf{x} \longleftrightarrow \mathbf{x}'$, and the camera matrices P, and P' and we want to compute $\mathbf{X} \in \mathbf{R}^3$ such that,

$$\mathbf{x} \equiv P\mathbf{X} \text{ and } \mathbf{x}' \equiv P'\mathbf{X},$$

where $\mathbf{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$, $\mathbf{x}' = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$ $P = \begin{pmatrix} \mathbf{p}^{1T} \\ \mathbf{p}^{2T} \\ \mathbf{p}^{3T} \end{pmatrix}$, and $P' = \begin{pmatrix} \mathbf{p}'^{3T} \\ \mathbf{p}'^{2T} \\ \mathbf{p}'^{3T} \end{pmatrix}$.

Triangulation methods

Linear triangulation methods

We have a correspondence $\mathbf{x} \longleftrightarrow \mathbf{x}'$, and the camera matrices P, and P' and we want to compute $\mathbf{X} \in \mathbf{R}^3$ such that,

$$\mathbf{x} \equiv P\mathbf{X} \text{ and } \mathbf{x}' \equiv P'\mathbf{X},$$

where $\mathbf{x} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$, $\mathbf{x}' = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$ $P = \begin{pmatrix} \mathbf{p}^{1T} \\ \mathbf{p}^{2T} \\ \mathbf{p}^{3T} \end{pmatrix}$, and $P' = \begin{pmatrix} \mathbf{p}'^{3T} \\ \mathbf{p}'^{2T} \\ \mathbf{p}'^{3T} \end{pmatrix}$.

The problem can be written as:

$$\mathbf{A}\mathbf{X} = 0,$$

where

$$\mathbf{A} = \begin{pmatrix} x\mathbf{p}^{3T} - \mathbf{p}^{1T} \\ y\mathbf{p}^{3T} - \mathbf{p}^{2T} \\ x'\mathbf{p}'^{3T} - \mathbf{p}'^{1T} \\ y'\mathbf{p}'^{3T} - \mathbf{p}'^{2T} \end{pmatrix}.$$

Triangulation methods

Homogeneous method (DLT)

$$\min_{\mathbf{X}} \|\mathbf{A}\mathbf{X}\|_2$$

$$\text{such that } \|\mathbf{X}\|_2 = 1.$$

Solution: The singular vector associated to the minimum singular value of **A**.

Normalization: Scaling and translation so that both pixel coordinates are in the interval $[-1, 1]$.

$$H = \begin{pmatrix} 2/\mathrm{nx} & 0 & -1 \\ 0 & 2/\mathrm{ny} & -1 \\ 0 & 0 & 1 \end{pmatrix}.$$

We build **A** from $\mathbf{x} = H\mathbf{x}$, $\mathbf{x}' = H\mathbf{x}'$, $P = HP$, and $P' = HP'$.

Triangulation methods

Inhomogeneous method

Since $\mathbf{X} \in \mathbb{P}^3$ only 3 dof (unknowns).

We assume that there are no points at infinity and write $\mathbf{X}$ as:

$\mathbf{X} = (X, Y, Z, 1)^T = (\tilde{\mathbf{X}}, 1)$

$$\min_{\tilde{\mathbf{X}}} \|\mathbf{A}'\tilde{\mathbf{X}} + \mathbf{a_4}\|_2$$

$$\text{where } \mathbf{A} = (\mathbf{A}'|\mathbf{a_4}), \text{ and } \mathbf{X} = (\tilde{\mathbf{X}}, 1)^T.$$

Solution:

(i) Find the SVD $\mathbf{A}' = \mathbf{U}\mathbf{D}\mathbf{V}^T$.

(ii) Set $\mathbf{a_4}' = \mathbf{U}^T\mathbf{a_4}$.

(iii) Find the vector $\mathbf{Y}$ defined by $y_i = \frac{-a'_{4_i}}{d_i}$, $i = 1, \ldots, n$, where d_i is the i diagonal element of $\mathbf{D}$.

(iv) The solution is $\tilde{\mathbf{X}} = \mathbf{V}\mathbf{Y}$.

Triangulation methods

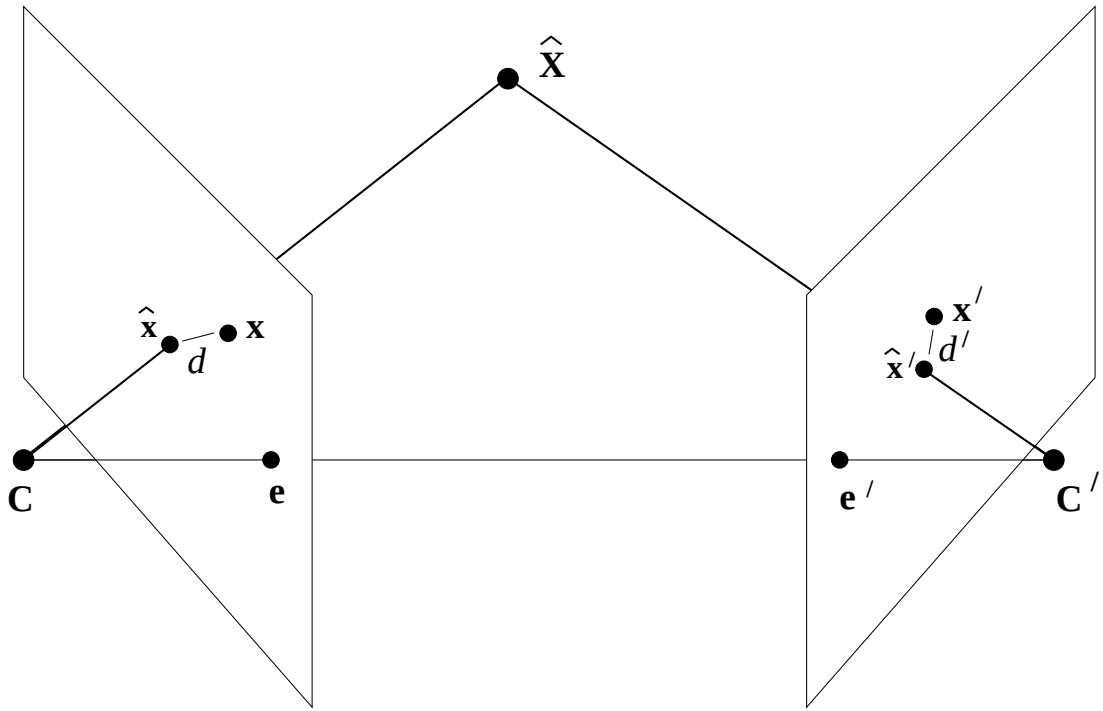
Advantages +, and disadvantages − of linear methods

- \+ Both are linear, non-iterative methods.
- \+ The homogeneous method often provides acceptable results and serves as an initialization for the geometric methods.
- \+ The homogeneous method generalizes easily to more than two views.
- − None of these two linear methods is projective-invariant. (Because of the conditions $\|\mathbf{X}\|_2 = 1$, and $\mathbf{X} = (\tilde{X}, 1)^T$, the point $\mathbf{X}$ solving the original problem will not correspond to a solution $H\mathbf{X}$ for the transformed problem.)
- − The inhomogeneous method does not handle 3D points at the infinity.

The recommended triangulation method would be one that is invariant to the projective frame of the cameras, and minimizes a geometric image error.

Triangulation methods

Geometric methods



$$\min_{\widehat{\mathbf{x}},\widehat{\mathbf{x}}'} d^2(\mathbf{x},\widehat{\mathbf{x}}) + d^2(\mathbf{x}',\widehat{\mathbf{x}}') = \min_{\widehat{\mathbf{x}},\widehat{\mathbf{x}}'} \|[\mathbf{x}] - [\widehat{\mathbf{x}}]\|_2^2 + \|[\mathbf{x}'] - [\widehat{\mathbf{x}}']\|_2^2$$

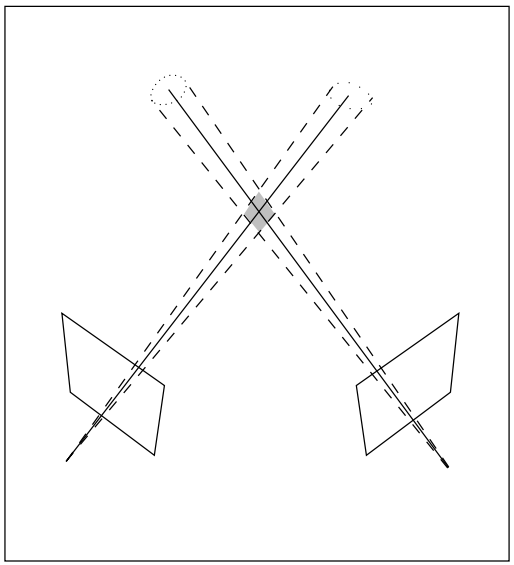
$$\text{such that } \widehat{\mathbf{x}}'^T F \widehat{\mathbf{x}} = 0.$$

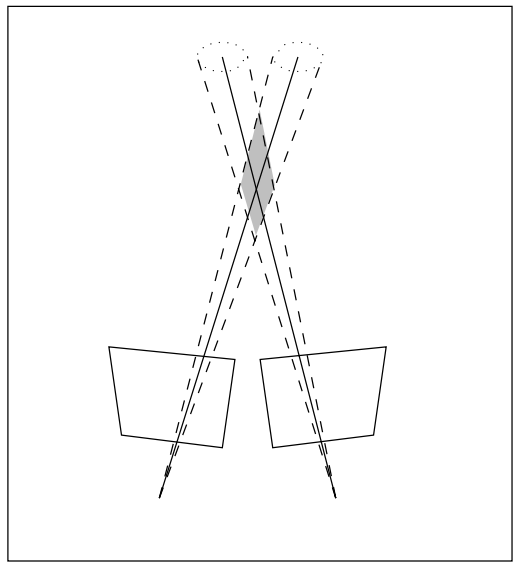
Then, get $\widehat{\mathbf{X}}$ from $\widehat{\mathbf{x}}$ and $\widehat{\mathbf{x}}'$ (the visual rays intersect).

Image source: [Hartley and Zisserman 2004]

Triangulation methods

Uncertainty in the 3D point localization





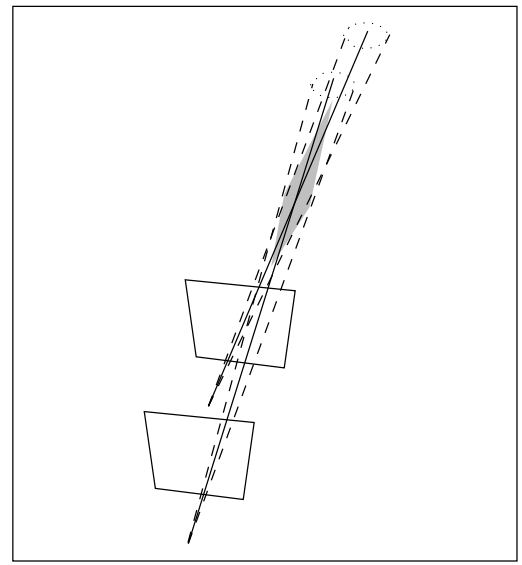


Image source: [Hartley and Zisserman 2004]

Outline

- Triangulation methods
- **Stereo and depth computation**
- New view synthesis

Stereo and depth computation

Stereoscope

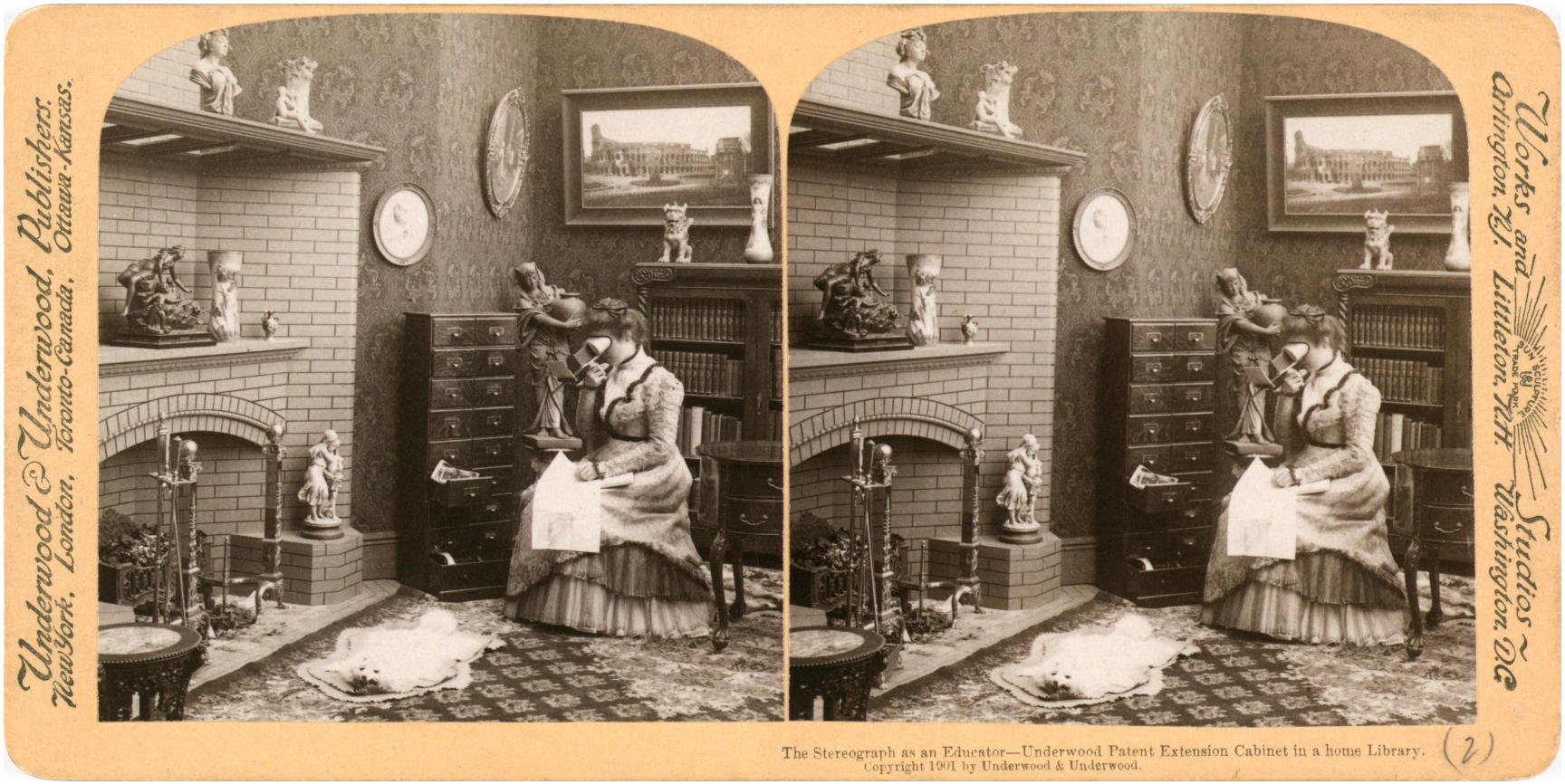


Image source: [Wikipedia]

Stereo and depth computation

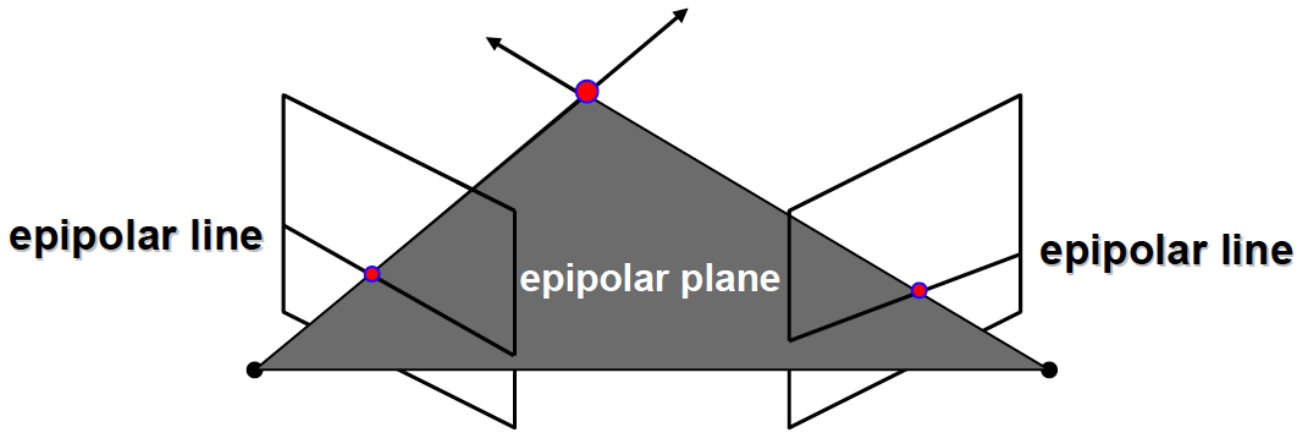
Anaglyph images

Image source: [S. Seitz]

Stereo and depth computation

Stereo correspondence

Determine pixel correspondences, i.e. pairs of pixels that correspond to the same scene point.



The **epipolar constraint** reduces the problem to 1D search along the epipolar lines.

Image source: [S. Seitz]

Stereo and depth computation

Stereo image rectification

Images are reprojected onto a common plane parallel to the baseline
Simplifies the correspondence problem: pixel motion is horizontal

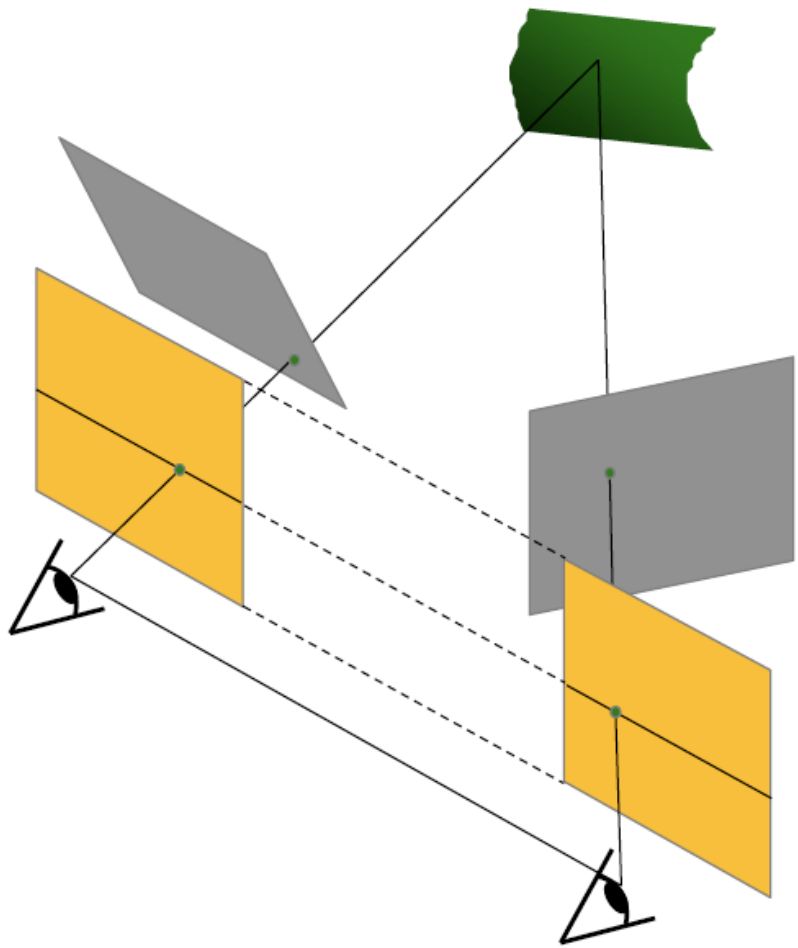
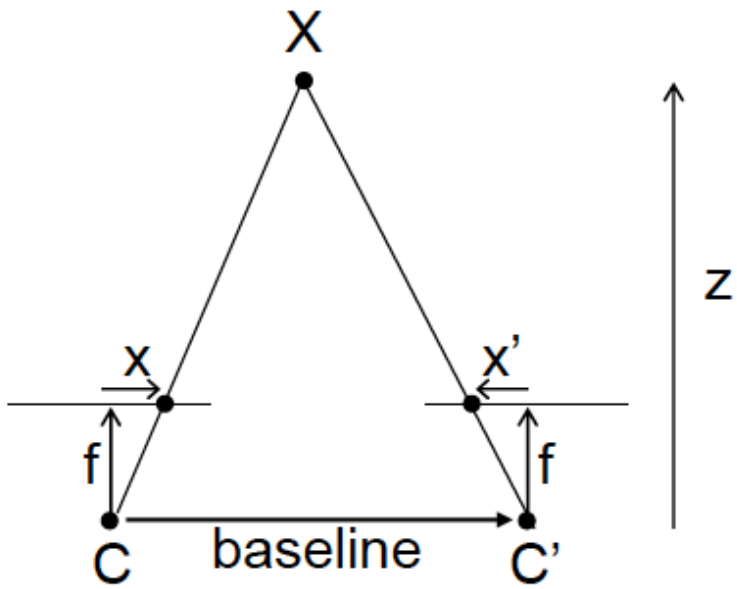


Image source: [S. Seitz]

Stereo and depth computation

Depth from disparity

Pair of recitified stereo images



$$disparity = x - x' = \frac{baseline * f}{z}$$

Image source: [S. Seitz]

Stereo and depth computation

Stereo matching algorithm

GOAL: Match pixels in conjugate epipolar lines

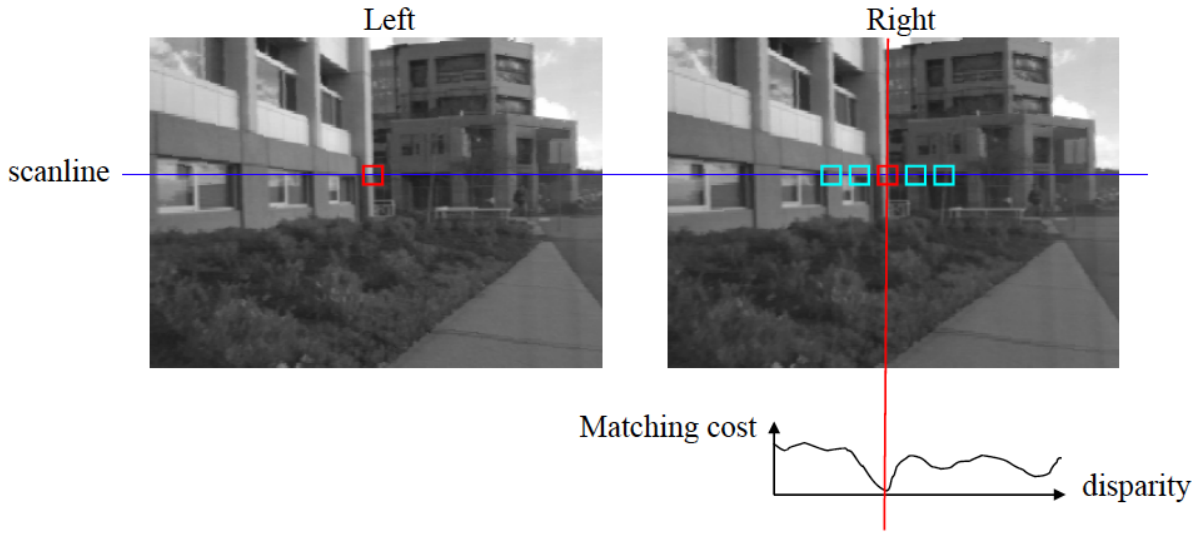
Several approaches (`http://www.middlebury.edu/stereo`).

Classification in two main approaches:

- Local methods
- Global methods

Stereo and depth computation

Local methods for stereo matching



BASIC IDEA:

For every pixel in the left image

1. Slide a window along the same line in the right image and compare its content to that of the reference window in the left image.
2. Pick pixel with minimum matching cost.

Matching cost: SSD, NCC, others

Image source: [S. Lazebnik]

Stereo and depth computation

Matching costs:

- SSD: Sum of Squared Differences

$$\sum_{(x,y)\in W} w(x,y)|I_1(x,y) - I_2(x_0+x, y_0+y)|^2$$

- NCC: Normalized Cross Correlation

$$\frac{\sum_{(x,y)\in W} w(x,y)(I_1(x,y) - \bar{I}_1)(I_2(x_0+x, y_0+y) - \bar{I}_2)}{\sigma_{I_1}\sigma_{I_2}}$$

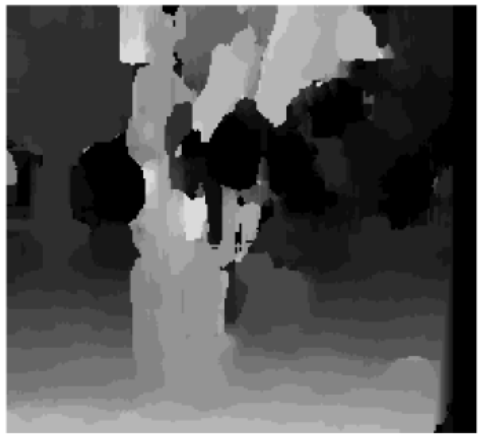
where
$\bar{I}_1 = \sum_{(x,y)\in W} w(x,y) I_1(x,y)$; $\bar{I}_2 = \sum_{(x,y)\in W} w(x,y) I_2(x_0+x, y_0+y)$;
$\sigma_{I_1} = \sqrt{\sum_{(x,y)\in W} w(x,y)(I_1(x,y) - \bar{I}_1)^2}$;
$\sigma_{I_2} = \sqrt{\sum_{(x,y)\in W} w(x,y)(I_2(x_0+x, y_0+y) - \bar{I}_2)^2}$
$w(x,y)$: normalized weight function, $\sum_{(x,y)\in W} w(x,y) = 1$

- Others: Census Transform, Rank Filters, Mutual Information, ...

Stereo and depth computation

Effect of the window size

W = 3



W = 20

Smaller window	Larger window
+ more detail	− less detail
− more noise	+ smoother disparity maps

Better results with adaptive windows or bilateral weights

Image source: [S. Seitz]

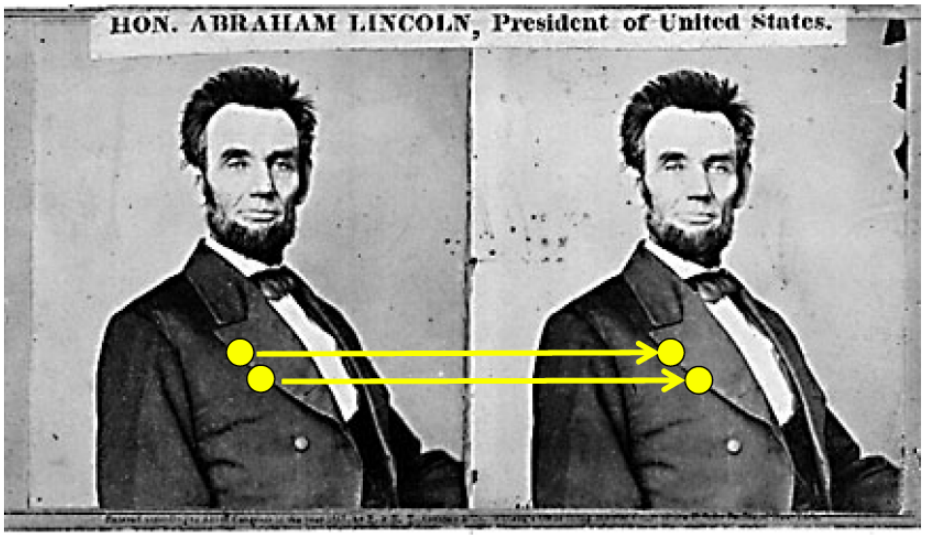
Stereo and depth computation

Failures of correspondence search

- Textureless areas
- Occlusions
- Repetitions (self-similarity)
- Non-Lambertian surfaces, specularities

Stereo and depth computation

Global methods: based on energy minimization



$$\min_d E_{\text{data}}(I_1, I_2, d) + E_{\text{regularity}}(d)$$

- **Data term:** d finds the good match.
  E.g. $L2$ data term (others exist)

$$E_{\text{data}}(I_1, I_2, d) = \sum_{(x,y)} |I_1(x,y) - I_2(x+d,y)|^2$$

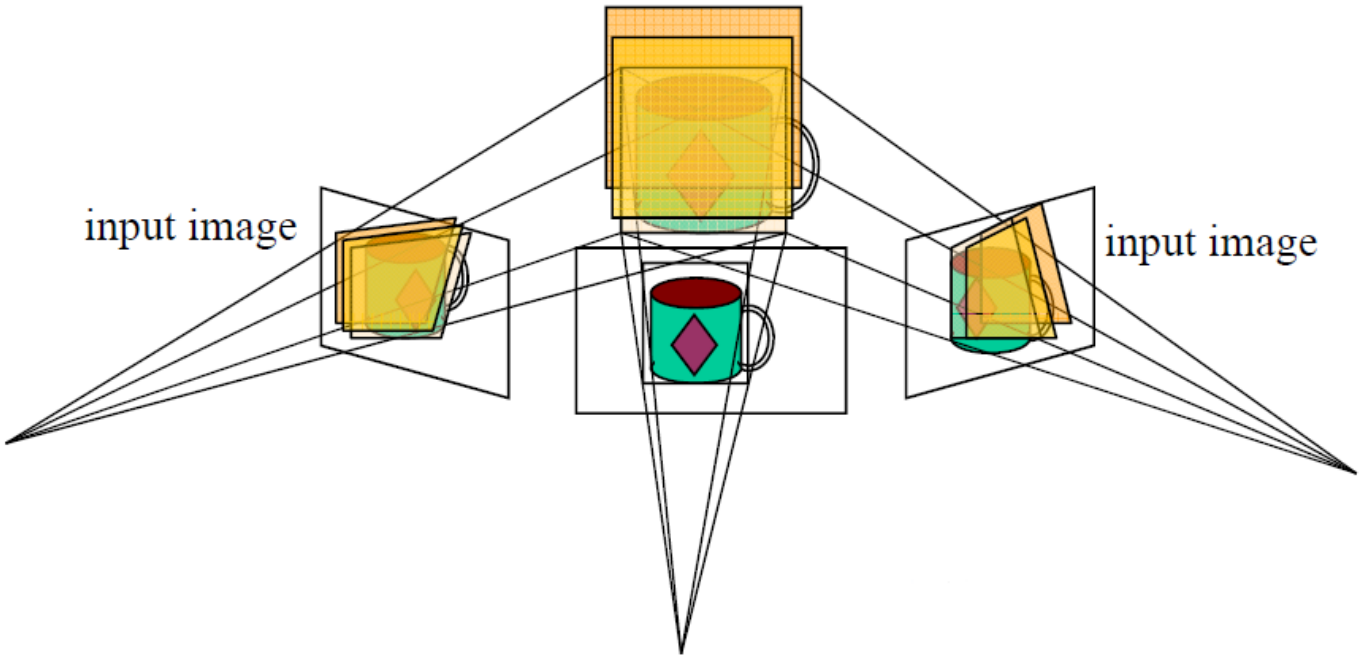
- **Regularity term:** close pixels should (usually) have similar d

$$E_{\text{regularity}}(d) = \sum_{(x,y)} \sum_{\text{neighbor pixels}(x_n,y_n)} \rho(|d(x,y) - d(x_n,y_n)|)$$

Image source: [S. Seitz]

Stereo and depth computation

Plane sweep algorithm



Cameras in general position (not necessarily rectified).

BASIC IDEA: For every pixel in the first image, travel its viewing ray, project it onto the second image, and compare the image similarities.
STRATEGY: Travel the viewing ray of all pixels at once by projecting the image onto fronto parallel planes that move forward.

Image source: [S. Seitz]

Stereo and depth computation

Plane sweep algorithm

For each sampling depth:

- Compute the fronto-parallel plane at the corresponding depth
- Compute the image to image homography given a plane
- Warp the second image according to the homography
- Compute the matching score (SSD or NCC)
- For each pixel keep the depth with best score

Stereo and depth computation

Plane sweep algorithm: fronto-parallel planes

A plane that is parallel to the image plane at a certain depth.
All points of the plane are at the same depth (distance to the image plane).

Stereo and depth computation

Plane sweep algorithm: fronto-parallel planes

A plane that is parallel to the image plane at a certain depth.
All points of the plane are at the same depth (distance to the image plane).

OBSERVATION: The depth is the 3rd coordinate of the point when expressed in the camera frame $\mathbf{x} = K[I|0]\mathbf{X}_{\text{cam}}$
(assuming last row of K: $(0, 0, 1)$).

Stereo and depth computation

Plane sweep algorithm: fronto-parallel planes

A plane that is parallel to the image plane at a certain depth.
All points of the plane are at the same depth (distance to the image plane).

OBSERVATION: The depth is the 3rd coordinate of the point when expressed in the camera frame $\mathbf{x} = K[I|0]\mathbf{X}_{\text{cam}}$
(assuming last row of K: $(0, 0, 1)$).

In the general case: $\mathbf{x} = P\mathbf{X} = K[R|t]\mathbf{X}$, where $\mathbf{X}_{\text{cam}} = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \mathbf{X}$

Then, the depth of $\mathbf{x}$: $d(\mathbf{x}) = (P\mathbf{X})_3 = p_3^T \mathbf{X}$

Stereo and depth computation

Plane sweep algorithm: fronto-parallel planes

A plane that is parallel to the image plane at a certain depth.
All points of the plane are at the same depth (distance to the image plane).

OBSERVATION: The depth is the 3rd coordinate of the point when expressed in the camera frame $\mathbf{x} = K[I|0]\mathbf{X}_{\text{cam}}$
(assuming last row of K: $(0, 0, 1)$).

In the general case: $\mathbf{x} = P\mathbf{X} = K[R|t]\mathbf{X}$, where $\mathbf{X}_{\text{cam}} = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \mathbf{X}$

Then, the depth of $\mathbf{x}$: $d(\mathbf{x}) = (P\mathbf{X})_3 = p_3^T \mathbf{X}$

Fronto-parallel plane: $\mathbf{\Pi} = p_3^T - (0, 0, 0, d)$
(plane equation: $\mathbf{\Pi}^T \mathbf{X} = 0$)

Stereo and depth computation

Plane sweep algorithm: image to image homography

Let $\mathbf{x}$ be a pixel of the first image in homogeneous coordinates.
We want to back-project the pixel into the 3D plane $\mathbf{\Pi}$ and reproject it to the second image.

Stereo and depth computation

Plane sweep algorithm: image to image homography

Let $\mathbf{x}$ be a pixel of the first image in homogeneous coordinates.
We want to back-project the pixel into the 3D plane $\mathbf{\Pi}$ and reproject it to the second image.

We have:

$$\begin{pmatrix} P \\ \mathbf{\Pi}^T \end{pmatrix} \mathbf{X} = \begin{pmatrix} \mathbf{x} \\ 0 \end{pmatrix} \rightarrow \mathbf{X} = \begin{pmatrix} P \\ \mathbf{\Pi}^T \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{x} \\ 0 \end{pmatrix} = A \begin{pmatrix} \mathbf{x} \\ 0 \end{pmatrix}$$

Stereo and depth computation

Plane sweep algorithm: image to image homography

Let $\mathbf{x}$ be a pixel of the first image in homogeneous coordinates.
We want to back-project the pixel into the 3D plane $\mathbf{\Pi}$ and reproject it to the second image.

We have:

$$\begin{pmatrix} P \\ \mathbf{\Pi}^T \end{pmatrix} \mathbf{X} = \begin{pmatrix} \mathbf{x} \\ 0 \end{pmatrix} \rightarrow \mathbf{X} = \begin{pmatrix} P \\ \mathbf{\Pi}^T \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{x} \\ 0 \end{pmatrix} = A \begin{pmatrix} \mathbf{x} \\ 0 \end{pmatrix}$$

Project $\mathbf{X}$ to the second image:

$$\mathbf{x}' = P' A \begin{pmatrix} \mathbf{x} \\ 0 \end{pmatrix} = P' \bar{A} \mathbf{x} = H \mathbf{x}$$

where $\bar{A}$ is the 4×3 submatrix of A.
We get $H = P' \bar{A}$

Outline

- Triangulation methods
- Stereo and depth computation
- **New view synthesis**

New view synthesis

GOAL: Synthetize a new view from a set of available images taken from different angles.

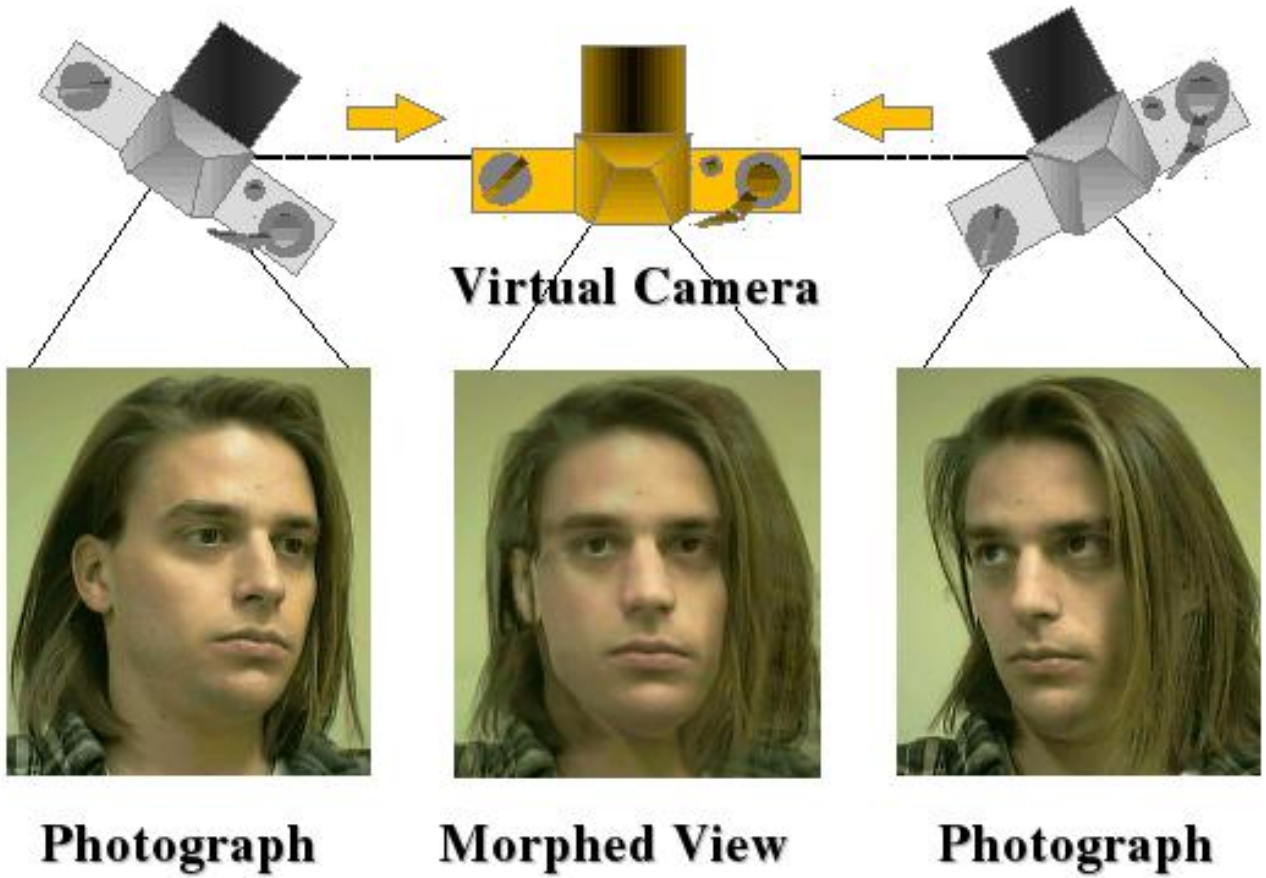


Image source: [Seitz and Dyer 1996]

New view synthesis

Classification of methods:

- **MBR: Model-Based Rendering**
  A 3D reconstruction of the scene is performed.
  + Easily solves the occlusion problem
  – Needs (several) accurately calibrated views (computationally expensive)
  – Due to errors in the 3D model, the texture mapping may create blur in the generated view.
- **IBR: Image-Based Rendering**
  Do not need an explicit 3D model, based on a proper warping of the existing views. The epipolar geometry has to be known.
  Fails at occlusions.
- **Hybrid models**
  Combination of both previous methods. Use IBR basically and the 3D model when ambigous situations arise (e.g. occlusions).

New view synthesis

Image-Based Rendering

Two different approaches:

- **Interpolation based on a dense set of samples**
  Samples of the plenoptic function $F(\phi, \theta, x, y, z)$ are acquired from different images and stored in a table. Then, to synthetise a new view the values of the table are interpolated.
  Fast method BUT needs many views as well as the knowledge of camera position and principal direction.
  Light field rendering [Levoy and Hanrahan 1996], lumigraph [Gortler et al. 1996].
- **Interpolation based on projective reconstruction and morphing**
  Based on the geometrical relationship between images (fundamental matrix or trifocal tensor) and morphing techniques.
  They can be applied with a few images.

New view synthesis

Interpolation based on projective reconstruction and morphing
[Seitz and Dyer 1996]

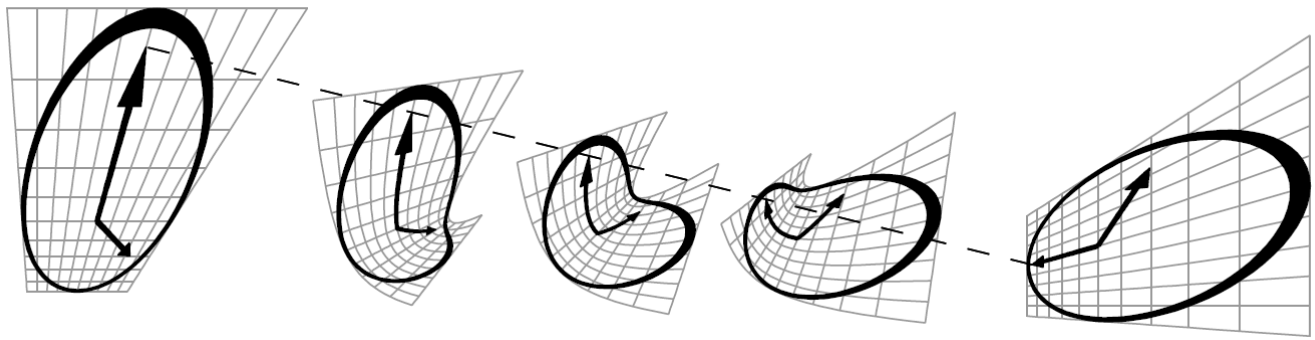


image morphing vs view morphing

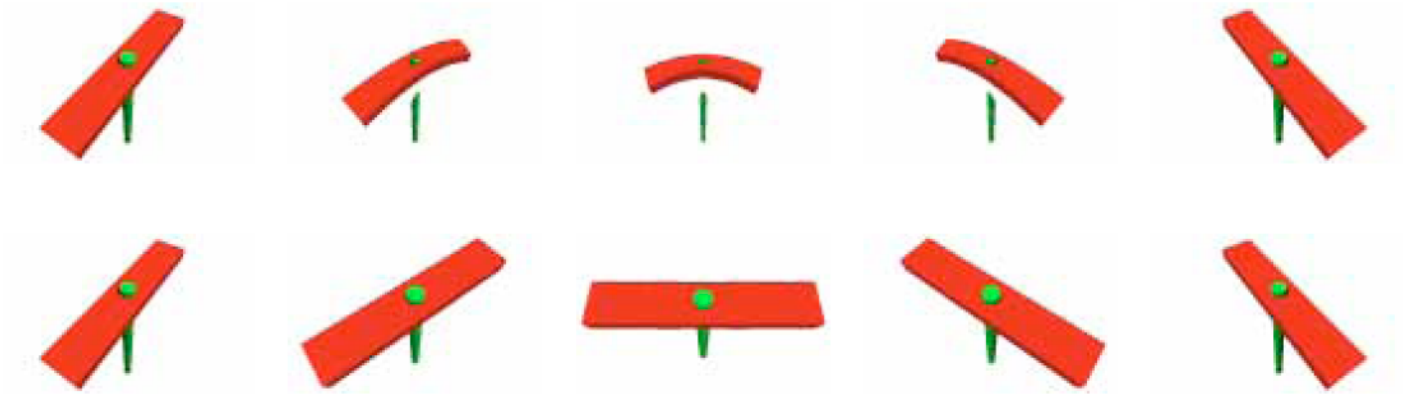


Image source: [Seitz and Dyer 1996]

New view synthesis

Interpolation based on projective reconstruction and morphing
[Seitz and Dyer 1996]

1. Prewarp
   - *align views*

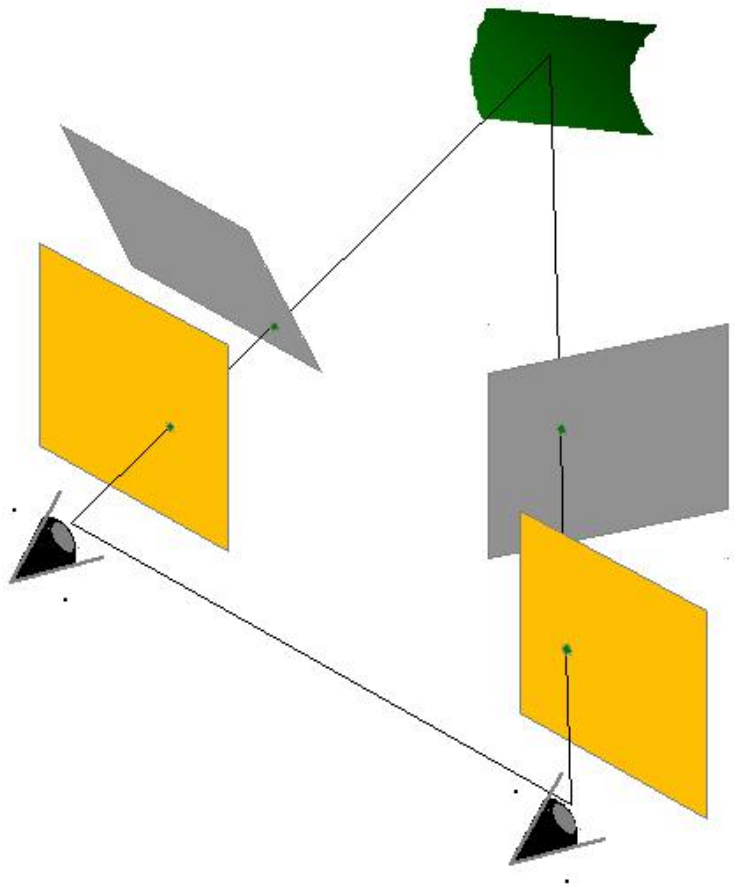


Image source: [Seitz and Dyer 1996]

New view synthesis

Interpolation based on projective reconstruction and morphing
[Seitz and Dyer 1996]

1. Prewarp
   ⇨ *align views*

2. Morph
   ➡ *move camera*

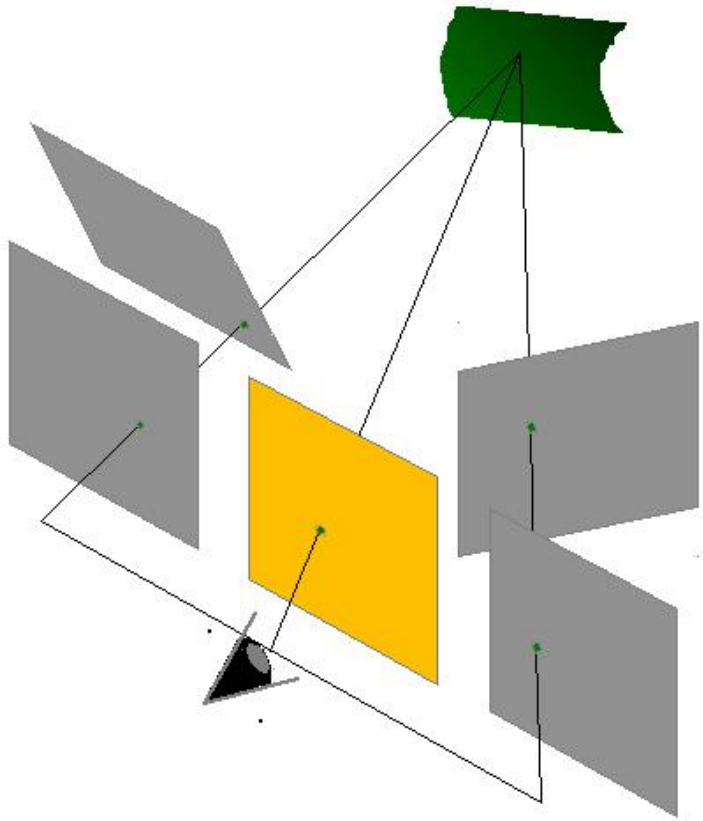


Image source: [Seitz and Dyer 1996]

New view synthesis

Interpolation based on projective reconstruction and morphing
[Seitz and Dyer 1996]

1. Prewarp
   ⇨ *align views*
2. Morph
   ⇨ *move camera*
3. Postwarp
   ⇨ *point camera*

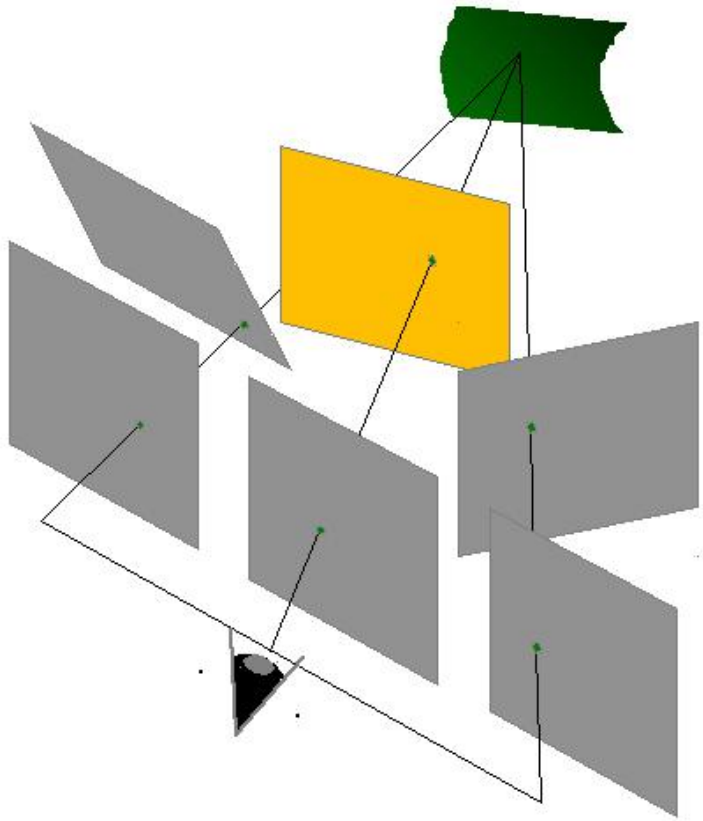


Image source: [Seitz and Dyer 1996]

New view synthesis

Special case of two parallel views:
The camera moves parallel to the image plane

$$P_0 = \begin{pmatrix} f_0 & 0 & 0 & 0 \\ 0 & f_0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, P_1 = \begin{pmatrix} f_1 & 0 & 0 & -f_1 C_X \\ 0 & f_1 & 0 & -f_1 C_Y \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$\mathbf{X}$ projects to $\mathbf{x}_0$ and $\mathbf{x}_1$ in images I_0 and I_1 respectively.
We have:

$$(1-s)\mathbf{x}_0 + s\mathbf{x}_1 = (1-s)\frac{1}{Z}P_0\mathbf{X} + s\frac{1}{Z}P_1\mathbf{X} = \frac{1}{Z}P_s\mathbf{X}$$

where $P_s = (1-s)P_0 + sP_1$

$$I(p) = (1-s)I_0(\mathbf{x}_0) + sI_1(\mathbf{x}_1)$$

$$C_s = (sC_x, sC_Y, 0)$$

$$f = (1-s)f_0 + sf_1$$

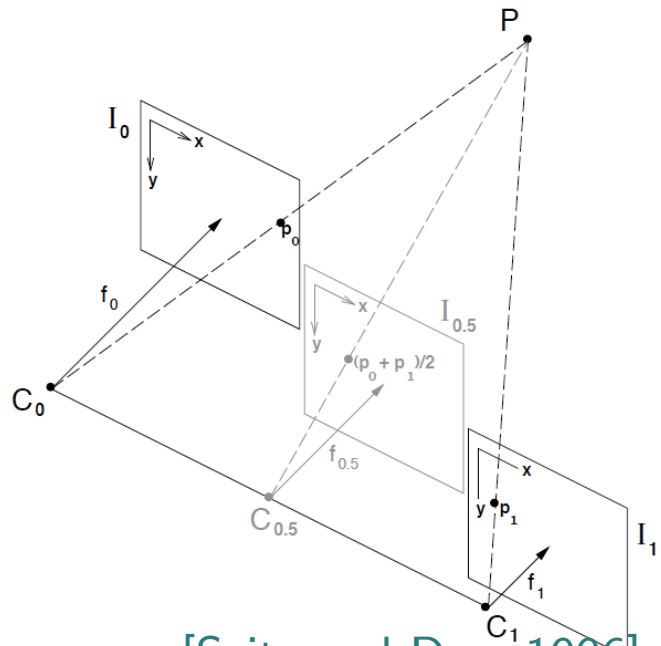


Image source: [Seitz and Dyer 1996]

New view synthesis

Interpolation based on projective reconstruction and morphing

Non-parallel views

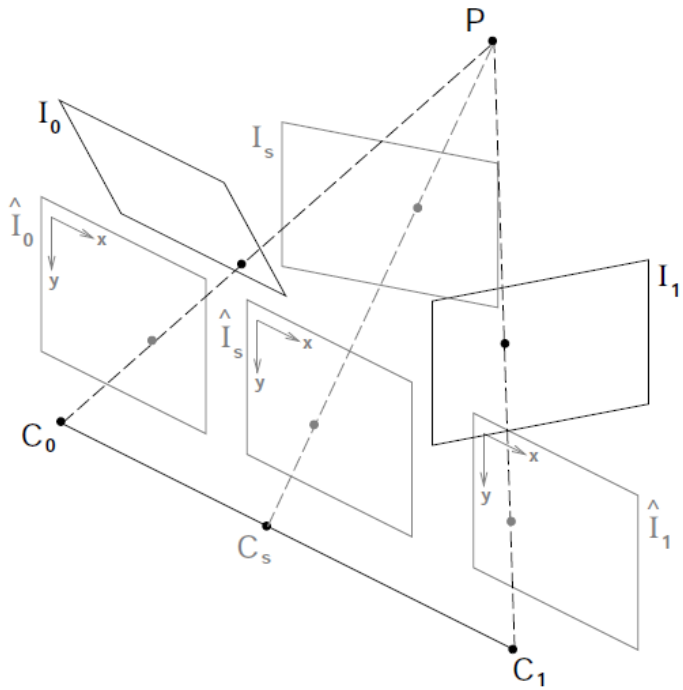
Property: Any two views that share the optical center are related by a planar projective transformation (homography).
Let I and $\hat{I}$ be two images with projection matrices $P = [H| - HC]$ and $\hat{P} = [\hat{H}| - \hat{H}C]$. The projections $\mathbf{x}$ and $\hat{\mathbf{x}}$ of any scene point $\mathbf{X}$ are related by:

$$\hat{H}H^{-1}\mathbf{x} = \hat{H}H^{-1}P\mathbf{X} = \hat{P}\mathbf{X} = \hat{\mathbf{x}}$$

New view synthesis

Interpolation based on projective reconstruction and morphing
[Seitz and Dyer 1996] algorithm steps:

- **Prewarp:** obtain images $\hat{I}_0$ and $\hat{I}_1$ by applying H_0^{-1} and H_1^{-1} to I_0 and I_1 respectively.
- **Morph:** form $\hat{I}_s$ by linear interpolation of position and colors of corresponding points in $\hat{I}_0$ and $\hat{I}_1$.
- **Postwarp:** obtain I_s by applying H_s to $\hat{I}_s$.



We choose: $\hat{P}_0 = [I|-C_0]$ and $\hat{P}_1 = [I|-C_1]$.
We know: $P_0 = [H_0|-H_0C_0]$ and $P_1 = [H_1|-H_1C_1]$.

Image source: [Seitz and Dyer 1996]

New view synthesis

Interpolation based on projective reconstruction and morphing
If the camera matrices are not known, then use image rectification.

Algorithm steps:

- Image rectification
- Computation of dense correspondences
- Linear interpolation of corresponding points
- Post-warping of the interpolated view

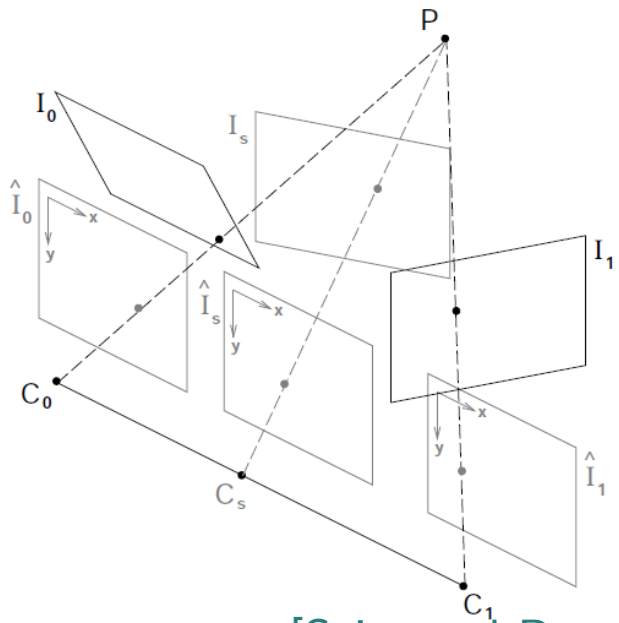


Image source: [Seitz and Dyer 1996]

New view synthesis

Interpolation based on projective reconstruction and morphing
[Seitz and Dyer 1996]

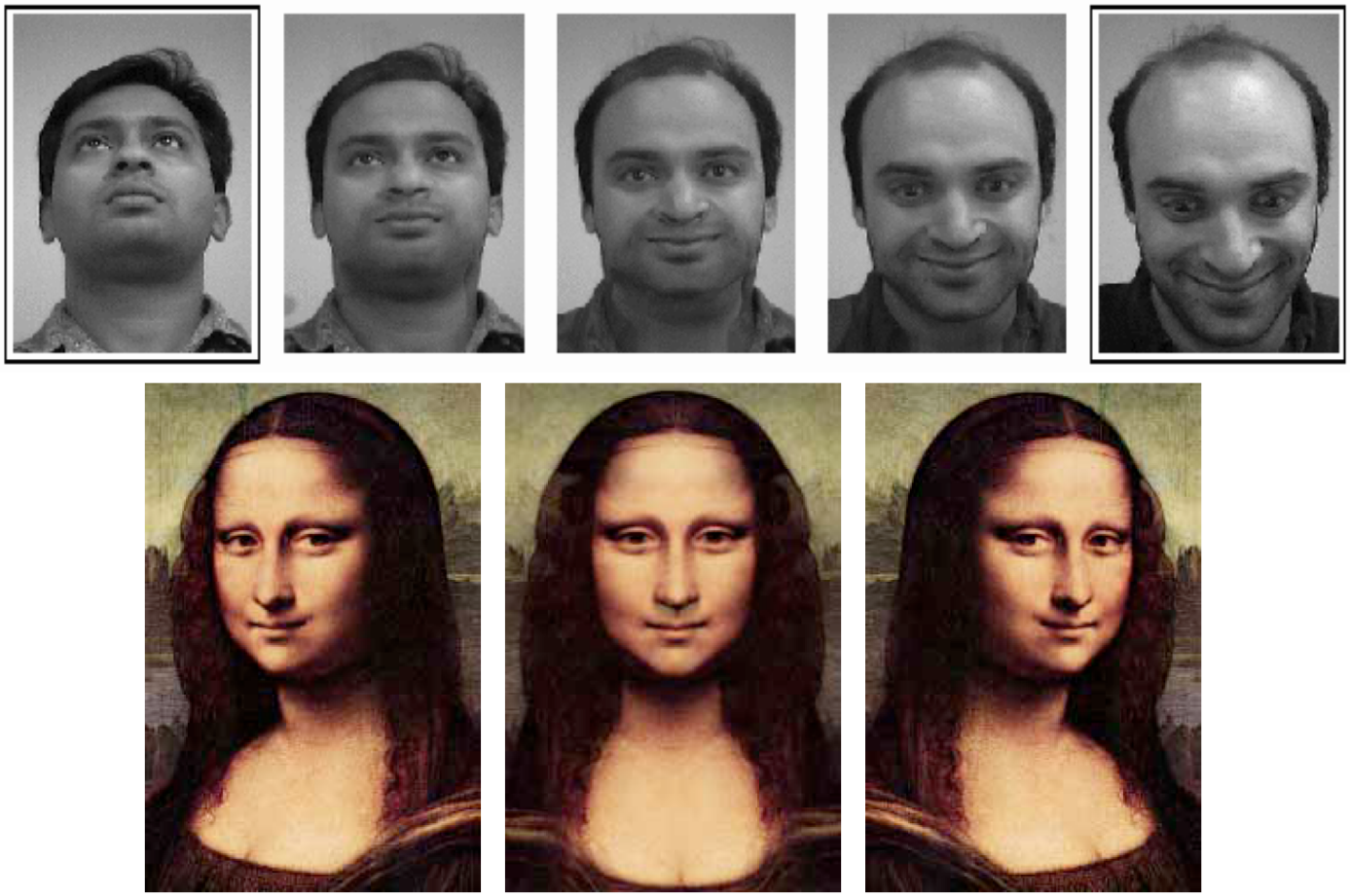
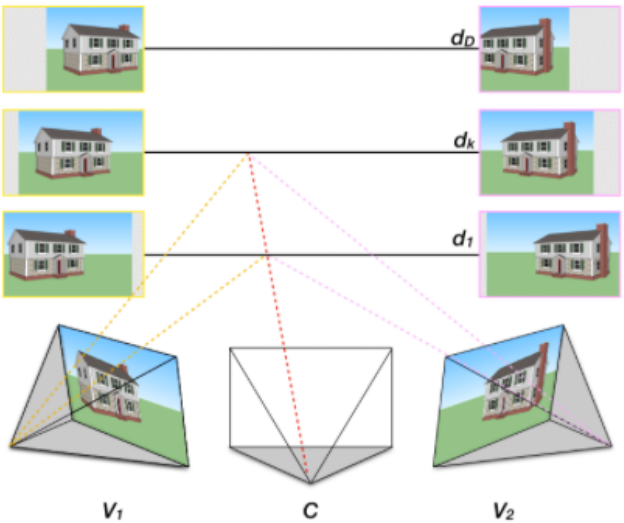
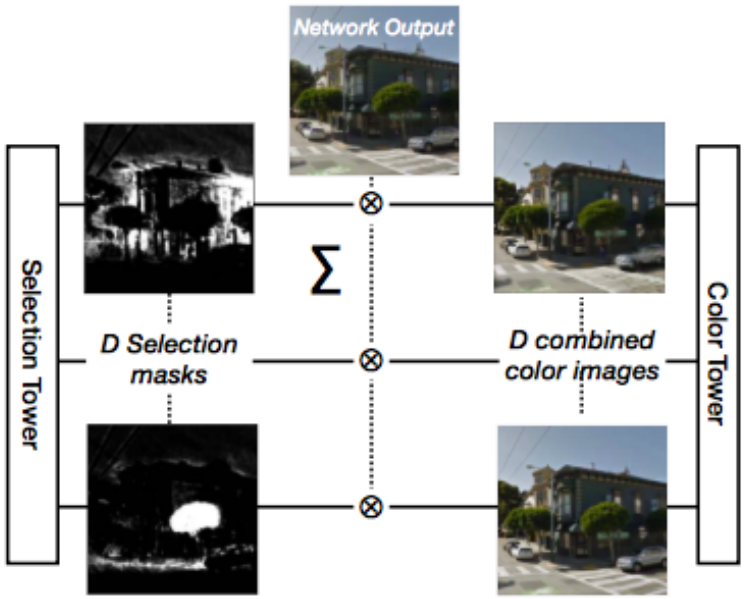


Image source: [Seitz and Dyer 1996]

New view synthesis

Image-Based Rendering with Deep Learning: DeepStereo





Images source: [Flynn et al. 2016]

New view synthesis

Image-Based Rendering with Deep Learning: DeepStereo

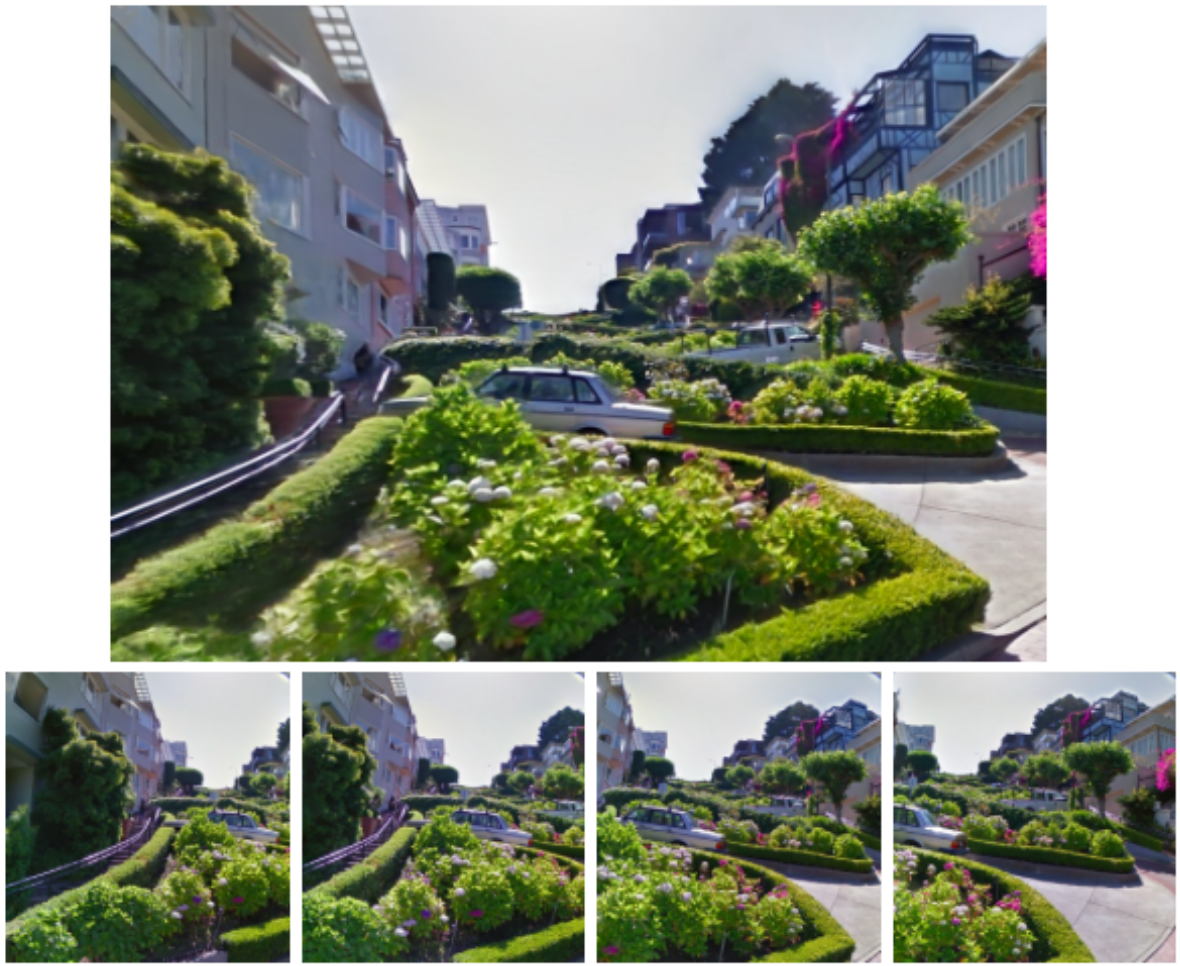


Image source: [Flynn et al. 2016]

New view synthesis

IBR with Deep Learning: Deep View Morphing

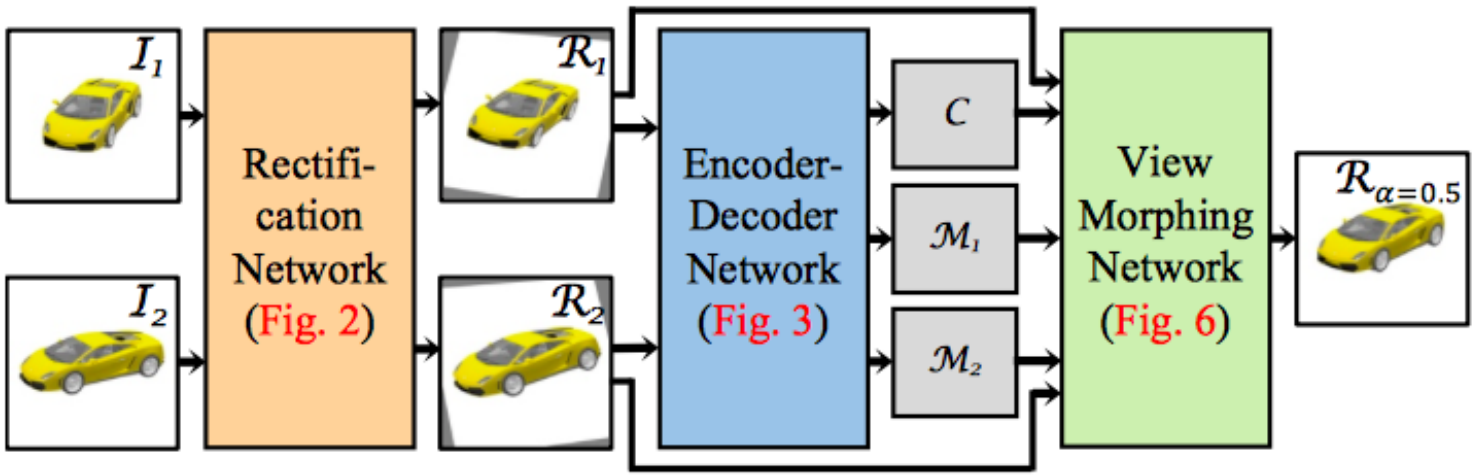
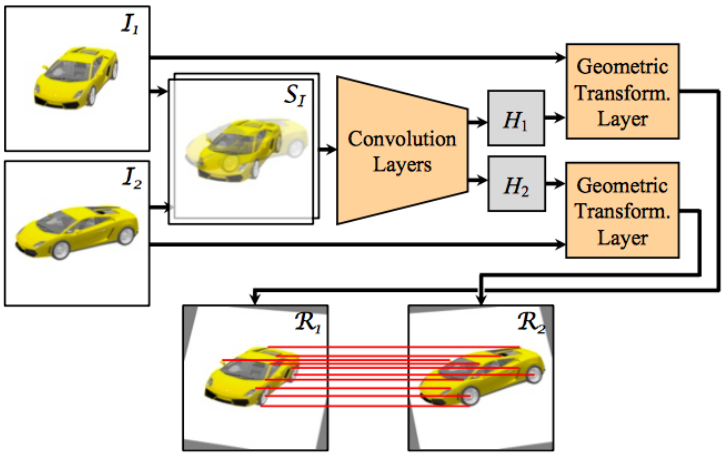


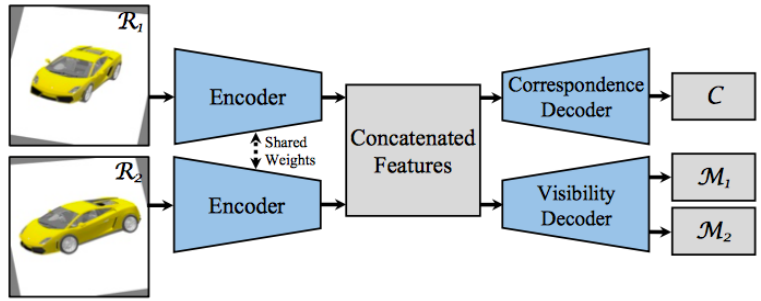
Image source: [Ji et al. 2017]

New view synthesis

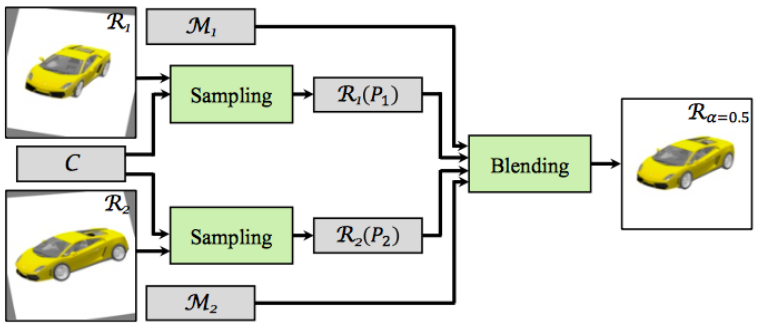
IBR with Deep Learning: Deep View Morphing



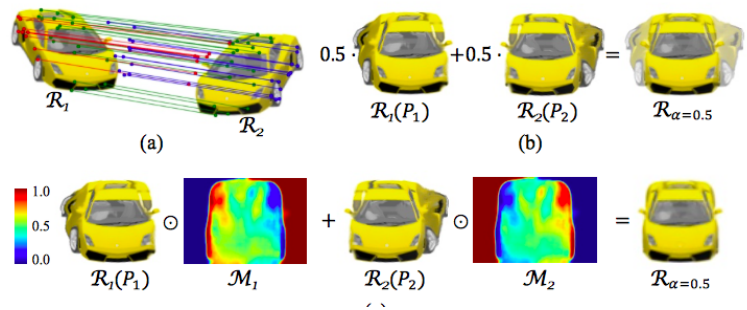
Rectificaction network



Encoder-Decoder network



View morphing (blending) network



Blending procedure

Image source: [Ji et al. 2017]

References

[Flynn et al. 2016] J. Flynn, I. Neulander, J. Philbin, and N. Snavely, DeepStereo: Learning to Predict New Views from the World's Imagery, Proc. CVPR 2016.

[Gortler et al. 1996] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen, The lumigraph, Proc. ACM SIGGRAPH 1996.

[Hartley and Zisserman 2004] R.I. Hartley and A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2004.

[Ji et al. 2017] D. Ji, J. Kwon, M. McFarland, and S. Savarese, Deep View Morphing, arXiv 2017.

[Levoy and Hanrahan 1996] M. Levoy, and P. Hanrahan, Light field rendering, Proc. ACM SIGGRAPH 1996.

[D. Scharstein and R. Szeliski 2002], A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International Journal of Computer Vision, 47,(1-3) 2002.

[Seitz and Dyer 1996] S. Seitz, and C. Dyer, View morphing, Proc. ACM SIGGRAPH 1996.