# Deep Learning For 3d Object Detection

*Brown Bag Session*

Hochschule Konstanz
University of Applied Sciences

INSTITUTE FOR OPTICAL SYSTEMS

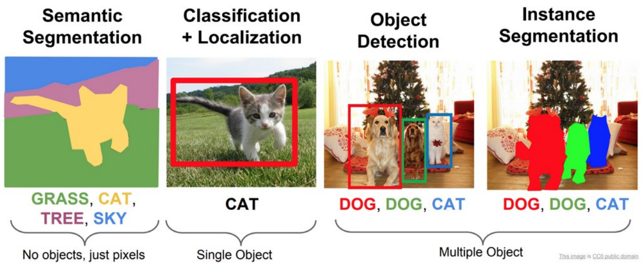

University of Applied Sciences Konstanz
Institute for Optical Systems
14.12.2020
Dennis Grießer

# Overview

- Motivation
- Depth reconstruction with stereo
- Depth reconstruction with multiple views
- 3d detection with multiple views

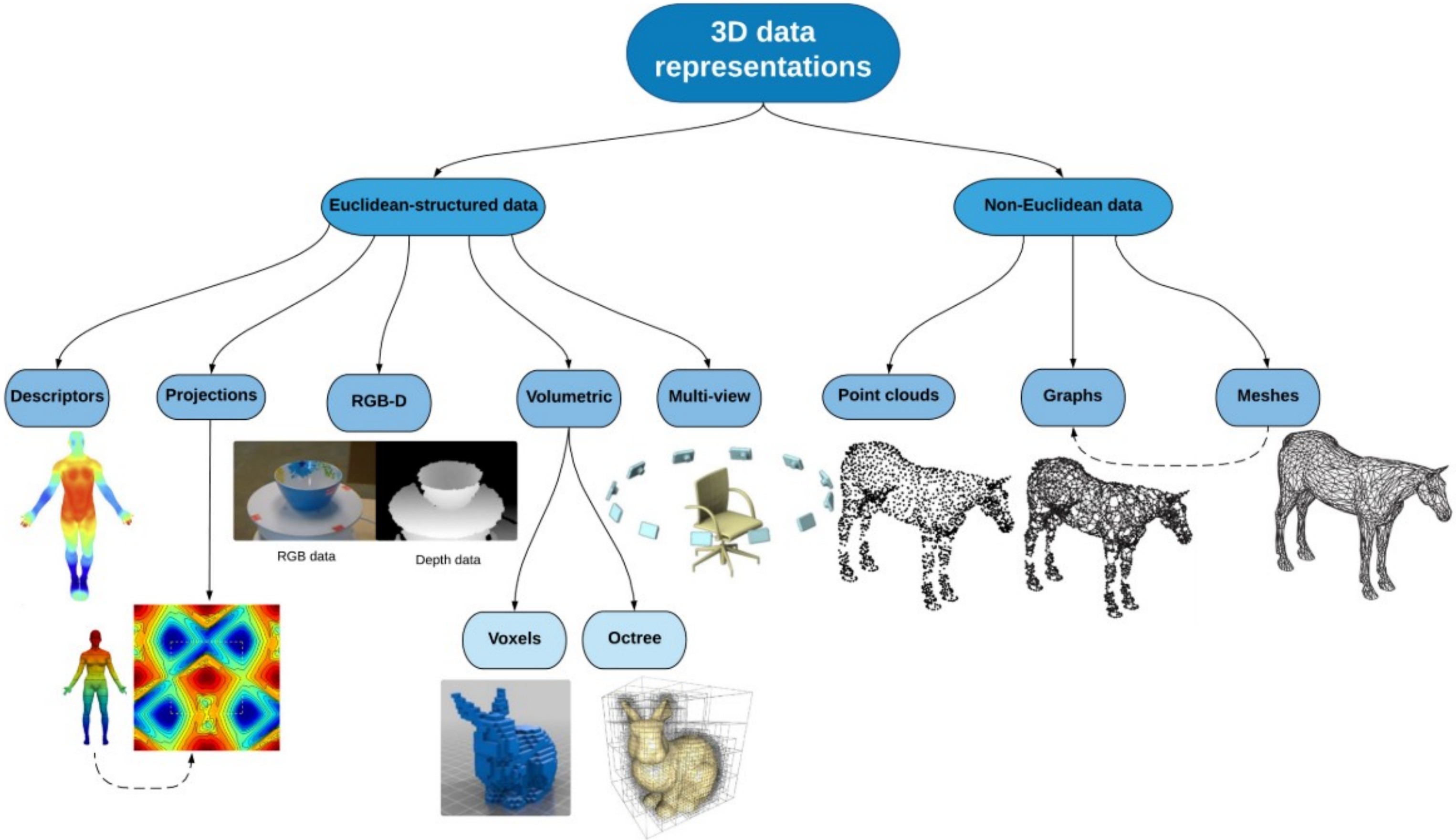
# Why 3d Deep Learning?

- Deep Learning on 2d data achieve impressive results in many tasks
  - Classification
  - Segmentation
  - Detection




  - ! Large amount of data is required
- Increased availability of affordable 3d data aquisation devices

# Which representation?



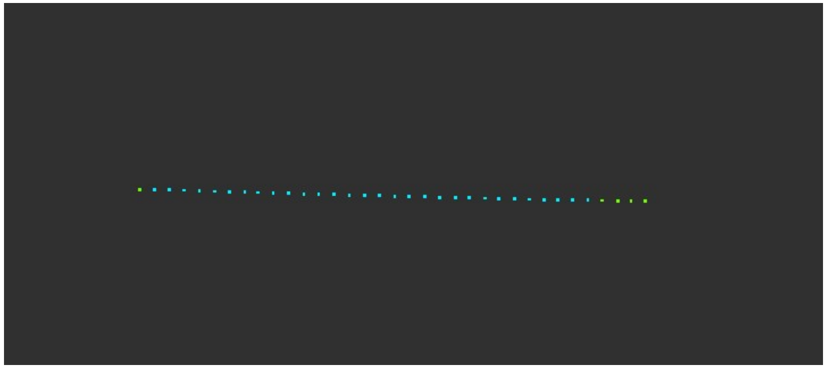

Ahmed et al., A survey on Deep Learning Advances on Different 3D DataRepresentations (2019)

# Why Multi-view?

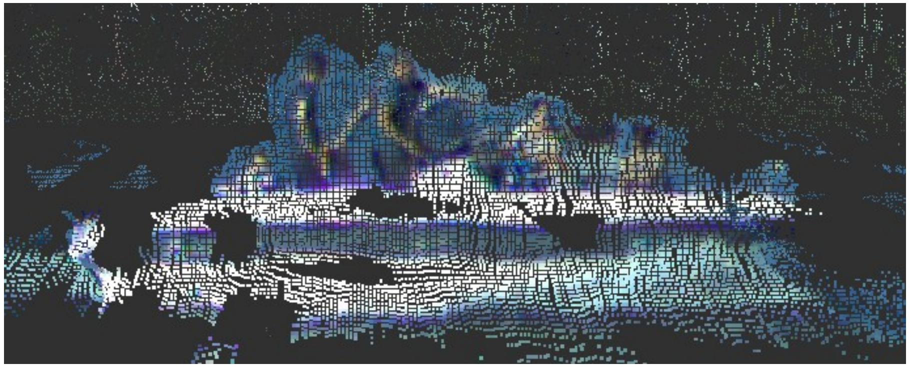

~16.000 €

~225 €

Elon Musk: “Anyone relying on lidar is doomed.” Experts: Maybe not



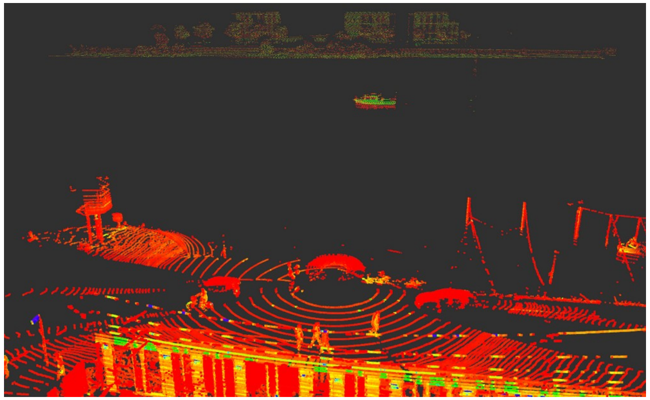
Lidar measurement



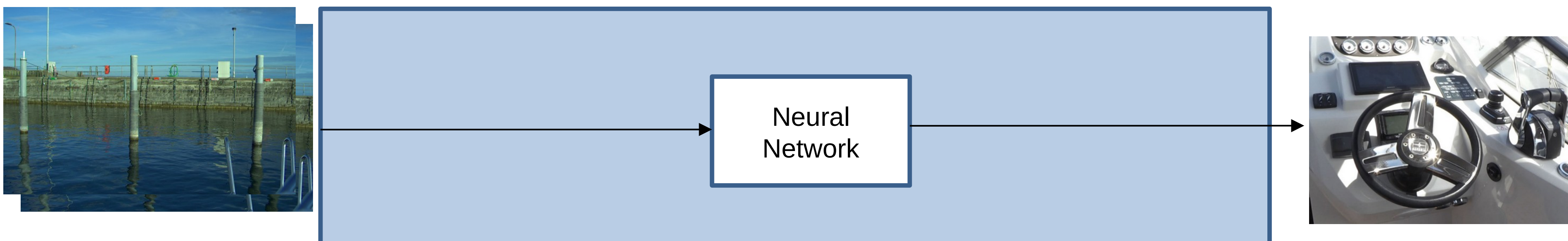
Stereo measurement

# MultiSenseLakePerceptor

What does the system see?

# End-to-End approach



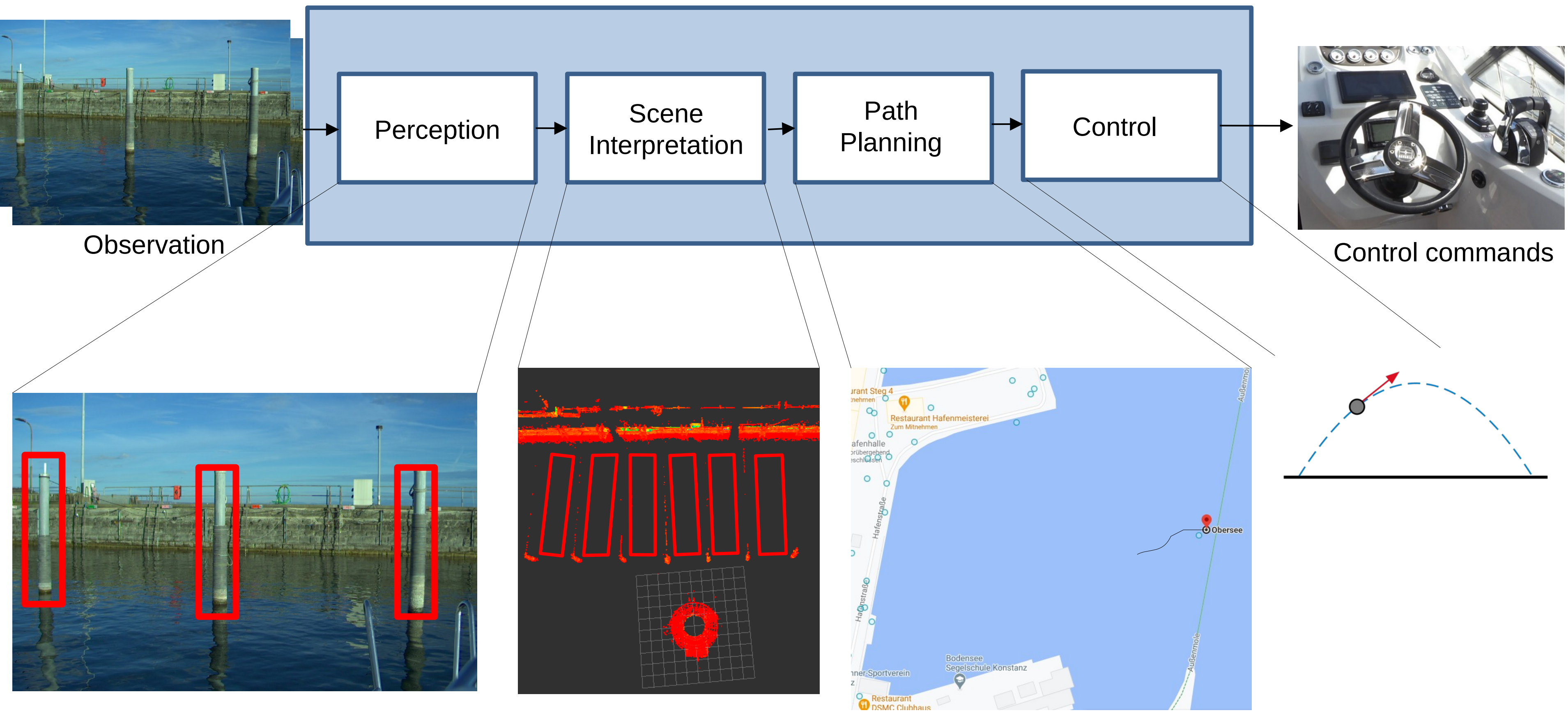

Observation

Control commands

- Simple Model
- End-to-End Training

- Interpretability
- Generalization

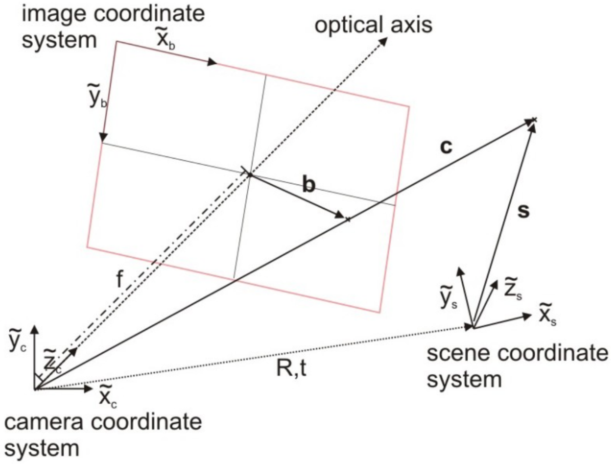
# Modular Approach



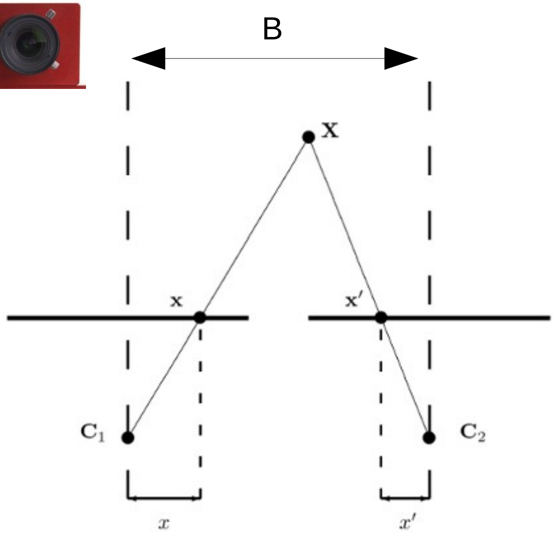
# Central projection

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \underbrace{\begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}}_{K} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

K R t



# Depth reconstruction


left image:

$$X = -z\frac{x_1}{f}$$

$$Y = -z\frac{y_1}{f}$$

right image:

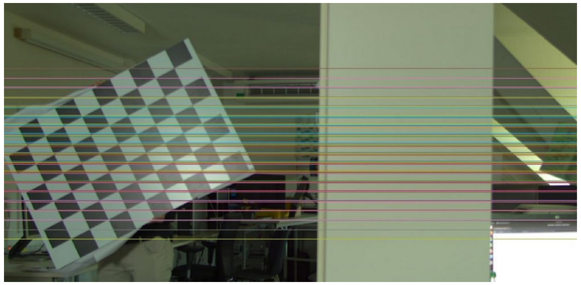
$$X = B - z\frac{x_2}{f}$$

$$Y = -z\frac{y_2}{f}$$

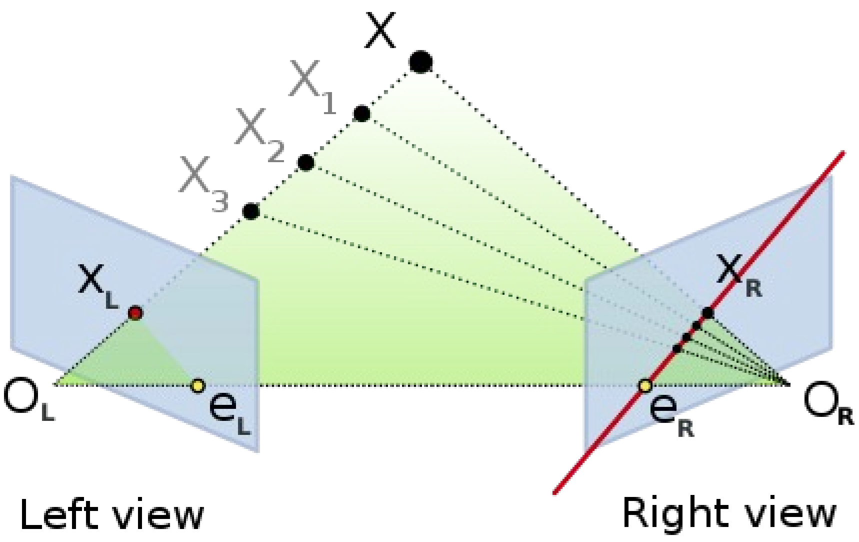
calculate Z:

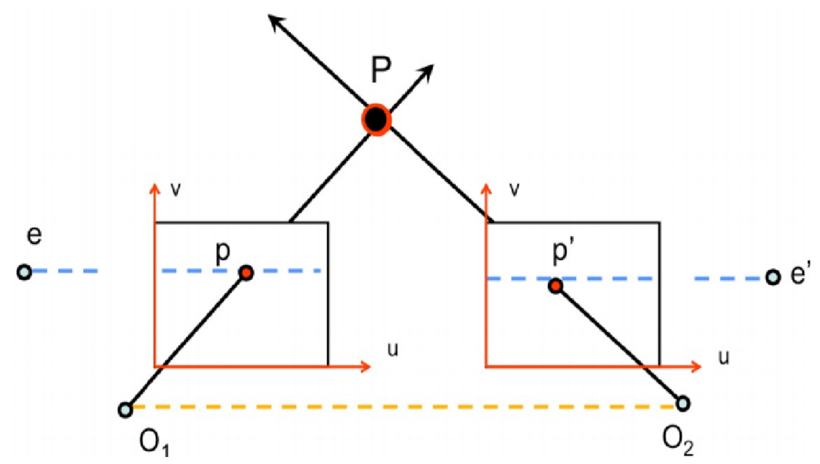
$$z\frac{x_1}{f} = B - z\frac{x_2}{f}$$

$$z = \frac{-f\,B}{x_1 - x_2}$$



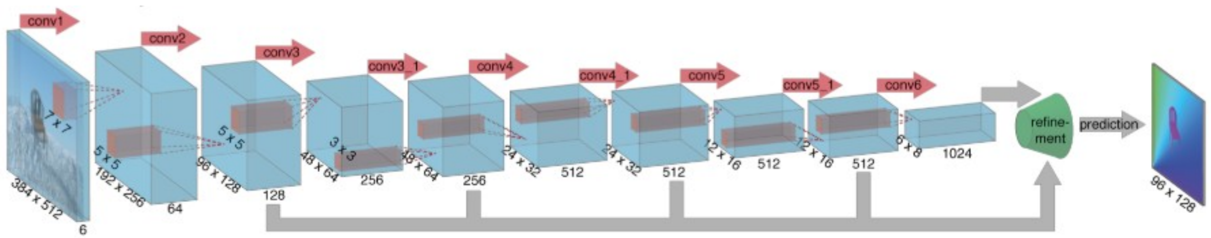
# Rectification



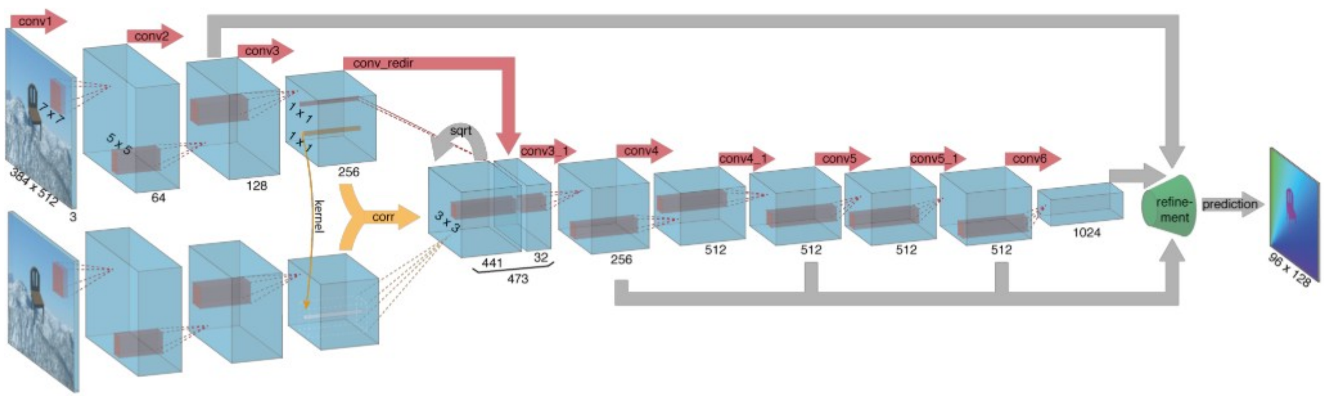



# DispNet

## DispNetSimple




## DispNetCorrelation

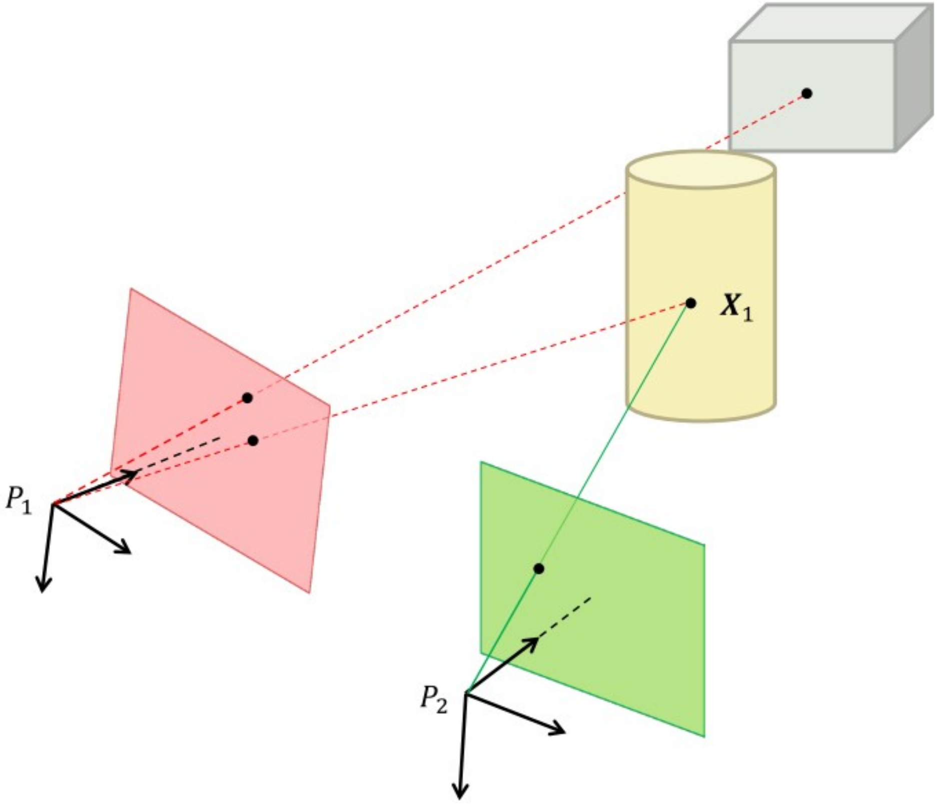



Dosovitskiy et al., FlowNet: Learning Optical Flow with Convolutional Networks (2015)

Mayer et al., A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation (2016

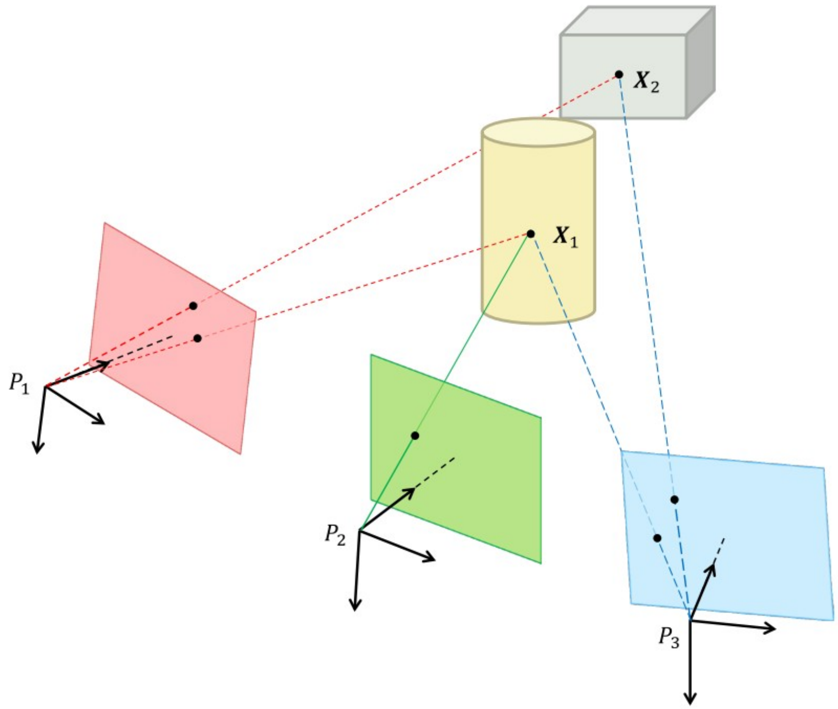
# Multiple-view stereo

- It is not always possible to find correct correspondences
  - E.g. due to occlusion



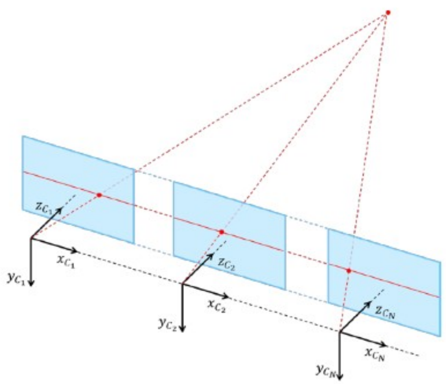
# Multiple-view stereo

- Therefore, add more views
  - Can be used to verify correspondences
  - Can make reconstruction more robust to occlusion

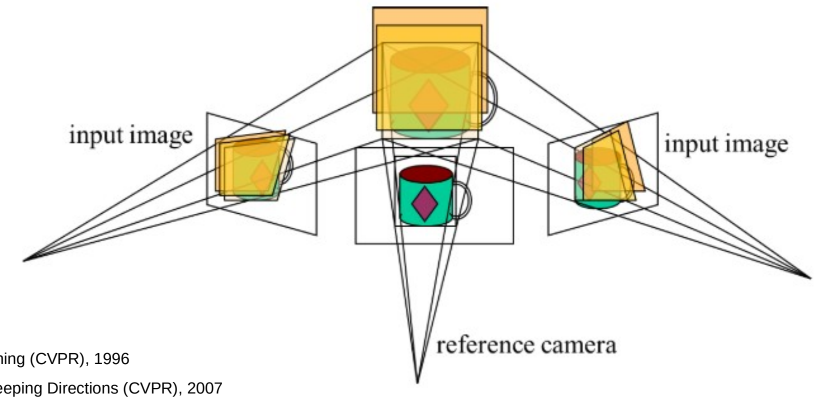


# Multiple-view stereo

1. Rectification of several cameras to a common plane
   – But rectification is complex for more views and large baselines



2. Plane sweep stereo
   – Select a reference view
   – Sweep some planes at different depths with respect to the reference camera




Collins, R.T. A space-sweep approach to true multi-image matching (CVPR), 1996
Gallup, D., Real-Time Plane-Sweeping Stereo with Multiple Sweeping Directions (CVPR), 2007

# Plane sweep

- Properties:
  - Algorithm works with with any number of cameras
  - Rectification is not needed
- Define a family of depth planes:

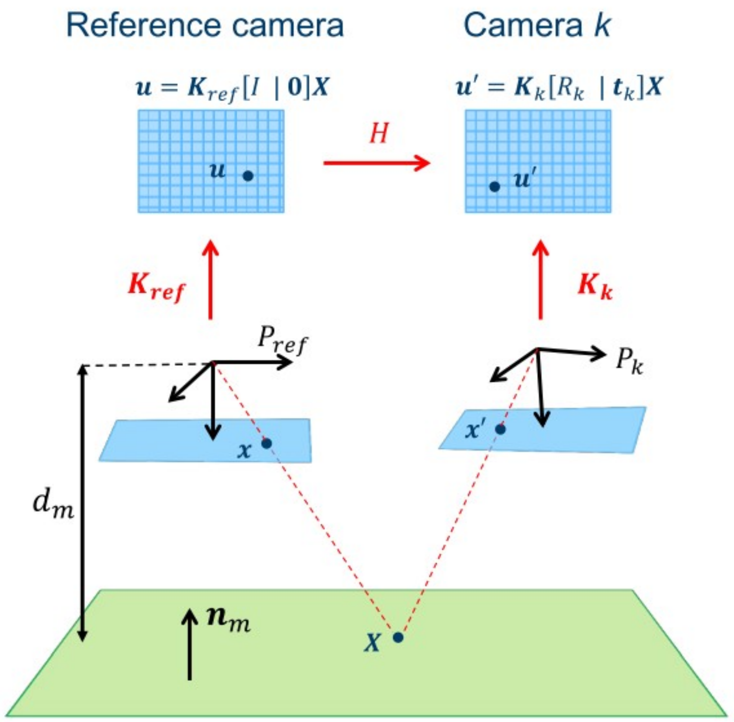
$$\prod_m = [n_m^T \; -d_m]$$

- Mapping from reference camera to camera k

$$H_{\prod_m, P_k} = K_k \left( R_k + \frac{t_k n_m}{d_m} \right) K_{ref}^{-1}$$

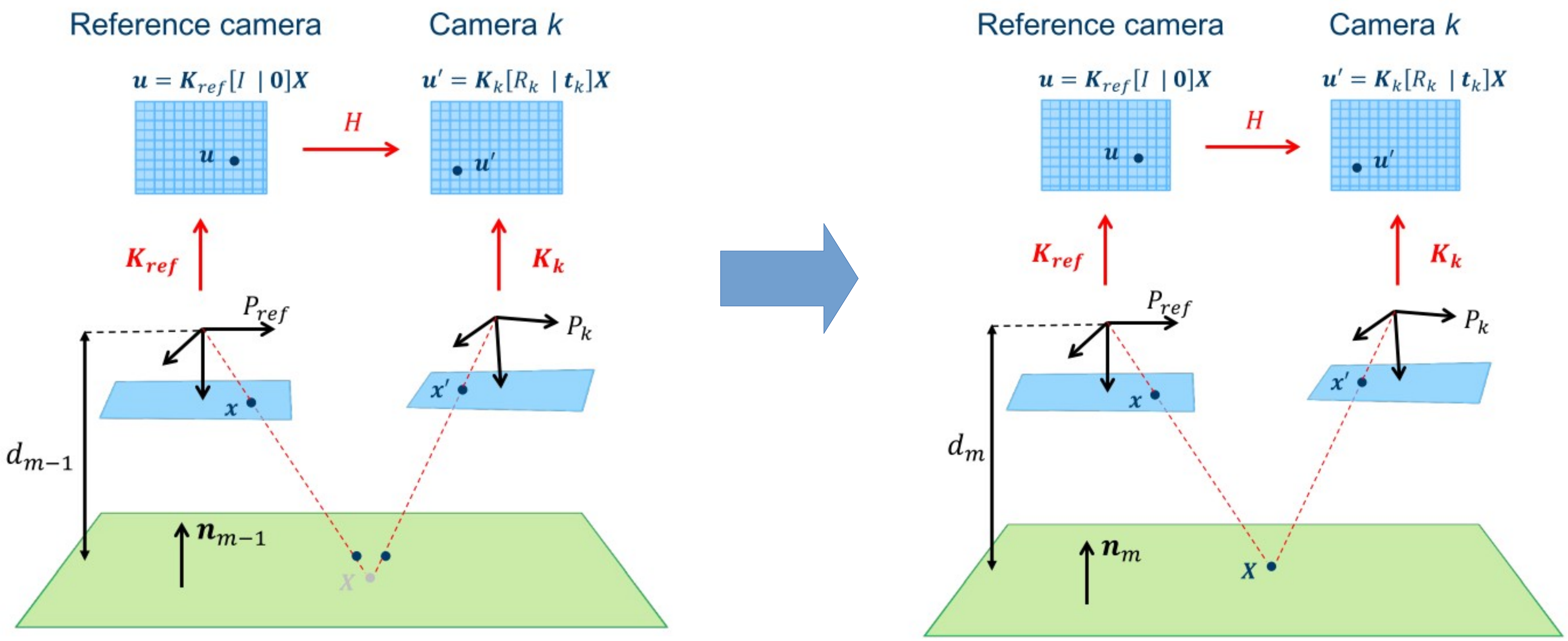
$$[x' \; y' \; w']^T = H_{\prod_m, P_k} [x \; y \; 1]^T$$

$$x_k = \frac{x'}{w'} \qquad y_k = \frac{y'}{w'}$$



# Plane sweep

- Sweep planes at different depths



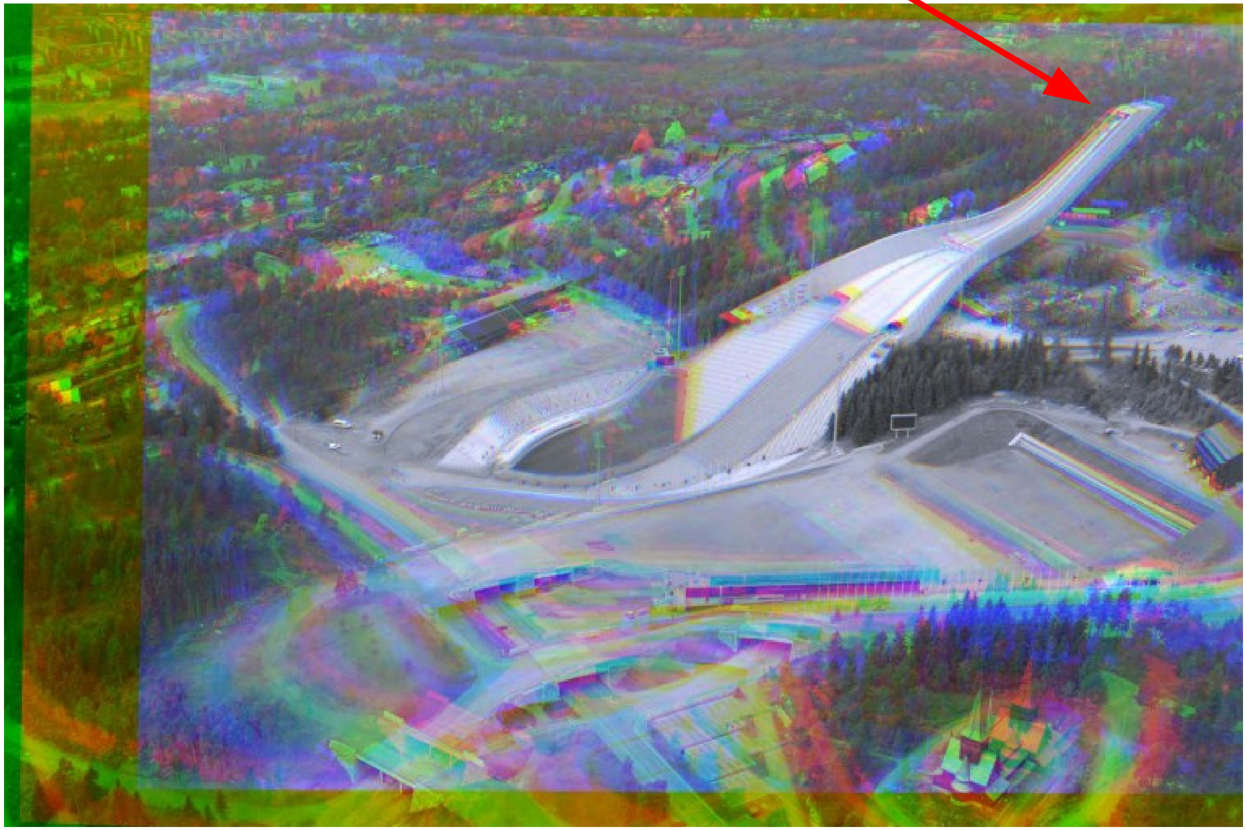
# Plane sweep for multiple views

Red:

Green:

Blue:

790 meter



sweep plane = 706 meter below reference camera

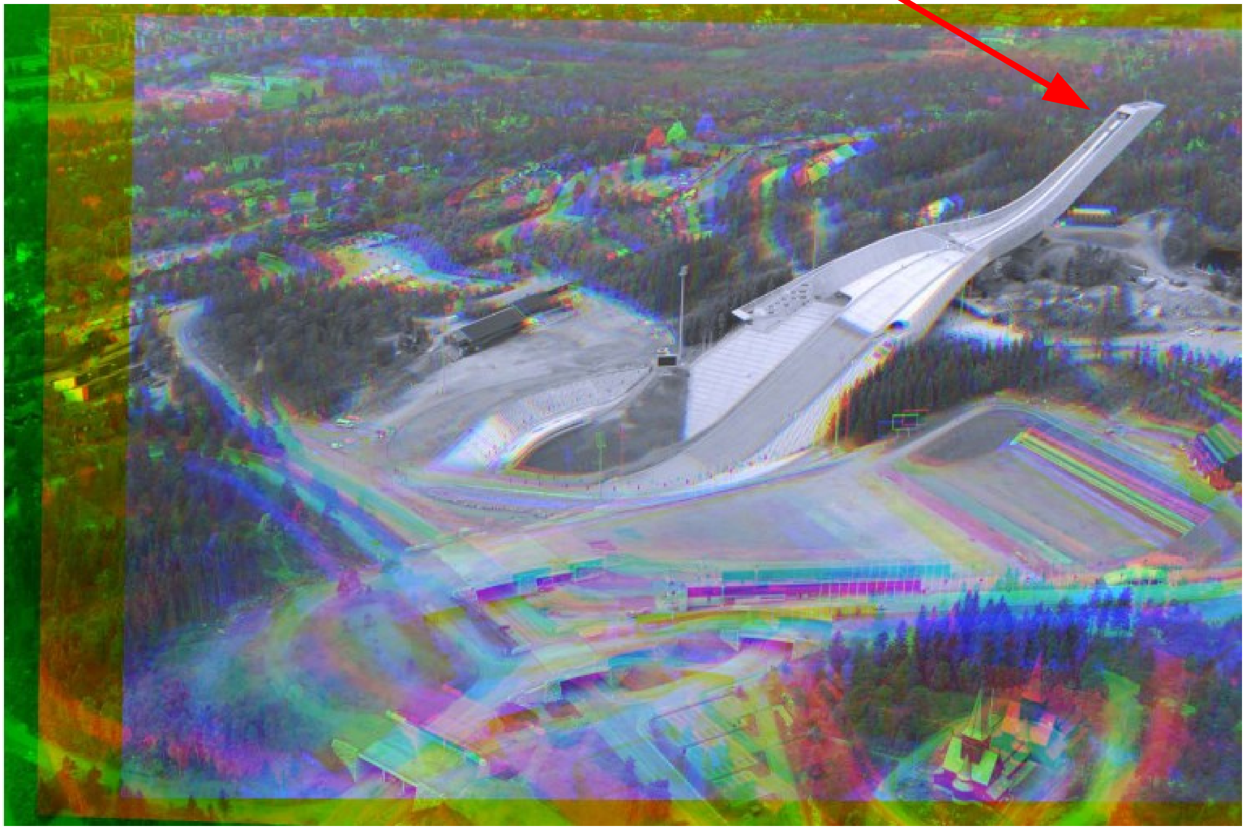
# Plane sweep for multiple views

Red:

Green:

Blue:

790 meter



sweep plane = 790 meter below reference camera

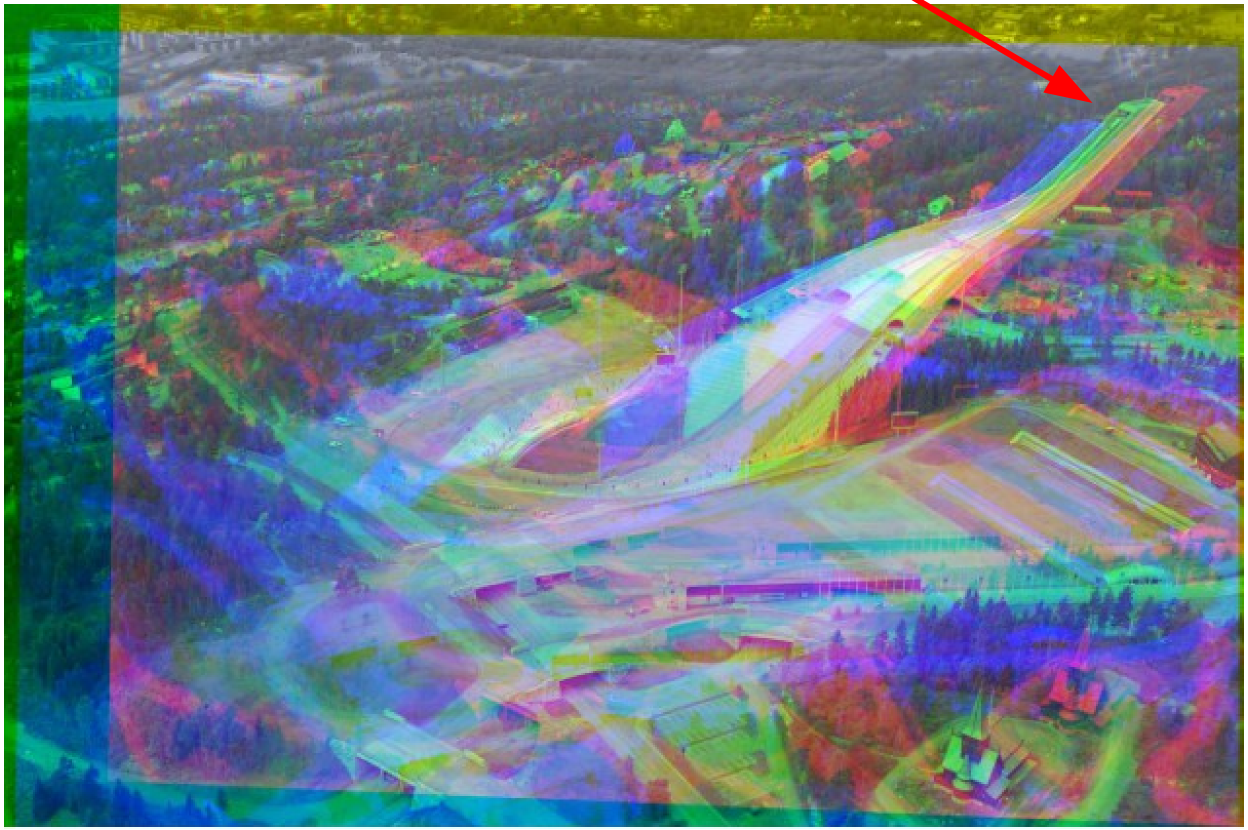
# Plane sweep for multiple views

**Red:**

**Green:**

**Blue:**

790 meter



sweep plane = 2168 meter below reference camera

# Plane sweep stereo

Algorithm:

1) Map each image to the reference image for each sweep plane with $H^{-1}_{\prod_m, P_k}$
2) Compute the similarity between Patches $W$ of the reference image and each warped image. Use e.g. normalized cross correlation

$$NCC(W_1, W_2) = \frac{\sum_x (W_1(x) - mean_1)(W_2(x) - mean_2)}{\sqrt{\sum_x (W_1(x) - mean_1)^2 \sum_x (W_2(x) - mean_2)^2 \text{¿}}}$$

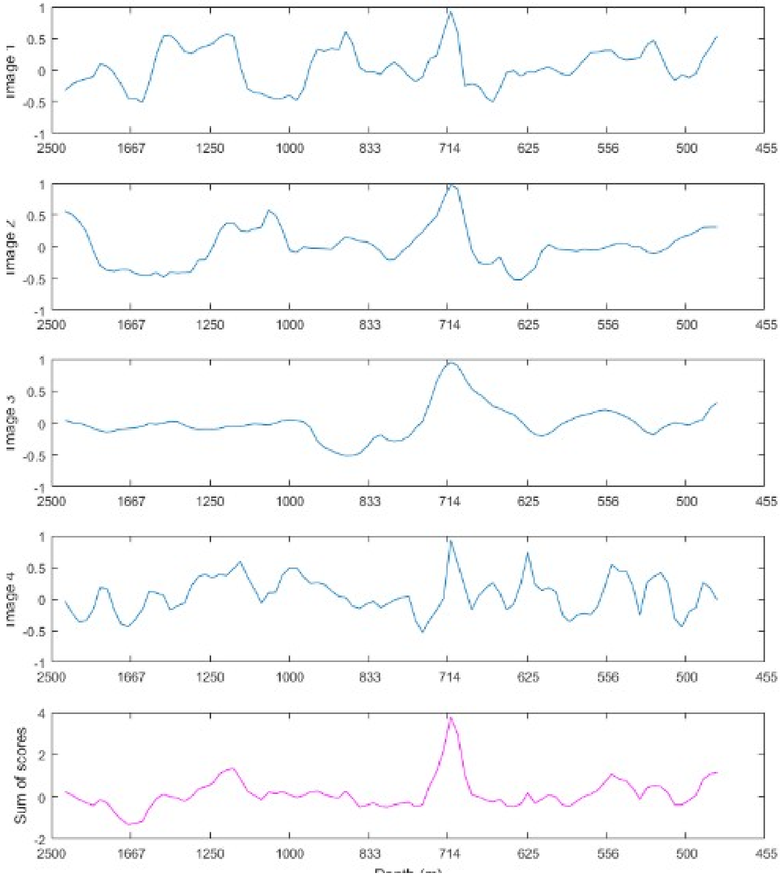
3) Do 2) for each camera and sum up

$$M(u, v, \prod_m) = \sum_k NCC_W(I_{ref}, I_{k,m})$$

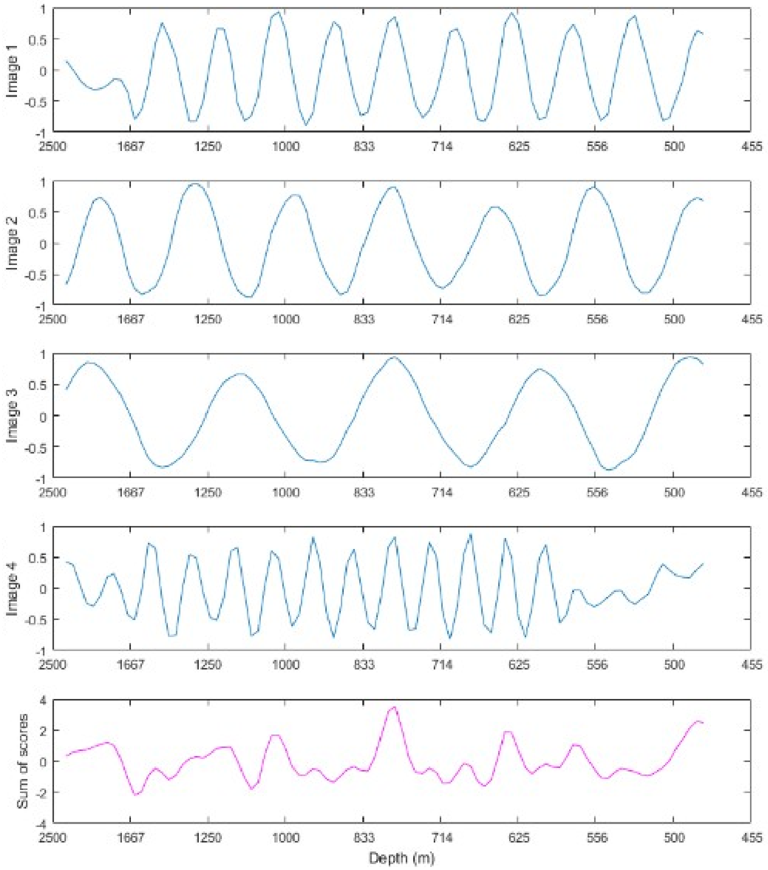
4) Select for each pixel the best depth plane

$$\prod(u, v) = \underset{m}{arg\,max}\, M(u, v, \prod_m)$$

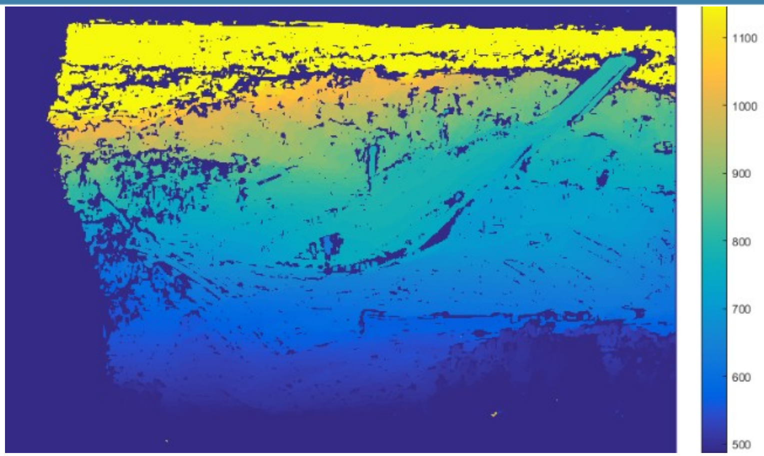
# Plane sweep example



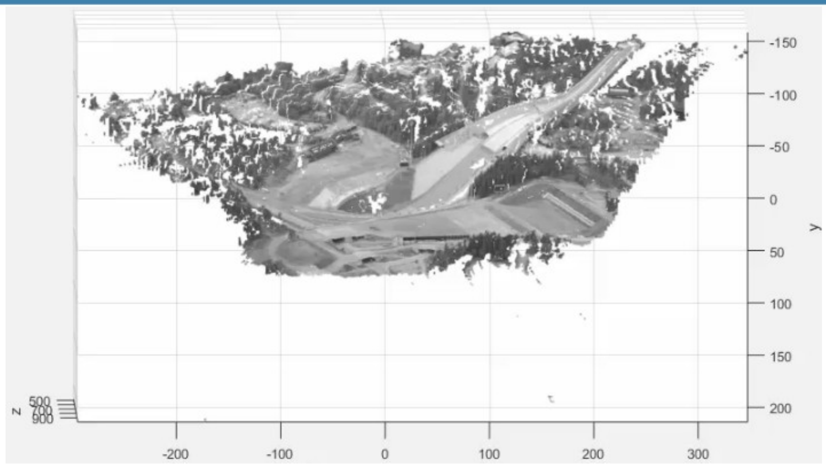
# Plane sweep in difficult areas



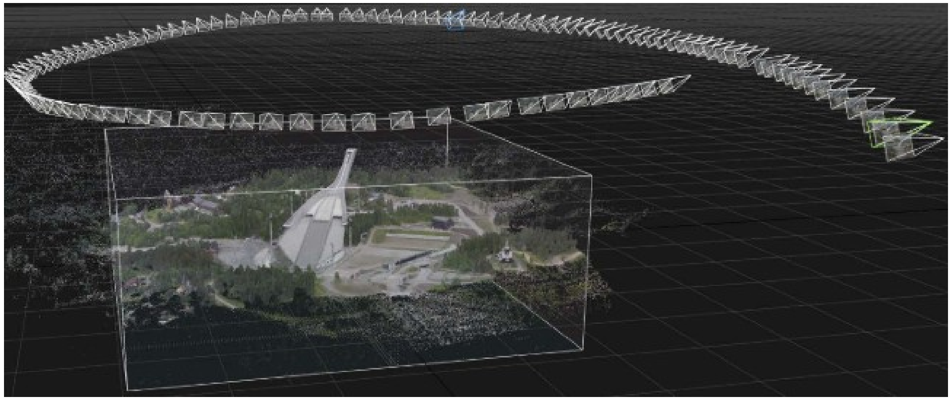
# Pointcloud reconstruction




Depth map




Pointcloud

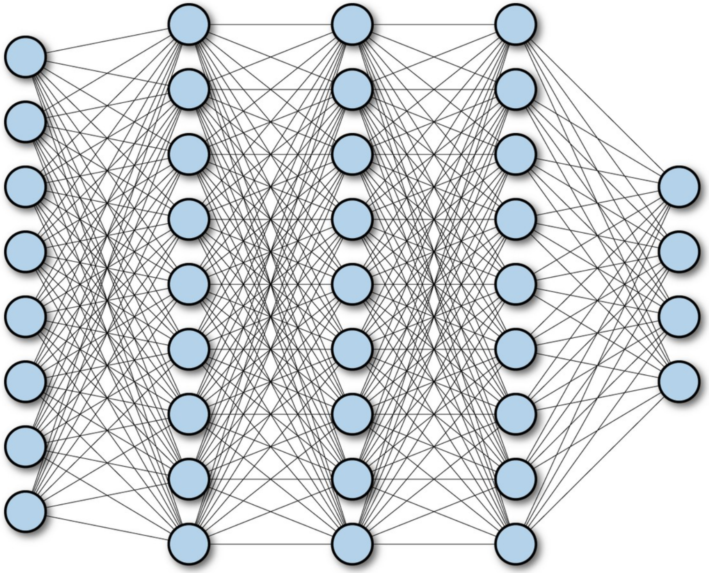


Choose a set of reference cameras

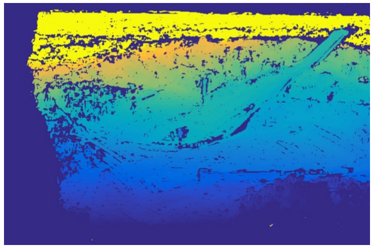
# Deep plane sweep stereo (DPSNet)

- End-to-End training
- Models the full plane sweep process

N-views



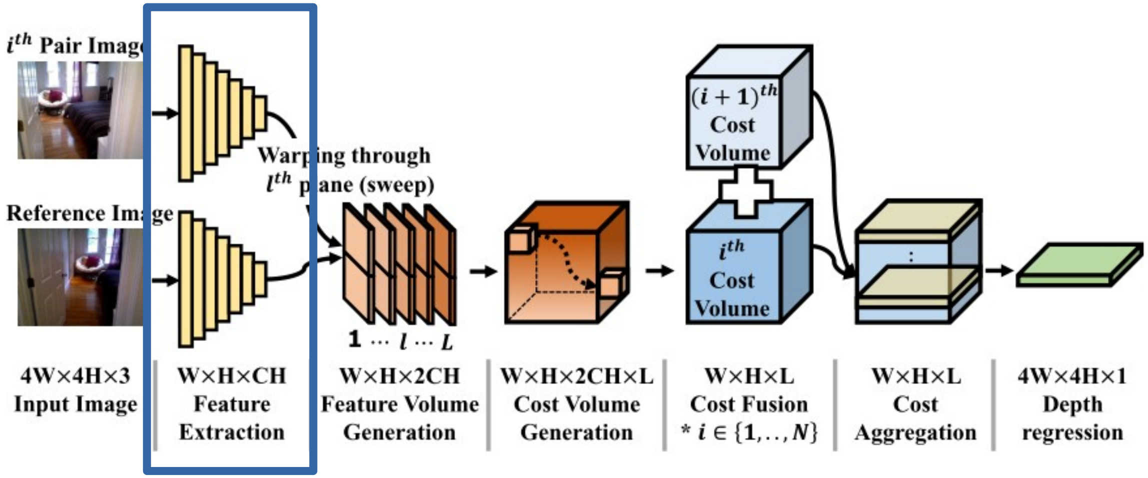
Neural Network



Disparity map

Im, S. et. al., DPSNET: END-TO-END DEEP PLANE SWEEP STEREO (ICLR), 2019

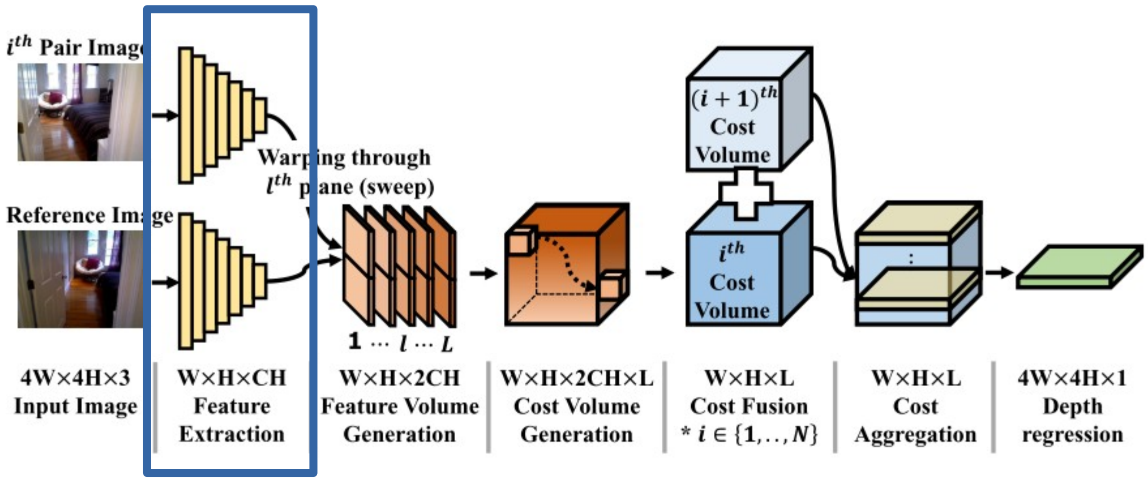
# DPSNet architecture



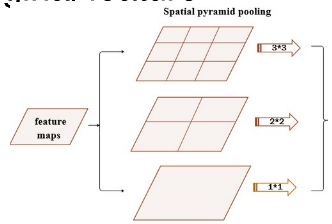

- 59 convolutional layer with batch normalization, ReLU and Residual connections
- Output: (B,F,H,W) Tensor, with

B: minibatchsize   F: number of features   H: height   W: width

# DPSNet architecture

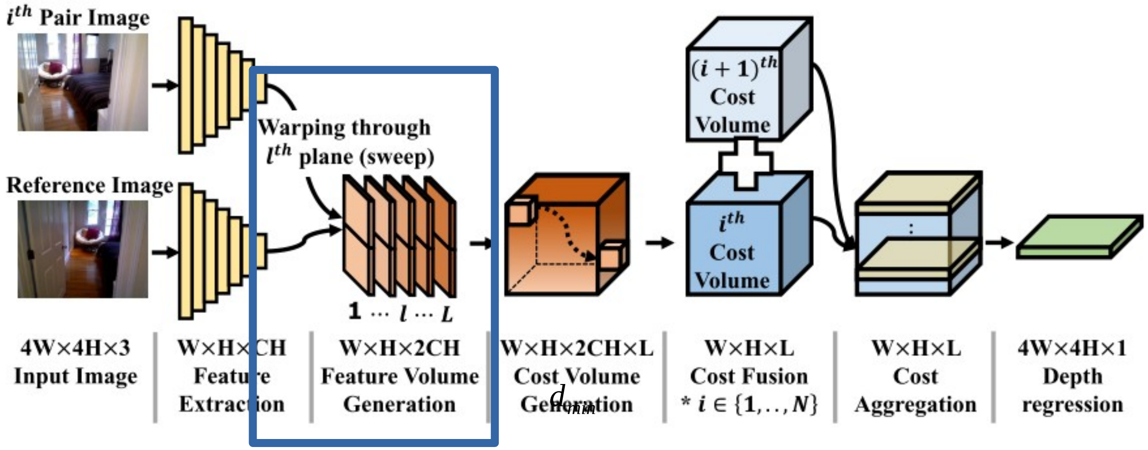



- Spatial pyramid pooling to extract multi-scale features (He et al. 2015)
  - Average pooling (16 x 16, 8 x 8, 4 x 4, 2 x 2)
  - Upsample the hierachical contextual information to the same size as the original feature map
  - Concatenate all feature maps
  - Final convolutional layer to get for each input image 32 features maps




He et al., Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition (2015)

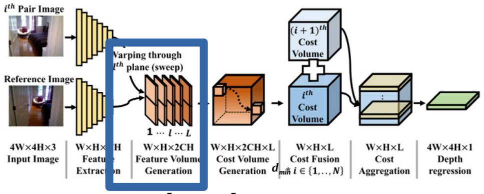
# DPSNet architecture




- Set the number of virtual planes perpendicular to the z-axis of the reference viewpoint and sample in the inverse-depth space:
  - L: total number of depth labels
  - $d_{min}$: minimum scene depth

$$d_l = \frac{L \cdot d_{min}}{l}, (l = 1, \ldots, L)$$

# DPSNet architecture



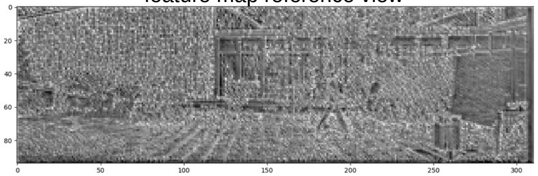

- Warp all features of the target view into reference view (same as in the classical plane sweep approach):

$$u_{ref} \sim K_{ref}\left[R_k | t_k\right]\begin{bmatrix}(K_k^{-1} u_k)\, d_l \\ 1\end{bmatrix}$$

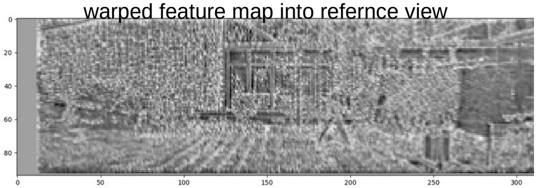
$u_{ref}, u_k$ homogenous coordinates of a pixel in reference view and target view k

input image refernce view

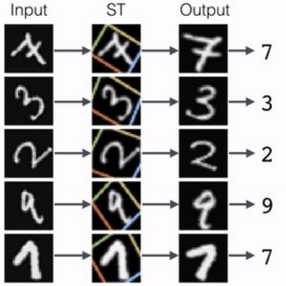
feature map reference view



warped feature map into refernce view

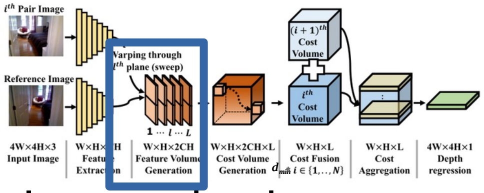


- Use a spatial transformer network for the warping process (Jaderberg et al. 2015)




Jaderberg et al., Spatial Transformer Networks (2015)

# Spatial Transformer Networks

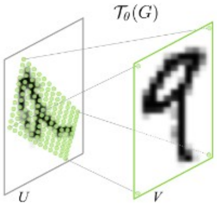


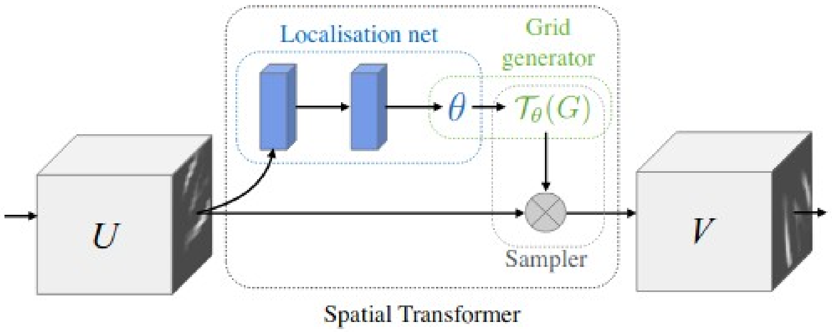
- Spatial transformer is a differentiable module, giving neural networks the ability to actively spatially transform feature maps

**Localization network** regresses transformation paramters

**Grid generator** uses the parameters to compute a sampling grid

$$\begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix} = \begin{bmatrix} \theta_{11} & \theta_{12} & \theta_{13} \\ \theta_{21} & \theta_{22} & \theta_{23} \end{bmatrix} \begin{pmatrix} x_i^t \\ y_i^t \\ 1 \end{pmatrix}$$



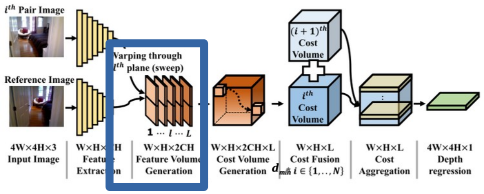




**Sampler** uses bilinear interpolation to produce output $V_i^c = \sum_n^H \sum_m^W U_{nm}^c \max(0, 1 - |x_i^s - m|) \max(0, 1 - |y_i^s - n|)$

# DPSNet architecture

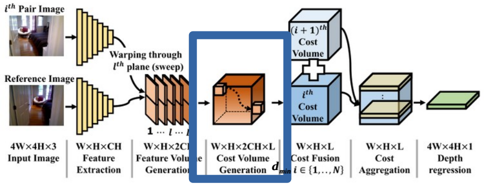
- Output Tensor after warping has shape of [B, 2F, D, H, W]

  D: number of depth planes



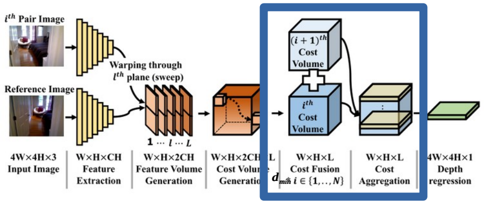

---

- Use a series of 3d convolutions to learn the cost volume generation
  - Output tensor of shape [B, D, H, W]

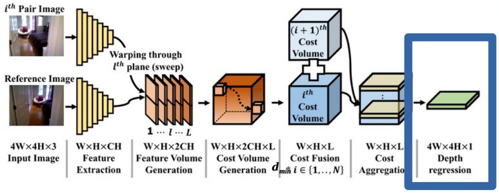



---

- In the training step use only one paired image
- In the testing step use any number of paired images by averaging the cost volumes



# DPSNet architecture




- Regress continuous depth values
  - But argmin function is:
    - Discrete and unable to produce sub-pixel disparity
    - Not differentiable

- Therefore, compute a soft argmin which is differentiable

$$\hat{d} = \frac{L \times d_{min}}{\tilde{l}} \qquad \tilde{l} = \sum_{l=1}^{L} l \times softmax(c_l)$$

- Training loss:

$$L(\theta) = \sum_x |\hat{d}_x^{\theta} - d_x^{g}|_H$$

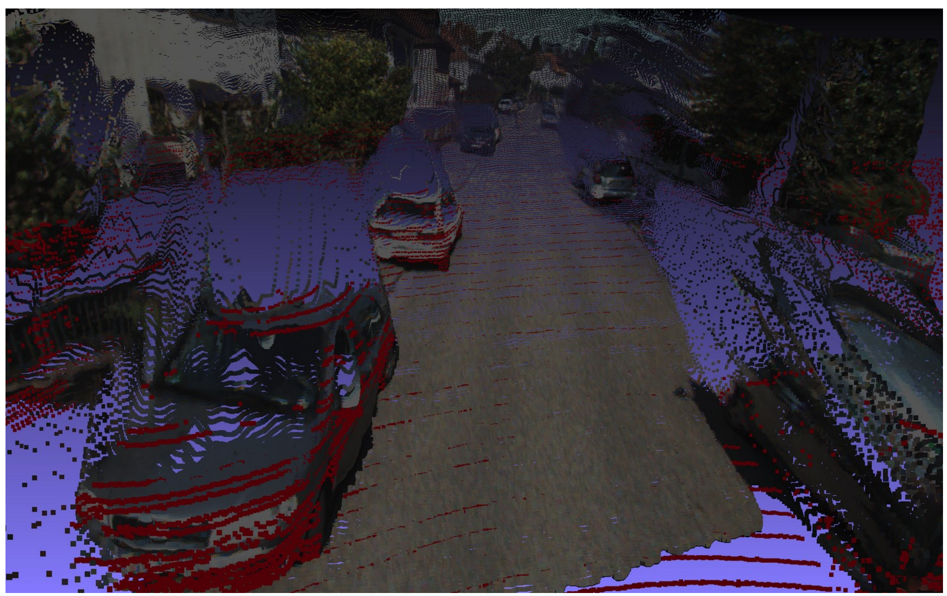
where H is SmoothL1 loss

# DPSNet Results

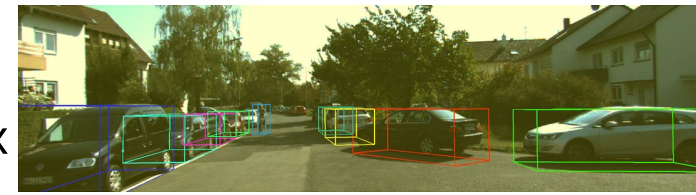
Input images

Disparity map

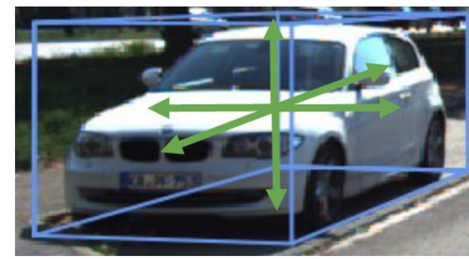
Pointcloud +
Lidar pointcloud
(red)



# 3d Object Detection



- Localize and classify objects in the scene
- Represent a detected object with a bounding box
  - Position (X, Y, Z)
  - Dimension (H, W, D)

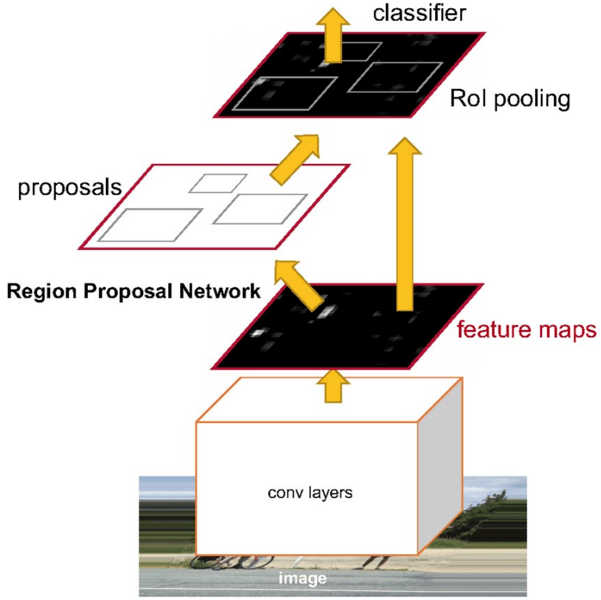


- Axis aligned and non axis aligned bounding boxes
- Kitti dataset provide ~7000 annotated frames + synced lidar, gps, imu data

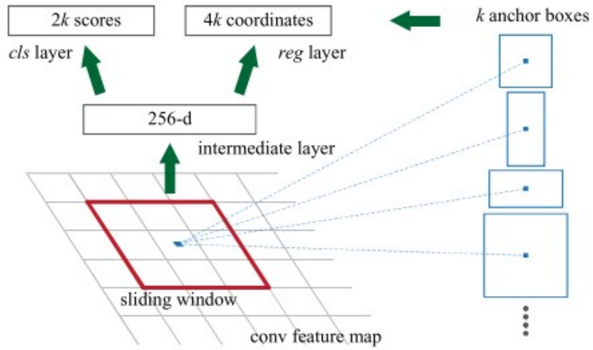
| | Method | Setting | Code | Moderate | Easy | Hard | Runtime | Environment | Compare |
|---|---|---|---|---|---|---|---|---|---|
| 1 | HRI-ADLab-HZ | | | **82.83 %** | 89.00 % | 76.00 % | 0.1 s | 1 core @ 2.5 Ghz (C/C++) | |
| 2 | SE-SSD | | | 82.54 % | 91.49 % | 77.15 % | 0.03 s | 1 core @ 2.5 Ghz (Python + C/C++) | |
| 3 | EA-M-RCNN(BorderAtt) | | | 82.33 % | 87.77 % | 77.37 % | 0.08 s | 1 core @ 2.5 Ghz (C/C++) | |
| 4 | HUAWEI Octopus | | | 82.13 % | 88.26 % | **77.41 %** | 0.1 s | 1 core @ 2.5 Ghz (C/C++) | |
| 5 | ADLAB | | | 82.08 % | 90.92 % | 77.36 % | 0.05 s | 1 core @ >3.5 Ghz (C/C++) | |
| . | | | | | | | | | |
| . | | | | | | | | | |
| . | | | | | | | | | |
| 111 | UM3D_TUM | | | 0.62 % | 0.45 % | 0.62 % | 0.05 s | 1 core @ 2.5 Ghz (Python) | |
| 112 | MonoRUn | | | 0.61 % | 1.01 % | 0.48 % | 0.07 s | GPU @ 2.5 Ghz (Python + C/C++) | |
| 113 | Shift R-CNN (mono) | | code | 0.29 % | 0.48 % | 0.31 % | 0.25 s | GPU @ 1.5 Ghz (Python) | |
| A. Naiden, V. Paunescu, G. Kim, B. Jeon and M. Leordeanu: Shift R-CNN: Deep Monocular 3D Object Detection With Closed-form Geometric Constraints. ICIP 2019. | | | | | | | | | |
| 114 | PVNet | | | 0.00 % | 0.00 % | 0.00 % | 0,1 s | 1 core @ 2.5 Ghz (Python) | |
| 115 | mBoW | | | 0.00 % | 0.00 % | 0.00 % | 10 s | 1 core @ 2.5 Ghz (C/C++) | |

# Recap 2d object detection

- Faster R-CNN for 2d object detection (Ren et al. 2015)



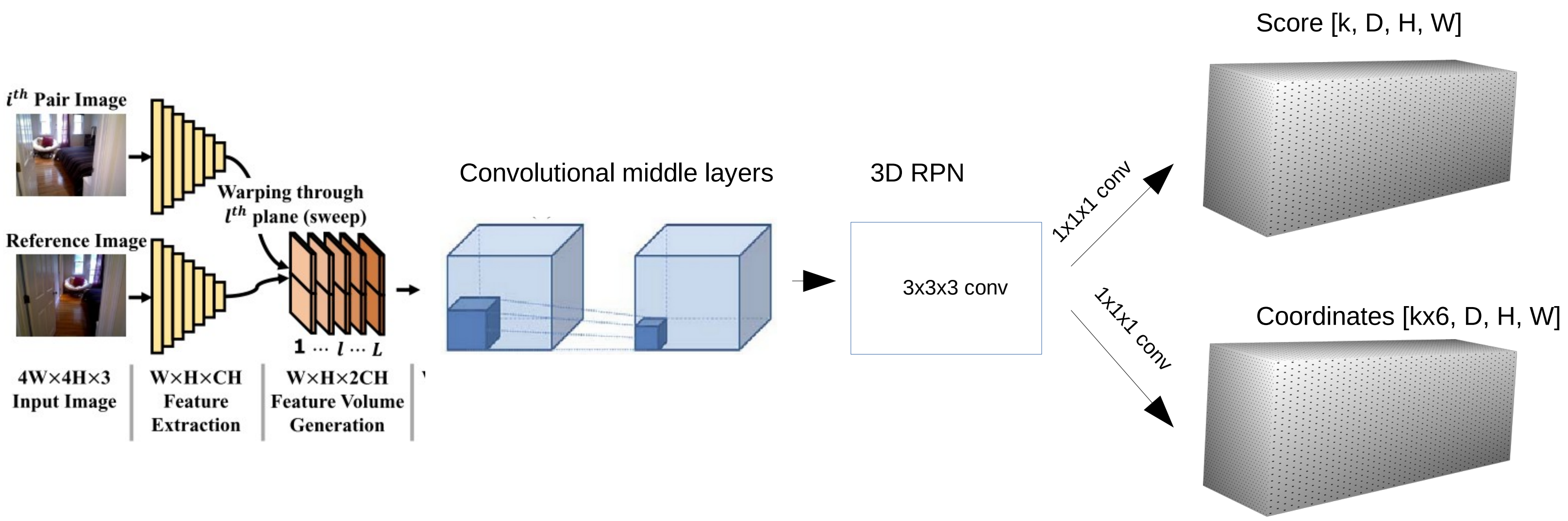

- Sliding Region Proposal Network




Ren et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks (2015)

# 3d region proposal

1) Use a muti-view backbone like first part of DPSNet
2) Convolutional middle layers
3) 3d Region Proposal Network



# Loss

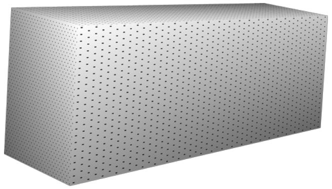
- Assign labels to output volume
  - Labels are (X, Y, Z, L, W, H) of the bounding box
  - Compute 3d position for each pixel in  the output volume

$$Z = D$$

$$X = -Z\frac{(x + c_x)}{f}$$

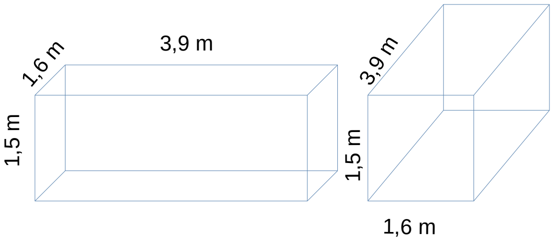
$$Y = -Z\frac{(y + c_y)}{f}$$

Coordinates [kx6, D, H, W]



Network output

- Define anchor boxes for the objects
  - e.g. for cars



# Loss

- Compute intersection over union between ground truth boxes and anchor boxes
- Define a residual vector $u^x$ with the positve anchor parameters $(x^a, y^a, z^a, l^a, w^a, h^a)$ and the ground truth parameters $(x^g, y^g, z^g, l^g, w^g, h^g)$ as

$$\Delta x = \frac{x_c^g - x_c^a}{d^a}, \Delta y = \frac{y_c^g - y_c^a}{d^a}, \Delta z = \frac{z_c^g - z_c^a}{h^a},$$
$$\Delta l = \log(\frac{l^g}{l^a}), \Delta w = \log(\frac{w^g}{w^a}), \Delta h = \log(\frac{h^g}{h^a}),$$
$$\text{where } d^a = \sqrt{(l^a)^2 + (w^a)^2}$$

Loss function, same as VoxelNet (Zhou and Tuzel):

$$L = \alpha \frac{1}{N_{\text{pos}}} \sum_i L_{\text{cls}}(p_i^{\text{pos}}, 1) + \beta \frac{1}{N_{\text{neg}}} \sum_j L_{\text{cls}}(p_j^{\text{neg}}, 0) + \frac{1}{N_{\text{pos}}} \sum_i L_{\text{reg}}(\mathbf{u}_i, \mathbf{u}_i^*)$$

Regression loss: Smooth L1

Classification: Binary Cross-entropy

Positive, if IoU > 0,6, negative if IoU < 0,3

Zhou and Tuzel, VoxelNet: End-to-End Learning for Point Cloud Based 3D object Detection

# Qualitative results

Red: Network output
Blue: Axis aligned ground truth

# References

- Many pictures and slides are from „Lecture 8.3 Multiple-view stereo, Trym Vegard Haavardsholm“
- Some slides are inspired by „KI & Autonomes Fahren: Sehen lernen um fahren zu lernen, Andreas Geiger“ https://www.youtube.com/watch?v=HKsqhHuQqxE&t=212s
- Lecture Robotics 2, Uni Freiburg, Barbera Frank

  http://ais.informatik.uni-freiburg.de/teaching/ws10/robotics2/pdfs/rob2-10-camera-calibration.pdf
- Lecture 6 Computer Vision, HTWG Konstanz, Matthias O. Franz
- See the references in the footnote of the slides

# Thanks for your attention!

*Dennis Grießer*

*Institute for Optical Systems*

*d.griesser@htwg-konstanz.de*