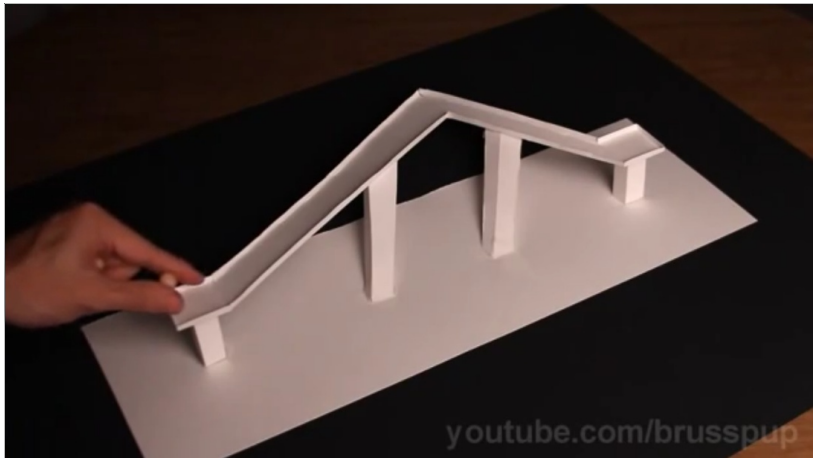
Machine Learning in Graphics and Vision

Prof. Dr.-Ing. Andreas Geiger

Autonomous Vision Group
MPI-IS / University of Tübingen

June 14, 2018

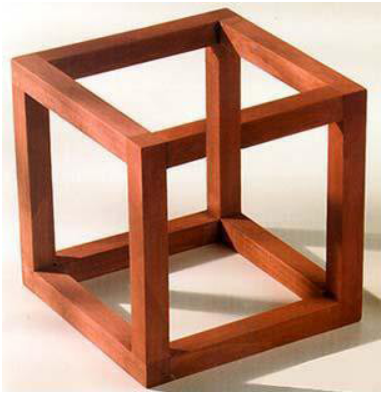
University of Tübingen
MPI for Intelligent Systems
Autonomous Vision Group


youtube.com/brusspup

How to recover 3D from an image?

In general, this is an ill-posed problem, but there are several cues:

- Occlusion
- Parallax
- Perspective
- Accomodation
- Stereopsis



How to recover 3D from an image?

In general, this is an ill-posed problem, but there are several cues:

- Occlusion
- Parallax
- Perspective
- Accomodation
- Stereopsis

How to recover 3D from an image?

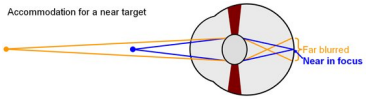
In general, this is an ill-posed problem, but there are several cues:

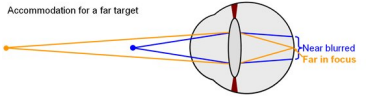
- Occlusion
- Parallax
- Perspective
- Accomodation
- Stereopsis

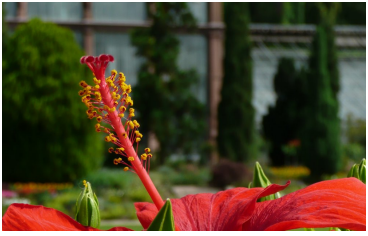
How to recover 3D from an image?

In general, this is an ill-posed problem, but there are several cues:

- Occlusion
- Parallax
- Perspective
- Accomodation
- Stereopsis



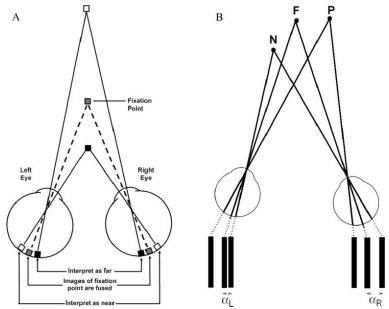




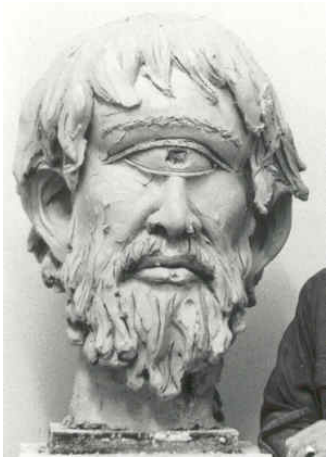
How to recover 3D from an image?

In general, this is an ill-posed problem, but there are several cues:

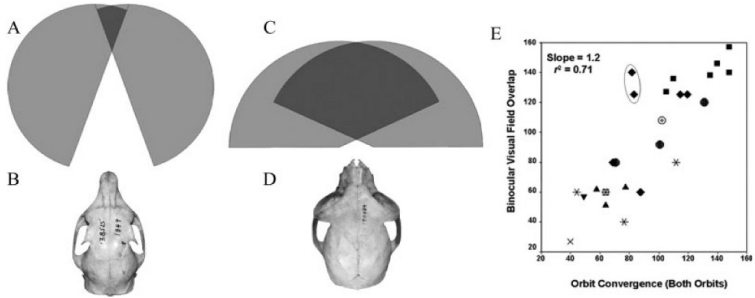
- Occlusion
- Parallax
- Perspective
- Accomodation
- Stereopsis



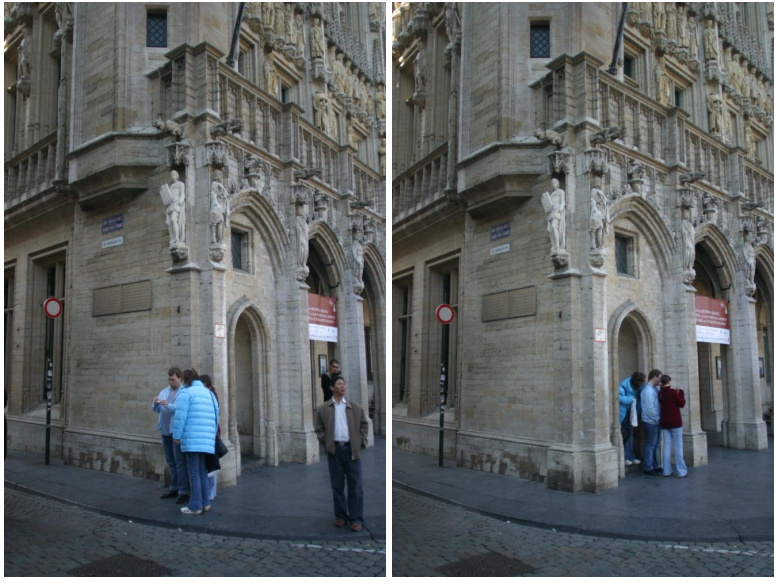
Why Binocular Stereopsis?



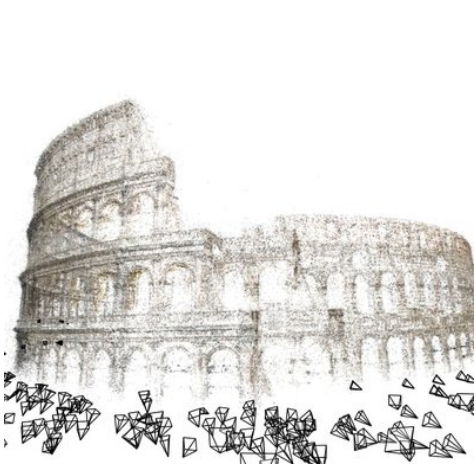
Why Binocular Stereopsis?



Two-View Stereo Matching



Multi-View Reconstruction

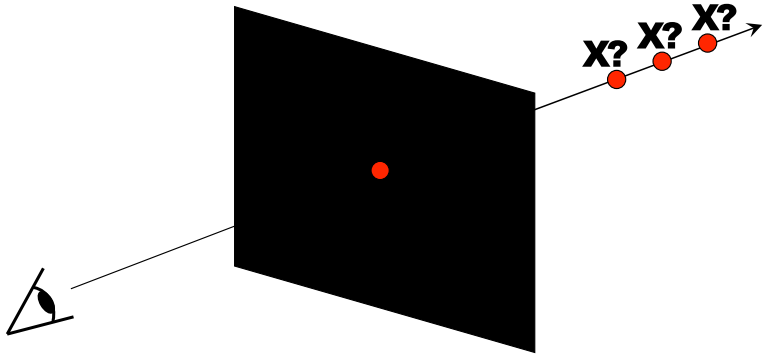


[Agarwal et al. 2011, Schönberger et al. 2016]

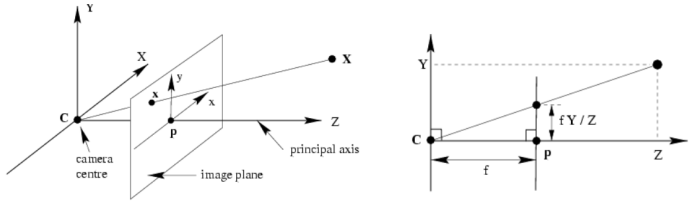
Projective Geometry

Projective Geometry

- Recovery of structure from one image is ambiguous



Pinhole Camera Model

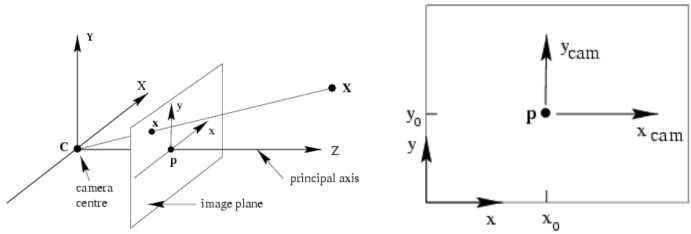


2D projection: Intersection of viewing ray with the image plane

$$\pi : \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f\,X/Z \\ f\,Y/Z \end{pmatrix}$$

- X, Y, Z: 3D coordinates
- x, y: 2D image coordinates
- f: focal length

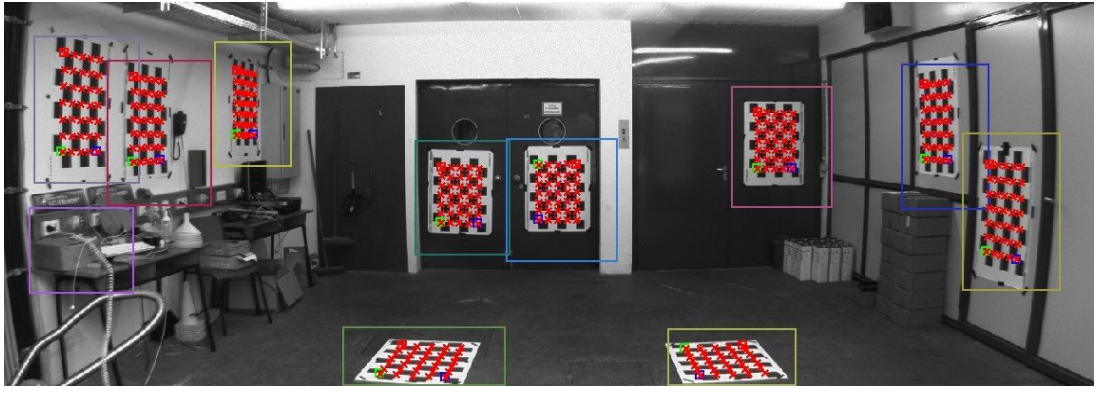
Pinhole Camera Model



- Principal point $\mathbf{p} = (c_x, c_y)^T$: Point where the principal axis intersects the image plane (origin of normalized coordinate system)

$$\pi : \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f\,X/Z + c_x \\ f\,Y/Z + c_y \end{pmatrix}$$

Camera Calibration



How to obtain these parameters for a given camera?

- Closed form approximation of intrinsics (f, c_x, c_y)
- Non-linear optimization of intrinsics (f, c_x, c_y)
- Online toolbox: `www.cvlibs.net/software/calibration`

Epipolar Geometry

Stereo Reconstruction

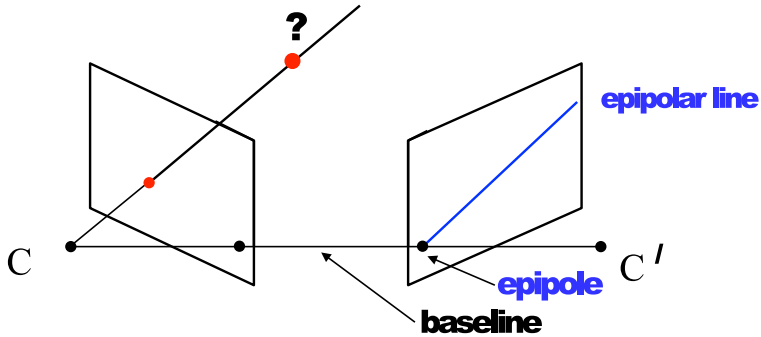
Task

- Construct a 3D model from 2 images

Pipeline:

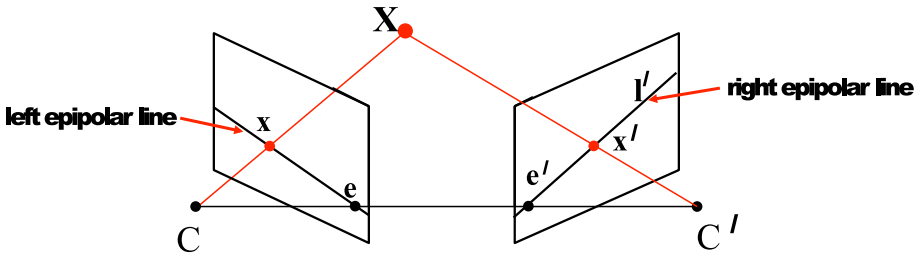
1. Calibrate camera intrinsically
2. Find camera poses using bundle adjustment (not discussed here)
3. Rectify the images given the calibration and the camera poses
4. Compute disparity map / depth map
5. Project points into 3D

Epipolar Geometry



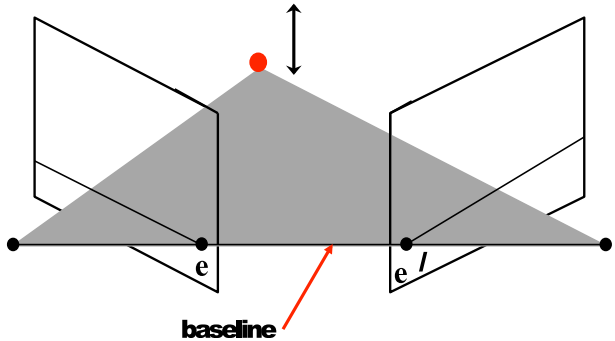
- Lets assume the camera parameters and geometry is known!
- Given a projection of a 3D point in the left image
- Where is it located in 3D?
- On the epipolar line defined by this point and the camera centers
- Reduces the search problem to 1D!

Epipolar Geometry



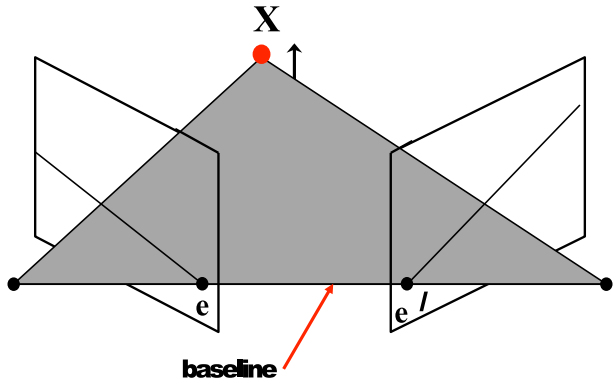
- $\overline{\mathbf{CC}'}$: Baseline (translation between cameras)
- $\mathbf{e}, \mathbf{e}'$: Epipole (intersection of image plane with baseline)
- $\mathbf{l}, \mathbf{l}'$: Epipolar line (intersection of image plane with epipolar plane)

Epipolar Geometry



- Each 3D point defines an epipolar plane (in combination with both camera centers). The set of planes is called “epipolar pencil”.
- All epipolar lines pass through the epipole

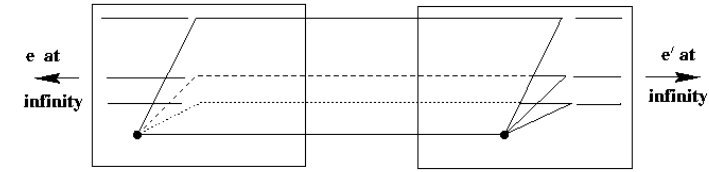
Epipolar Geometry

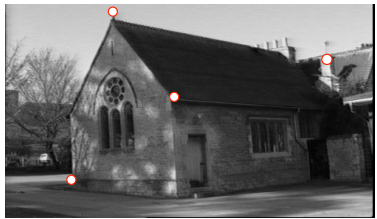


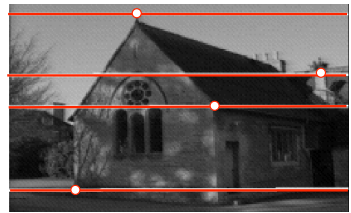
- Each 3D point defines an epipolar plane (in combination with both camera centers). The set of planes is called “epipolar pencil”.
- All epipolar lines pass through the epipole

Epipolar Geometry

What if both cameras face the same direction?

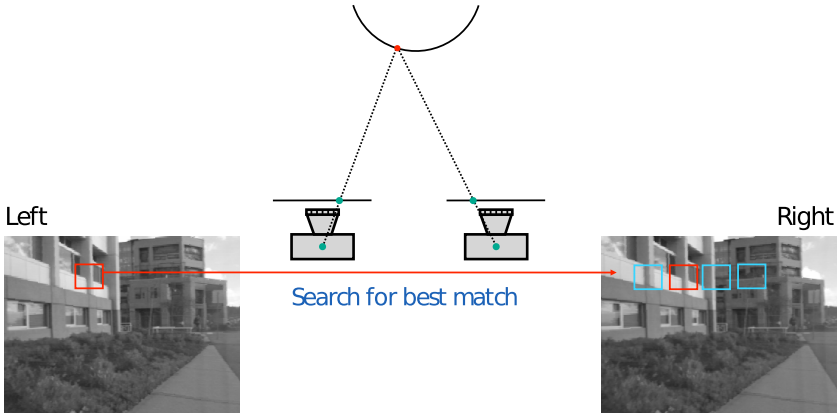






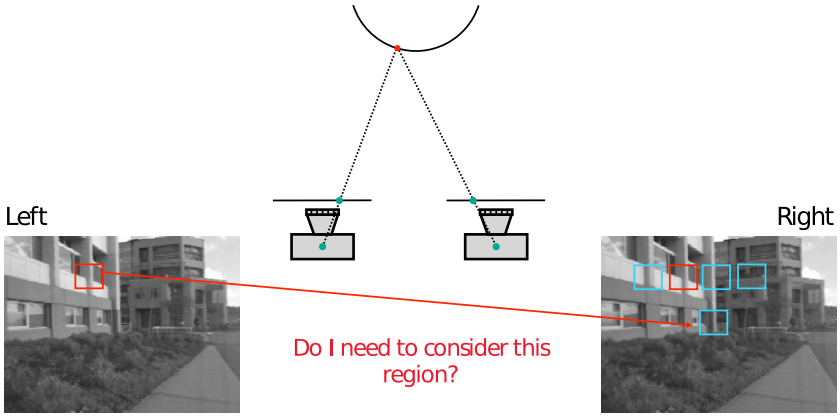
- Epipoles are at infinity, epipolar lines are parallel
- Correspondences along “scanlines” (simplifies computation)

Image Disparity



- The displacement between pixels is called “disparity”: $d = x - x'$

Image Disparity



- The displacement between pixels is called “disparity”: $d = x - x'$

Triangulation

How to recover a 3D point from two corresponding image points?

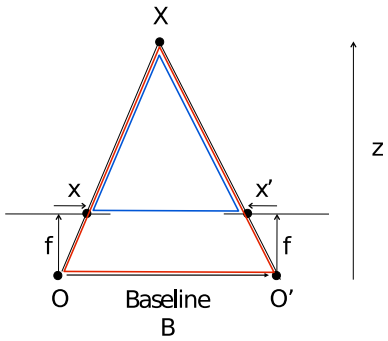
- Equal triangles (only when image planes are parallel)
- Using the definition $d = x - x'$:

$$\frac{Z - f}{B - (x - x')} = \frac{Z}{B}$$

$$ZB - fB = ZB - Z(x - x')$$

$$Z = \frac{fB}{x - x'} = \frac{fB}{d}$$

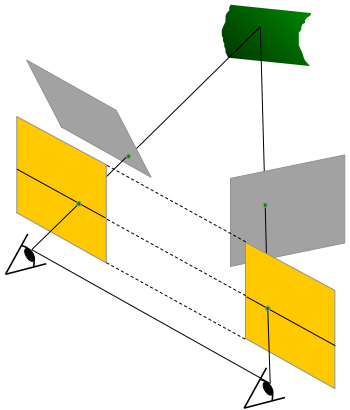
$$d = \frac{fB}{Z}$$



Rectification

What if the images are not in the required setup?

- Rewarp them such that they are! (“Rectification”)
- Map both image planes to a common plane parallel to the baseline by “virtually rotating” to the cameras



Correspondences Unrectified vs. Rectified

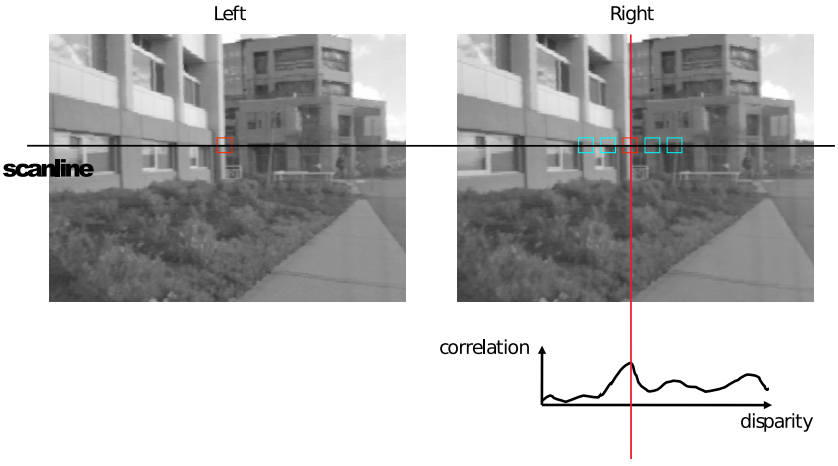
Block Matching

Correspondence Ambiguity

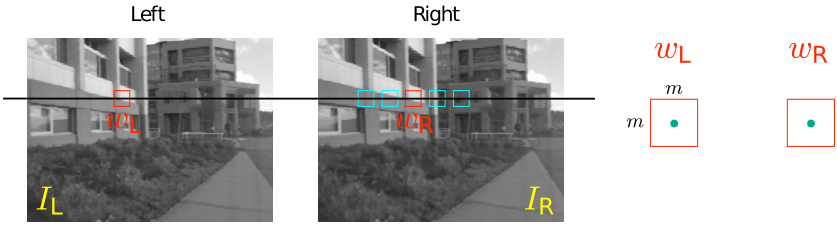
How to determine if two image points correspond?

- Assume that they look “similar”
- A single pixel does not reveal the local structure (ambiguities)
- Compare a small image region/patch!
- But even then the task is difficult:

Normalized Correlation



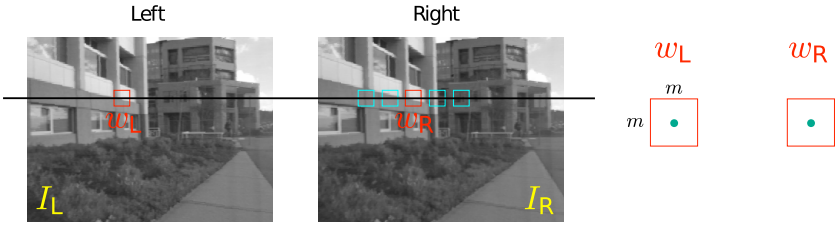
Similarity Metrics



- w_L and w_R are corresponding $m \times m$ windows of pixels
- We can write them as vectors: $\mathbf{w}_L, \mathbf{w}_R \in \mathbb{R}^{m^2}$
- Cross-Correlation:

$$\text{CC}(x, y, d) = \mathbf{w}_L(x, y)^T \mathbf{w}_R(x - d, y)$$

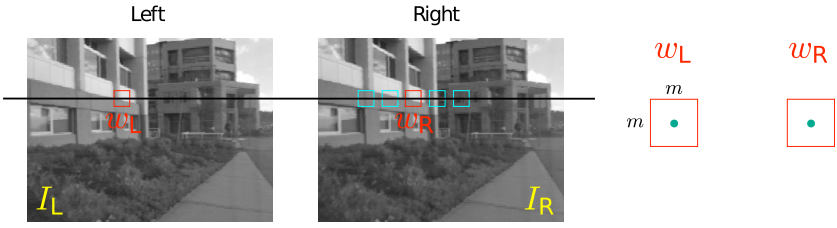
Similarity Metrics



- w_L and w_R are corresponding $m \times m$ windows of pixels
- We can write them as vectors: $\mathbf{w}_L, \mathbf{w}_R \in \mathbb{R}^{m^2}$
- Normalized Cross-Correlation:

$$\text{NCC}(x, y, d) = \frac{\left(\mathbf{w}_L(x, y) - \bar{\mathbf{w}}_L(x, y)\right)^T \left(\mathbf{w}_R(x - d, y) - \bar{\mathbf{w}}_R(x - d, y)\right)}{\|\mathbf{w}_L(x, y) - \bar{\mathbf{w}}_L(x, y)\|_2 \, \|\mathbf{w}_R(x - d, y) - \bar{\mathbf{w}}_R(x - d, y)\|_2}$$

Similarity Metrics



- w_L and w_R are corresponding $m \times m$ windows of pixels
- We can write them as vectors: $\mathbf{w}_L, \mathbf{w}_R \in \mathbb{R}^{m^2}$
- Sum of squared differences (SSD):

$$\text{SSD}(x, y, d) = \|\mathbf{w}_L(x, y) - \mathbf{w}_R(x - d, y)\|_2^2$$

Block Matching

$m = 3$ $m = 20$

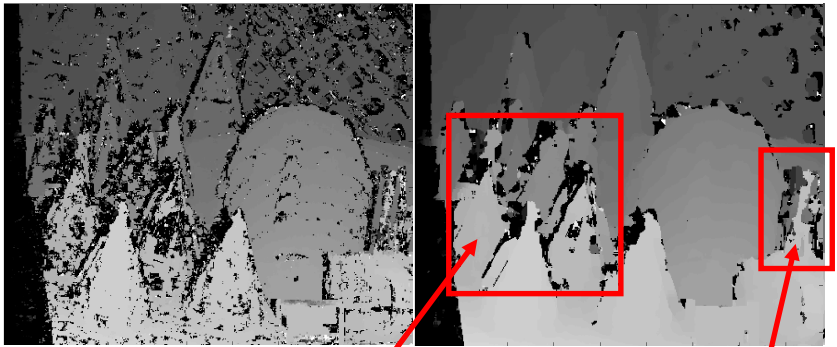
Block Matching:

- Choose some disparity range $[0, d_{max}]$
- Winner-takes-all (WTA): For all pixels $\mathbf{x} = (x, y)$ try all possible disparities d and choose the one that maximizes the cross-correlation or minimizes the SSD)
- Do this for both images and apply left-right consistency check to remove outliers

Challenges:

- Which window size to choose? Tradeoff: Ambiguity $\leftrightarrow$ Bleeding!
- Block matching = fronto-parallel assumption (often invalid!)

Effect of Window Size



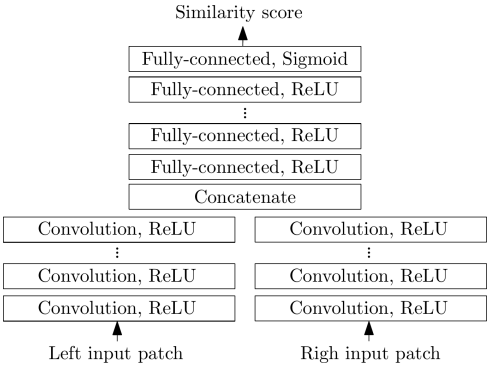
Siamese Networks for Stereo Matching

Siamese Networks for Stereo Matching

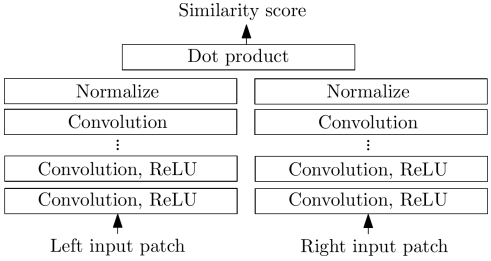
- Hand crafted features and similarity metrics do not take into consideration all relevant geometric and radiometric invariances
- The world is too complex to specify this by hand
- Computing the matching cost can be treated as supervised learning problem

Left patch	Right patch	Label
		Good match
		Bad match
⋮		⋮

Siamese Networks for Stereo Matching



- Learn features & sim. metric
- Potentially more expressive
- Slow at inference time



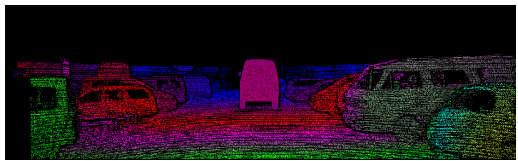
- Learn features, use dot-product
- Can be applied convolutionally
- Fast at inference time

[Zbontar & LeCun, JMLR 2016]

Siamese Networks for Stereo Matching

Overview

- Train CNN patch-wise based on images with ground truth disparity maps (*e.g.*, KITTI Stereo 2015: `http://www.cvlibs.net/datasets/kitti/`)



- Calculate features for each of the two images using trained model
- Correlate features between both images (dot product)
- Find maximum for every pixel or apply post-processing (*e.g.*, MRF)

Training

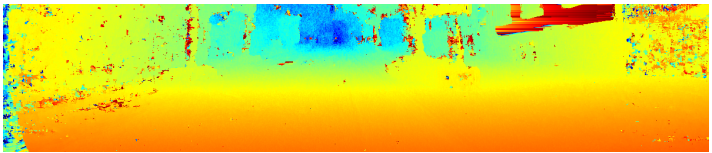
The training set is composed of patch triplets:

$$< \mathcal{P}^L(\mathbf{p}),\ \mathcal{P}^R(\mathbf{q}_{\text{neg}}),\ \mathcal{P}^R(\mathbf{q}_{\text{pos}}) >$$

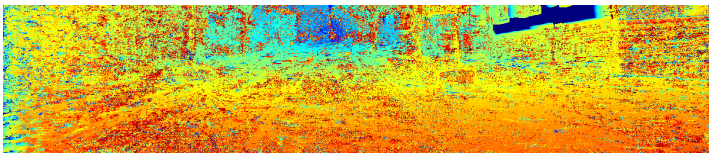
- $\mathcal{P}^L(\mathbf{p})$ is a patch from the left image centered at $\mathbf{p} = (x, y)$
- $\mathcal{P}^R(\mathbf{q})$ is a patch from the right image centered at $\mathbf{q} = (x, y)$
- Negative example: $\mathbf{q}_{\text{neg}} = (x - d + o_{\text{neg}}, y)$
  - o_{neg} chosen randomly from $\{-N_{\text{hi}}, \ldots, -N_{\text{lo}}, N_{\text{lo}}, \ldots, N_{\text{hi}}\}$
- Positive example: $\mathbf{q}_{\text{pos}} = (x - d + o_{\text{pos}}, y)$
  - o_{pos} chosen randomly from $\{-P_{\text{hi}}, \ldots, P_{\text{hi}}\}$
- Minimize hinge loss: $\mathcal{L} = \max(0, m + s_- - s_+)$
  - s_-/s_+ is the output of the network for the negative/positive example
  - The loss is zero when the similarity of the positive example is greater than the similarity of the negative example by at least margin m

Matching Cost

Left input image

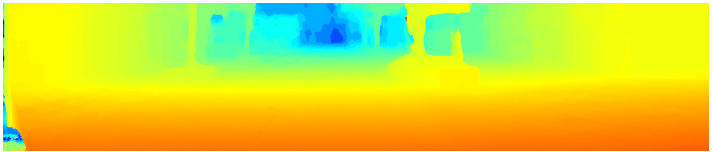


Convolutional neural network

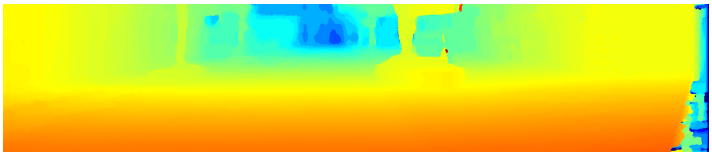


Census transform

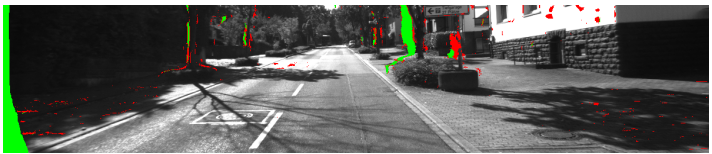
MRF and Left-Right Consistency Check



Disparity map for the left image



Disparity map for the right image

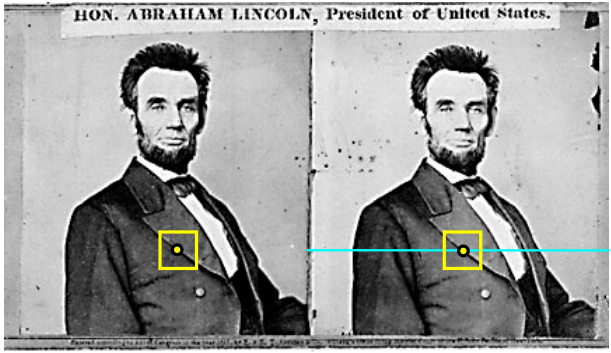


Left-right consistency check

Runtime

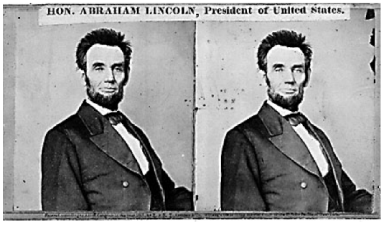
- Original version implemented in CUDA and Lua/Torch7
- Run on Nvidia GTX Titan GPU
- Training takes 5 hours
  - 45 million training examples
  - 16 epochs
  - Stochastic gradient descent with batch size of 128
- Inference for a single pair of images takes 6 seconds / 100 seconds
  - 1 second / 95 seconds for the neural network (depending on architecture)
  - 3 seconds for semiglobal matching
  - 2 seconds for cost aggregation and post-processing

The Underlying Assumption: Similarity Constraint



- Corresponding regions in both images should look similar ...
- ... and non-corresponding regions should look different.
- When will the similarity constraint fail?

Similarity Constraint: Failure Cases



Textureless surfaces

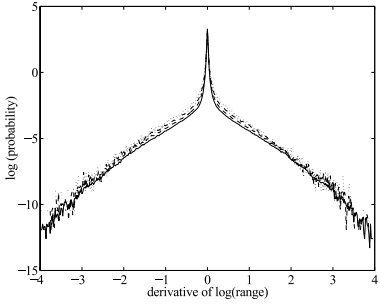
Occlusions, repetition

Non-Lambertian surfaces, specularities

Spatial Regularization

How does the real world look like?

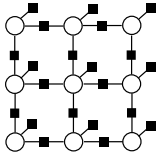
- Leverage real-world statistics
- E.g.: Brown range image database [Mumford et al.]



Stereo MRF

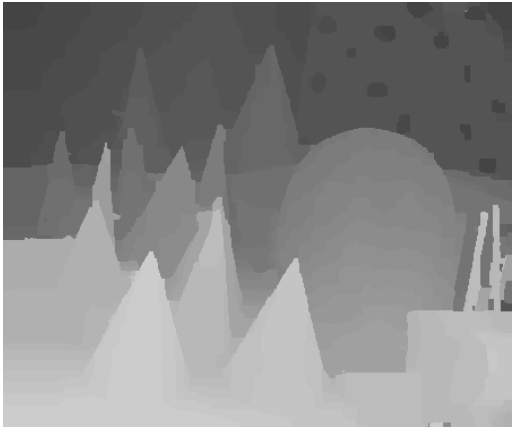
- Specify a loopy Markov Random Field (MRF) on a grid and solve for the whole disparity map $\mathbf{D}$ at once. MAP solution = minimum of energy.

$$p(\mathbf{D}) \propto \exp\left\{ -\sum_i \psi_{data}(d_i) - \lambda \sum_{i \sim j} \psi_{smooth}(d_i, d_j) \right\}$$

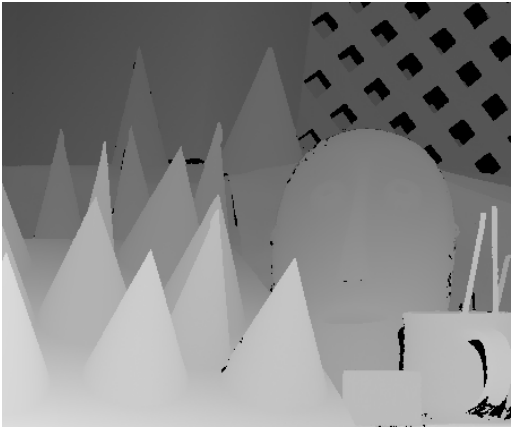


- $i \sim j$: neighboring pixels on a 4-connected grid
- Unary terms: Matching cost $\psi_{data}(d)$
- Pairwise terms: Smoothness between adjacent pixels, e.g.:
  - Potts: $\psi_{smooth}(d, d') = [d \neq d']$
  - Truncated l_1: $\psi_{smooth}(d, d') = \min(|d - d'|, \tau)$
- Solve MRF approximately using BP / graph cuts (see lecture 10/11)

Stereo MRF – Results



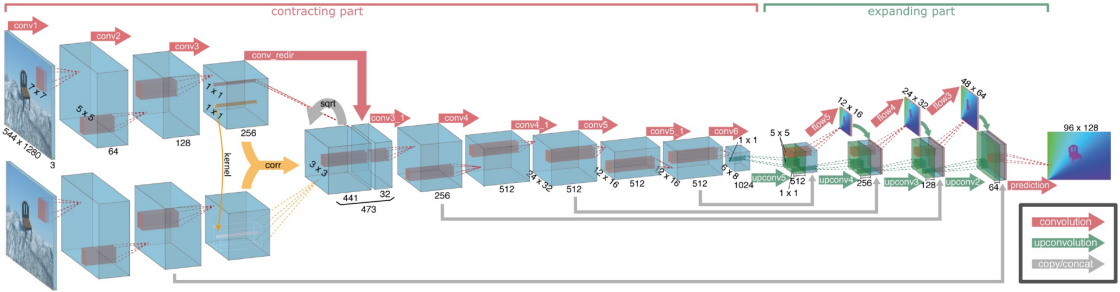
Inference Results



Ground Truth

End-to-End Learning for Disparity Estimation

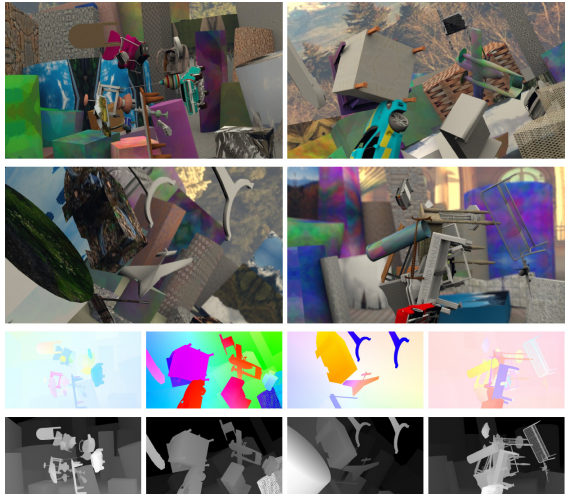
DispNet



[Mayer *et al.*, CVPR 2016]

- Encoder: Convolution with stride, decoder: Upconvolution
- Skip-Connections (to retain details)
- Correlation layer (40px displacement corresponding to 160px in input image)
- Multi-scale loss (disparity error in pixels)
- Curriculum learning (present training samples in easy-to-hard order)

Synthetic Datasets



Flying Things

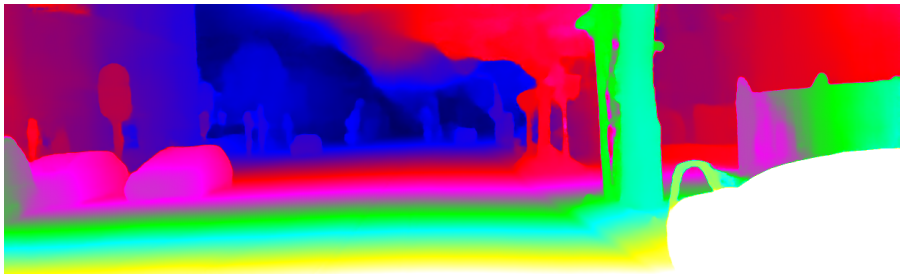
Synthetic Datasets

Monkaa

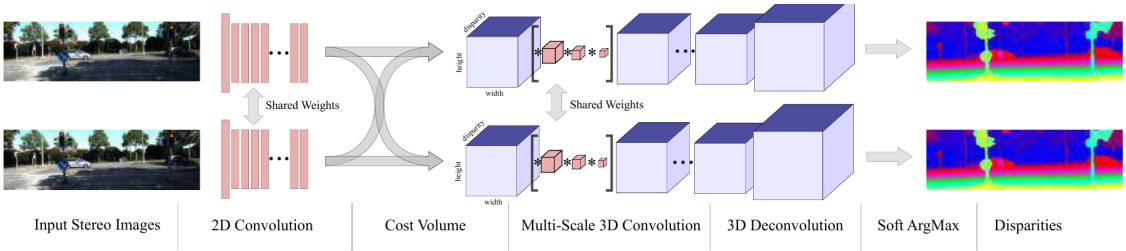
Synthetic Datasets

MPI Sintel

DispNet Results on KITTI Dataset



GC-Net



[Kendall *et al.*, ICCV 2017]

- Key idea: calculate cost volume and apply 3D convolutions
- Proposes softargmin operator to convert cost into disparity estimate

$$\text{softargmin} = \sum_{d=0}^{D_{max}} d \cdot \sigma(-cost_d)$$

- Slightly better performance but large memory requirements

Questions?