AI70601/CSE61101

3D Vision and Machine Perception

Prof. Kyungdon Joo

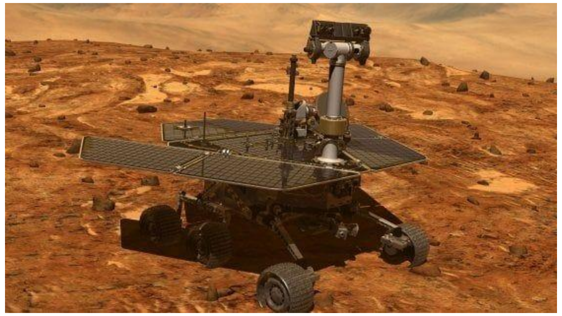
3D Vision & Robotics Lab.

AI Graduate School (AIGS) & Computer Science and Engineering (CSE)

Some materials, figures, and slides (used for this course) are from textbooks, published papers, and other open lectures

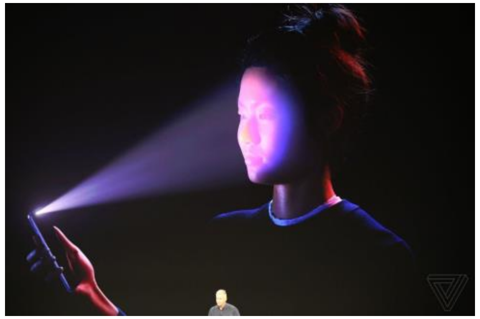
Depth (3D) sensing

- Depth is a crucial cue for many computer vision applications



Robotic (NASA)

Autonomous driving (Google)



Biometric (Apple)

Drones (DJI)

Gaming (Microsoft)

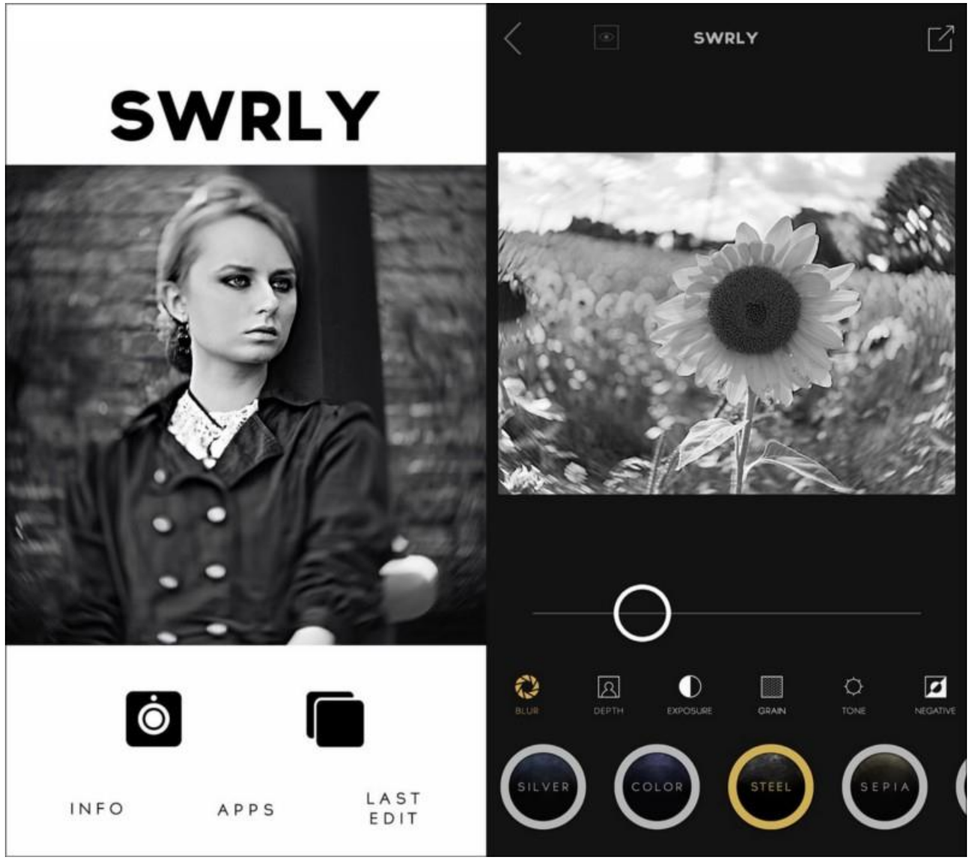
Augmented Reality (Microsoft)

Slide credit: CVPR 2019 Tutorial: Learning-based depth estimation from stereo and monocular images: successes, limitations and future challenges

“3D photo” in a mobile industry

Cinema4D image generation

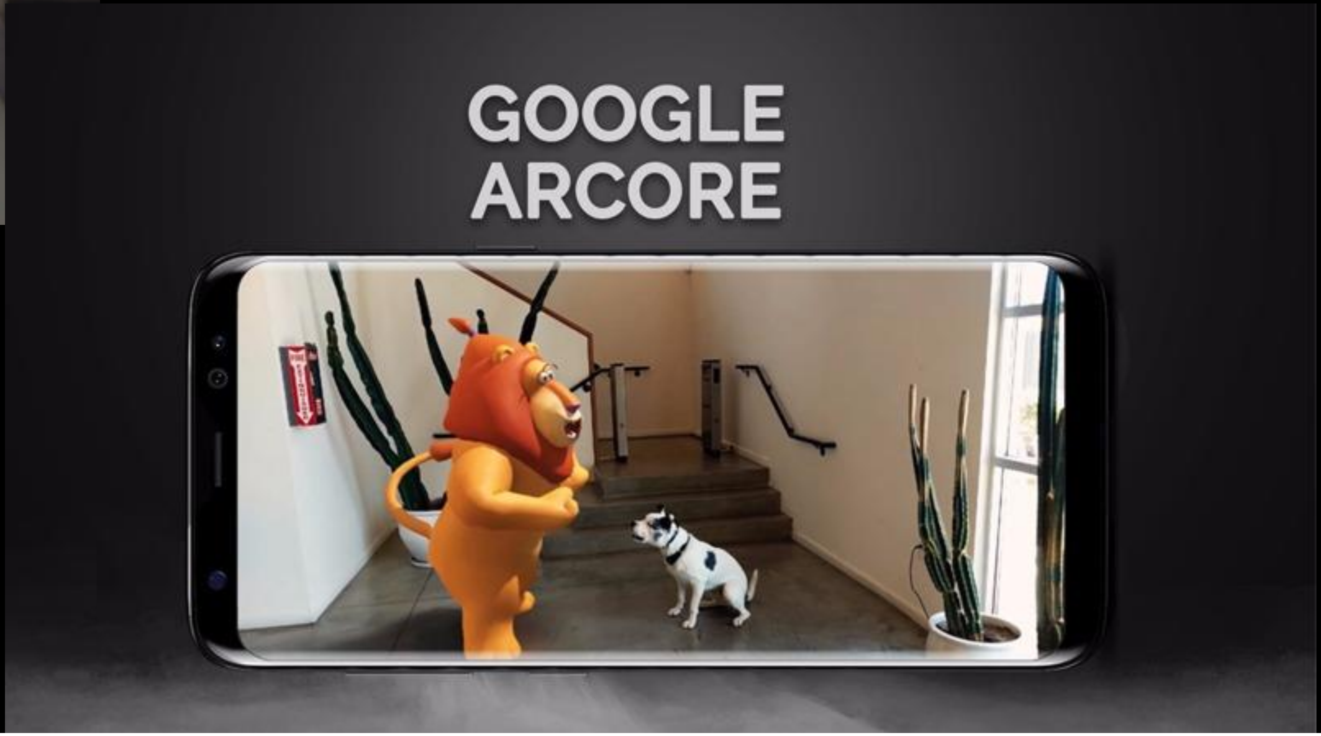
Mobile AR



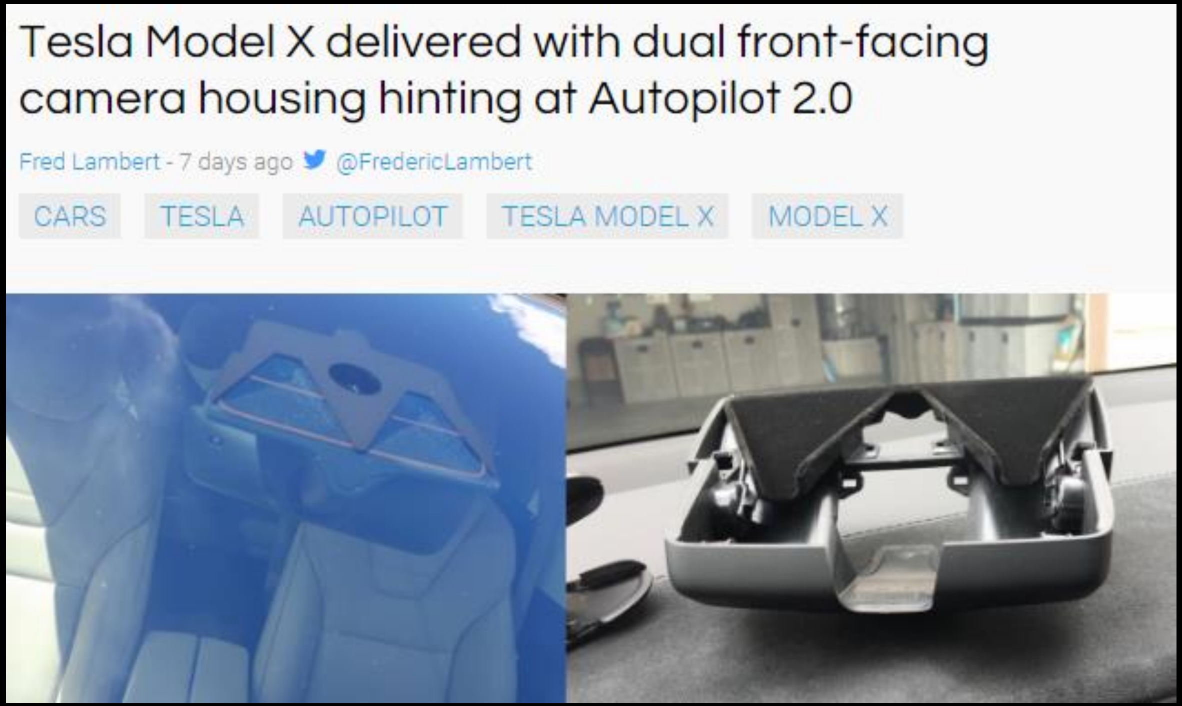
Out-focusing with swirly blur filter

Mobile application

AR development kit
Google vs Apple

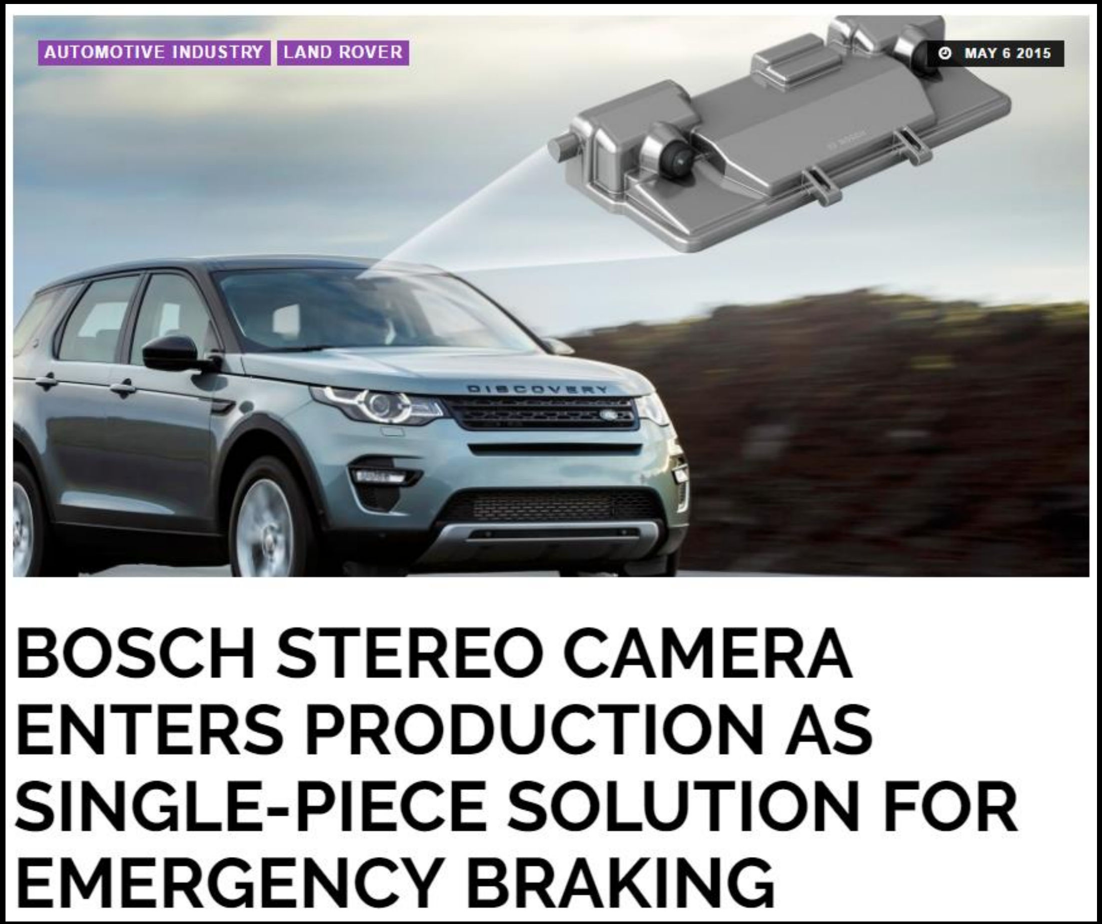


Autonomous driving application


Tesla Model X delivered with dual front-facing camera housing hinting at Autopilot 2.0

Fred Lambert - 7 days ago @FredericLambert

CARS TESLA AUTOPILOT TESLA MODEL X MODEL X

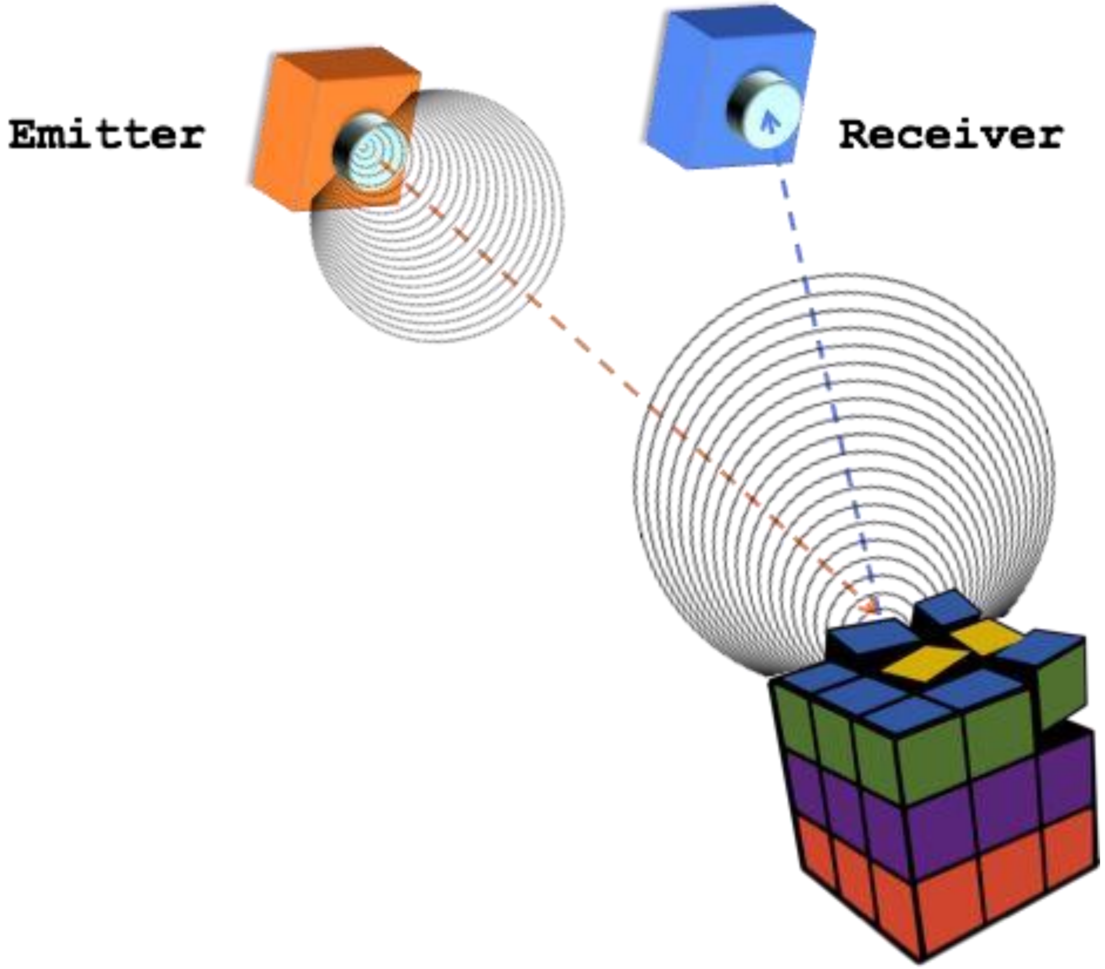

AUTOMOTIVE INDUSTRY LAND ROVER

MAY 6 2015

BOSCH STEREO CAMERA ENTERS PRODUCTION AS SINGLE-PIECE SOLUTION FOR EMERGENCY BRAKING

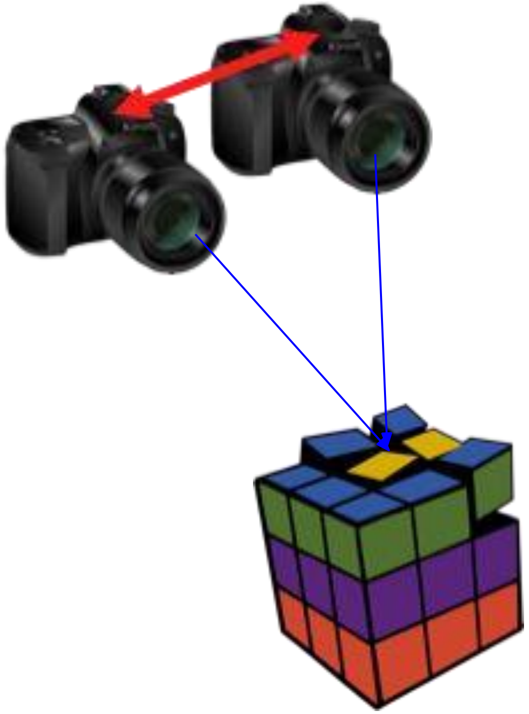
Active depth sensing

- Depth is perceived by perturbing the sensed environment:
  - LiDAR (e.g., Velodyne)
  - Structured light (e.g., Kinect 1)
  - Active stereo (e.g., Intel RealSense)

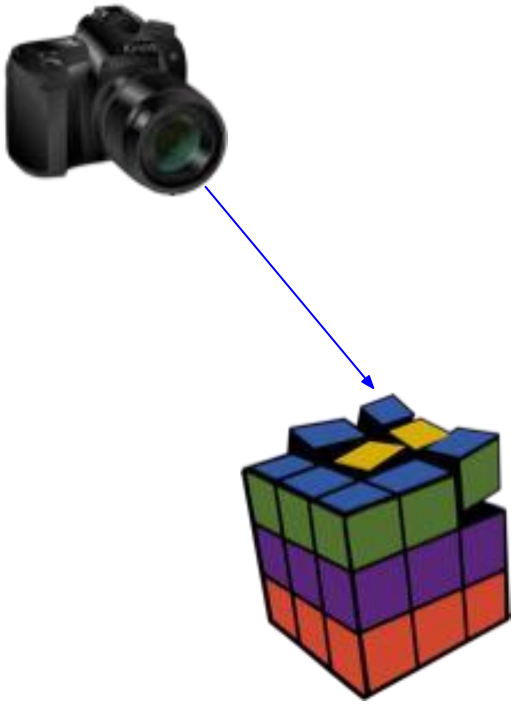


Slide credit: CVPR 2019 Tutorial: Learning-based depth estimation from stereo and monocular images: successes, limitations and future challenges

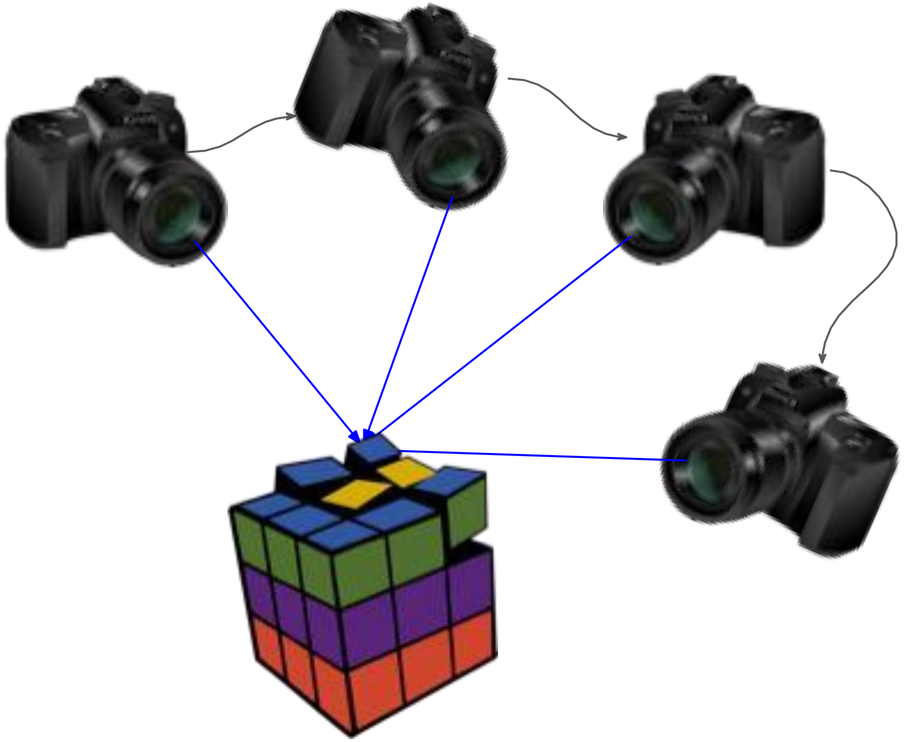
Passive depth sensing



Stereo



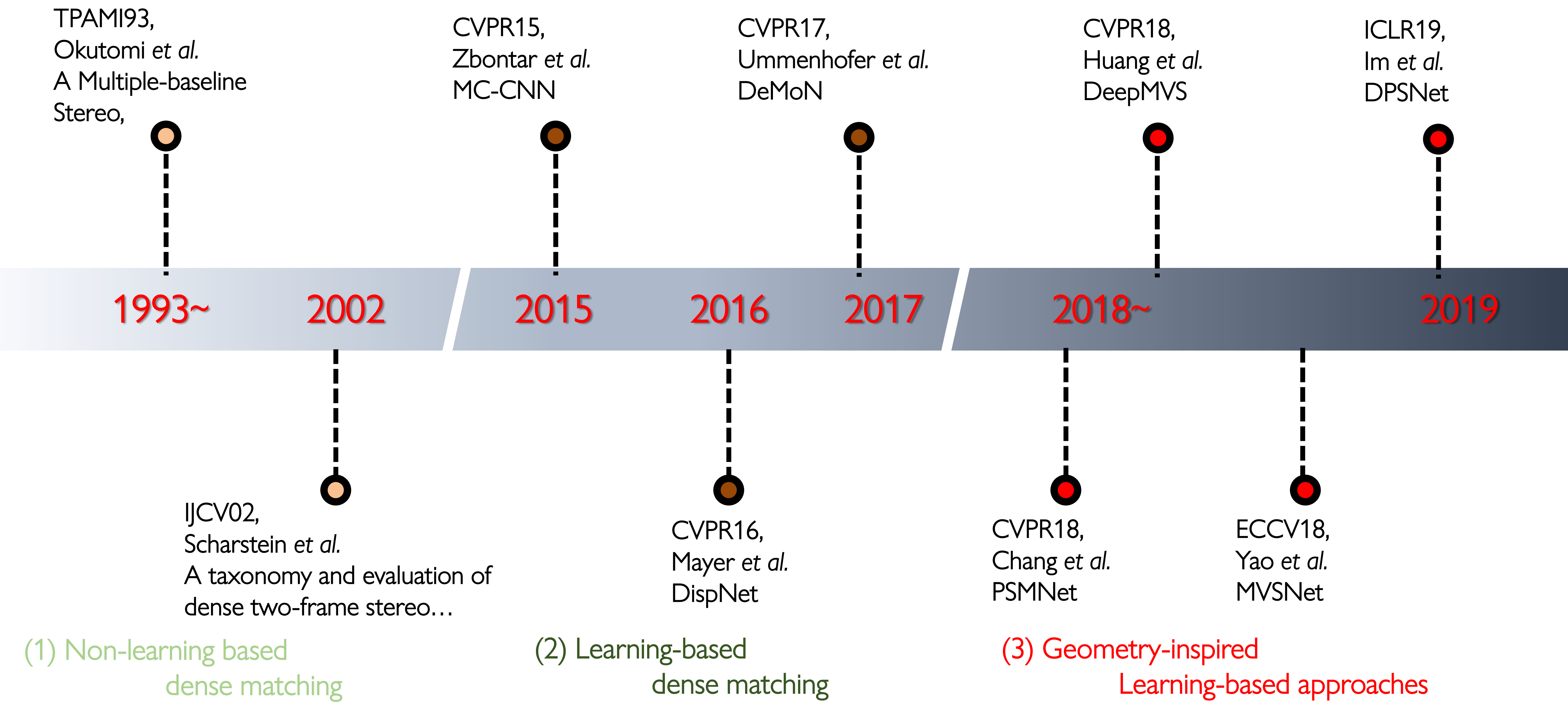
Monocular



Multi-view stereo

Slide credit: CVPR 2019 Tutorial: Learning-based depth estimation from stereo and monocular images: successes, limitations and future challenges

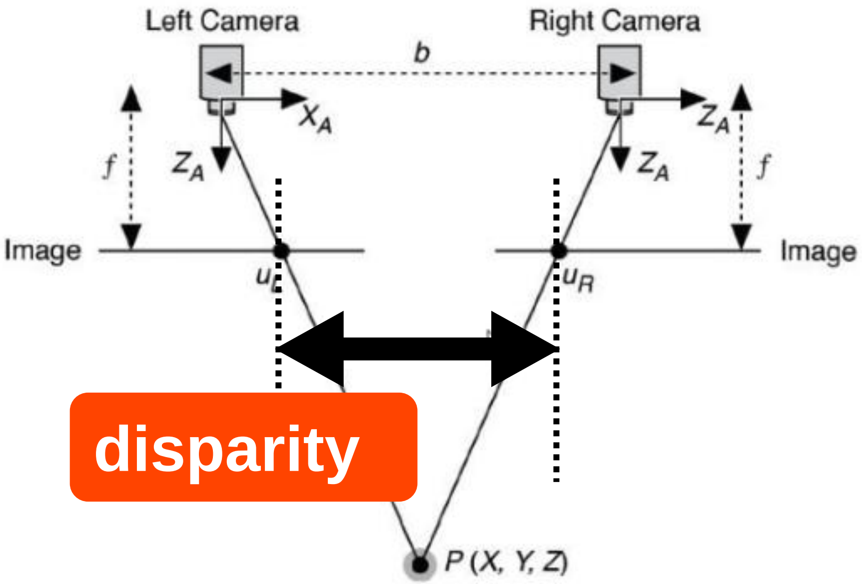
Landmarks of Stereo Matching



Traditional Stereo Matching

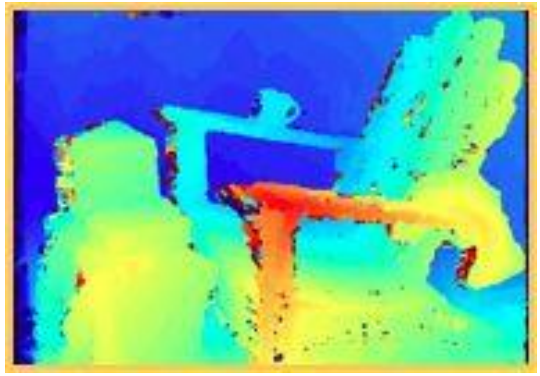
Stereo matching

- Given two (or more) images of the same scene, aims at inferring **depth**
- The **disparity** is the difference between x coordinates of corresponding points



Left (Reference)

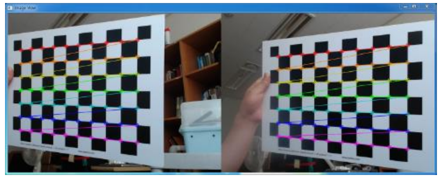
Right (Target)



Disparity map

Overall procedure of stereo matching

- Procedure

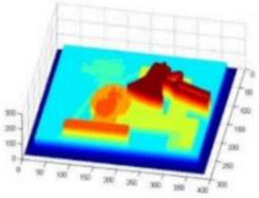


Calibration

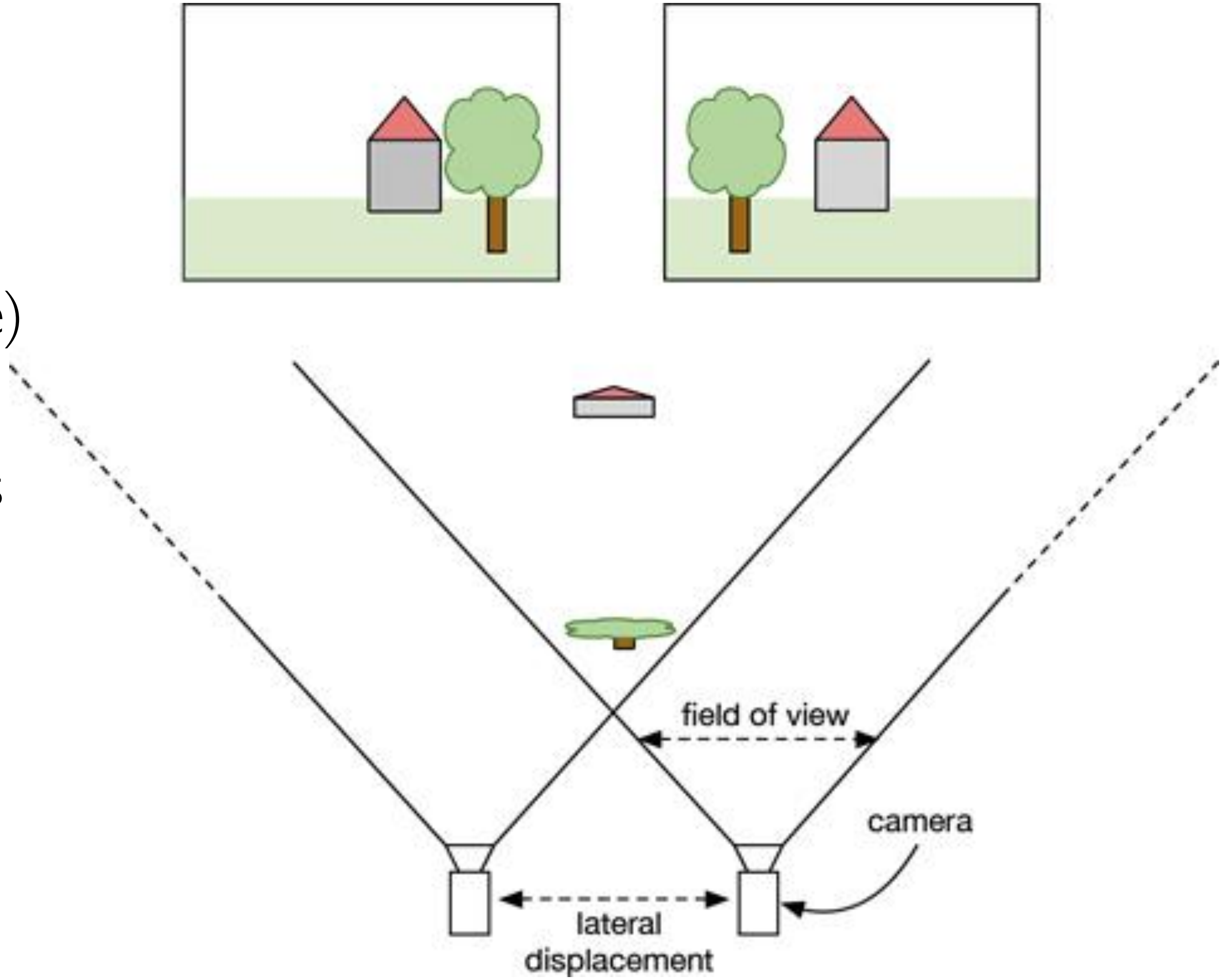
Stereo images (capture)

Rectified stereo images

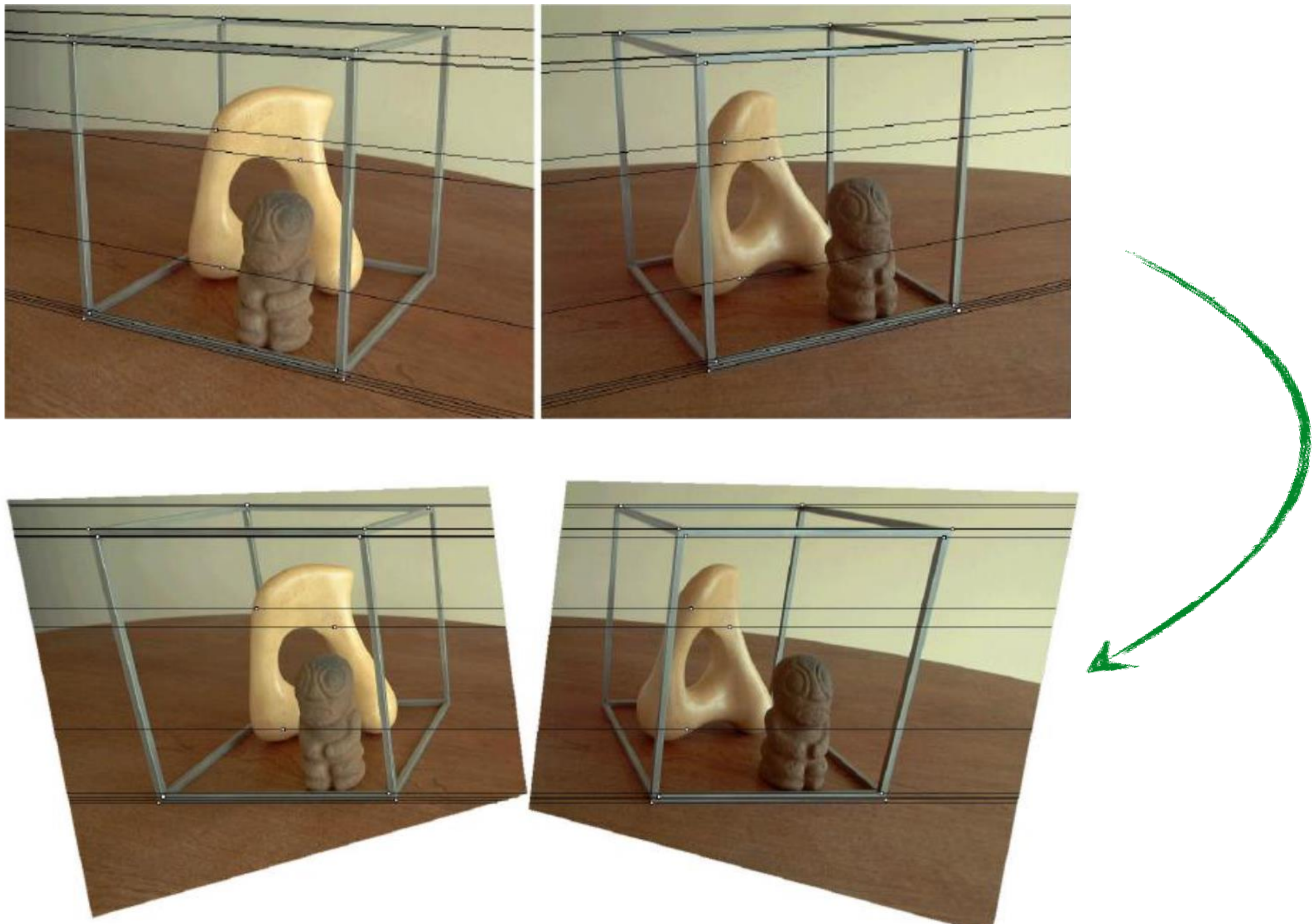
Disparity map



Depth map

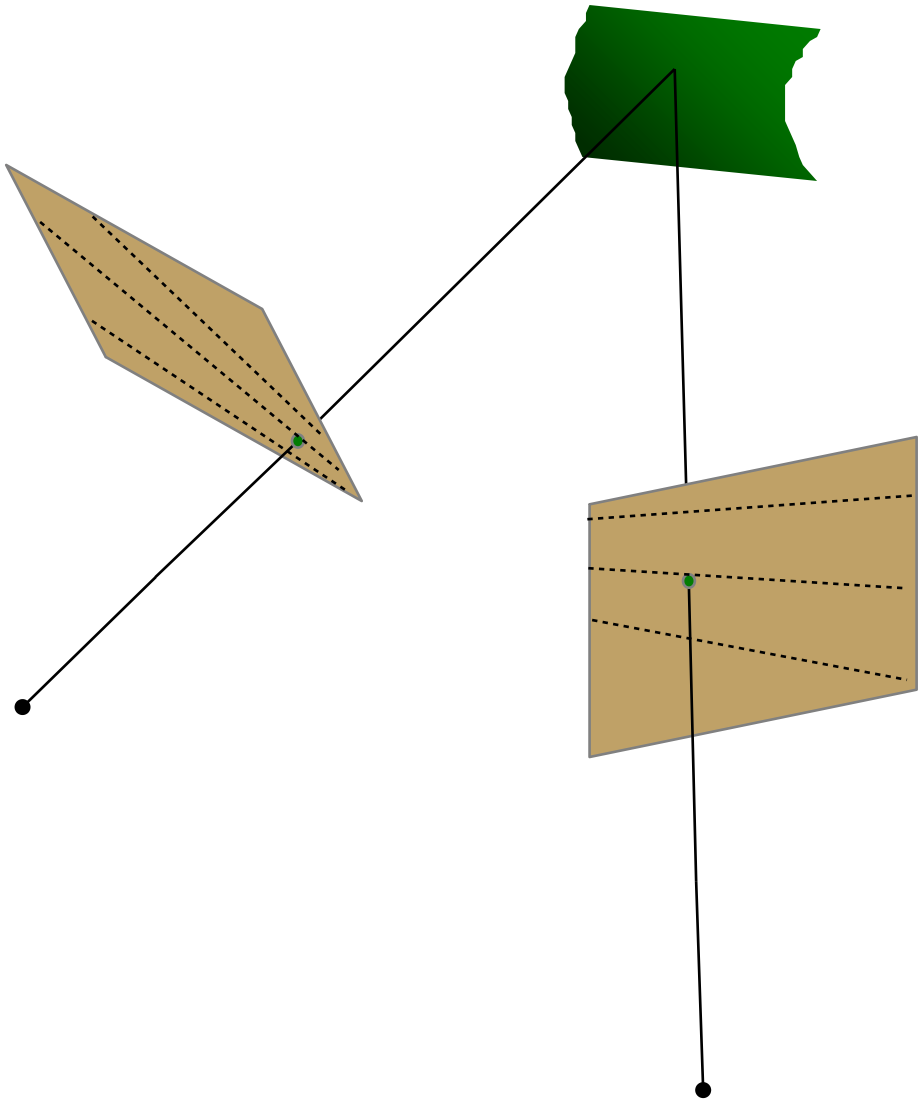


Stereo rectification



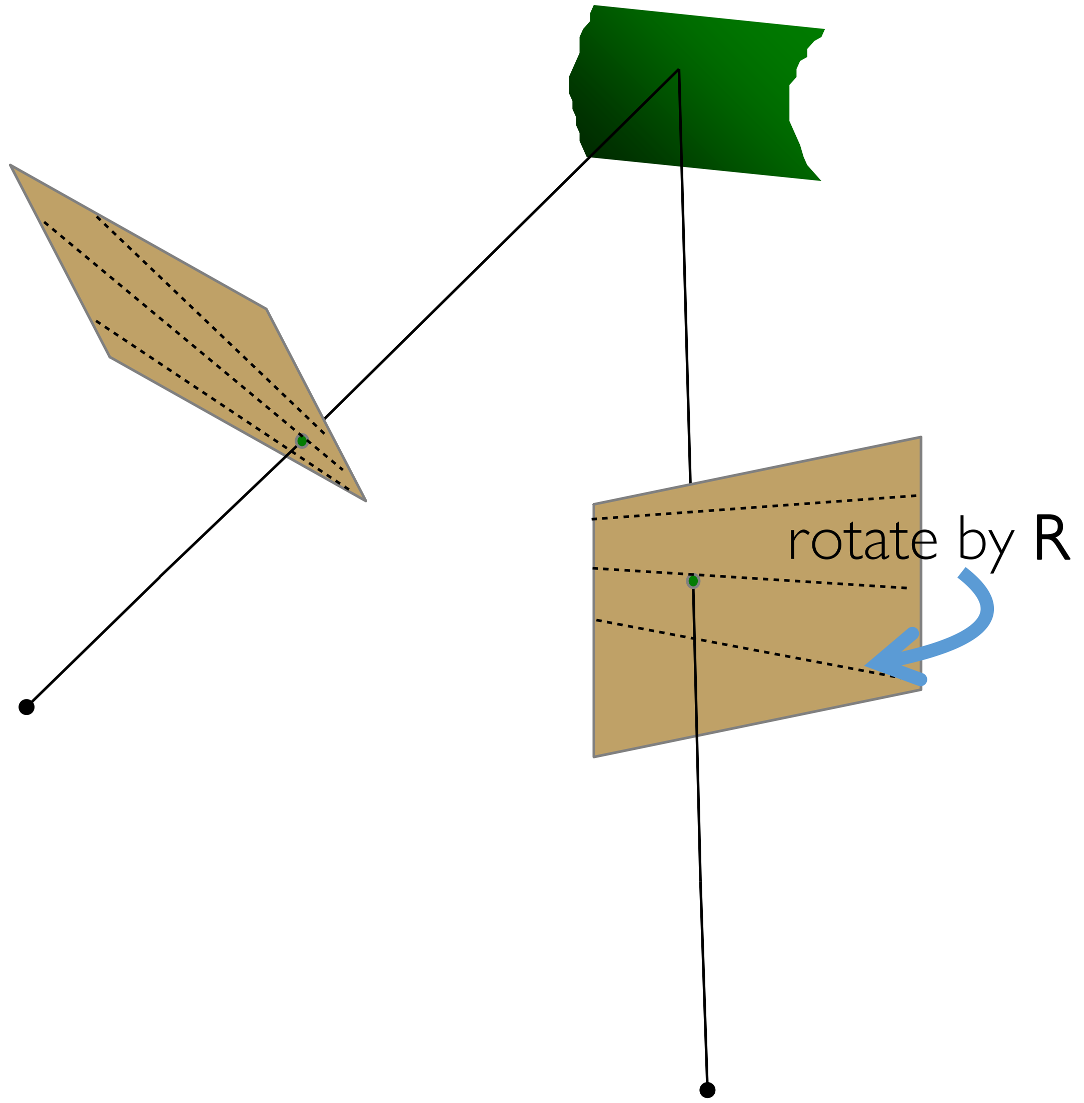
- Stereo Rectification:

1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{rect}$
4. Scale both images by **H**



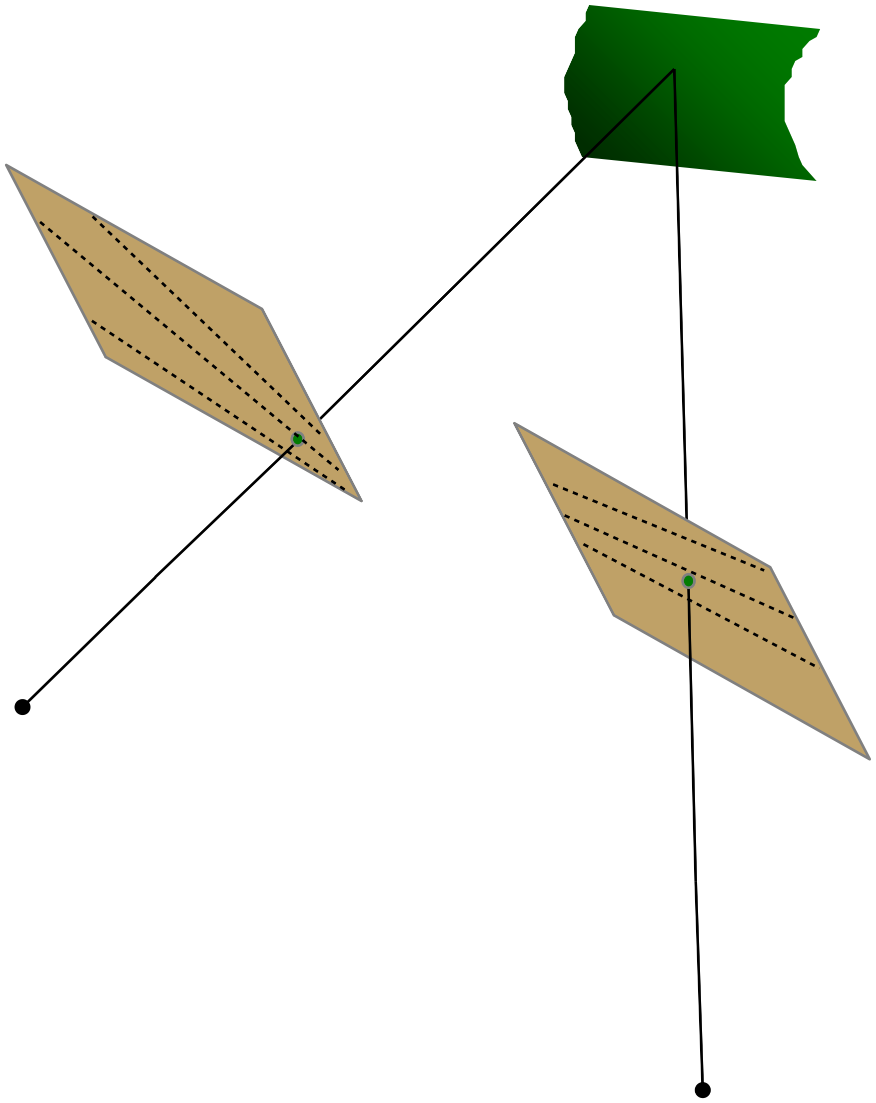
- Stereo Rectification:

1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{rect}$
4. Scale both images by **H**



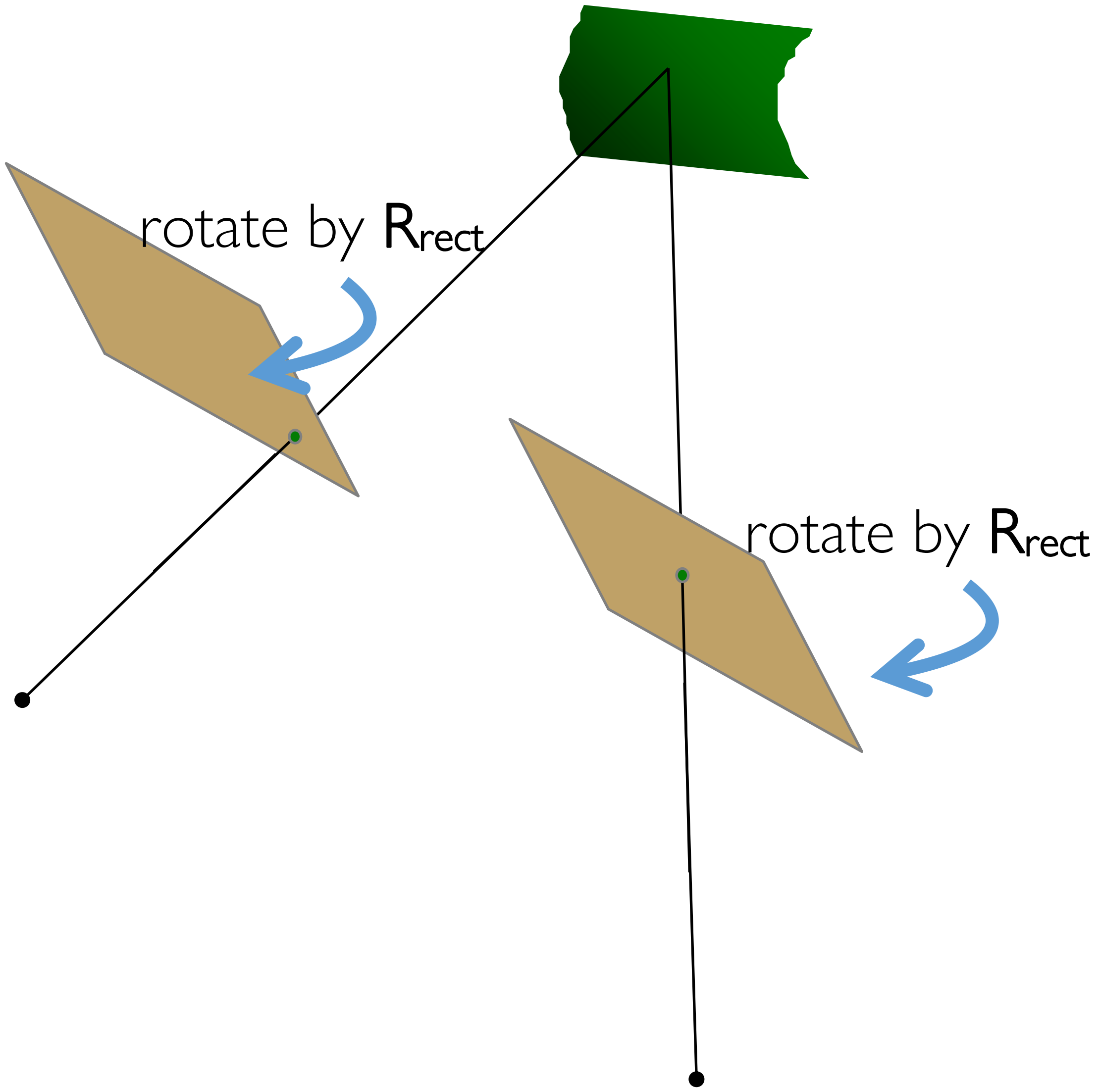
- Stereo Rectification:

1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{rect}$
4. Scale both images by **H**



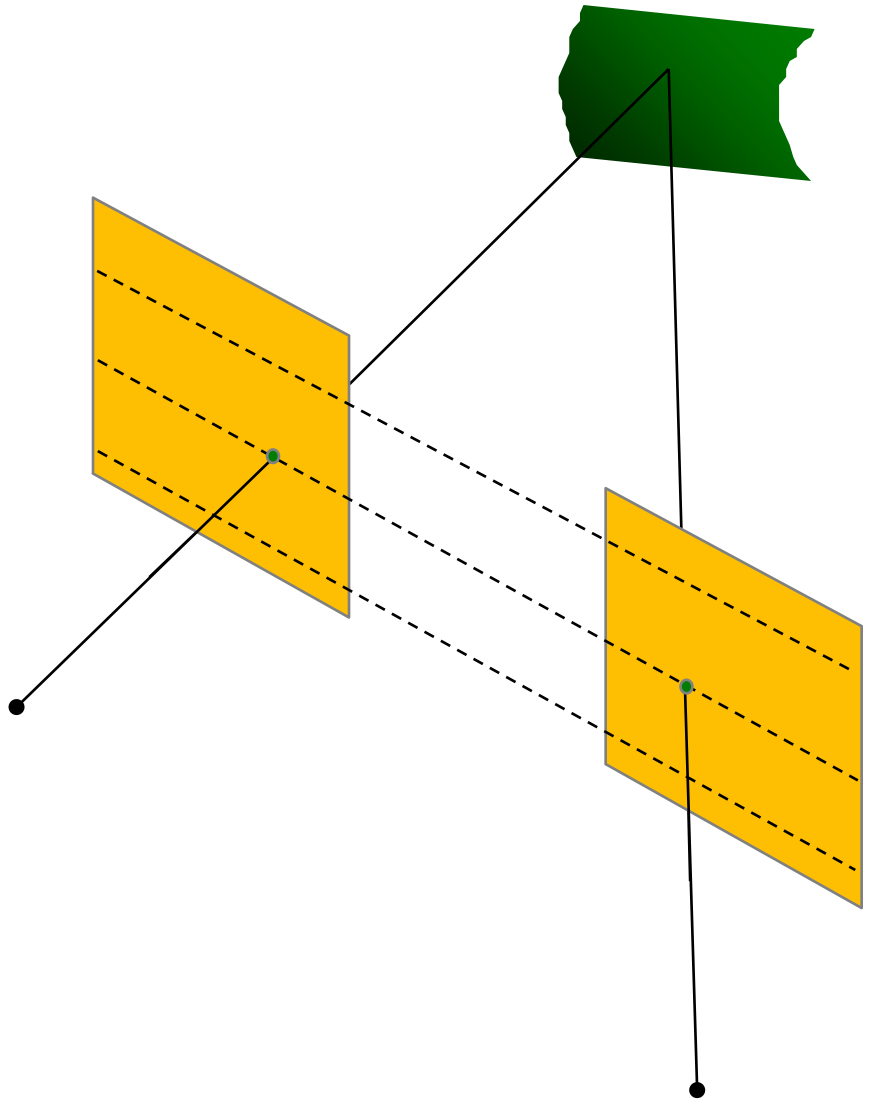
- Stereo Rectification:

  1. Compute **E** to get **R**
  2. Rotate right image by **R**
  3. Rotate both images by $\mathbf{R}_{\mathbf{rect}}$
  4. Scale both images by **H**



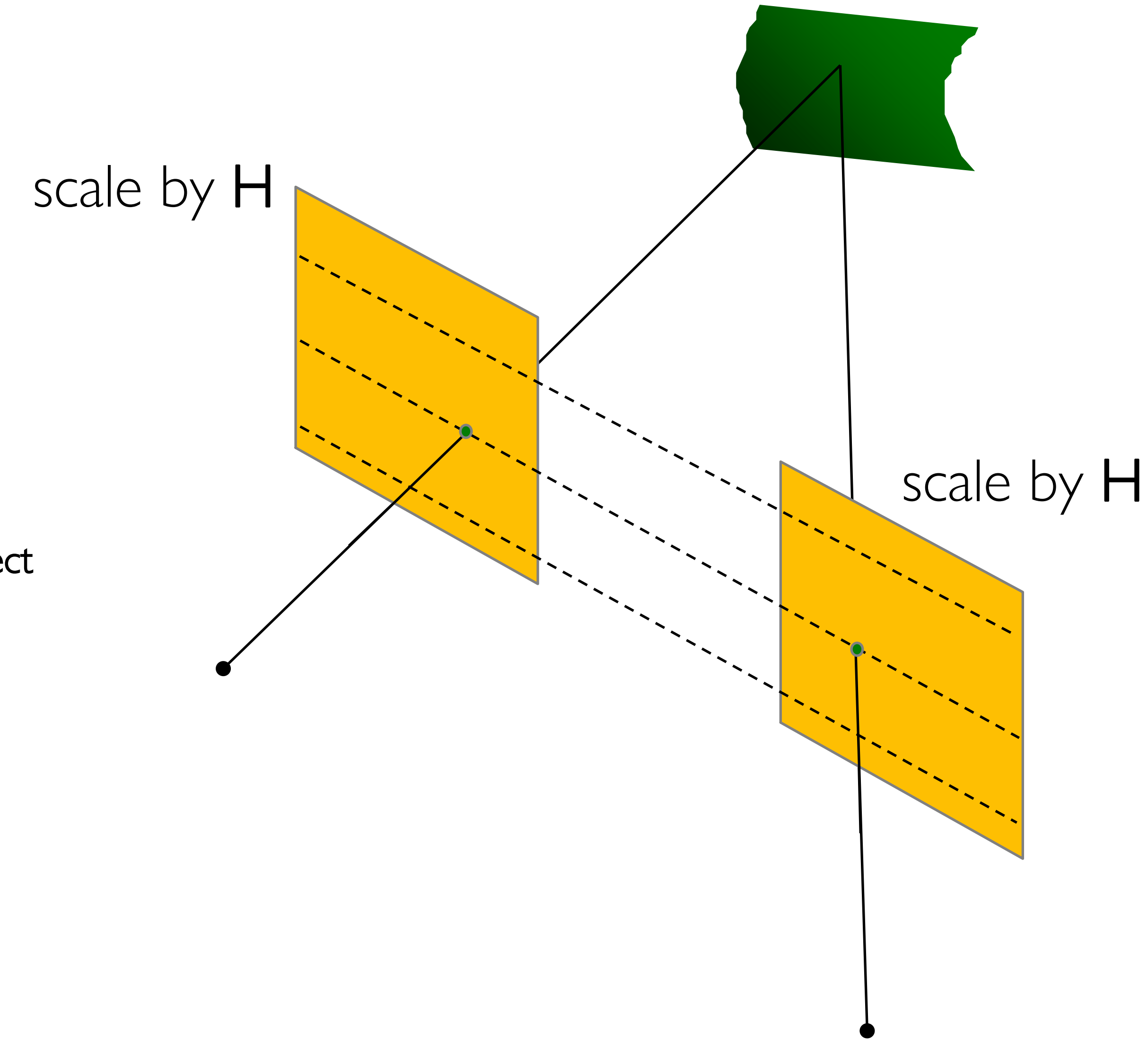
- Stereo Rectification:

1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R_{rect}}$
4. Scale both images by **H**



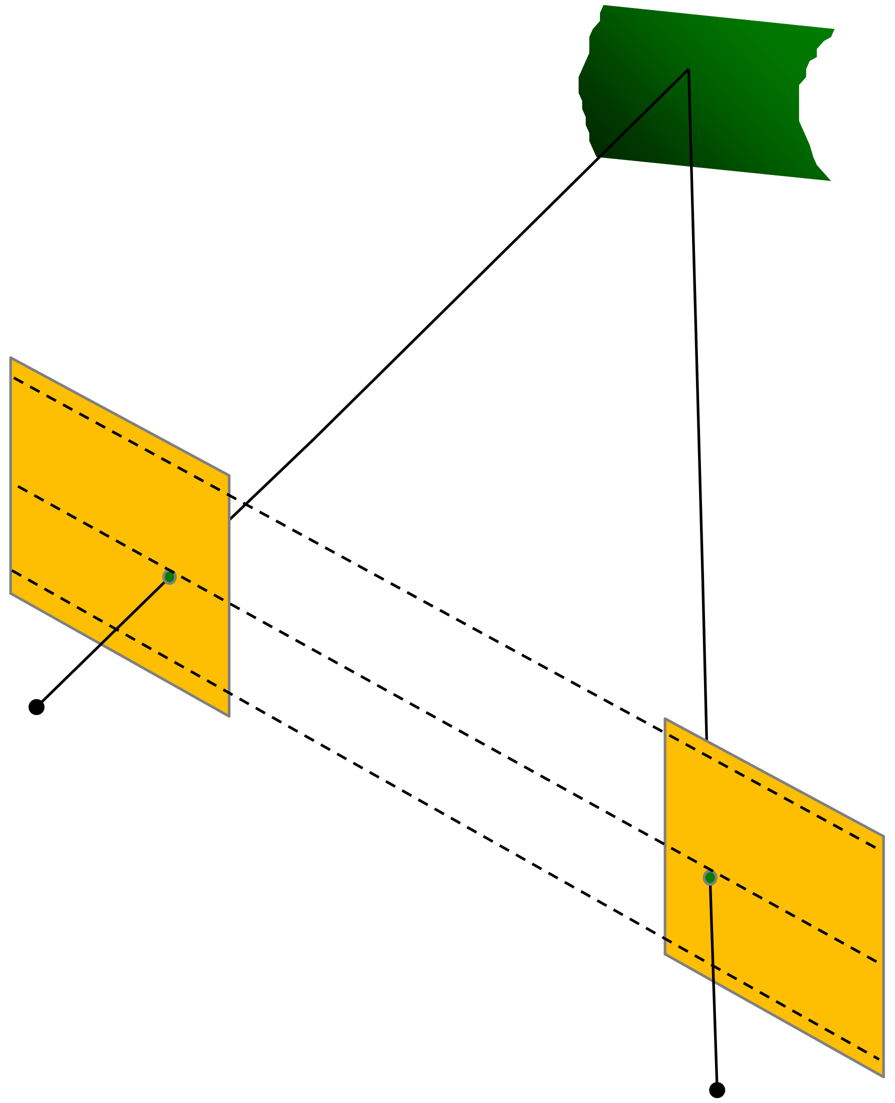
- Stereo Rectification:

1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R}_{rect}$
4. Scale both images by **H**

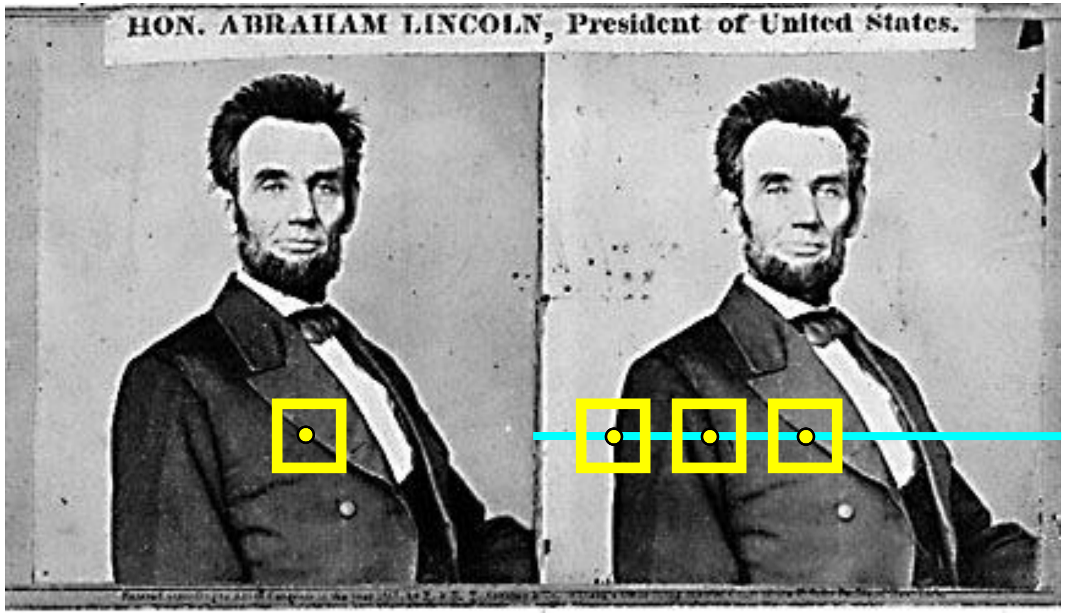


- Stereo Rectification:

1. Compute **E** to get **R**
2. Rotate right image by **R**
3. Rotate both images by $\mathbf{R_{rect}}$
4. Scale both images by **H**



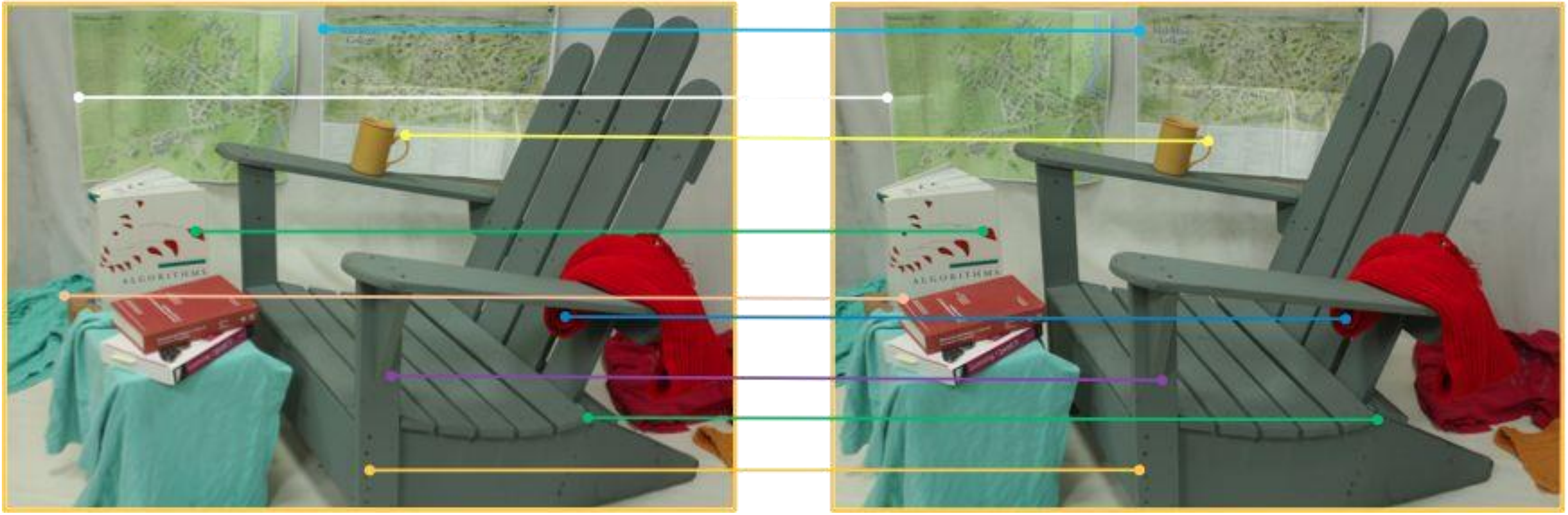
Overview of disparity (depth) estimation in stereo setup



1. Rectify images
   (make epipolar lines horizontal)
2. For each pixel
   - Find epipolar line
   - Scan line for best match *("correspondence problem")*
   - Compute depth from disparity ($Z = \frac{bf}{d}$)

Correspondence problem

- Finding homologous points is crucial (and challenging)
- Stereo pairs are typically rectified (homologous points into the same scanline)
- Once found corresponding points, depth is inferred by a simple triangulation



Slide credit: CVPR 2019 Tutorial: Learning-based depth estimation from stereo and monocular images: successes, limitations and future challenges

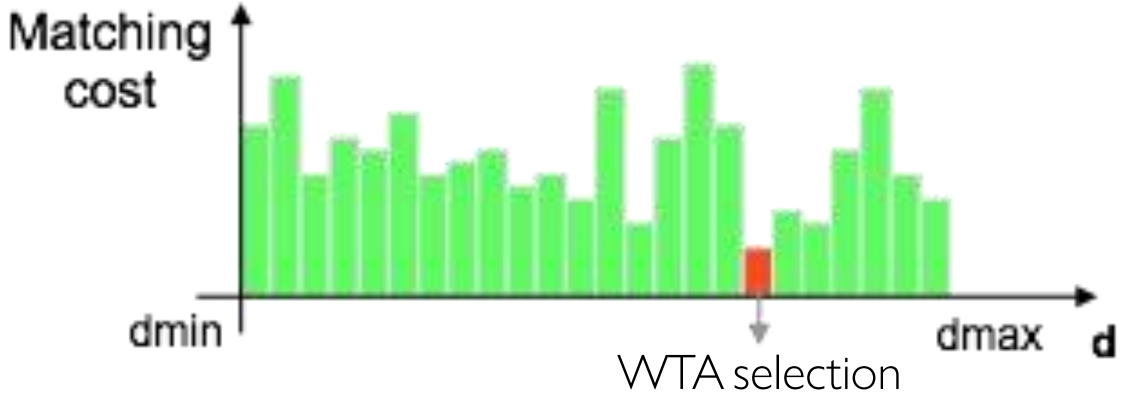
How to find homologous points?

- Looking for similar points/patches along scanlines
- Corresponding points are sought within a prefixed (disparity) range [dmin,dmax]

Slide credit: CVPR 2019 Tutorial: Learning-based depth estimation from stereo and monocular images: successes, limitations and future challenges

How to evaluate similarity between two points?

- Given a point p_R in the reference image, at each potential correspondence p_T in $[d_{min}, d_{max}]$ in the target image is associated a *score*
- Such score is referred to as ***matching cost*** $C(p_R, p_T, d)$, with d in $[d_{min}, d_{max}]$
- Pointwise matching cost (e.g., $|I(p_R) - I(p_T)|$)
- Patch based matching cost (e.g., average $|I(p_R) - I(p_T)|$ on a patch)

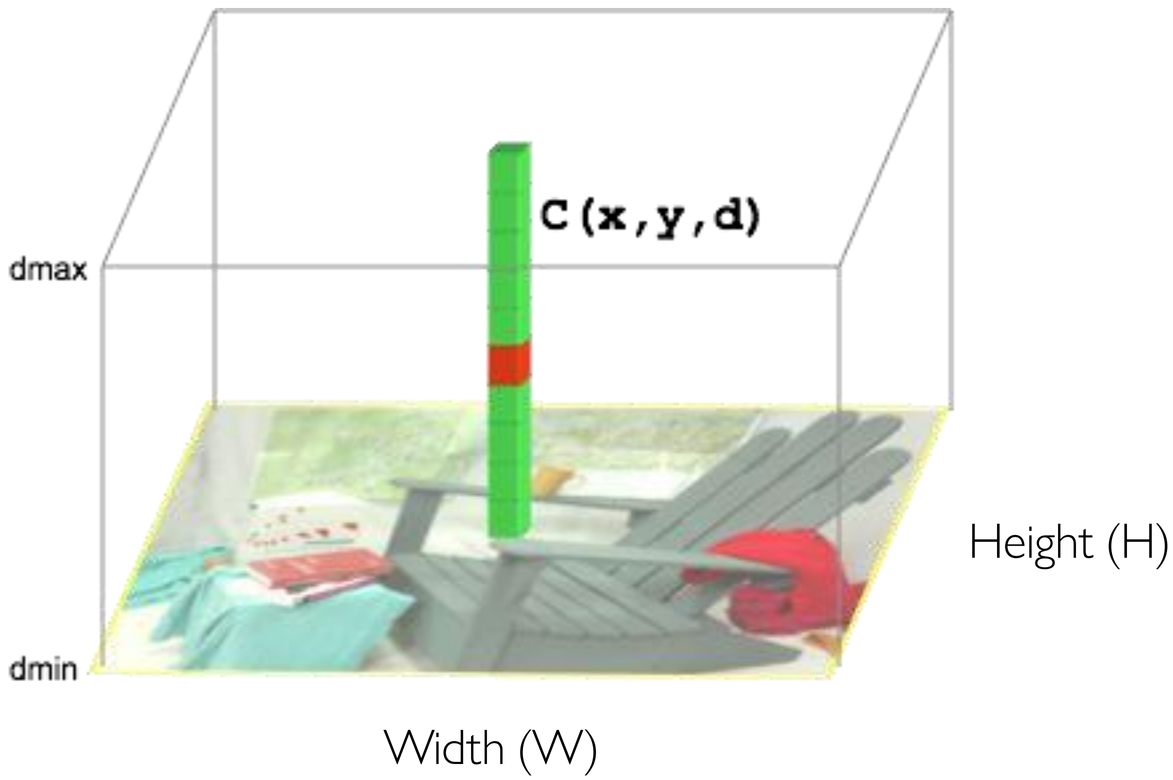


- Each p_R is assumed as uncorrelated to its neighbors
- Often, disparity selection consists in selecting the minimum score (WTA)

Slide credit: CVPR 2019 Tutorial: Learning-based depth estimation from stereo and monocular images: successes, limitations and future challenges

Cost volume or DSI (Disparity Space Image)

- The data structure containing all matching costs $C(p_R, p_T, d)$, with d in $[d_{min}, d_{max}]$



Slide credit: CVPR 2019 Tutorial: Learning-based depth estimation from stereo and monocular images: successes, limitations and future challenges

Summary: Traditional Stereo Matching

- Procedure of stereo matching [2]

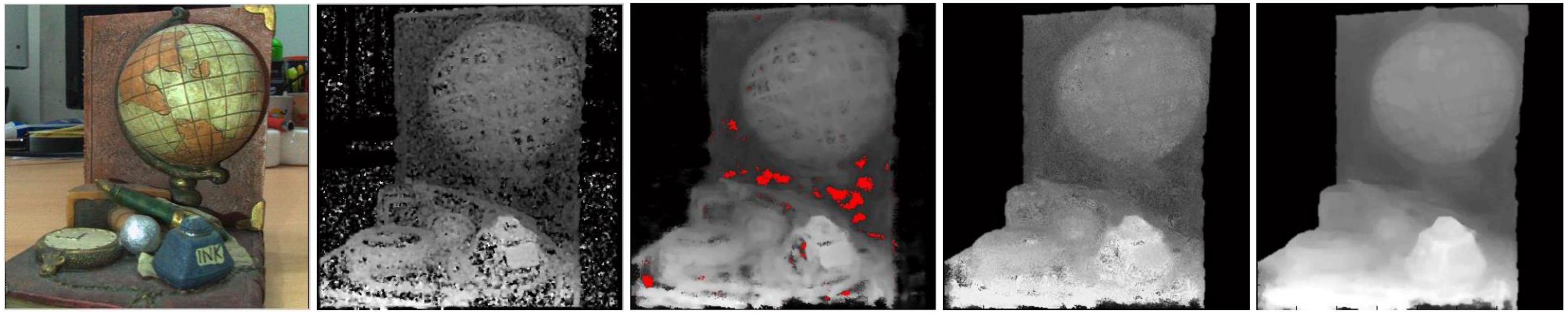
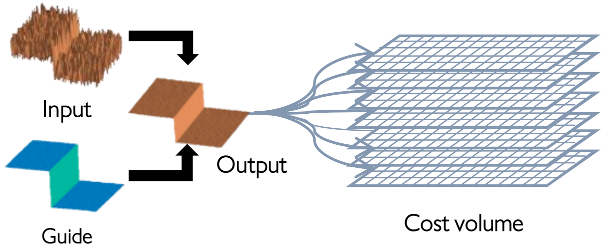


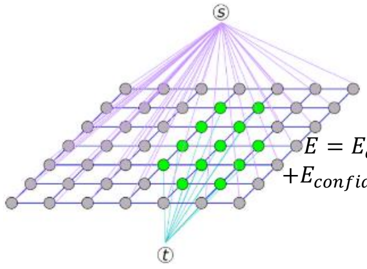
Image | Cost volume | Cost aggregation | Multi-label optimization | Refinement

Parametric – AD, SAD, BT, mean filter, Laplacian of Gaussian, Bilateral filtering, ZSAD, NCC, ZNCC
Nonparametric – Rank filter, Softrank filter, Census filter, Ordinal
Mutual Information – Hierarchical MI

Cost computation

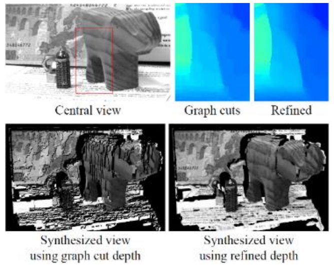


Cost aggregation



$$E = E_{data} + E_{smooth} + E_{confidnece} + E_{reliability}$$

Graph-cuts



$$l_r - \frac{C(l_+) + C(l_-)}{2(C(l_+ + C(l_-) - 2C(l_r)))}$$

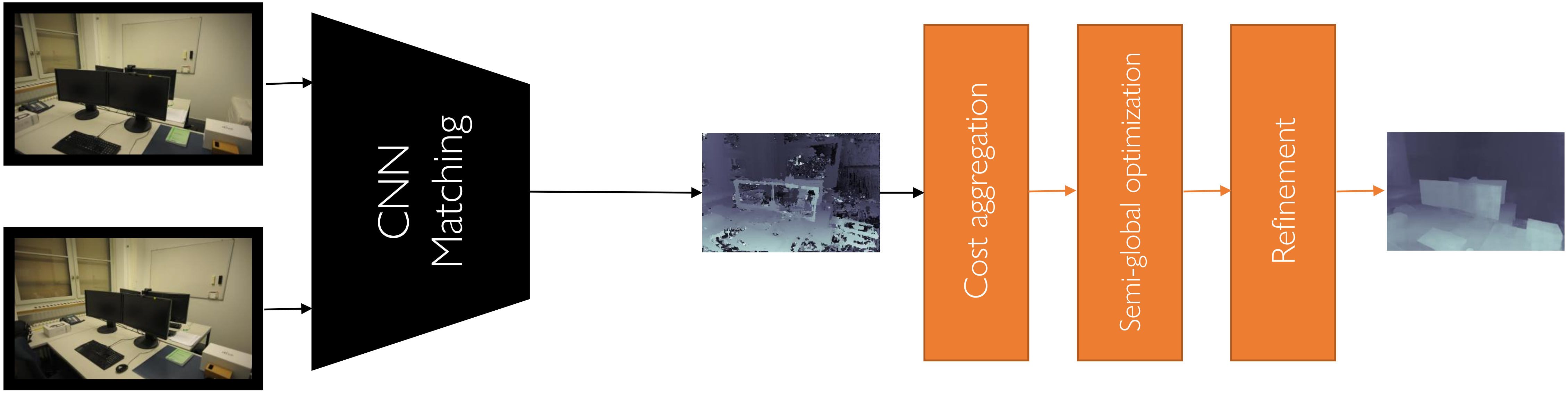
Iterative refinement

[1] Scharstein, Daniel, and Richard Szeliski. "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms." *International journal of computer vision* 47.1-3 (2002): 7-42.
[2] Jeon, Hae-Gon, et al. "Accurate depth map estimation from a lenslet light field camera." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

Deep Learning-based Stereo Matching

1st Generation of Learning-based Matching

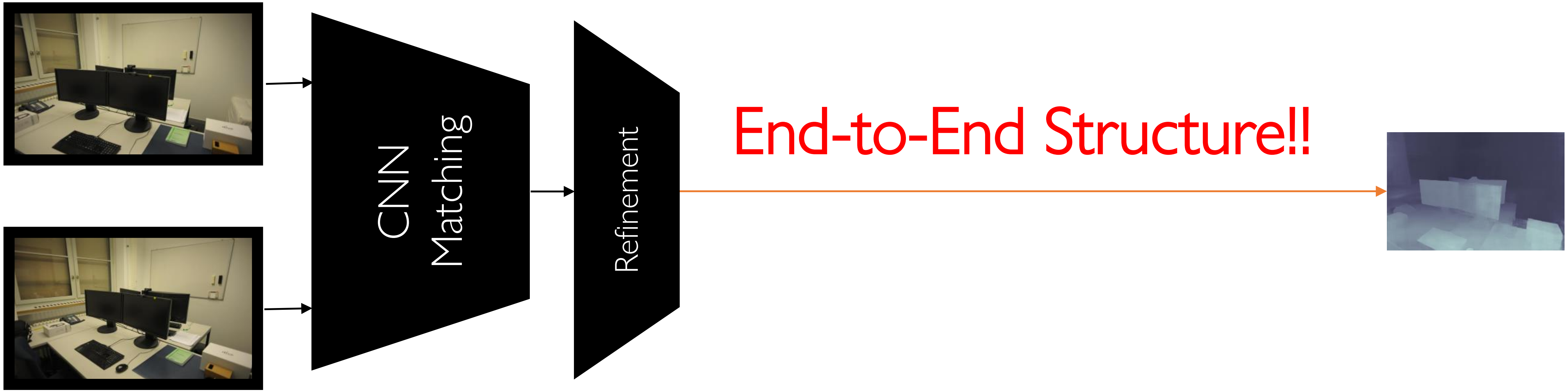
- The role of CNN: (1) Matching



CVPR15, MC-CNN, Zbontar *et al.*
ICCV15, Deep Embed, Chen *et al.*
CVPR16, Content-CNN, Lua *et al.*

2nd Generation of Learning-based Matching

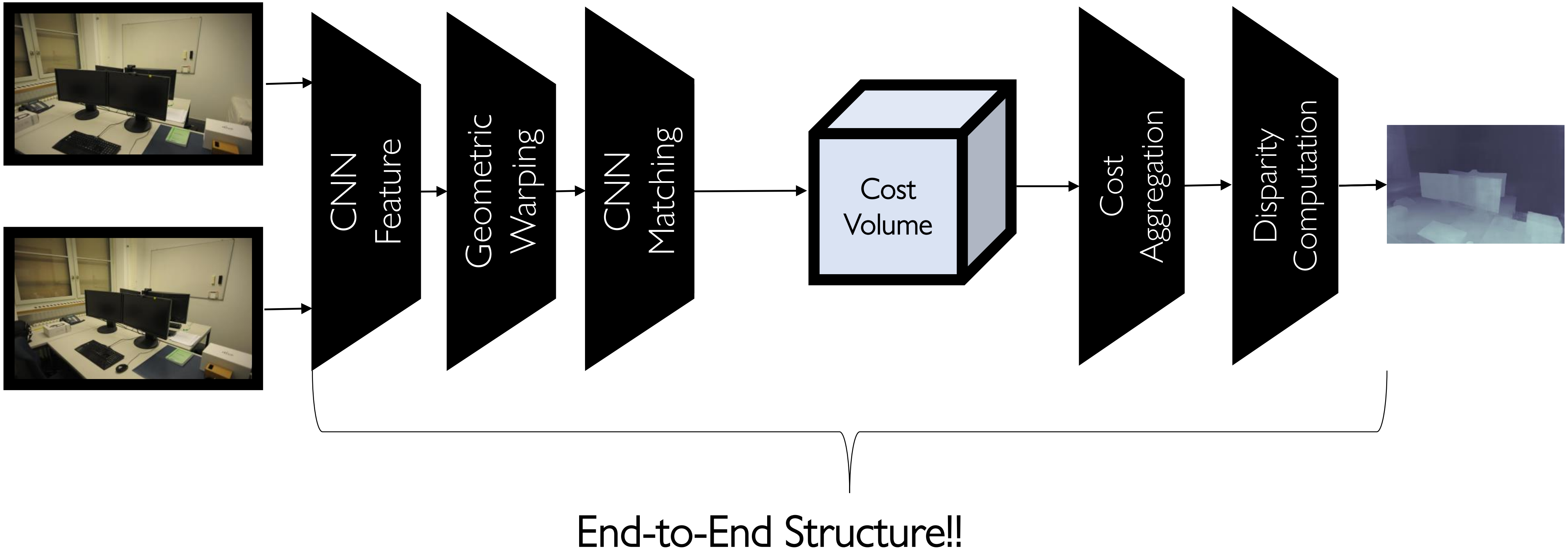
- The role of CNN: (1) Matching, (2) Refinement



CVPR16, DispNet, Mayer *et al.*
CVPR17, DeMoN, Ummenhofer *et al.*

3rd Generation of Learning-based Matching

- The role of CNN: (1) Geometry-inspired Matching, (2) Cost aggregation, (3) Disparity Computation, (4) Refinement

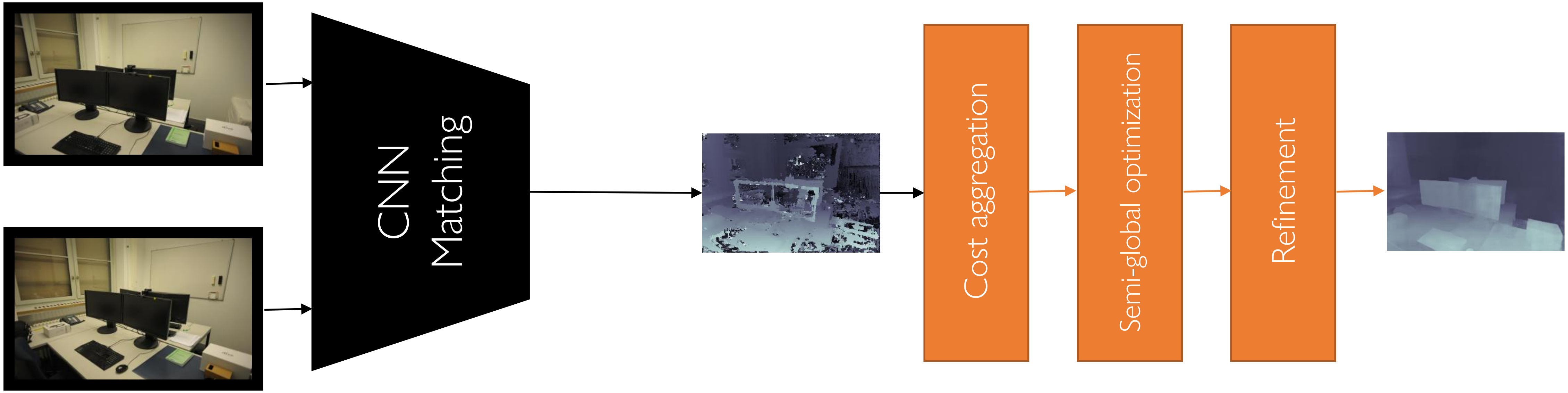


CVPR18, DeepMVS, Huang *et al.*
ECCV18, MVSNet, Yao *et al.*
ICLR19, DPSNet, Im *et al.*

1st Generation of Learning-based Matching

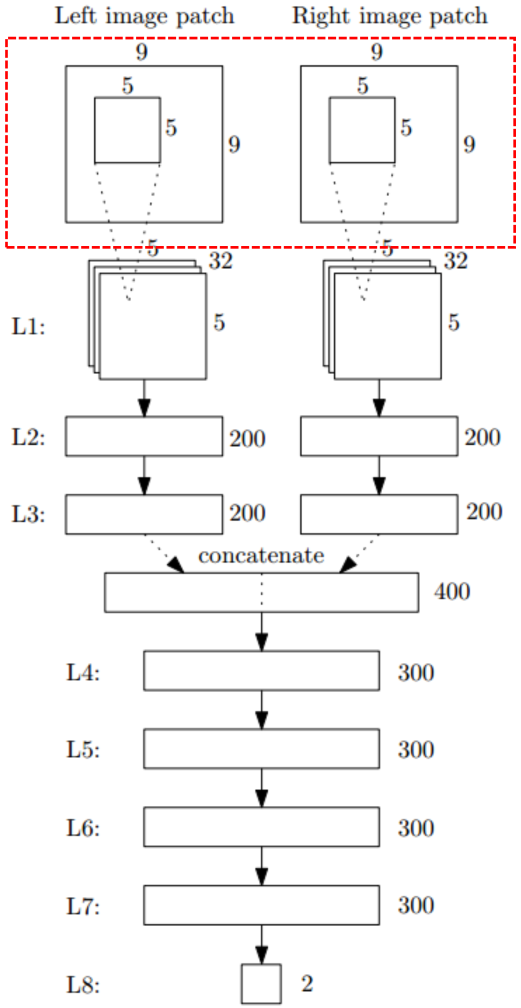
MC-CNN (Zbontar and LeCun, CVPR 15)

- Computing the Stereo Matching Cost with a Convolutional Neural Network
- The role of CNN: (1) Matching cost computation



CVPR15, MC-CNN, Zbontar *et al.*
ICCV15, Deep Embed, Chen *et al.*
CVPR16, Content-CNN, Lua *et al.*

MC-CNN (Zbontar and LeCun, CVPR 15)

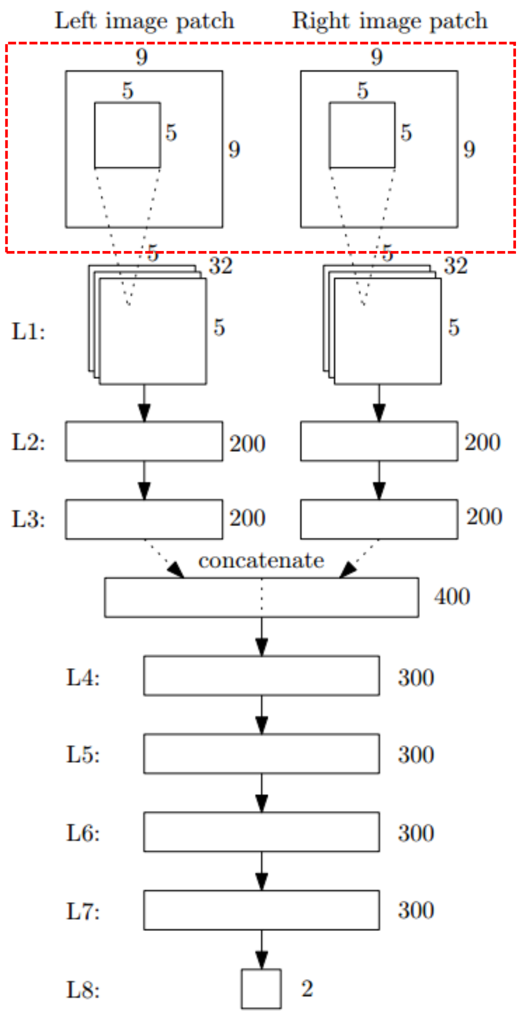


Dataset generation

L

R

MC-CNN (Zbontar and LeCun, CVPR 15)

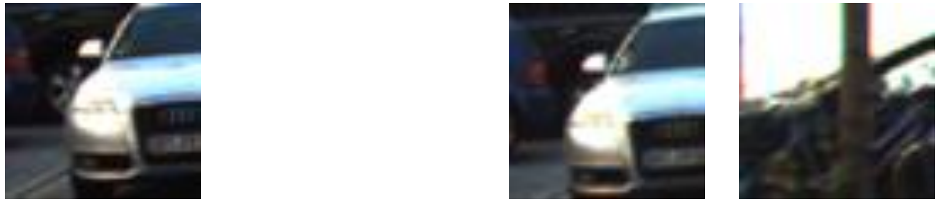


Dataset generation

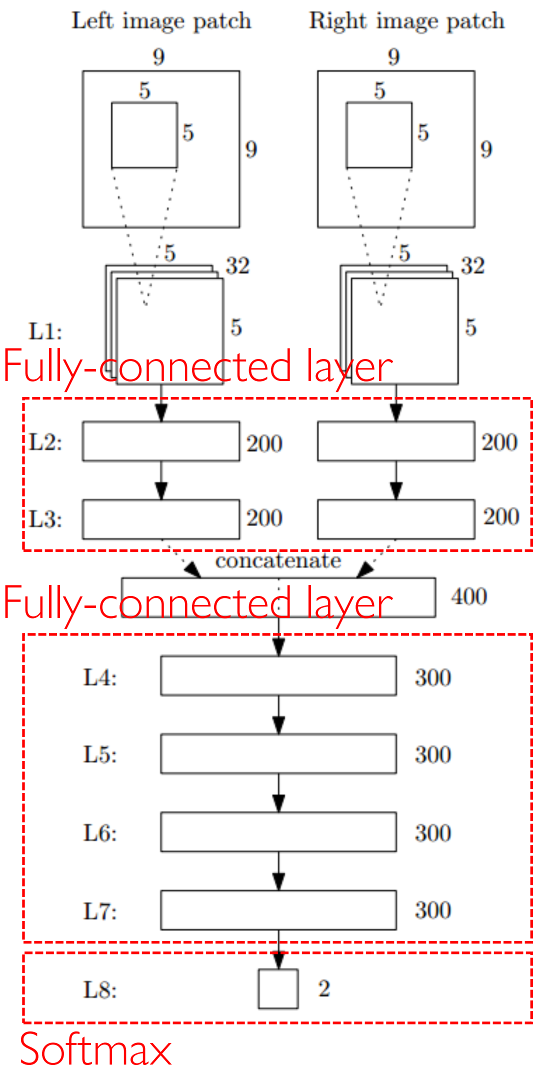
reference

Positive

Negative



MC-CNN (Zbontar and LeCun, CVPR 15)



- ReLU follow each layer
- 600 thousand parameters

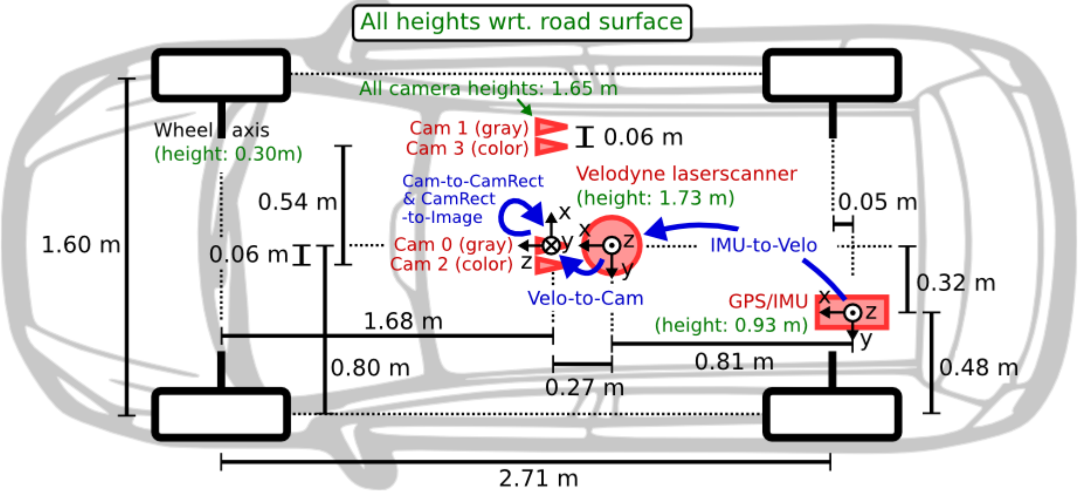
67 seconds

+

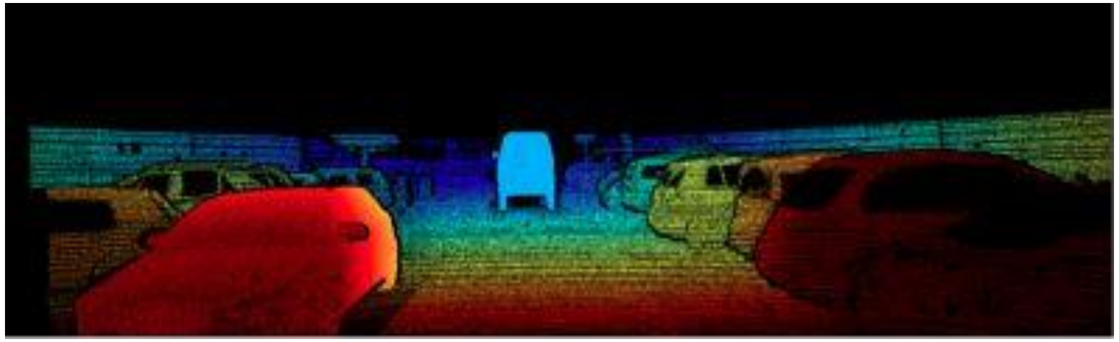
- cost aggregation
- semi-global optimization
- refinement

KITTI 2015 Benchmark

- **Sensor setup:** GPS/IMU, LiDAR, grayscale/color cameras



- **Various tasks for autonomous vehicles:**
  stereo, optical flow, scene flow, depth, odometry, object, tracking, semantics, etc.



MC-CNN (Zbontar and LeCun, CVPR 15)

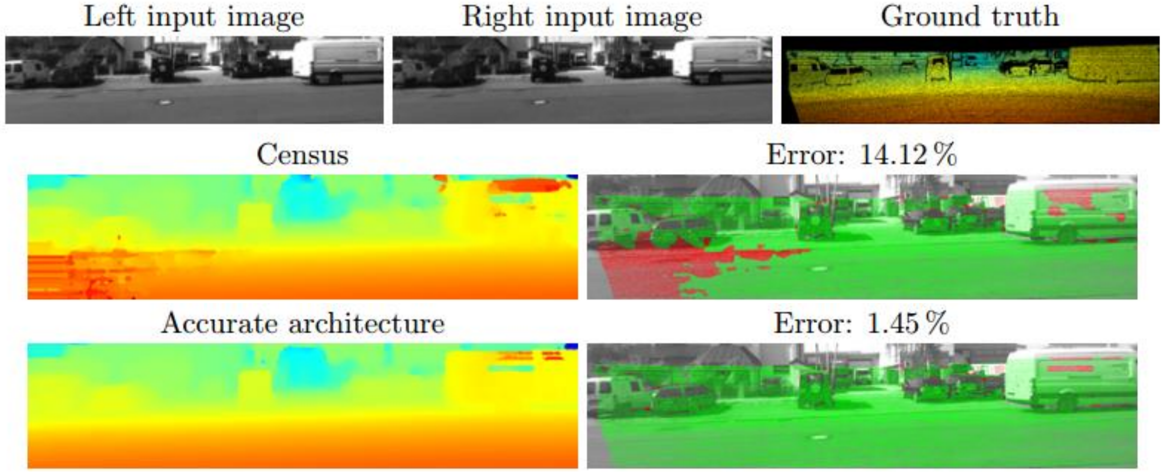
- KITTI 2015 Benchmark

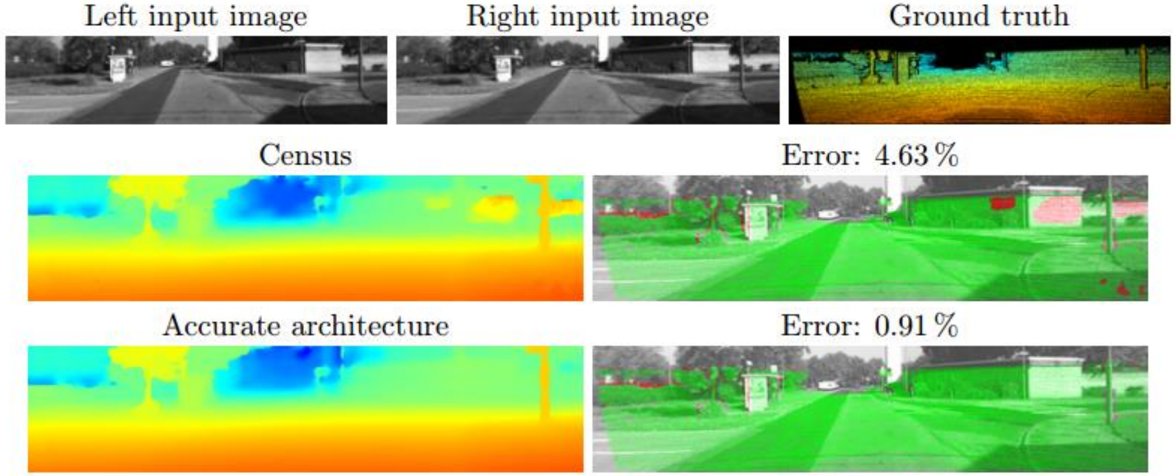
2019.03.25

	Method	Setting	Code	D1-bg	D1-fg	D1-all	Density	Runtime	Environment	Compare
111	MC-CNN-acrt		code	2.89 %	8.88 %	3.89 %	100.00 %	67 s	Nvidia GTX Titan X (CUDA, Lua/Torch7)	

J. Zbontar and Y. LeCun: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. Submitted to JMLR .

- Results comparison

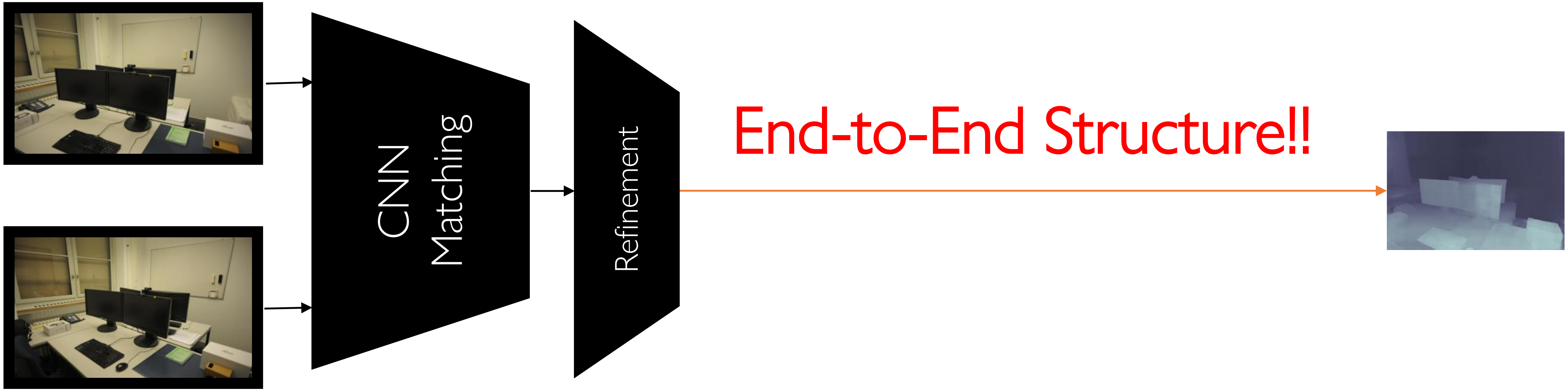




2nd Generation of Learning-based Matching

DispNet (Mayer *et al.* CVPR 16)

- A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation (CVPR16)
- The role of CNN: (1) Matching, (2) Refinement

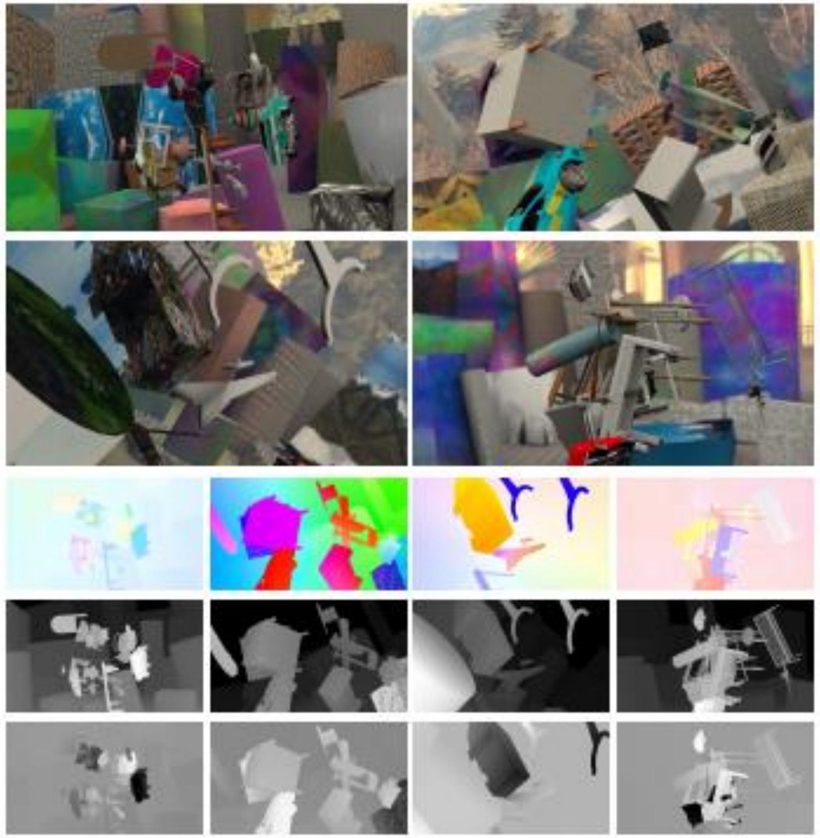


CVPR16, DispNet, Mayer *et al.*
CVPR17, DeMoN, Ummenhofer *et al.*

DispNet (Mayer *et al.* CVPR 16)

- New dataset ***"FlyingThings3D", "Monkaa", "Driving"***

KITTI 2015 *Driving* (ours)

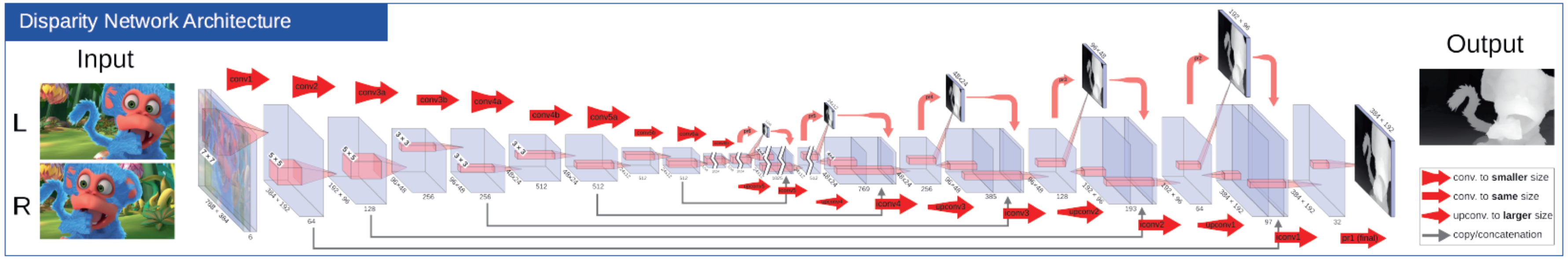


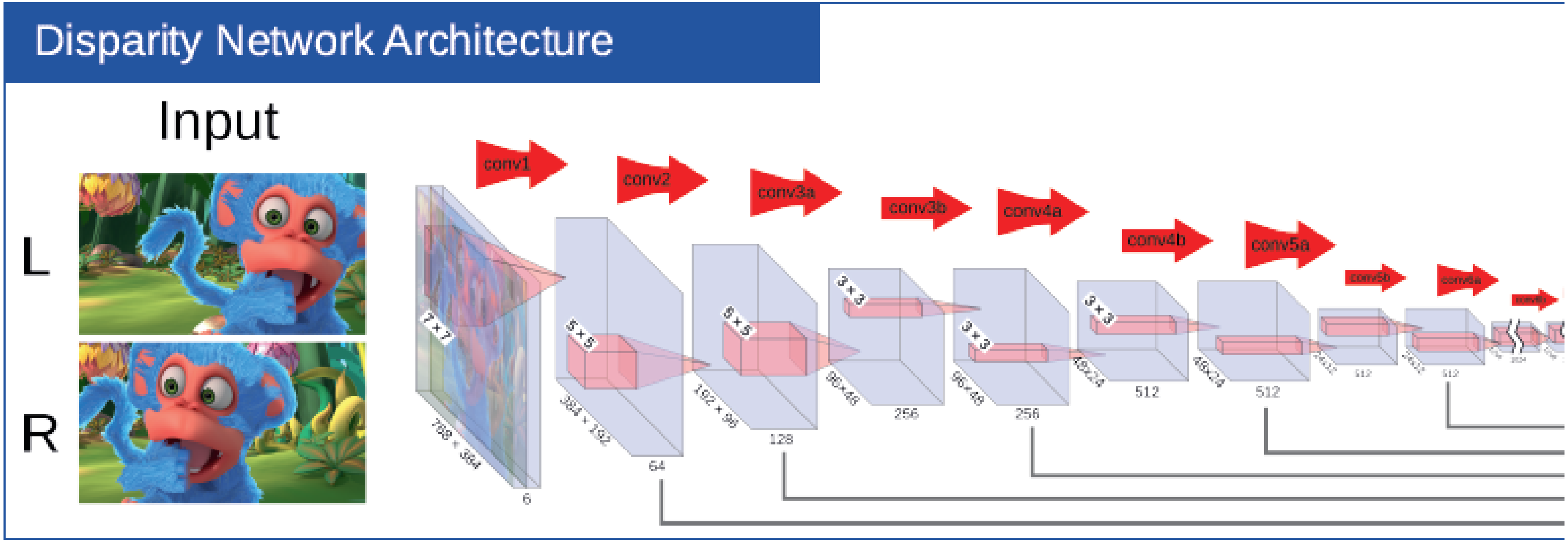
Name	Kernel	Str.	Ch I/O	InpRes	OutRes	Input
conv1	7×7	2	6/64	768×384	384×192	Images
conv2	5×5	2	64/128	384×192	192×96	conv1
conv3a	5×5	2	128/256	192×96	96×48	conv2
conv3b	3×3	1	256/256	96×48	96×48	conv3a
conv4a	3×3	2	256/512	96×48	48×24	conv3b
conv4b	3×3	1	512/512	48×24	48×24	conv4a
conv5a	3×3	2	512/512	48×24	24×12	conv4b
conv5b	3×3	1	512/512	24×12	24×12	conv5a
conv6a	3×3	2	512/1024	24×12	12×6	conv5b
conv6b	3×3	1	1024/1024	12×6	12×6	conv6a
pr6+loss6	3×3	1	1024/1	12×6	12×6	conv6b
upconv5	4×4	2	1024/512	12×6	24×12	conv6b
iconv5	3×3	1	1025/512	24×12	24×12	upconv5+pr6+conv5b
pr5+loss5	3×3	1	512/1	24×12	24×12	iconv5
upconv4	4×4	2	512/256	24×12	48×24	iconv5
iconv4	3×3	1	769/256	48×24	48×24	upconv4+pr5+conv4b
pr4+loss4	3×3	1	256/1	48×24	48×24	iconv4
upconv3	4×4	2	256/128	48×24	96×48	iconv4
iconv3	3×3	1	385/128	96×48	96×48	upconv3+pr4+conv3b
pr3+loss3	3×3	1	128/1	96×48	96×48	iconv3
upconv2	4×4	2	128/64	96×48	192×96	iconv3
iconv2	3×3	1	193/64	192×96	192×96	upconv2+pr3+conv2
pr2+loss2	3×3	1	64/1	192×96	192×96	iconv2
upconv1	4×4	2	64/32	192×96	384×192	iconv2
iconv1	3×3	1	97/32	384×192	384×192	upconv1+pr2+conv1
pr1+loss1	3×3	1	32/1	384×192	384×192	iconv1

DispNet (Mayer *et al.* CVPR 16)

- Network details

Name	Kernel	Str.	Ch I/O	InpRes	OutRes	Input
conv1	7×7	2	6/64	768×384	384×192	Images
conv2	5×5	2	64/128	384×192	192×96	conv1
conv3a	5×5	2	128/256	192×96	96×48	conv2
conv3b	3×3	1	256/256	96×48	96×48	conv3a
conv4a	3×3	2	256/512	96×48	48×24	conv3b
conv4b	3×3	1	512/512	48×24	48×24	conv4a
conv5a	3×3	2	512/512	48×24	24×12	conv4b
conv5b	3×3	1	512/512	24×12	24×12	conv5a
conv6a	3×3	2	512/1024	24×12	12×6	conv5b
conv6b	3×3	1	1024/1024	12×6	12×6	conv6a
pr6+loss6	3×3	1	1024/1	12×6	12×6	conv6b
upconv5	4×4	2	1024/512	12×6	24×12	conv6b
iconv5	3×3	1	1025/512	24×12	24×12	upconv5+pr6+conv5b
pr5+loss5	3×3	1	512/1	24×12	24×12	iconv5
upconv4	4×4	2	512/256	24×12	48×24	iconv5
iconv4	3×3	1	769/256	48×24	48×24	upconv4+pr5+conv4b
pr4+loss4	3×3	1	256/1	48×24	48×24	iconv4
upconv3	4×4	2	256/128	48×24	96×48	iconv4
iconv3	3×3	1	385/128	96×48	96×48	upconv3+pr4+conv3b
pr3+loss3	3×3	1	128/1	96×48	96×48	iconv3
upconv2	4×4	2	128/64	96×48	192×96	iconv3
iconv2	3×3	1	193/64	192×96	192×96	upconv2+pr3+conv2
pr2+loss2	3×3	1	64/1	192×96	192×96	iconv2
upconv1	4×4	2	64/32	192×96	384×192	iconv2
iconv1	3×3	1	97/32	384×192	384×192	upconv1+pr2+conv1
pr1+loss1	3×3	1	32/1	384×192	384×192	iconv1

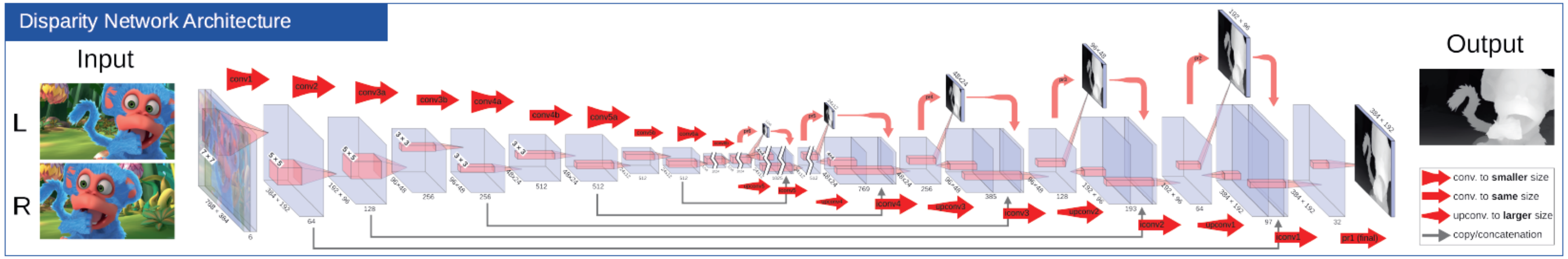


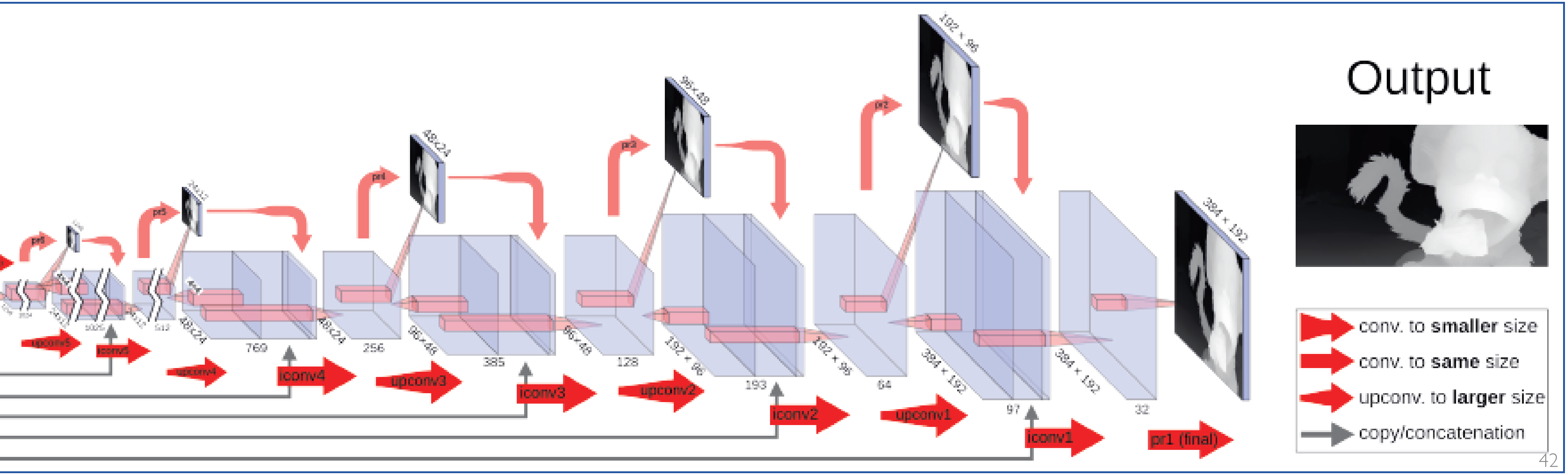


DispNet (Mayer *et al.* CVPR 16)

- Network details

Name	Kernel	Str.	Ch I/O	InpRes	OutRes	Input
conv1	7×7	2	6/64	768×384	384×192	Images
conv2	5×5	2	64/128	384×192	192×96	conv1
conv3a	5×5	2	128/256	192×96	96×48	conv2
conv3b	3×3	1	256/256	96×48	96×48	conv3a
conv4a	3×3	2	256/512	96×48	48×24	conv3b
conv4b	3×3	1	512/512	48×24	48×24	conv4a
conv5a	3×3	2	512/512	48×24	24×12	conv4b
conv5b	3×3	1	512/512	24×12	24×12	conv5a
conv6a	3×3	2	512/1024	24×12	12×6	conv5b
conv6b	3×3	1	1024/1024	12×6	12×6	conv6a
pr6+loss6	3×3	1	1024/1	12×6	12×6	conv6b
upconv5	4×4	2	1024/512	12×6	24×12	conv6b
iconv5	3×3	1	1025/512	24×12	24×12	upconv5+pr6+conv5b
pr5+loss5	3×3	1	512/1	24×12	24×12	iconv5
upconv4	4×4	2	512/256	24×12	48×24	iconv5
iconv4	3×3	1	769/256	48×24	48×24	upconv4+pr5+conv4b
pr4+loss4	3×3	1	256/1	48×24	48×24	iconv4
upconv3	4×4	2	256/128	48×24	96×48	iconv4
iconv3	3×3	1	385/128	96×48	96×48	upconv3+pr4+conv3b
pr3+loss3	3×3	1	128/1	96×48	96×48	iconv3
upconv2	4×4	2	128/64	96×48	192×96	iconv3
iconv2	3×3	1	193/64	192×96	192×96	upconv2+pr3+conv2
pr2+loss2	3×3	1	64/1	192×96	192×96	iconv2
upconv1	4×4	2	64/32	192×96	384×192	iconv2
iconv1	3×3	1	97/32	384×192	384×192	upconv1+pr2+conv1
pr1+loss1	3×3	1	32/1	384×192	384×192	iconv1





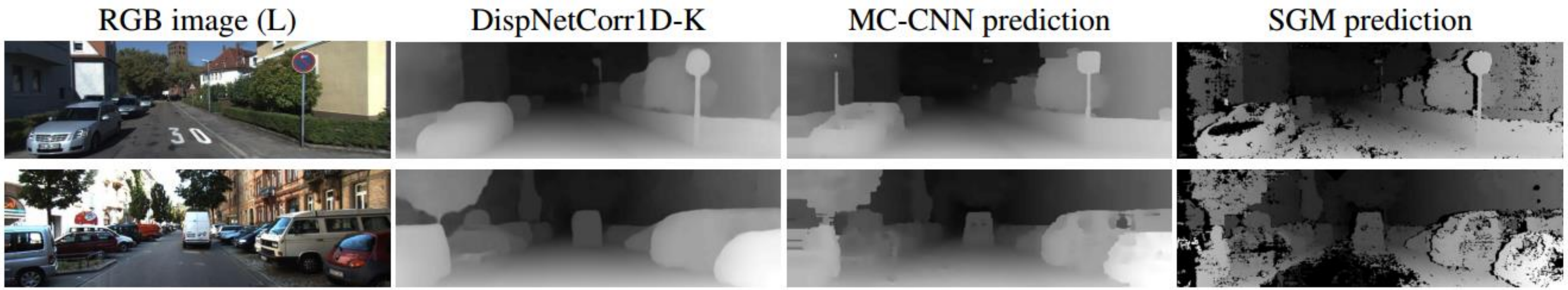
DispNet (Mayer *et al.* CVPR 16)

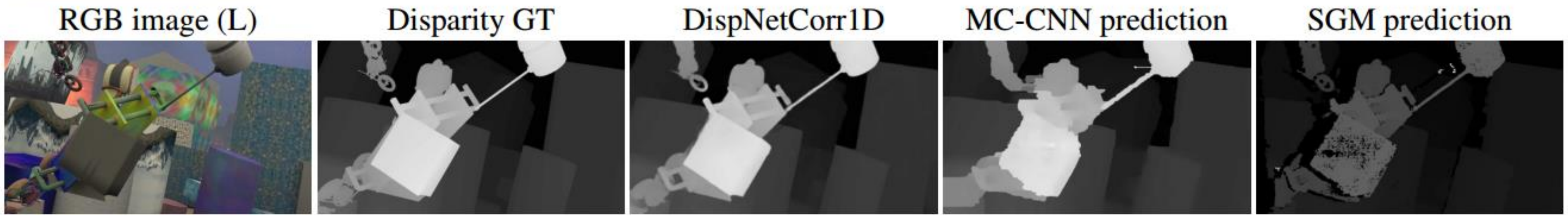
- KITTI 2015 Benchmark

2019.03.25

	Method	Setting	Code	D1-bg	D1-fg	D1-all	Density	Runtime	Environment	Compare
111	MC-CNN-acrt		code	2.89 %	8.88 %	3.89 %	100.00 %	67 s	Nvidia GTX Titan X (CUDA, Lua/Torch7)	
J. Zbontar and Y. LeCun: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. Submitted to JMLR .										
116	DispNetC		code	4.32 %	4.41 %	4.34 %	100.00 %	0.06 s	Nvidia GTX Titan X (Caffe)	
N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy and T. Brox: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. CVPR 2016.										
121	Content-CNN			3.73 %	8.58 %	4.54 %	100.00 %	1 s	Nvidia GTX Titan X (Torch)	
W. Luo, A. Schwing and R. Urtasun: Efficient Deep Learning for Stereo Matching. CVPR 2016.										

- Results comparison



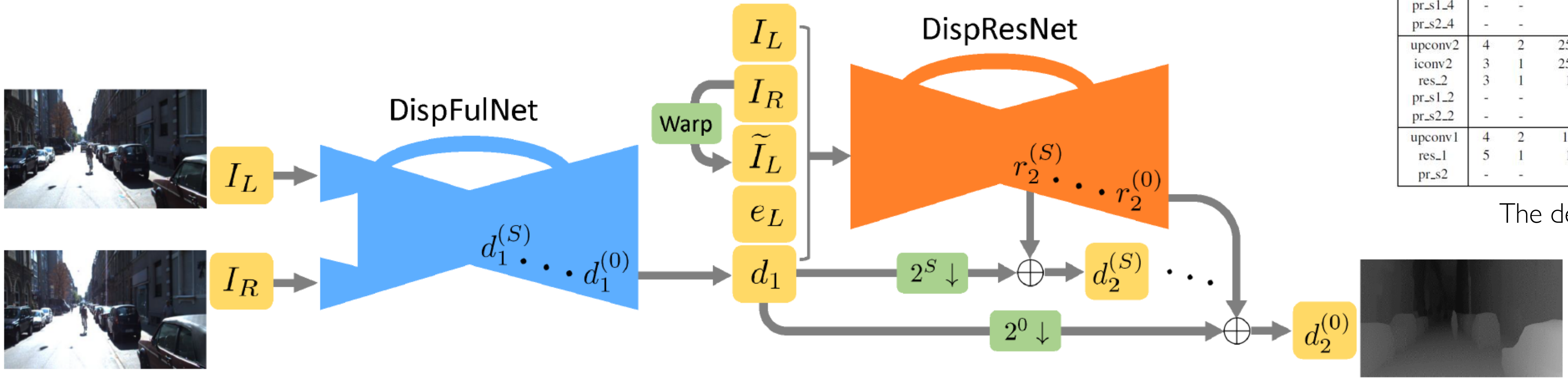


CRL (Pang *et al.* ICCVW17)

- Cascaded Residual Learning: A Two-stage Convolutional Neural Network for Stereo Matching
- Idea: Design two network:
  - **Network 1:** Initial depth estimation network (Dispnet with extra up-convolution modules)
  - **Network 2:** Refinement network (DispResNet)

Layer	K	S	Channels	I	O	Input Channels
conv1	5	1	13/64	1	1	left+right+left_s+err+pr_s1
conv2	5	2	64/128	1	2	conv1
conv2_1	3	1	128/128	2	2	conv2
conv3	3	2	128/256	2	4	conv_3_1
conv3_1	3	1	256/256	4	4	conv3
conv4	3	2	256/512	4	8	conv3_1
conv4_1	3	1	512/512	8	8	conv4
conv5	3	2	512/1024	8	16	conv4_1
conv5_1	3	1	1024/1024	16	16	conv5
res_16	3	1	1024/1	16	16	conv5_1
pr_s1_16	-	-	1/1	1	16	pr_s1
pr_s2_16	-	-	1/1	16	16	pr_s1_16+res_16
upconv4	4	2	1024/512	16	8	conv5_1
iconv4	3	1	1025/512	8	8	upconv4+conv4_1+pr_s2_16
res_8	3	1	512/1	8	8	iconv4
pr_s1_8	-	-	1/1	1	8	pr_s1
pr_s2_8	-	-	1/1	8	8	pr_s1_8+res_8
upconv3	4	2	512/256	8	4	iconv4
iconv3	3	1	513/256	4	4	upconv3+conv3_1+pr_s2_8
res_4	3	1	256/1	4	4	iconv3
pr_s1_4	-	-	1/1	1	4	pr_s1
pr_s2_4	-	-	1/1	4	4	pr_s1_4+res_4
upconv2	4	2	256/128	4	2	iconv3
iconv2	3	1	257/128	2	2	upconv2+conv2_1+pr_s2_4
res_2	3	1	128/1	2	2	iconv2
pr_s1_2	-	-	1/1	1	2	pr_s1
pr_s2_2	-	-	1/1	2	2	pr_s1_2+res_2
upconv1	4	2	128/64	2	1	iconv2
res_1	5	1	129/1	1	1	upconv1+conv1+pr_s2_2
pr_s2	-	-	1/1	1	1	pr_s1+res_1

The detailed architecture

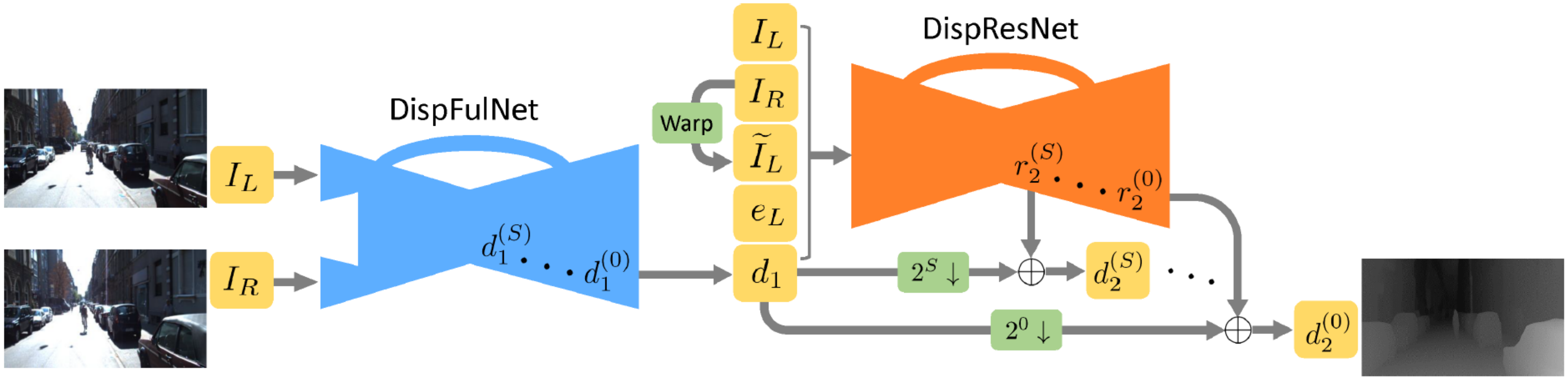


CRL (Pang *et al.* ICCVW17)

1. DispFulNet
   (1) Input: Stereo images I_L, I_R
   (2) Outputs: Initial Disparity d_1 + Warped Right image $\tilde{I}_L$, error e_L

$$\tilde{I}_L(x, y) = I_L(x + d_1(x, y), y)$$
$$e_L = |I_L - \tilde{I}_L(x, y)|$$

2. DispResNet
   (1) Input: Stereo images I_L, I_R, Initial Disparity d_1 + Warped Right image $\tilde{I}_L$, error e_L
   (2) Outputs: Residual Disparity d_2



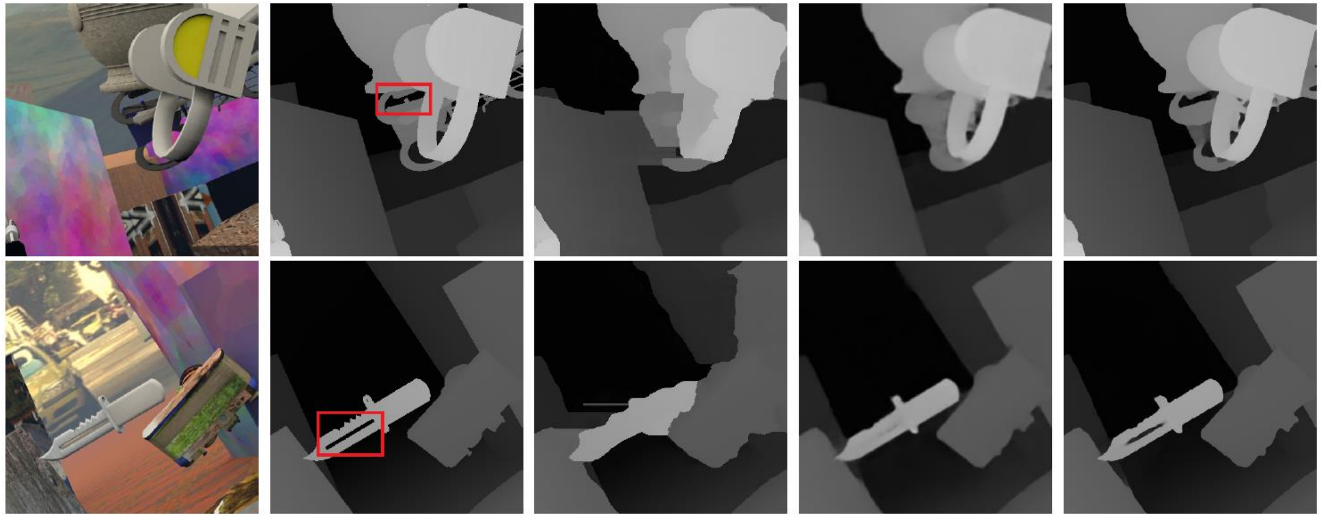
CRL (Pang *et al.* ICCVW17)

- KITTI 2015 Benchmark

2019.03.25

	Method	Setting	Code	D1-bg	D1-fg	D1-all	Density	Runtime	Environment	Compare
71	CRL		code	2.48 %	3.59 %	2.67 %	100.00 %	0.47 s	Nvidia GTX 1080	
J. Pang, W. Sun, J. Ren, C. Yang and Q. Yan: Cascade residual learning: A two-stage convolutional neural network for stereo matching. ICCV Workshop on Geometry Meets Deep Learning 2017.										
111	MC-CNN-acrt		code	2.89 %	8.88 %	3.89 %	100.00 %	67 s	Nvidia GTX Titan X (CUDA, Lua/Torch7)	
J. Zbontar and Y. LeCun: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. Submitted to JMLR .										
116	DispNetC		code	4.32 %	4.41 %	4.34 %	100.00 %	0.06 s	Nvidia GTX Titan X (Caffe)	
N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy and T. Brox: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. CVPR 2016.										
121	Content-CNN			3.73 %	8.58 %	4.54 %	100.00 %	1 s	Nvidia GTX Titan X (Torch)	
W. Luo, A. Schwing and R. Urtasun: Efficient Deep Learning for Stereo Matching. CVPR 2016.										

- Results comparison

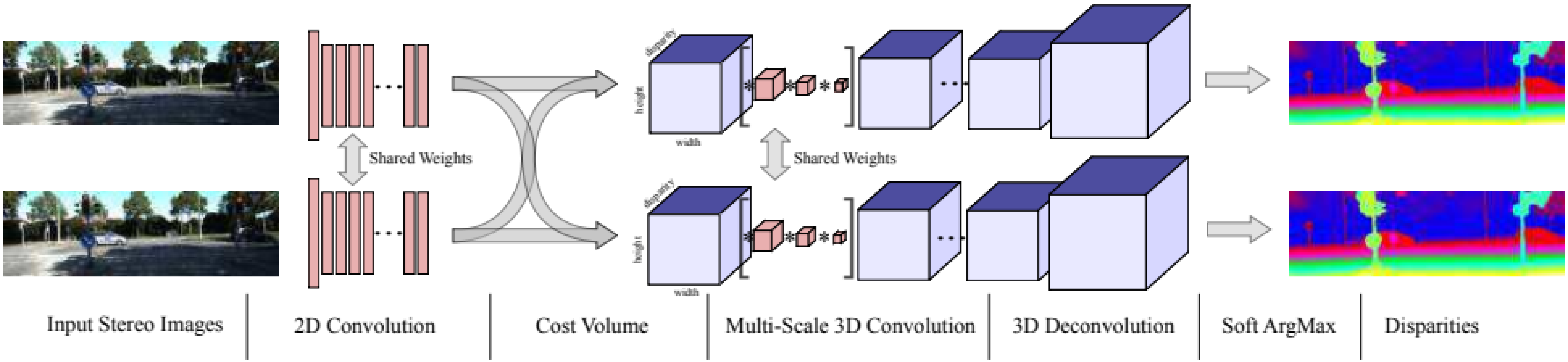


Left image | Ground truth disparity | MC-CNN-fst | DispNetC | CRL

3rd Generation of Learning-based Matching

GCNet (Kendall *et al.* ICCV17)

- End-to-End Learning of Geometry and Context for Deep Stereo Regression
- First learning-based approach
    - Cost volume generation
    - WTA strategy using softmax



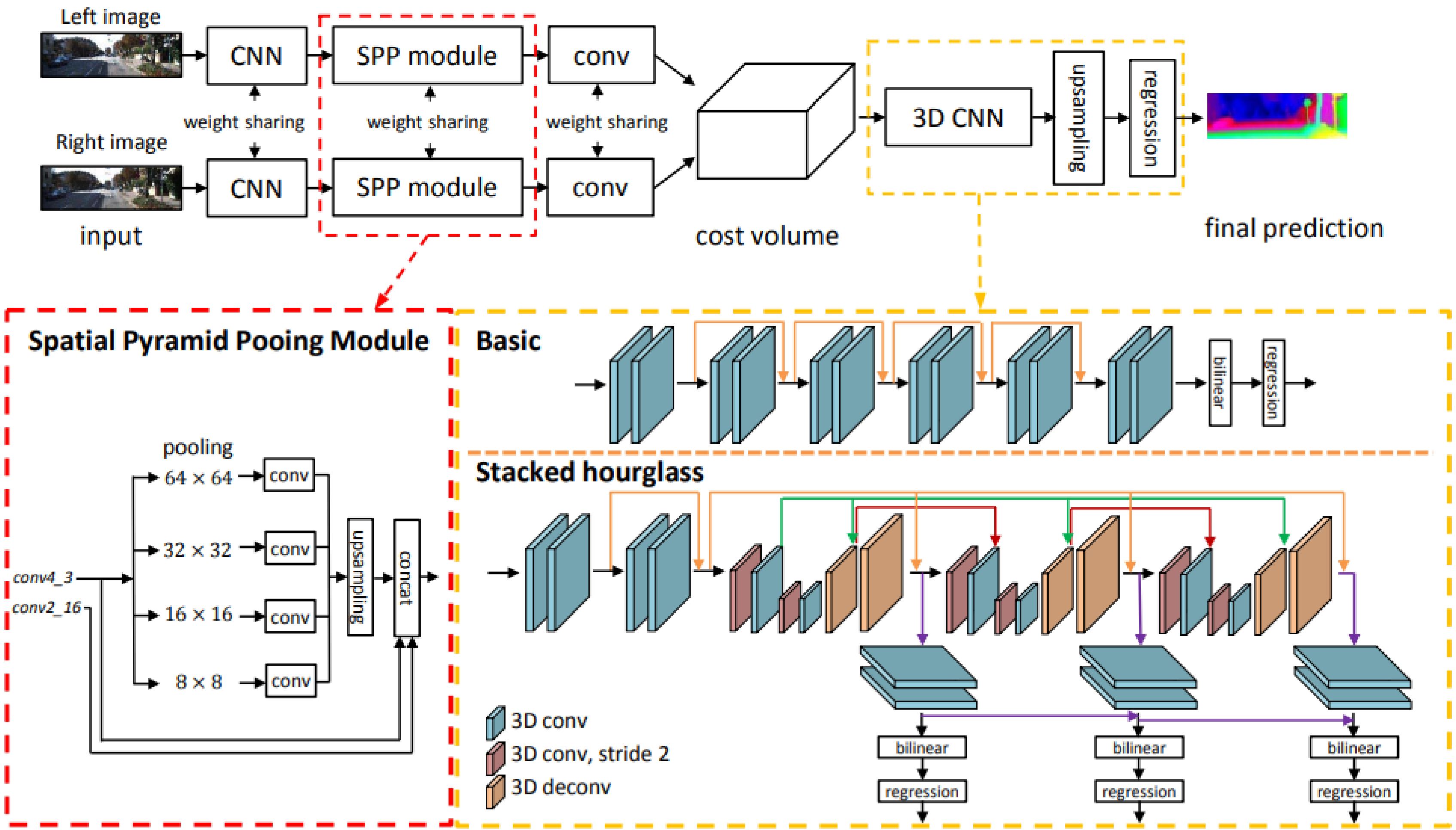
PSMNet (Chang *et al.* CVPR 18)

(1) SPP module

(2) Cost volume
(Concatenate left-right features across each disparity level)

(3) 3D CNN
(Basic vs Stacked hourglass)

(4) Regression
(SoftMax & weighted sum)



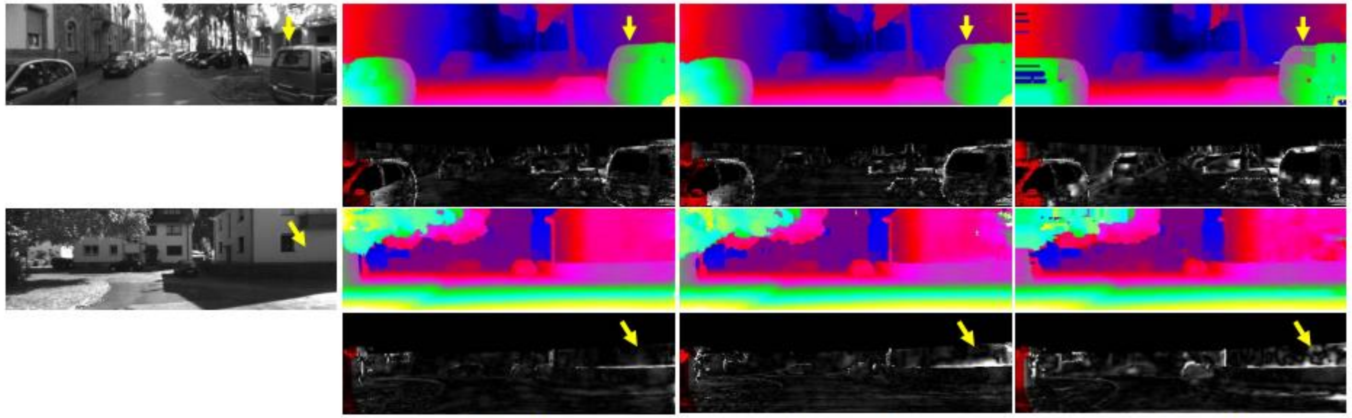
PSMNet (Chang *et al.* CVPR 18)

- KITTI 2015 Benchmark

2019.03.25

	Method	Setting	Code	D1-bg	D1-fg	D1-all	Density	Runtime	Environment	Compare
11	PSMNet_R			1.62 %	3.79 %	1.98 %	100.00 %	0.5 s	GPU @ 2.5 Ghz (Python)	
	iResNet-i2e2			2.10 %	3.64 %	2.36 %	100.00 %	0.25 s	Nvidia Titan X (Pascal)	
J. Pang Z. Liang, Y. Feng, Y. Guo and H. Liu: Learning Deep Correspondence through Prior and Posterior Feature Constancy. arXiv preprint arXiv:1712.01039 2017.										
71	CRL		code	2.48 %	3.59 %	2.67 %	100.00 %	0.47 s	Nvidia GTX 1080	
J. Pang, W. Sun, J. Ren, C. Yang and Q. Yan: Cascade residual learning: A two-stage convolutional neural network for stereo matching. ICCV Workshop on Geometry Meets Deep Learning 2017.										
111	MC-CNN-acrt		code	2.89 %	8.88 %	3.89 %	100.00 %	67 s	Nvidia GTX Titan X (CUDA, Lua/Torch7)	
J. Zbontar and Y. LeCun: Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches. Submitted to JMLR .										
116	DispNetC		code	4.32 %	4.41 %	4.34 %	100.00 %	0.06 s	Nvidia GTX Titan X (Caffe)	
N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy and T. Brox: A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation. CVPR 2016.										
121	Content-CNN			3.73 %	8.58 %	4.54 %	100.00 %	1 s	Nvidia GTX Titan X (Torch)	
W. Luo, A. Schwing and R. Urtasun: Efficient Deep Learning for Stereo Matching. CVPR 2016.										

- Results comparison



(a) PSMNet (b) GC-Net (c) MC-CNN