Fundamentals of Machine Learning

Chenyi Chen

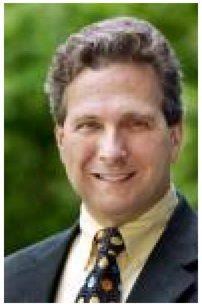
What's learning?

- Example problem: face recognition

Prof. K

Prof. F

Prof. P



Prof. V

Chenyi

- Training data: a collection of images and labels (names)

Who is this guy?

- Evaluation criterion: correct labeling of new images

What's learning?

- Example problem: scene classification


road


road


sea


mountain


city

- a few labeled training images


What's the label of this image?

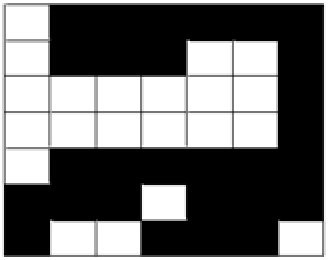
- goal to label yet unseen image

Why learning?

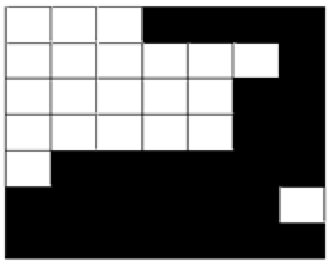
- The world is very complicated
- We don’t know the exact model/mechanism between input and output
- Find an approximate (usually simplified) model between input and output through learning
- Principles of learning are “universal”
  - society (e.g., scientific community)
  - animal (e.g., human)
  - machine

A Taste of Machine Learning

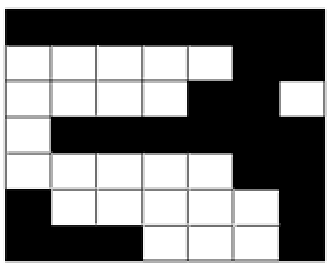
Learning, biases, representation



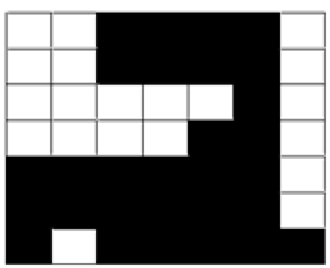
“yes”



“yes”



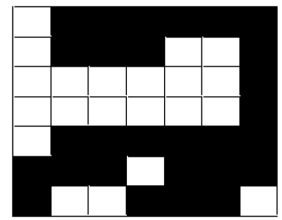
“no”



?

Representation

- There are many ways of presenting the same information

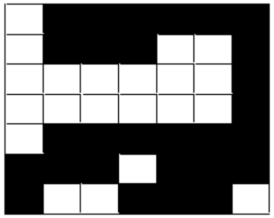


0111111001110010000000100000001001111110111011111001110111110001

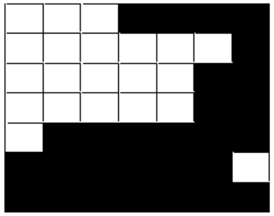
- The choice of representation may determine whether the learning task is very easy or very difficult

Representation

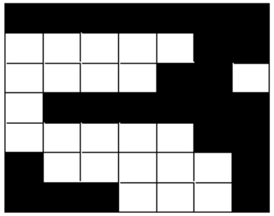
0111111001110010000000100000001001111110111011111001110111110001	"yes"
0001111100000011000001110000011001111110111111001111111101111111	"yes"
1111111000000110000011000111111000000111000000111110001101110001	"no"



"yes"



"yes"

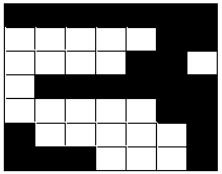


"no"

Hypothesis class

- Representation: examples are binary vectors of length $d = 64$

$$\mathbf{x} = [111\ldots0001]^T \;=$$



True label

and labels $y \in \{-1, 1\}$ ("no","yes")

- The mapping from examples to labels is a "linear classifier"

$$\hat{y} = \text{sign}\,(\,\theta \cdot \mathbf{x}\,) = \text{sign}\,(\,\theta_1 x_1 + \ldots + \theta_d x_d\,)$$

where θ is a vector of *parameters* we have to learn from examples.

Estimated label

We want to minimize the difference between them!

Estimation

$\mathbf{x}$	y
0111111001110010000000100000001001111110111011111001110111110001	+1
0001111100000011000001110000011001111110111111001111111100000011	+1
1111111000000110000011000111111000000111100000111110001101111111	−1
. . .	. . .

- How do we adjust the parameters θ based on the labeled examples?

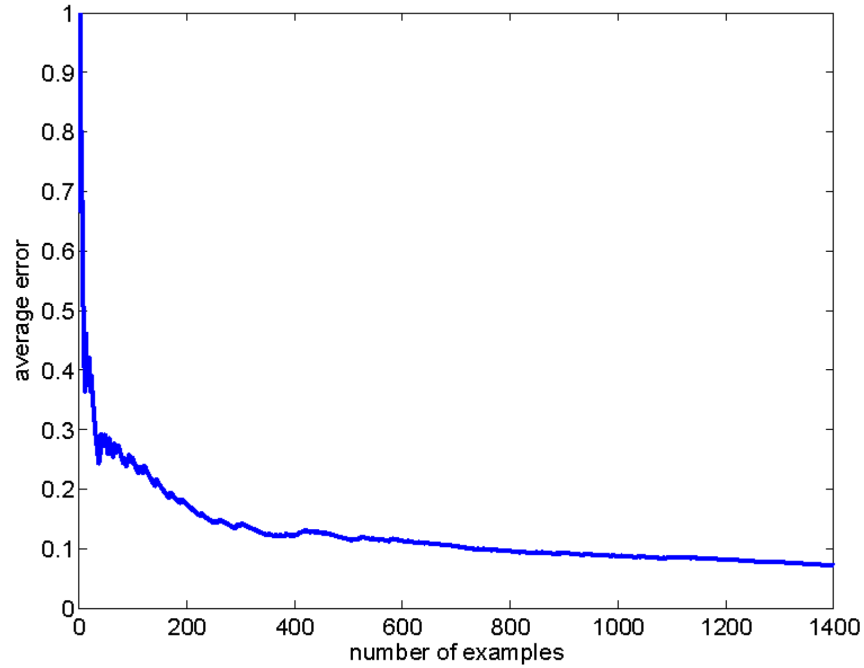
$$\hat{y} = \text{sign}\,(\,\theta \cdot \mathbf{x}\,)$$

For example, we can simply refine/update the parameters whenever we make a mistake:

$$\theta_i \leftarrow \theta_i + y\,x_i,\ \ i = 1, \ldots, d \quad \text{if prediction was wrong}$$

Evaluation

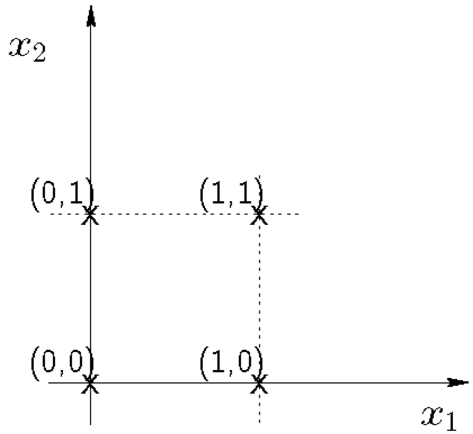
- Does the simple mistake driven algorithm work?



(average classification error as a function of the number of examples and labels seen so far)

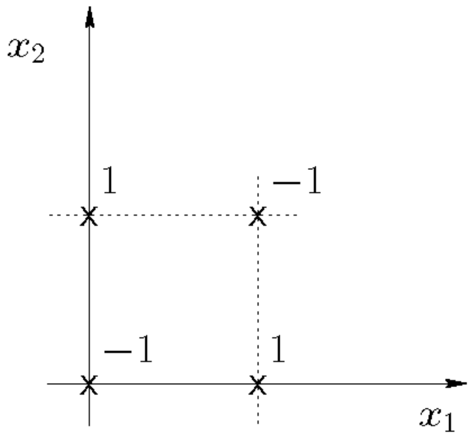
Model selection

- The simple linear classifier cannot solve all the problems (e.g., XOR)



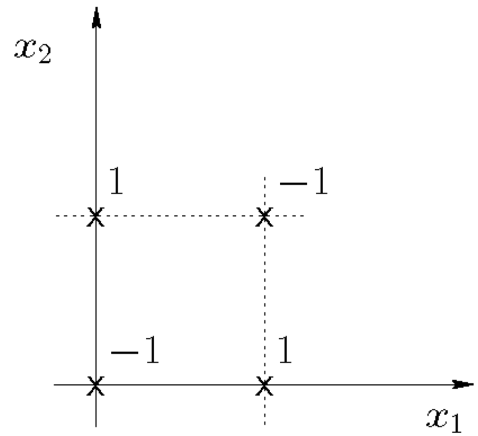
Model selection

- The simple linear classifier cannot solve all the problems (e.g., XOR)



Model selection

- The simple linear classifier cannot solve all the problems (e.g., XOR)

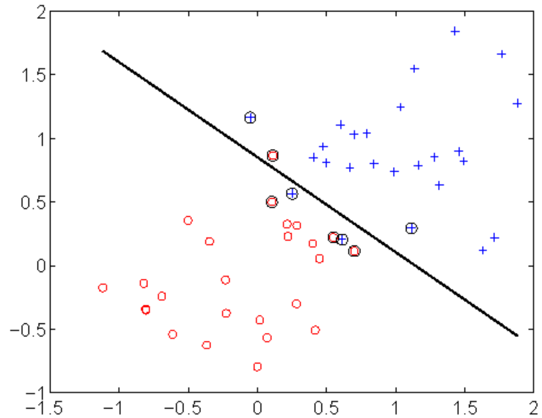


- Can we rethink the approach to do even better?

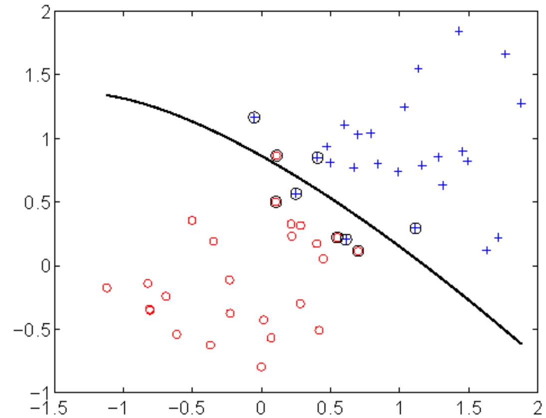
  We can, for example, add "polynomial experts"

$$\hat{y} = \text{sign}\left(\theta_1 x_1 + \ldots + \theta_d x_d + \theta_{12} x_1 x_2 + \ldots \right)$$

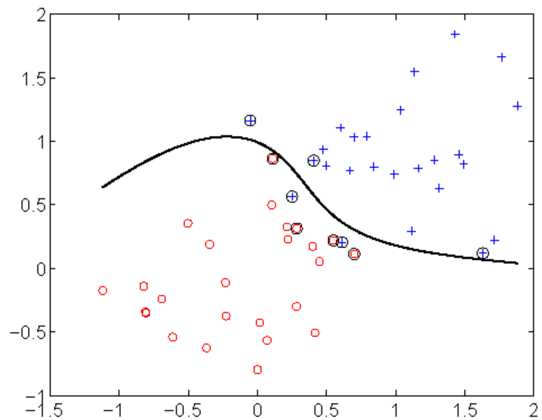
Model selection cont'd



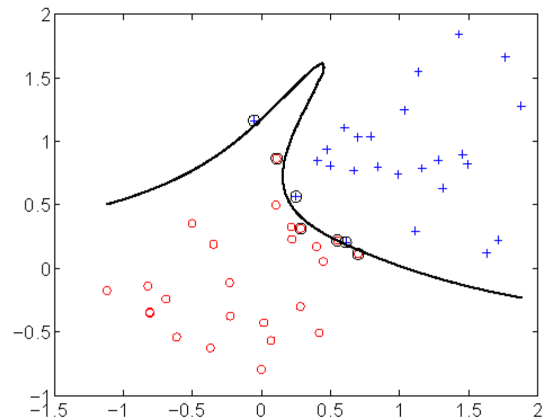
linear



2^{nd} order polynomial



4^{th} order polynomial



8^{th} order polynomial

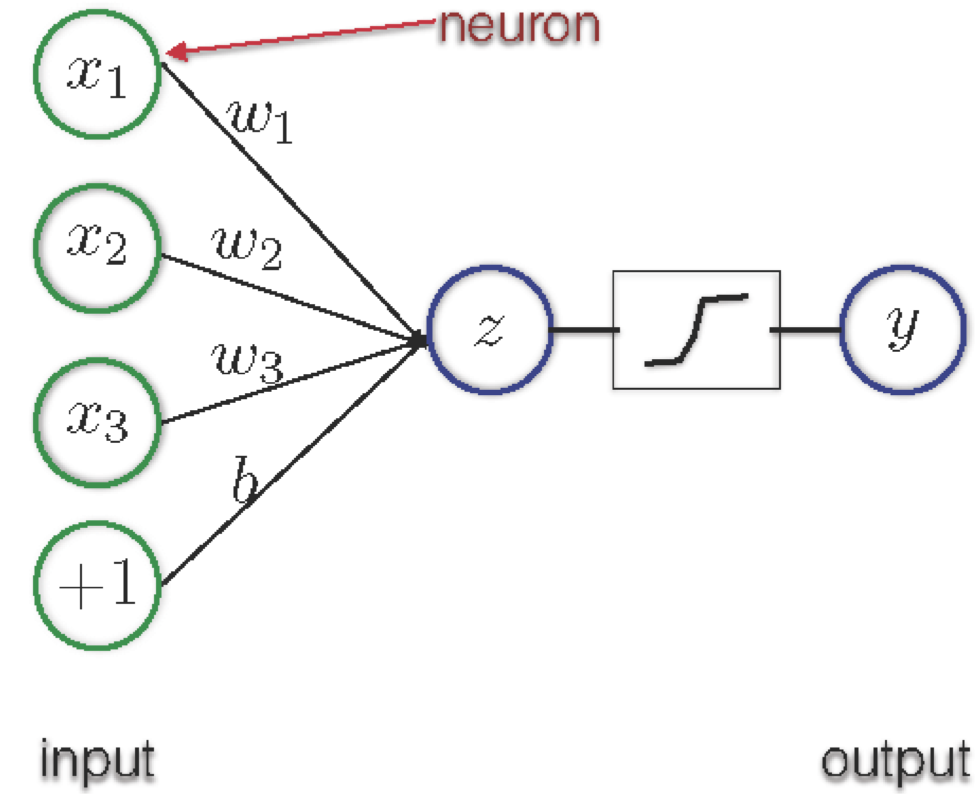
Types of learning problems (not exhaustive)

- *Supervised* learning: explicit feedback in the form of examples and target labels
  - goal to make predictions based on examples (classify them, predict prices, etc)
- *Unsupervised* learning: only examples, no explicit feedback
  - goal to reveal structure in the observed data
- *Semi-supervised* learning: limited explicit feedback, mostly only examples
  - tries to improve predictions based on examples by making use of the additional "unlabeled" examples
- *Reinforcement* learning: delayed and partial feedback, no explicit guidance
  - goal to minimize the cost of a sequence of actions (policy)

Artificial Neural Network

Neural Networks

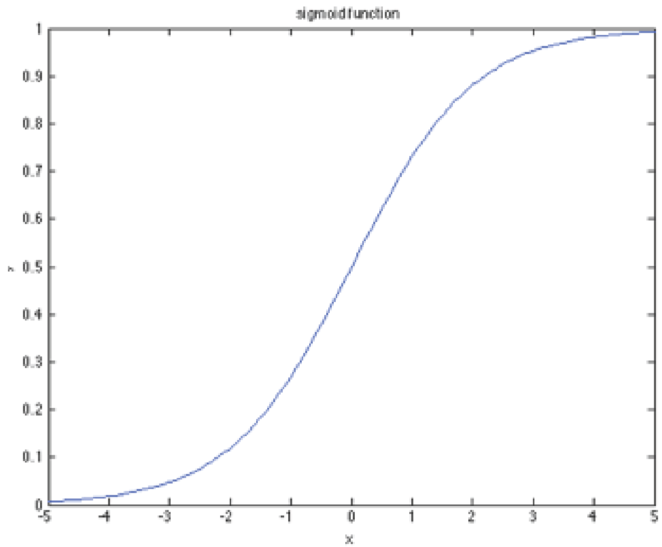
basic building blocks



$$z = \sum_i x_i w_i + b, \quad y = f(z)$$

where f is a activation function:

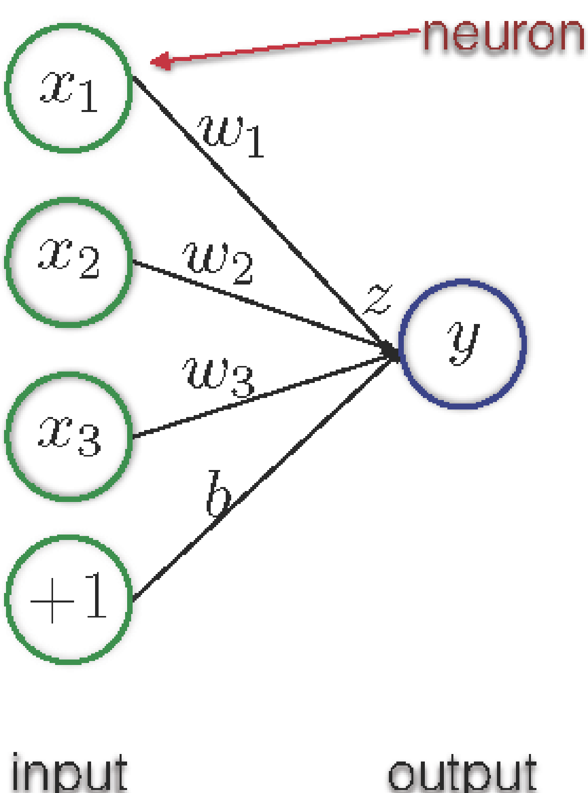
$$f(z) = \sigma(z) = \frac{1}{1 + exp(-z)}$$



- sigmoid is bounded between 0 and 1
- monotonically increasing
- differentiation: $\sigma'(z) = \sigma(z) * (1 - \sigma(z))$

Neural Networks

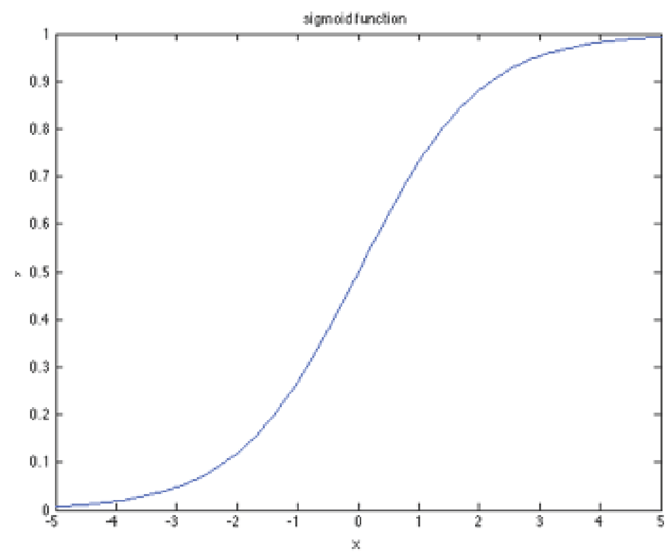
basic building blocks



$$z = \sum_i x_i w_i + b, \quad y = f(z)$$

where f is a activation function:

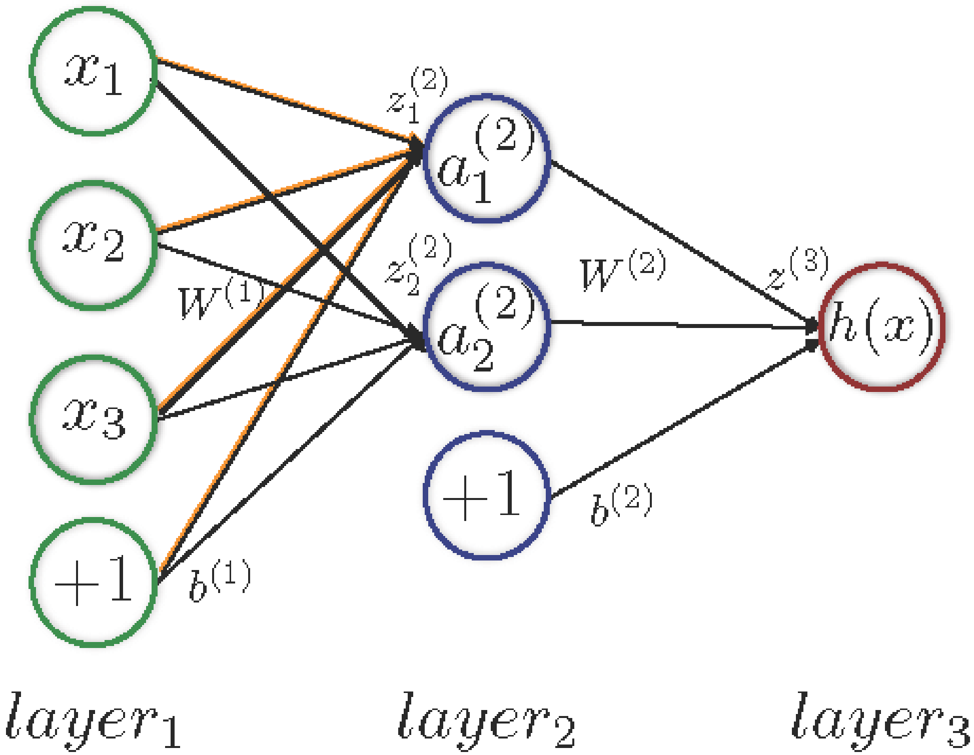
$$f(z) = \sigma(z) = \frac{1}{1 + exp(-z)}$$



- sigmoid is bounded between 0 and 1
- monotonically increasing
- differentiation: $\sigma'(z) = \sigma(z) * (1 - \sigma(z))$

Neural Networks

representation



feed-forward

for each neuron in the next layer:

$$z_i^{(2)} = \sum_{j=1}^{n} w_{ij}^{(1)} x_j + b_i^{(1)}, \quad a_i^{(2)} = f(z_i^{(2)})$$

$$z_1^{(2)} = w_{11}^{(1)} x_1 + w_{21}^{(1)} x_2 + w_{31}^{(1)} x_3 + b_1^{(1)}, \quad a_1^{(2)} = f(z_1^{(2)})$$

compactly:

$$z^{(2)} = W^{(1)} x + b^{(1)}, \quad a^{(2)} = f(z^{(2)})$$

$$z^{(3)} = W^{(2)} x + b^{(2)}, \quad a^{(3)} = f(z^{(3)})$$

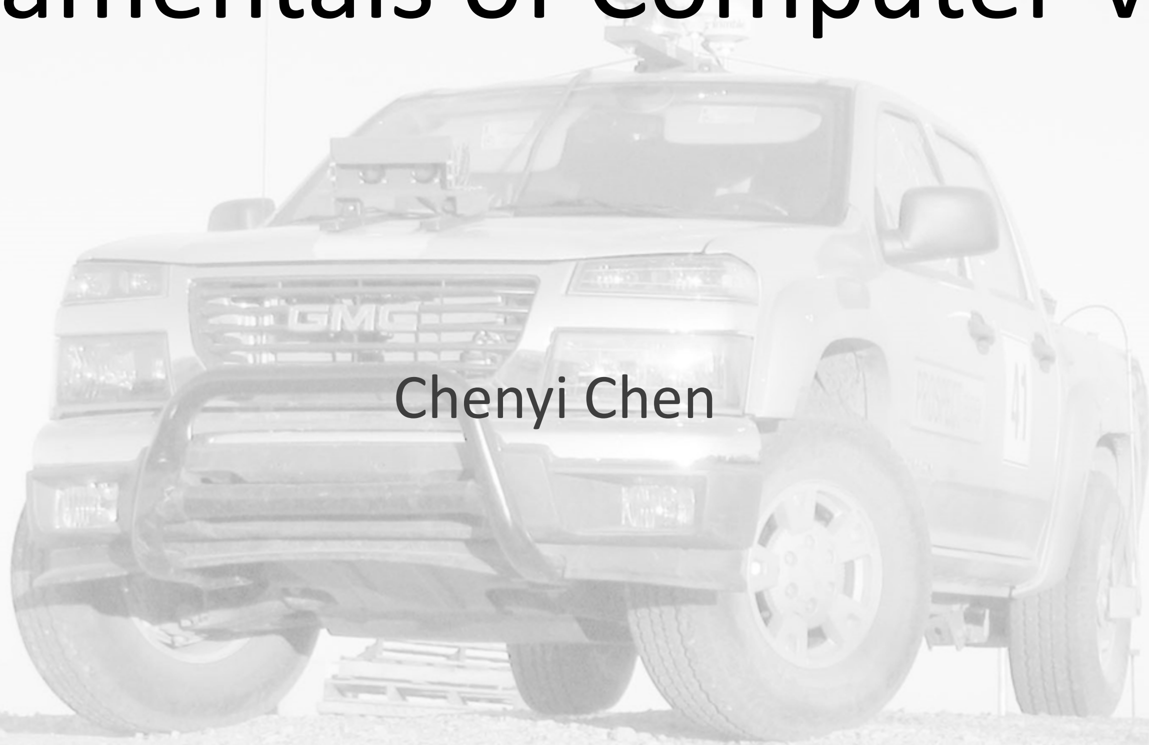
globally models a function:

$$\hat{y} = h_{W,b}(x)$$

where W and b are model parameters

Fundamentals of Computer Vision

Chenyi Chen



What is Computer Vision?

- Input: images
- Output: information about the world

What is Computer Vision?

Example:

- What is in this image?
- Who is in this image?
- Where are they?
- What are they doing?

What is Computer Vision?

Other questions:

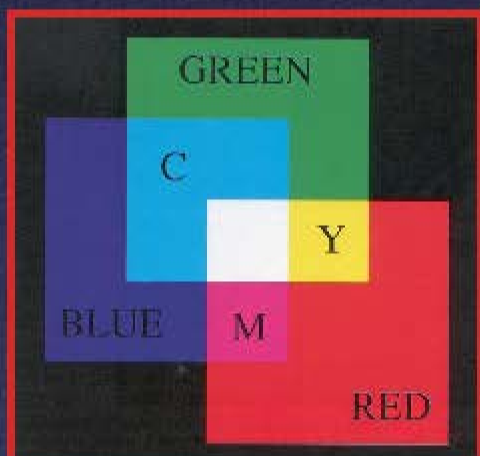
- What camera settings were used?
- Which pixels go with which objects?
- What is the scene description in 3D?

How Do We Represent Colors Digitally?

Common color models

- ➢ **RGB**
- CMY
- HLS
- HSV
- XYZ
- Others

R	G	B	Color
0.0	0.0	0.0	Black
1.0	0.0	0.0	Red
0.0	1.0	0.0	Green
0.0	0.0	1.0	Blue
1.0	1.0	0.0	Yellow
1.0	0.0	1.0	Magenta
0.0	1.0	1.0	Cyan
1.0	1.0	1.0	White
0.5	0.0	0.0	
1.0	0.5	0.5	
1.0	0.5	0.0	
0.5	0.3	0.1	



Colors are additive

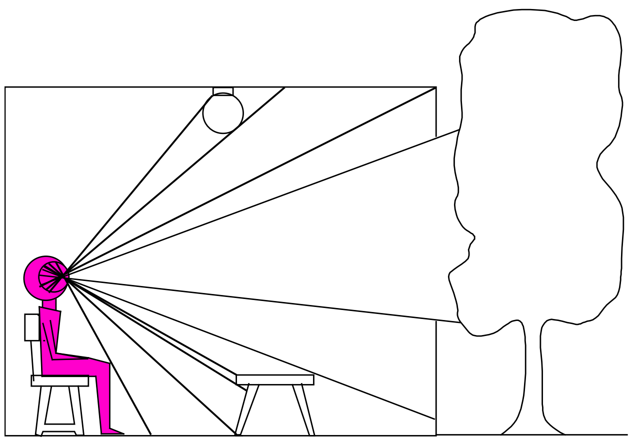
Camera Projection

Camera Projection

Camera Projection

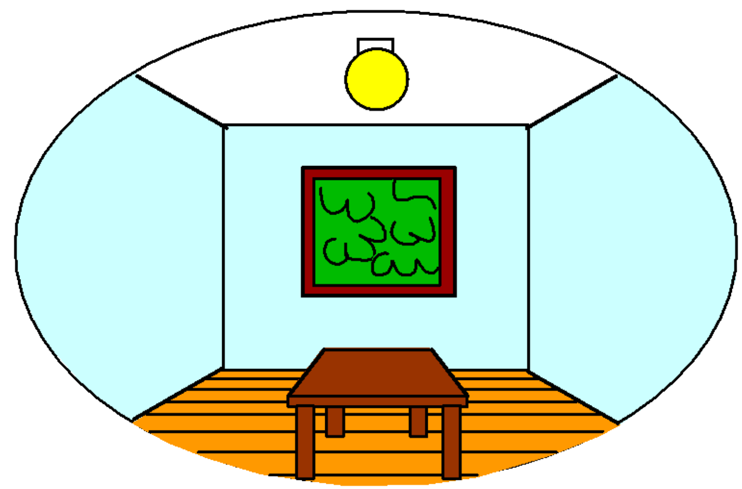
Dimensionality Reduction Machine (3D to 2D)

3D world



Point of observation

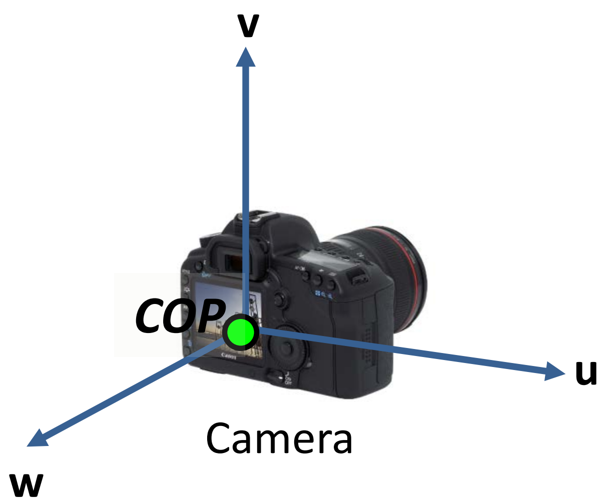
2D image



What have we lost?

Slide by A. Efros

A Tale of Two Coordinate Systems

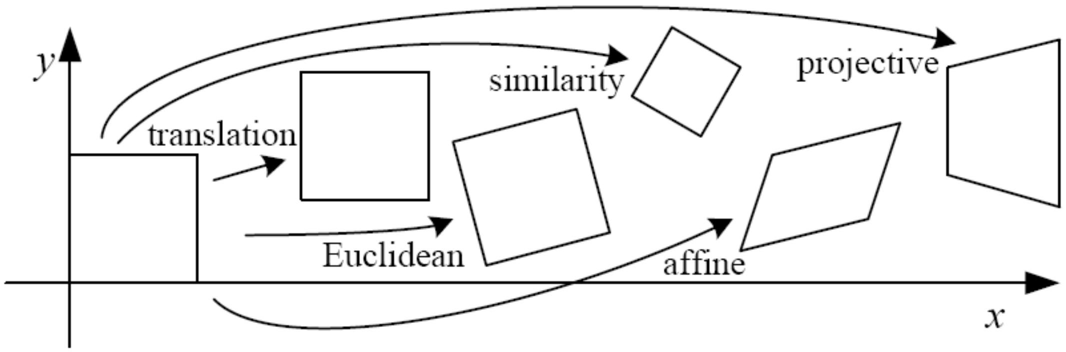


"The World"

Two important coordinate systems:

1. *World* coordinate system
2. *Camera* coordinate system

Geometric Transformations



What is the geometric relationship between these two images?

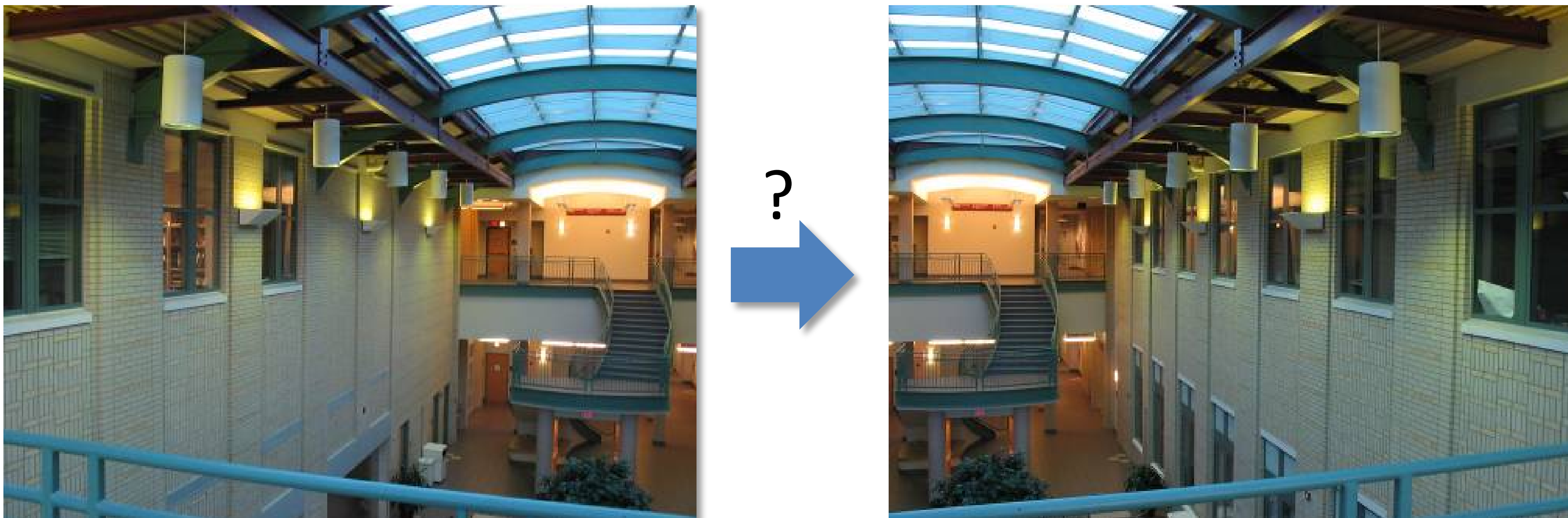


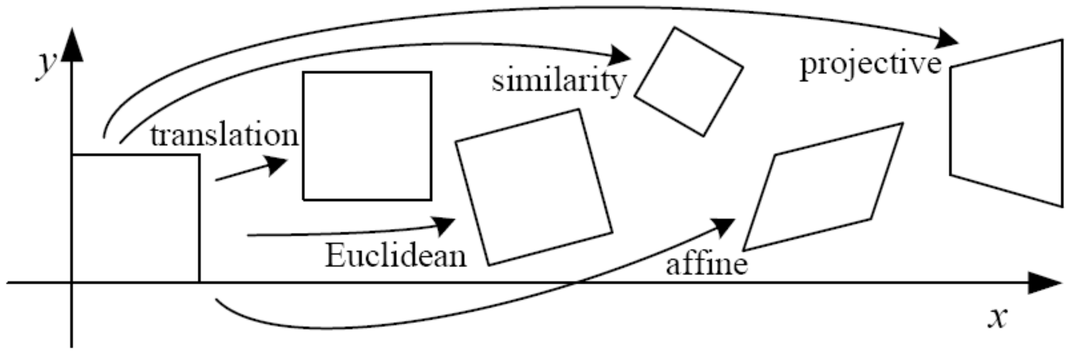
Image alignment

Why don’t these image line up exactly?

What is the geometric relationship between these two images?

Very important for creating mosaics!

2D image transformations



Name	Matrix	# D.O.F.	Preserves:	Icon
translation	$\begin{bmatrix} \boldsymbol{I} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	2	orientation $+ \cdots$	
rigid (Euclidean)	$\begin{bmatrix} \boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	3	lengths $+ \cdots$	
similarity	$\begin{bmatrix} s\boldsymbol{R} \mid \boldsymbol{t} \end{bmatrix}_{2\times 3}$	4	angles $+ \cdots$	
affine	$\begin{bmatrix} \boldsymbol{A} \end{bmatrix}_{2\times 3}$	6	parallelism $+ \cdots$	
projective	$\begin{bmatrix} \tilde{\boldsymbol{H}} \end{bmatrix}_{3\times 3}$	8	straight lines	

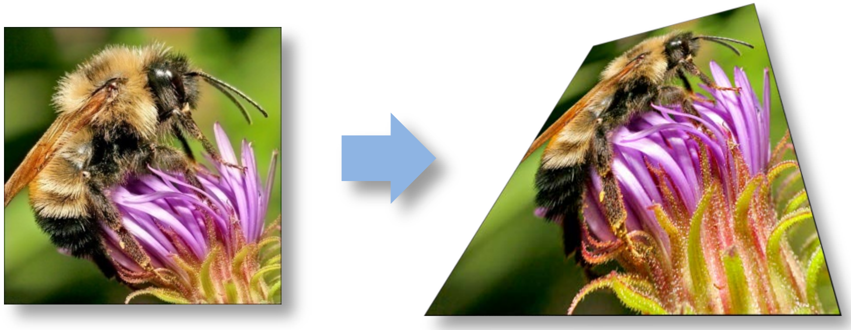
These transformations are a nested set of groups

- Closed under composition and inverse is a member

Projective Transformations / Homographies

$$\mathbf{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

Called a *homography*
(or *planar perspective map*)



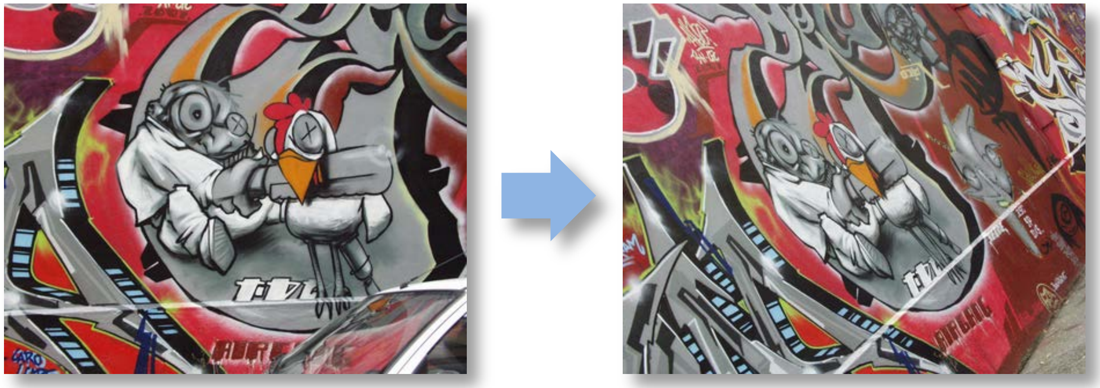
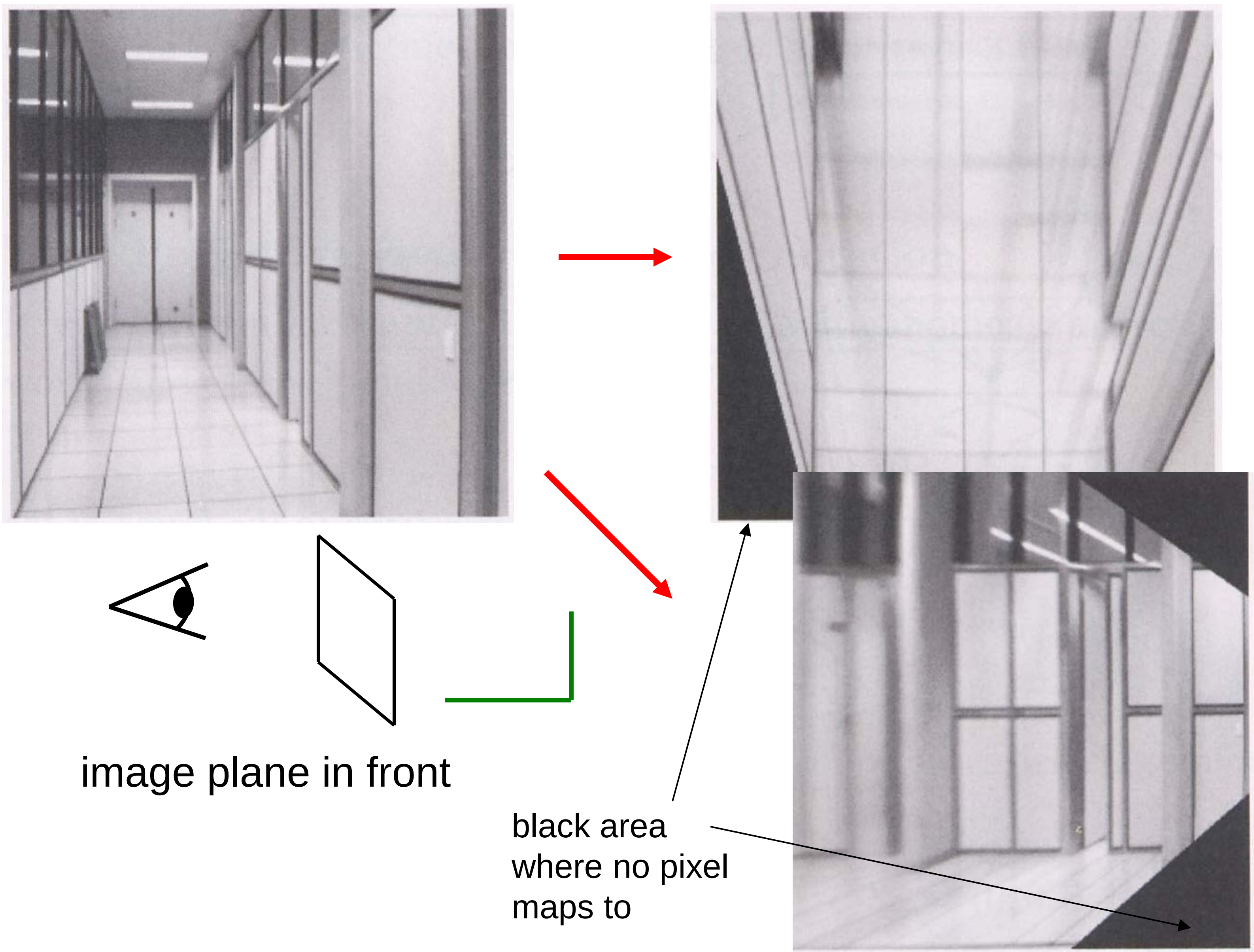


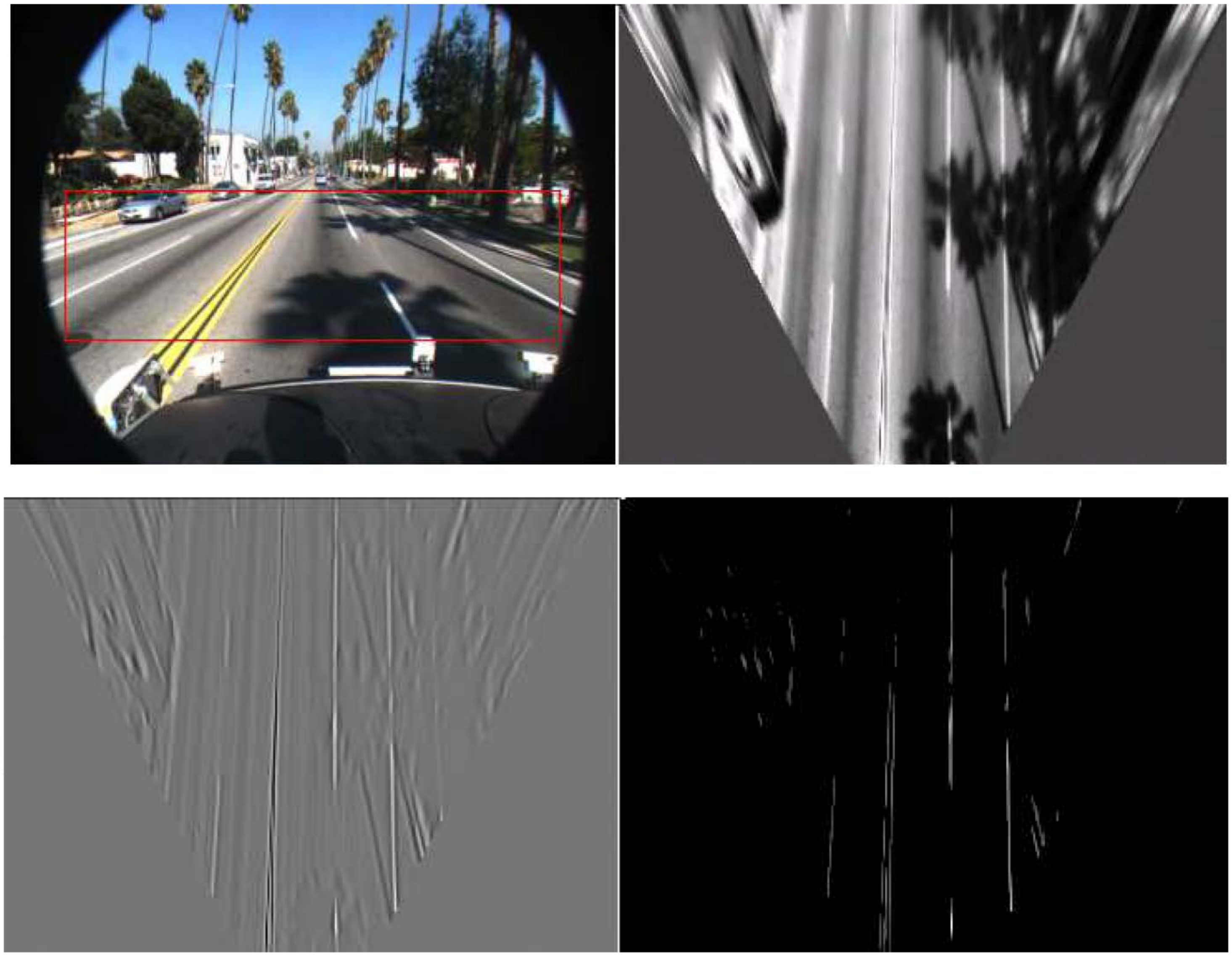
Image warping with homographies



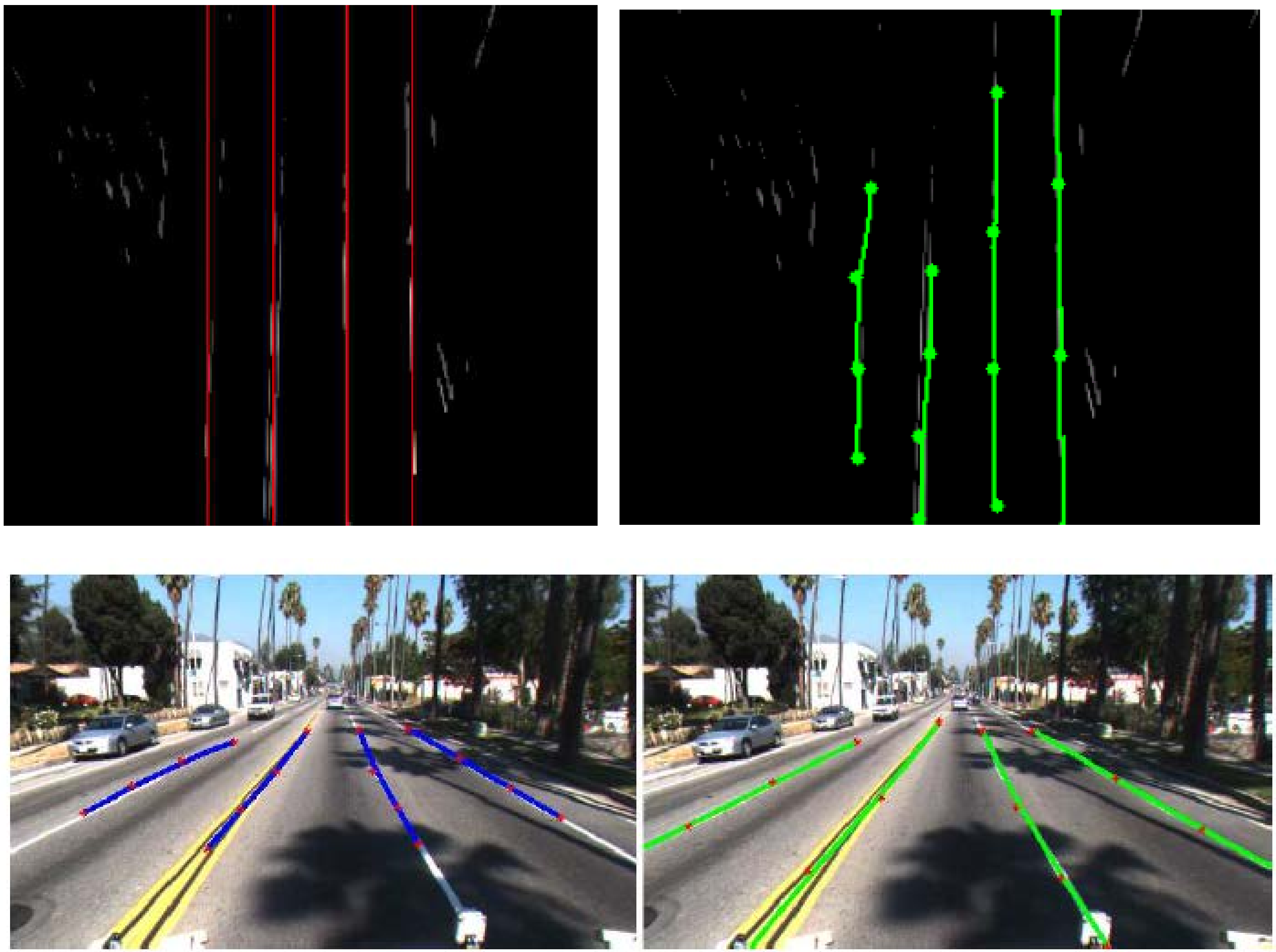
Homographies

A Quick Application: Lane Detection

Lane Detection



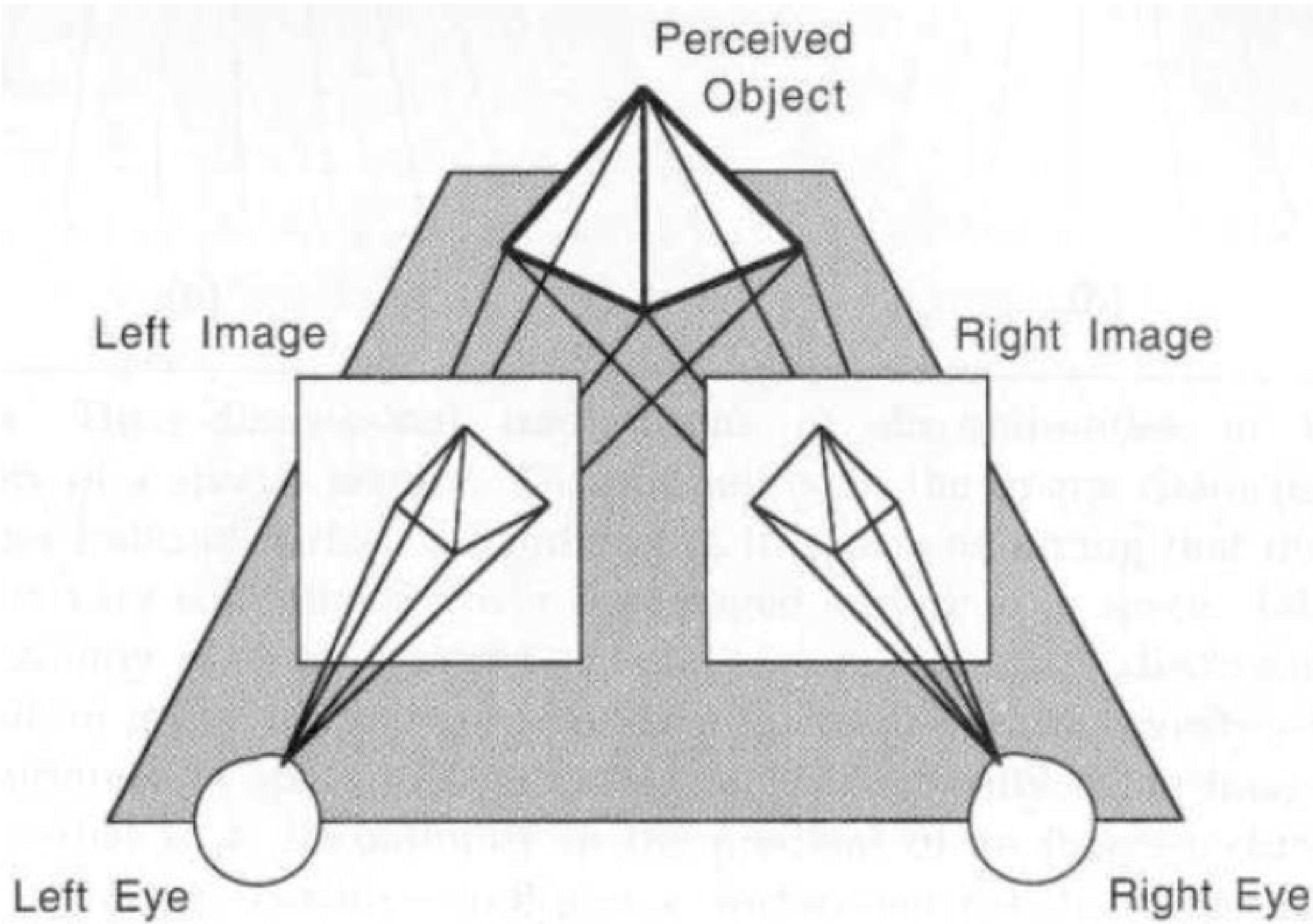
Lane Detection



Stereo Vision

Binocular Stereo Reconstruction

Recover dense 3D structure of a scene using two images from different viewpoints



Bebis

Binocular Stereo Reconstruction

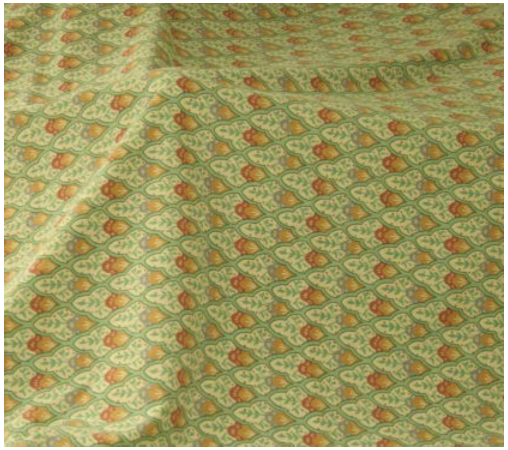
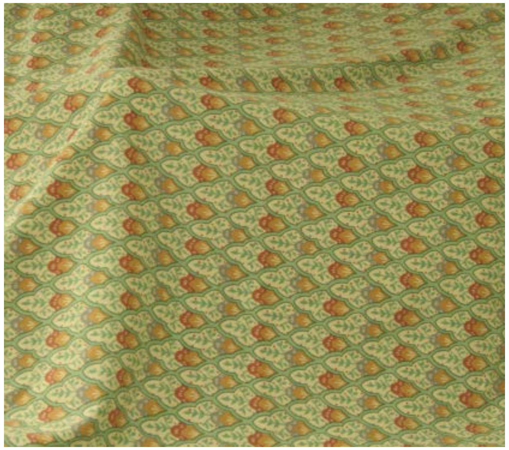
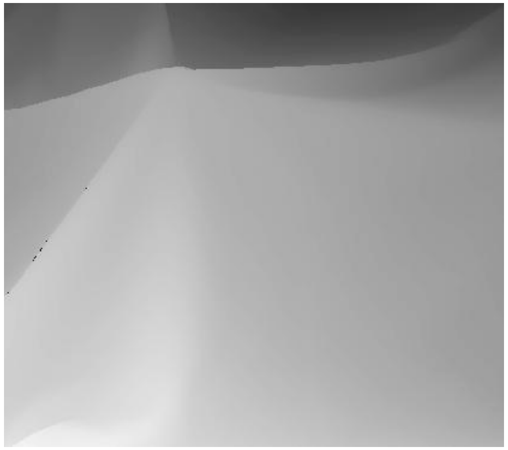
Recover dense 3D structure of a scene using two images from different viewpoints

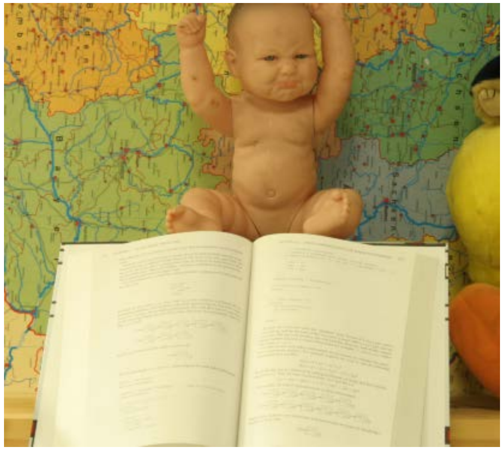
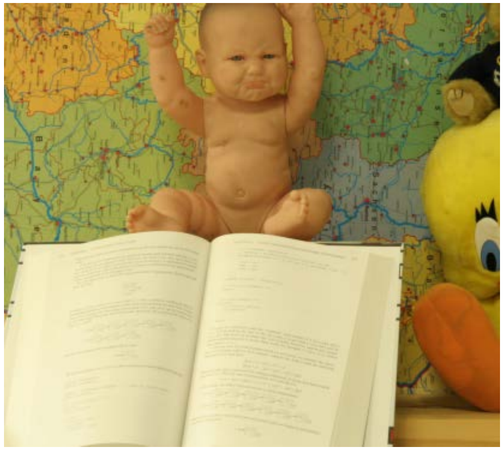
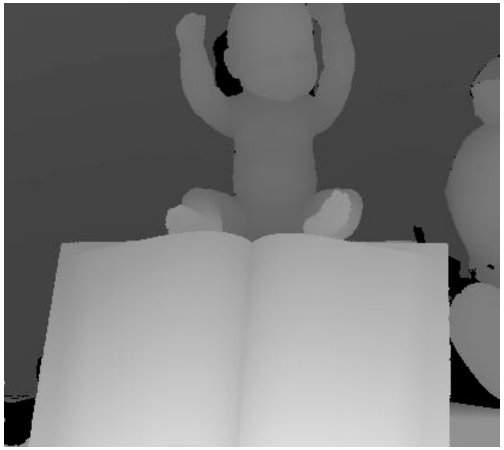
image 1

image 2

Dense depth map

A Taste of Stereo Vision

Applications

Scene modeling

Segmentation

Human-computer interaction

Autonomous driving

View interpolation

etc.

Autonomous Driving

Figure 1: Mercedes-Benz S-class vehicle with stereo camera system behind the wind shield.

Franke et al., "How Cars Learned to See"

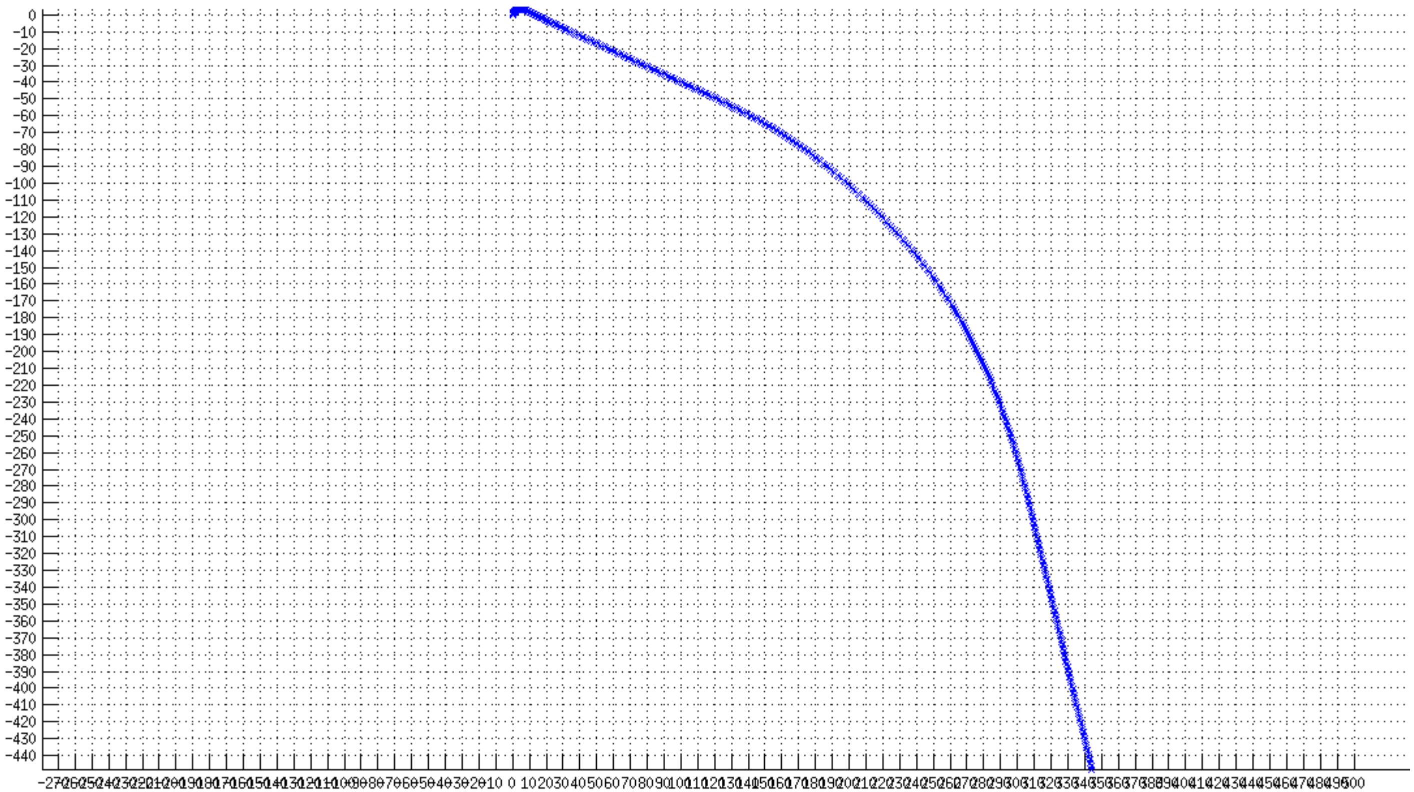
Visual Odometry, Structure-from-Motion, 3D Street Scene Reconstruction

KITTI Datasets

- Stereo images
- Grayscale
- Color
- Rectified
- 1382*512
- 10 FPS

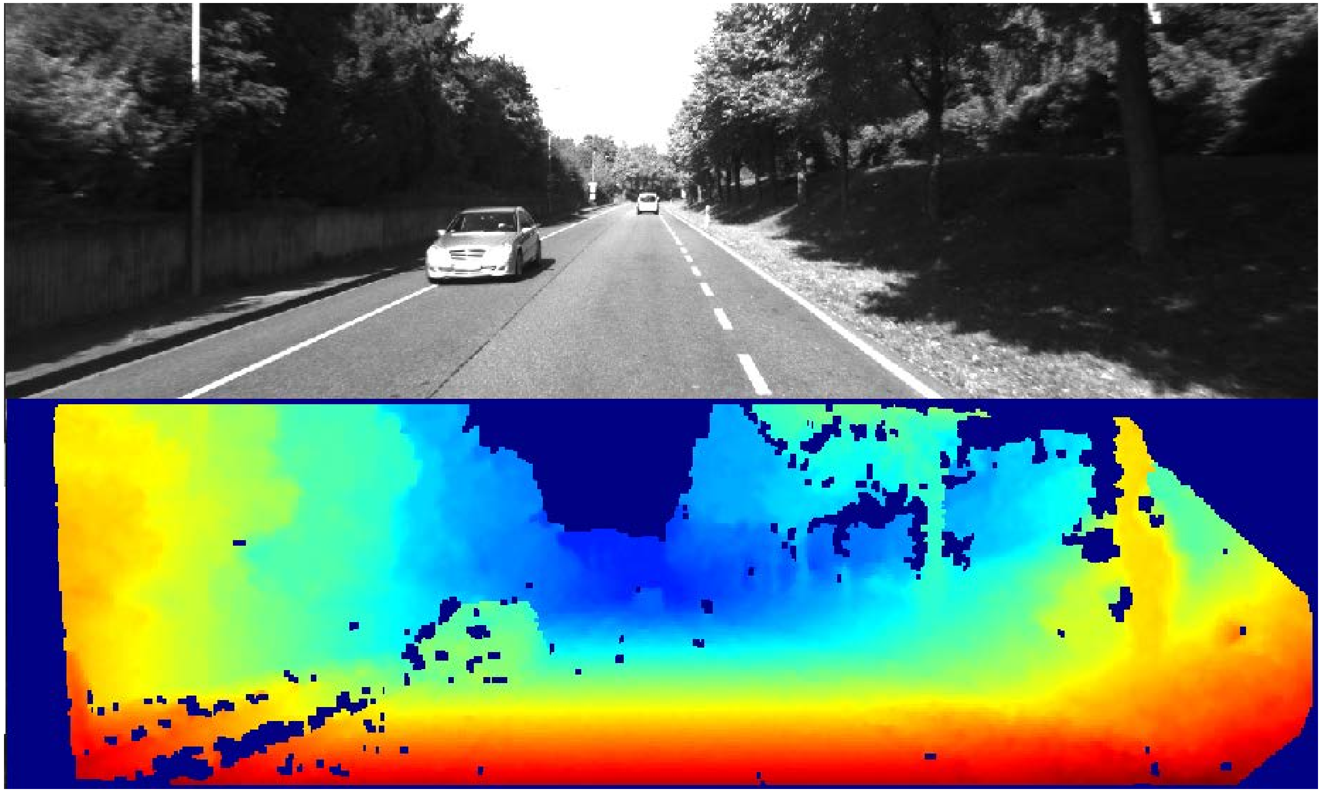
Visual Odometry

- Visual odometry computes the trajectory of the vehicle only based on image sequences (LIBVISO2)



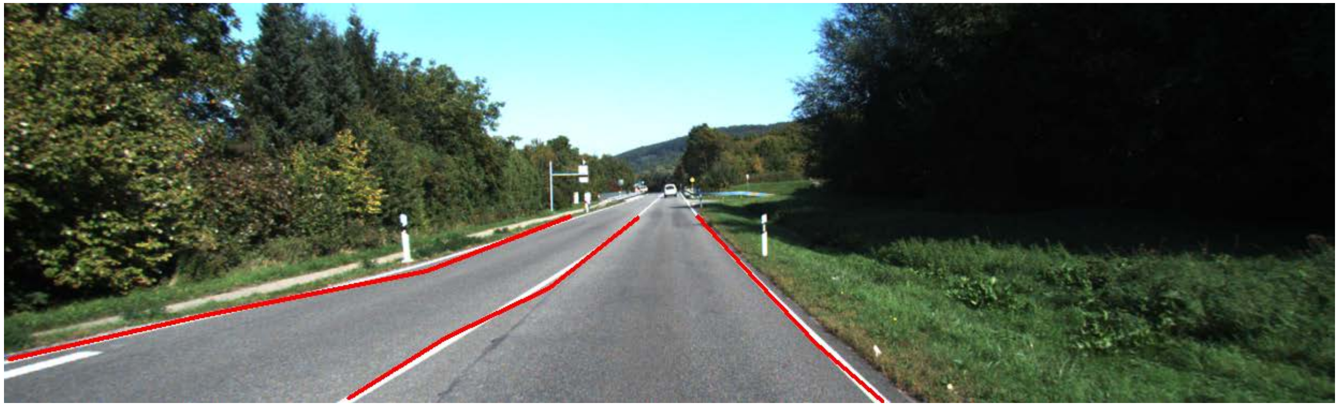
Depth Map

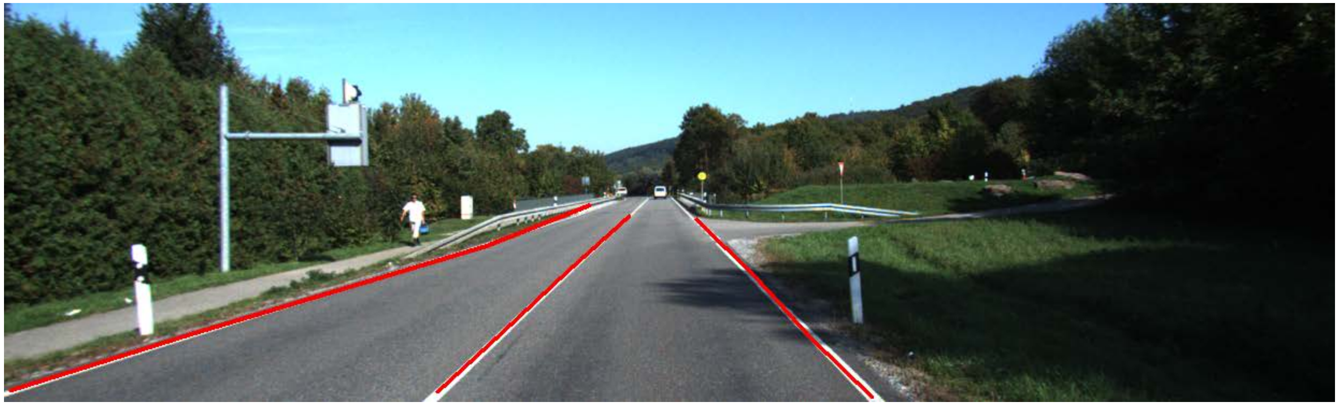
- Disparity map is computed from grayscale stereo image pairs (LIBELAS)
- Depth map can be derived from disparity map and camera model



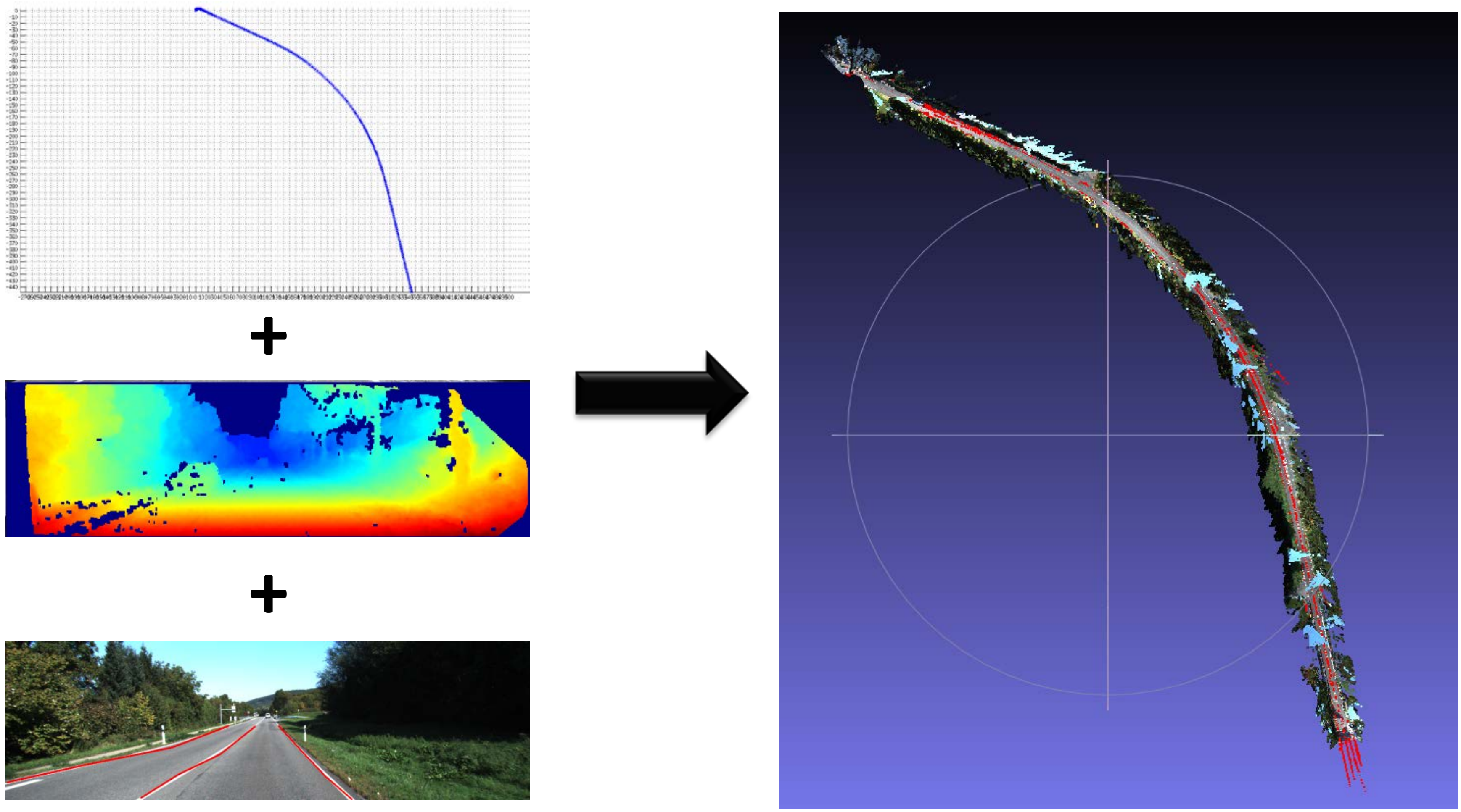
Lane Detection

- Projecting lane markers on the road (Caltech Lane Detector)



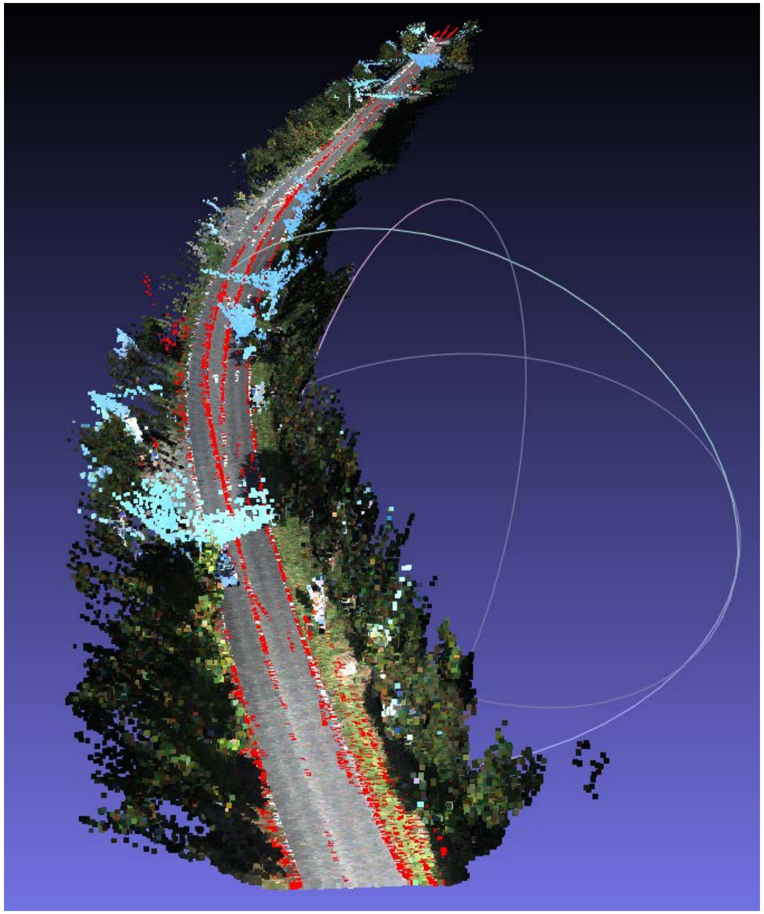


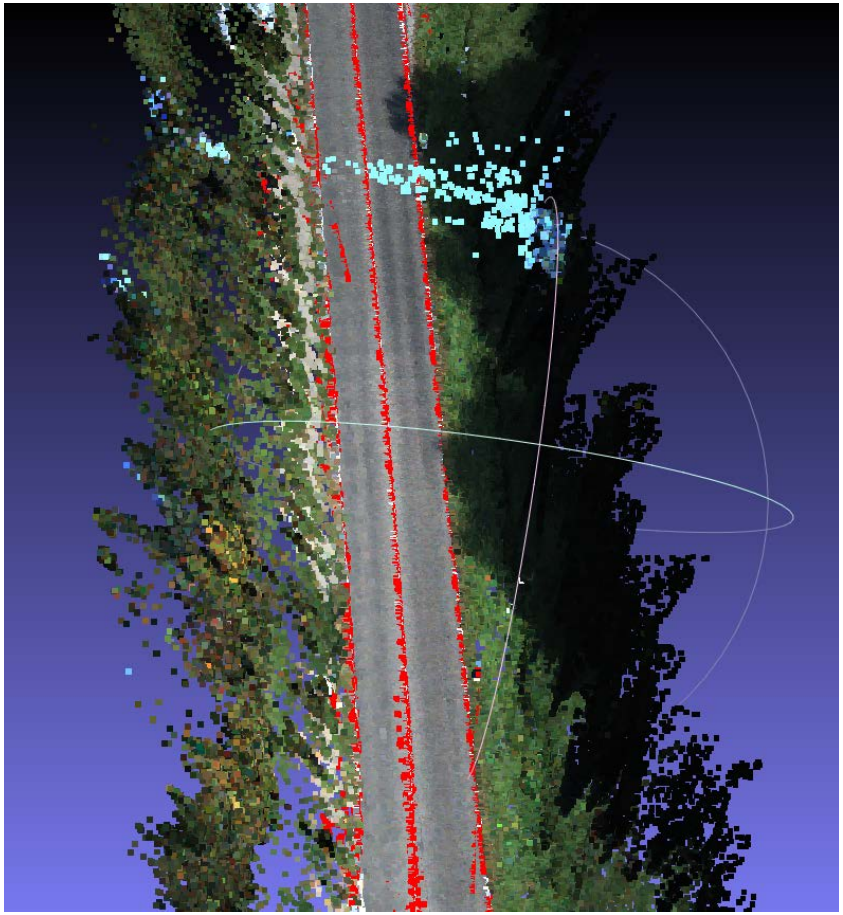
3D Street Scene Reconstruction



3D Street Scene Reconstruction

- Dense reconstruction on *run_70*





Reconstruction with Non-Stereo Images/Structure-from-Motion

- Triangulation: tracking a same point in three (or more) frames, its spatial position can be determined

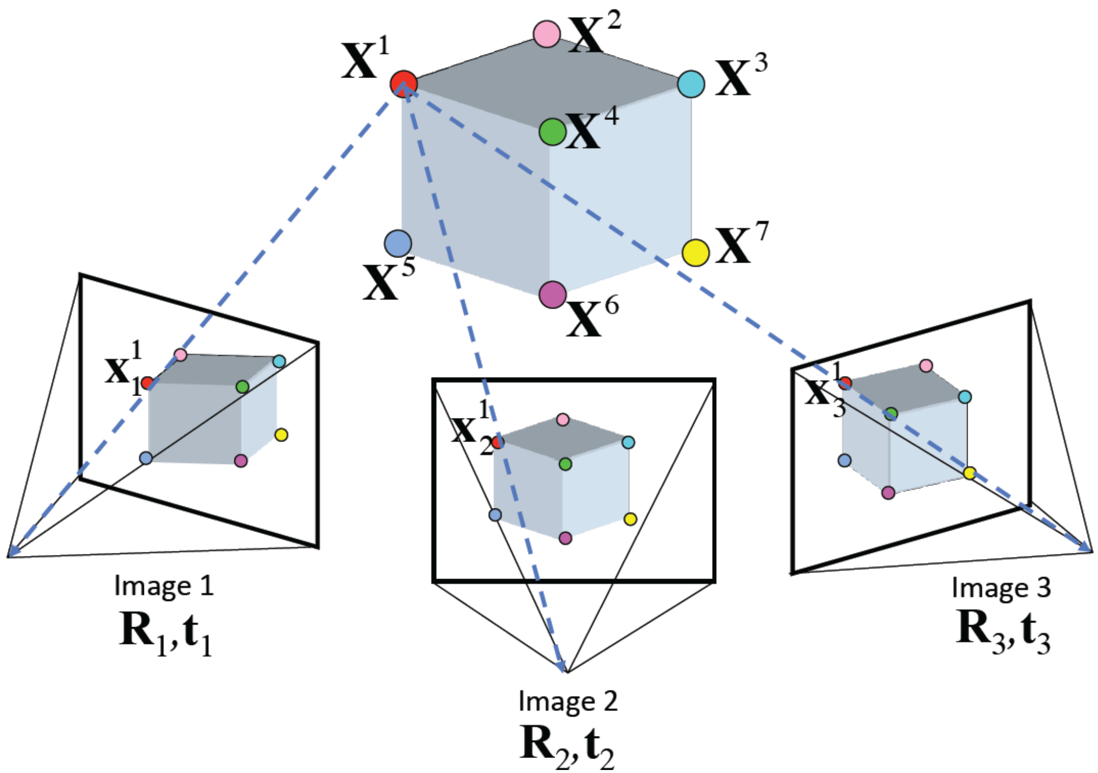
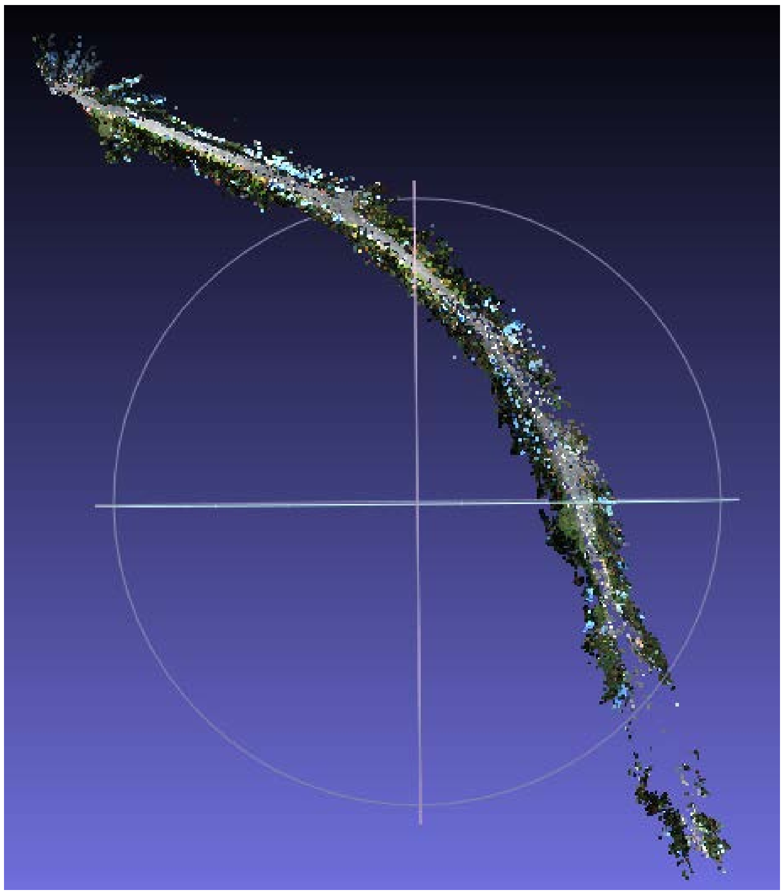
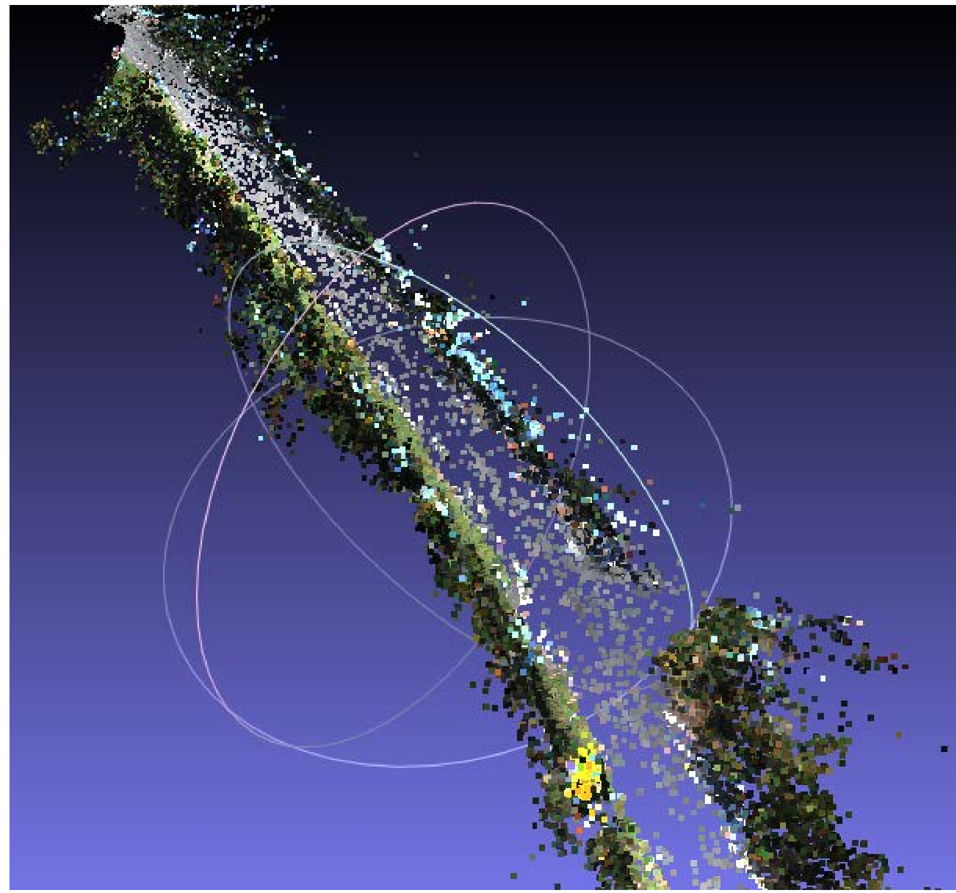


Figure courtesy of Jianxiong Xiao

Sparse Reconstruction with Non-Stereo Images

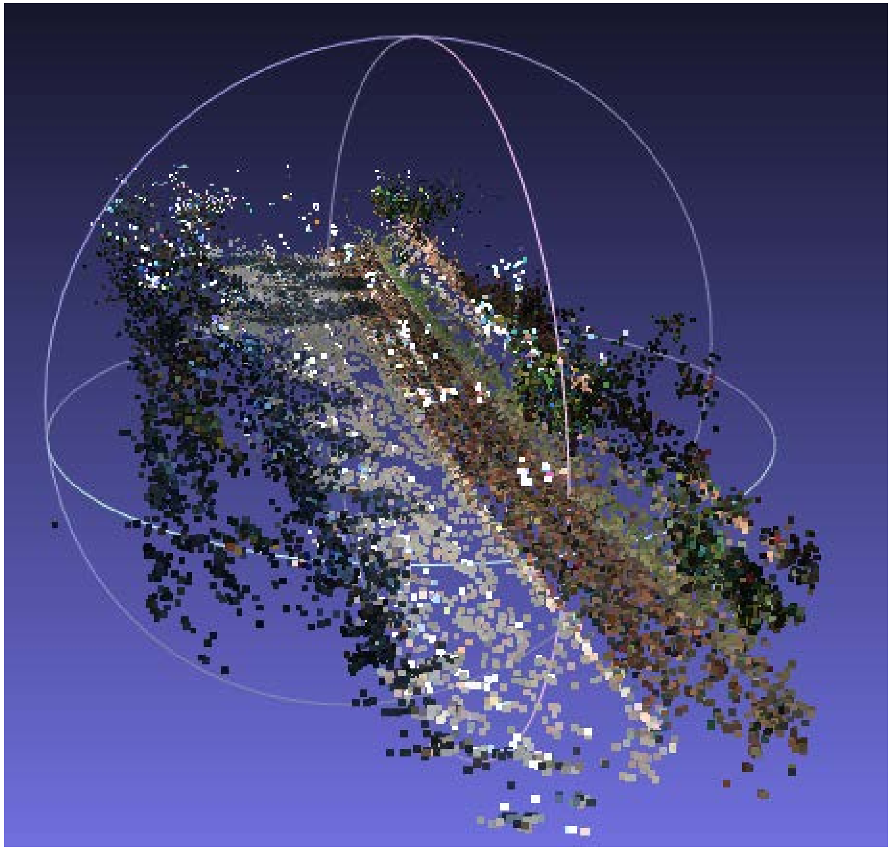
- Sparse reconstruction on *run_70*





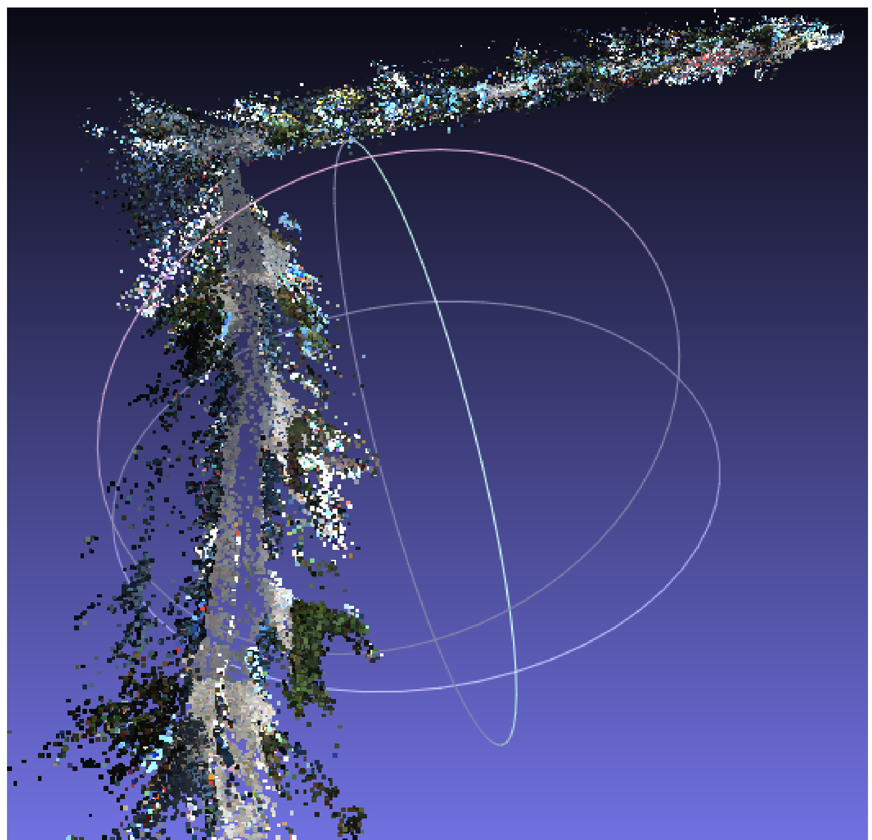
Sparse Reconstruction with Non-Stereo Images

- *run_1*



Sparse Reconstruction with Non-Stereo Images

- *run_9*



Other Demos for Structure-from-Motion

- https://www.youtube.com/watch?v=i7ierVkXYa8
- https://www.youtube.com/watch?v=vpTEobpYoTg

Other Demos for Structure-from-Motion

Other Demos for Structure-from-Motion

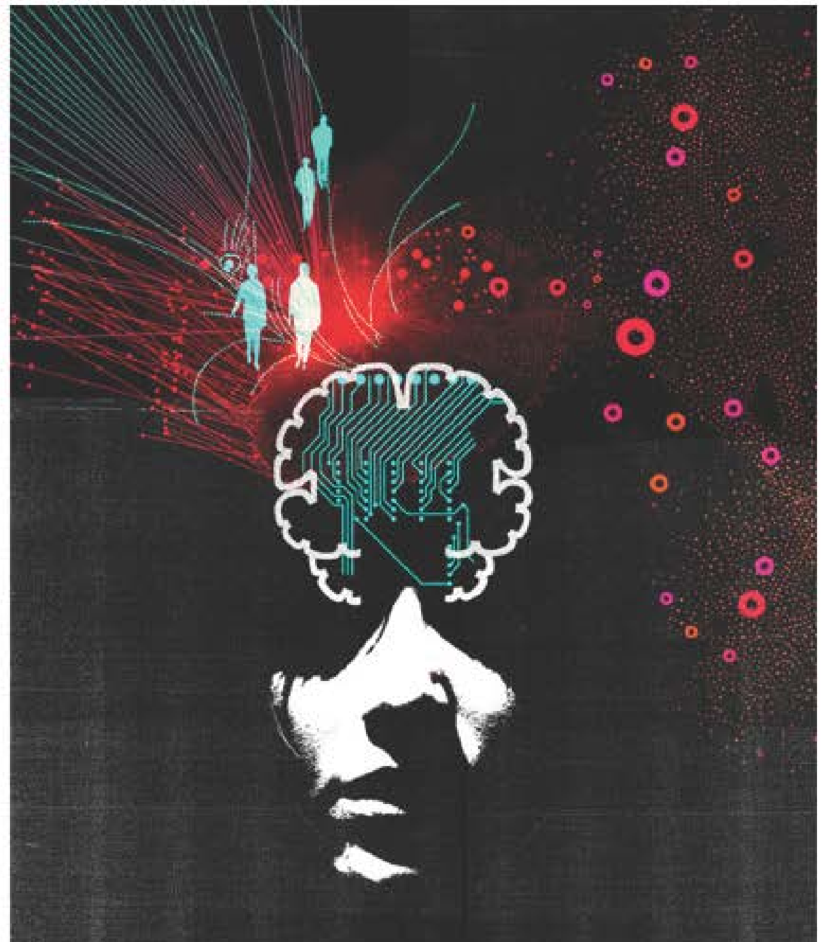
Deep Learning

MIT Technology Review

10 BREAKTHROUGH TECHNOLOGIES 2013

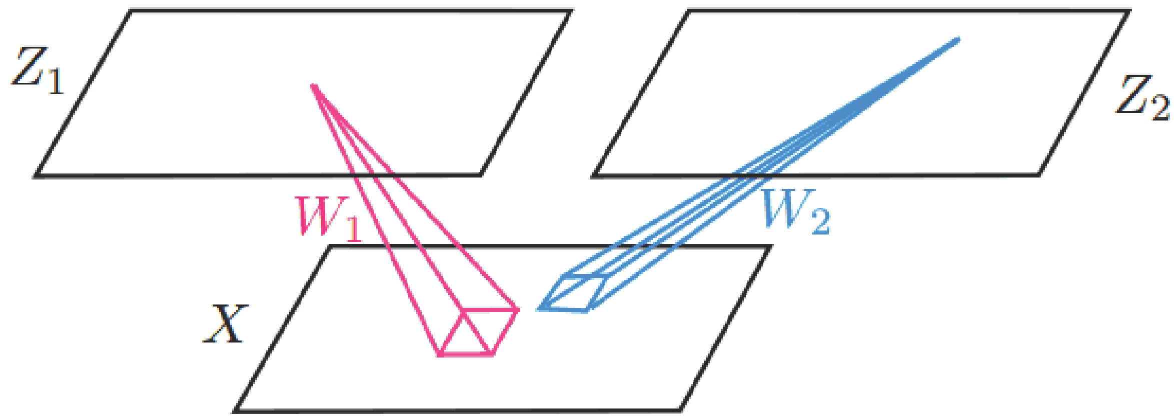
Deep Learning

With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart.



Convolutional Neural Networks

detection layers



convolution

$$Z_i = \sigma(W_i * X)$$

Convolutional Neural Networks

pooling :

- reduce the size of representations
- allow small translation invariance.

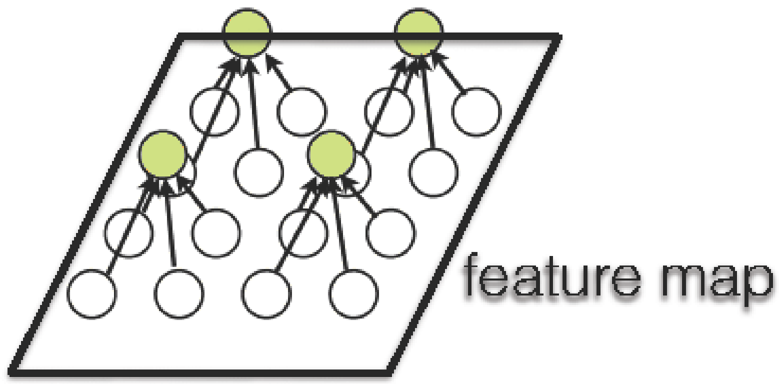
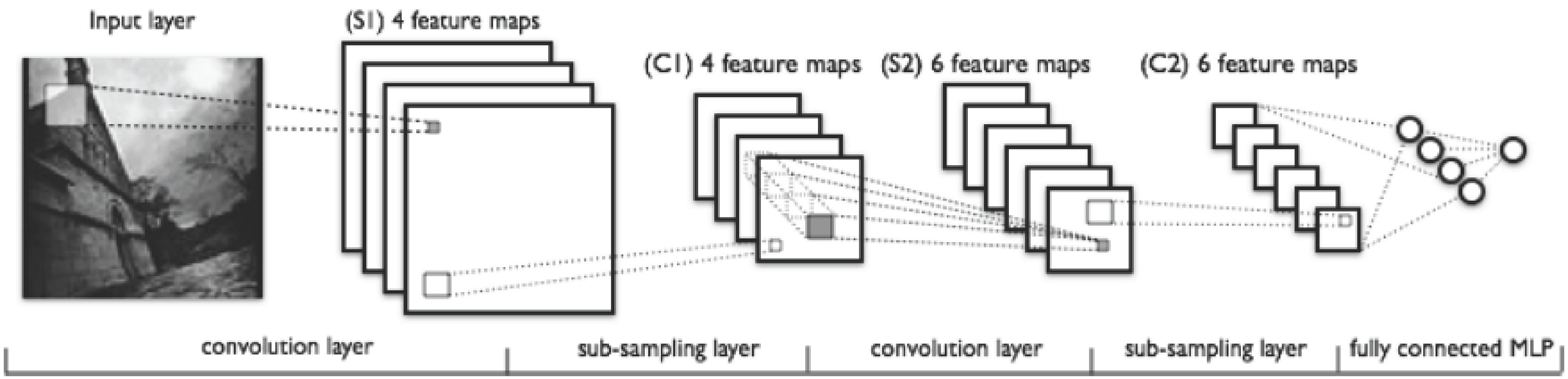


figure from roger grosse tutorial

common pooling techniques:
max pooling, average pooling

Convolutional Neural Networks

convolution network is just a combination of convolution layers, pooling layers and fully connected layers

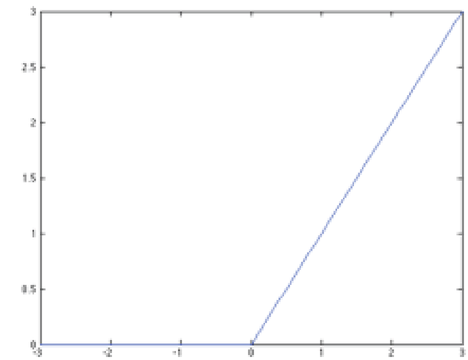


(figure from LeNet tutorial, http://deeplearning.net/tutorial/lenet.html)

Convolutional Neural Networks

details:

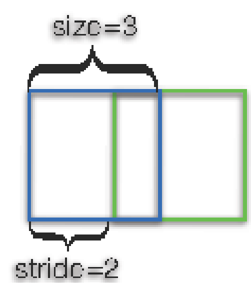
activation function : ReLU $f(z) = max(0, z)$



local normal contrast: $$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=\max(0, i-n/2)}^{\min(N-1, i+n/2)} (a^j_{x,y})^2 \right)^\beta$$

normalize at the same spatial location
over different feature maps

overlapping pooling: pooling size 3 with stride 2



Convolutional Neural Networks

reduce over-fitting:

training 1.2 million images for 60 million parameters

data augmentation:

- extracting random 224*224 patches from the 256*256 images and horizontal refections. This results in an increase by a factor of 2048.

- altering the intensities of the RGB channels, object identity is invariant to the illumination. Add each RGB pixel in each training image by

$$[\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3][\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]^T$$

where p_i are the 3 - 1 eigenvectors and λ_i are eigen values of 3 - 3 covariance matrix of RGB pixels, α_i are random numbers

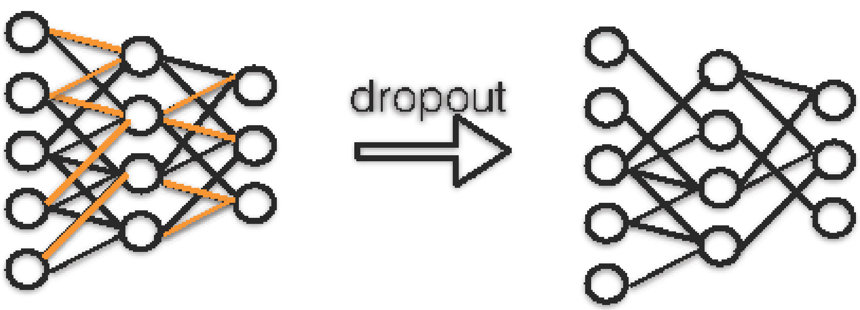
Convolutional Neural Networks

reduce over-fitting:

dropout:

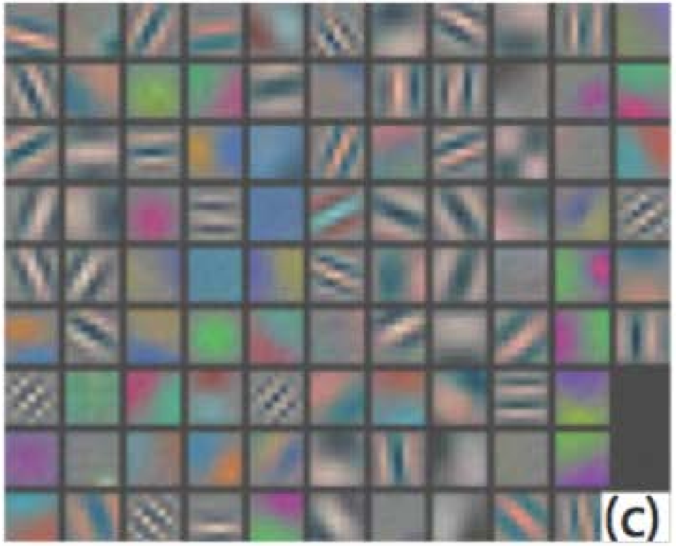
an efficient way to combining different model predictions.

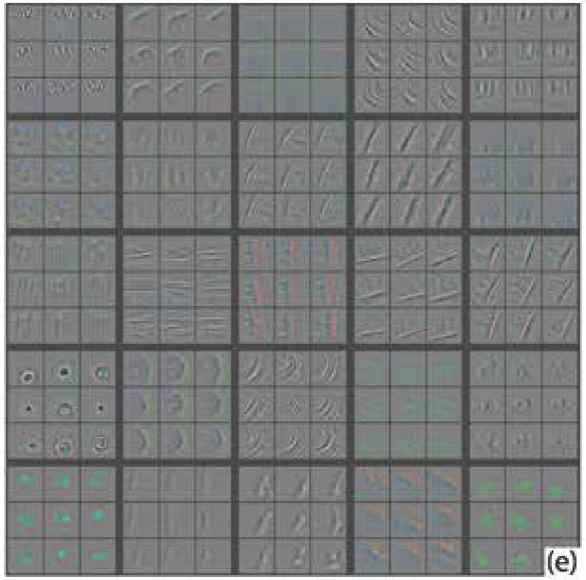
- for each forward pass and each back propagation, randomly defunctions every neuron with probability 0.5. (setting the activation to zero)
- testing: half all the weights, and use all of the neurons to predict



Convolutional Neural Networks

filter visualization:





filters learned

(Matthew D. Zeiler et al 2013)

The end of all the fundamentals