Understanding Real World Indoor Scenes: Geometry and Semantics

Ankur Handa

University of Cambridge and Imperial College London

October 16, 2016

Outline

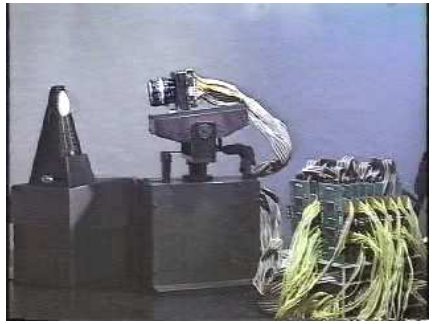
- Background
- SceneNet: Repository of Labelled Synthetic Indoor Scenes
- Results on Semantic Segmentation on Real World Data
- gvnn: Neural Network Library for Geometric Vision
- Future Ideas

Outline

- Background
- SceneNet: Repository of Labelled Synthetic Indoor Scenes
- Results on Semantic Segmentation on Real World Data
- gvnn: Neural Network Library for Geometric Vision
- Future Ideas

Real-time tracking



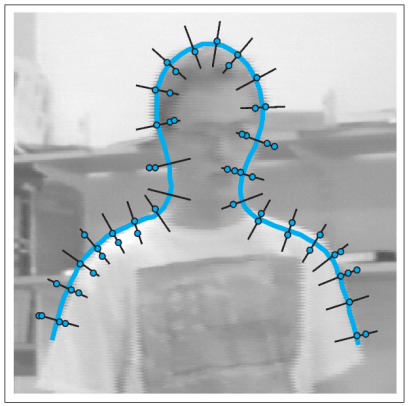
Real-time tracking

- High frame-rate seems better but.. today most advanced real-time tracking is at 10–60Hz.
- Why? Should we increase the frame-rate in real, modern advanced tracking problems?

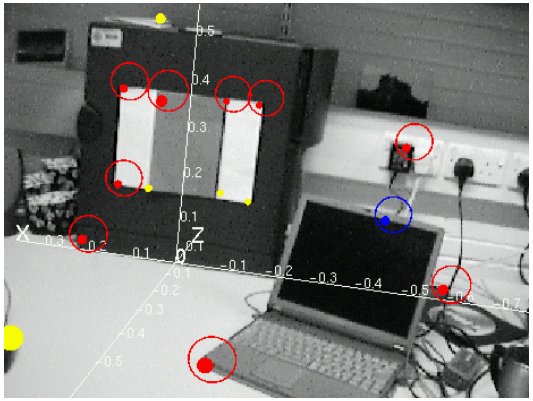
Outline

- We are going to experimentally investigate this using photo-realistic synthetic videos.
- We are going to consider a particular problem of camera tracking against a known 3D rigid model.

Tracking



Isard and Blake, IJCV 1998

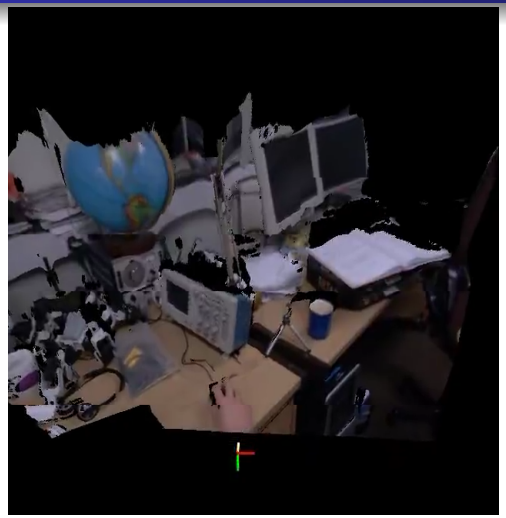


Davison, ICCV 2003

What is tracking?

- Visual tracking means estimating model parameters sequentially from video.
- Tracking is different from "detection" independently applied to each frame since we can benefit from prediction — which gets better as frame-rate increases.
- With prediction the search for correspondence can be local or guided — prediction from the previous frame can reduce the search for correspondences in the next frame.

3D tracking by whole image alignment



DTAM, 2011

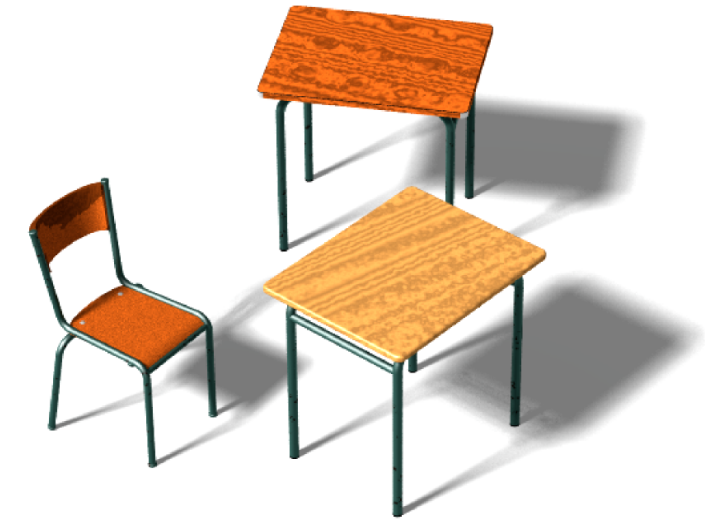
KinectFusion, 2011

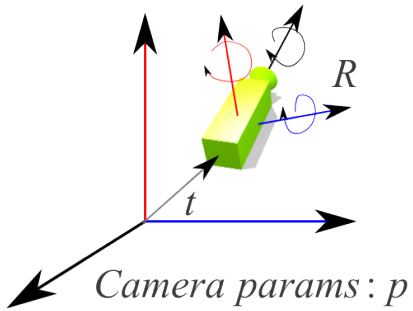
Audras et al., 2011

Dense Tracking

- We focus on camera tracking in mostly rigid scenes.
- Dense alignment has active search embedded in it — the basin of convergence is directly related to the baseline joining the images. As the baseline decreases convergence is quicker.
- Previous frame estimates for initialising the optimisation in current frame. The higher the frame-rate, better the initialisation.

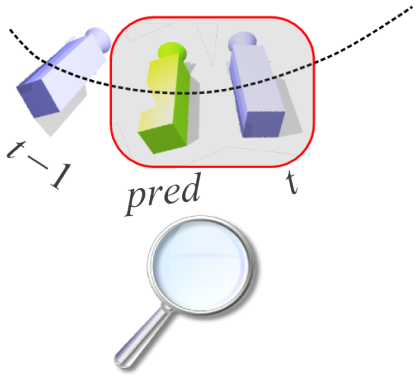
Dense 3D tracking: Background



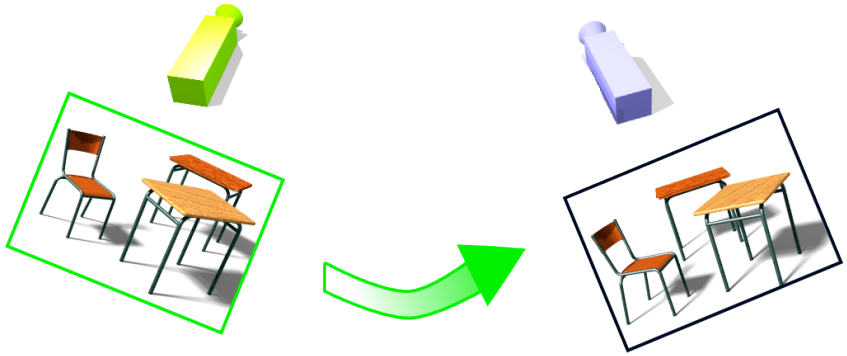


$Camera\ params: p$

$p \sim [R, t]$



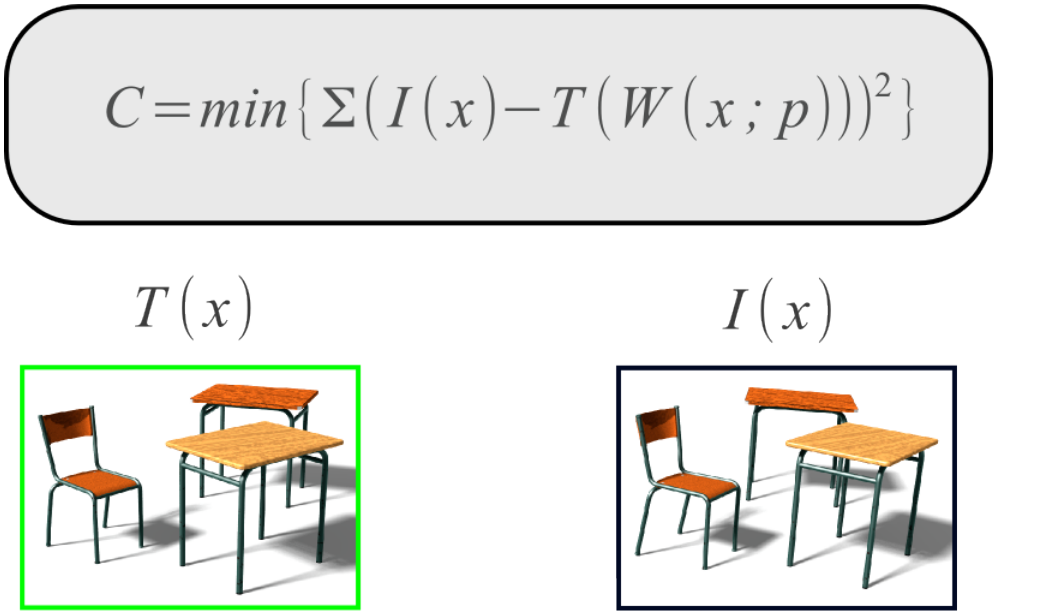
$T(x)$: *Predicted Image* $I(x)$: *Current image*



Dense 3D image alignment: Set-up

Dense 3D tracking: Background

$$C = min\{\Sigma(I(x) - T(W(x;p)))^2\}$$

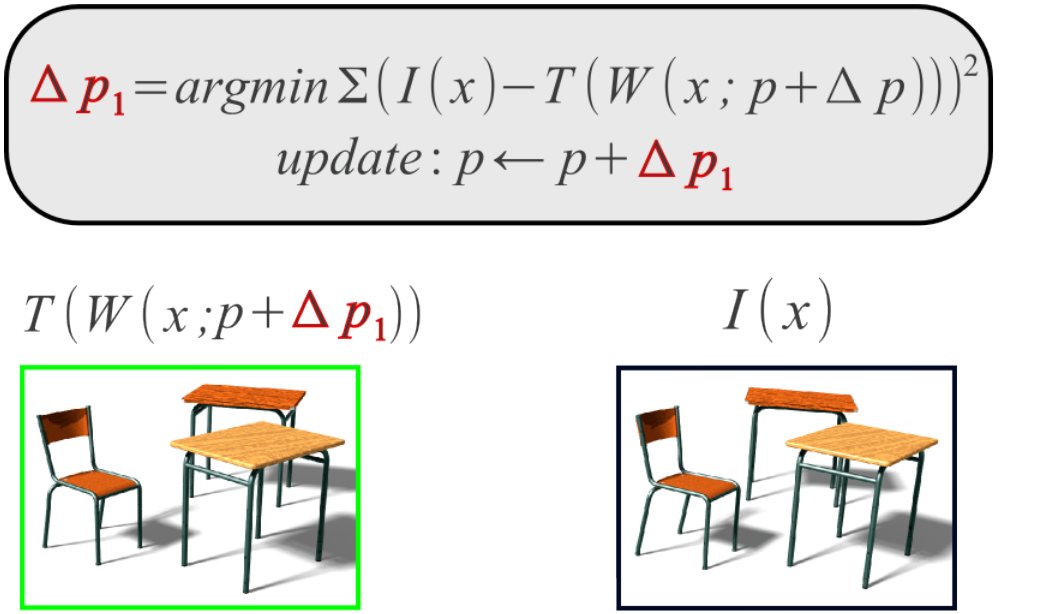


$T(x)$: Predicted Image
$I(x)$: Current image
p: Camera params
$W(x;p)$: Warp

Dense 3D image alignment: Initialisation

Dense 3D tracking: Background

$$\Delta p_1 = \text{argmin} \Sigma (I(x) - T(W(x; p + \Delta p)))^2$$
$$\text{update}: p \leftarrow p + \Delta p_1$$

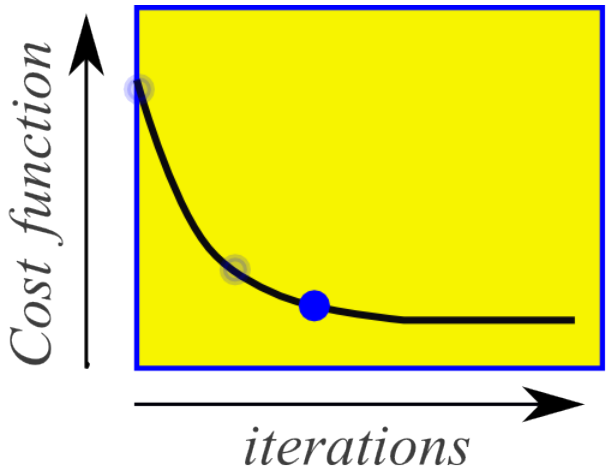


$T(x)$: Predicted Image
$I(x)$: Current image
p: Camera params
$W(x; p)$: Warp

Dense 3D image alignment: Step 1

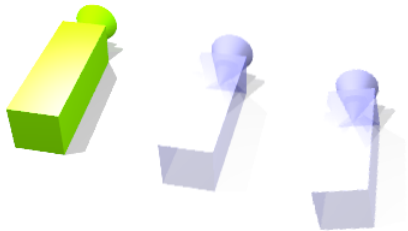
Dense 3D tracking: Background

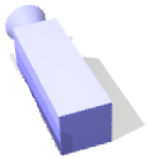
$$\Delta p_2 = \text{argmin} \Sigma (I(x) - T(W(x; p + \Delta p)))^2$$
$$\text{update}: p \leftarrow p + \Delta p_2$$



$T(W(x; p + \Delta p_2))$

$I(x)$



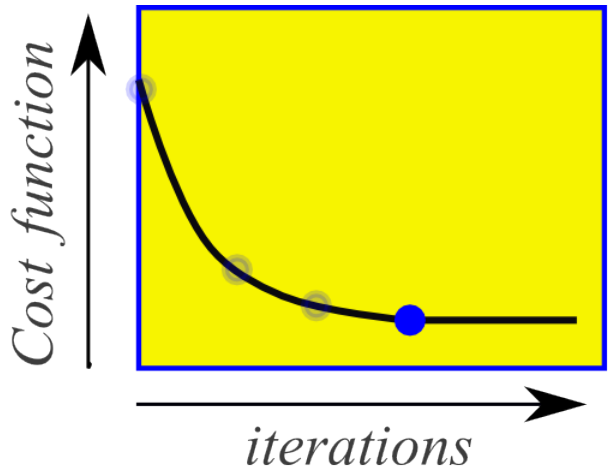


$T(x)$: Predicted Image
$I(x)$: Current image
p: Camera params
$W(x; p)$: Warp

Dense 3D image alignment: Step 2

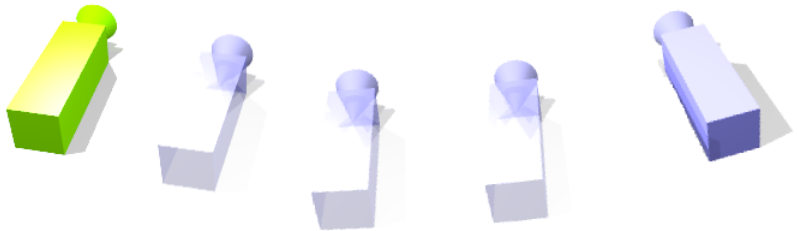
Dense 3D tracking: Background

$$\Delta p_3 = \text{argmin} \Sigma (I(x) - T(W(x; p + \Delta p)))^2$$
$$\text{update}: p \leftarrow p + \Delta p_3$$



$T(W(x; p + \Delta p_3))$

$I(x)$

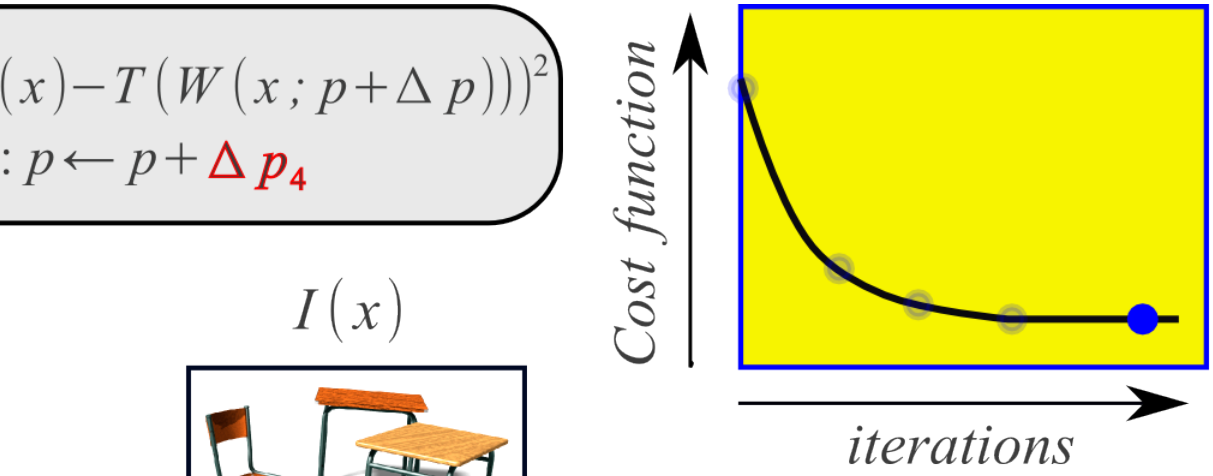


$T(x)$: Predicted Image
$I(x)$: Current image
p: Camera params
$W(x; p)$: Warp

Dense 3D image alignment: Step 3

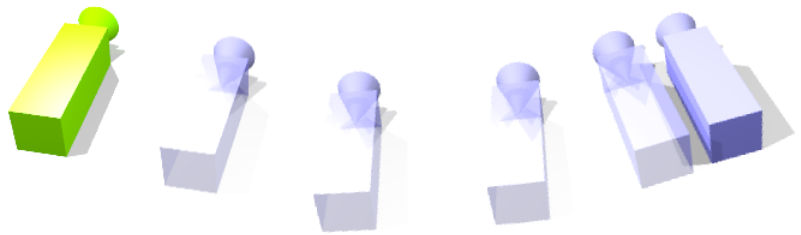
Dense 3D tracking: Background

$$\Delta p_4 = \text{argmin} \Sigma (I(x) - T(W(x; p + \Delta p)))^2$$
$$update: p \leftarrow p + \Delta p_4$$



$T(W(x; p + \Delta p_4))$

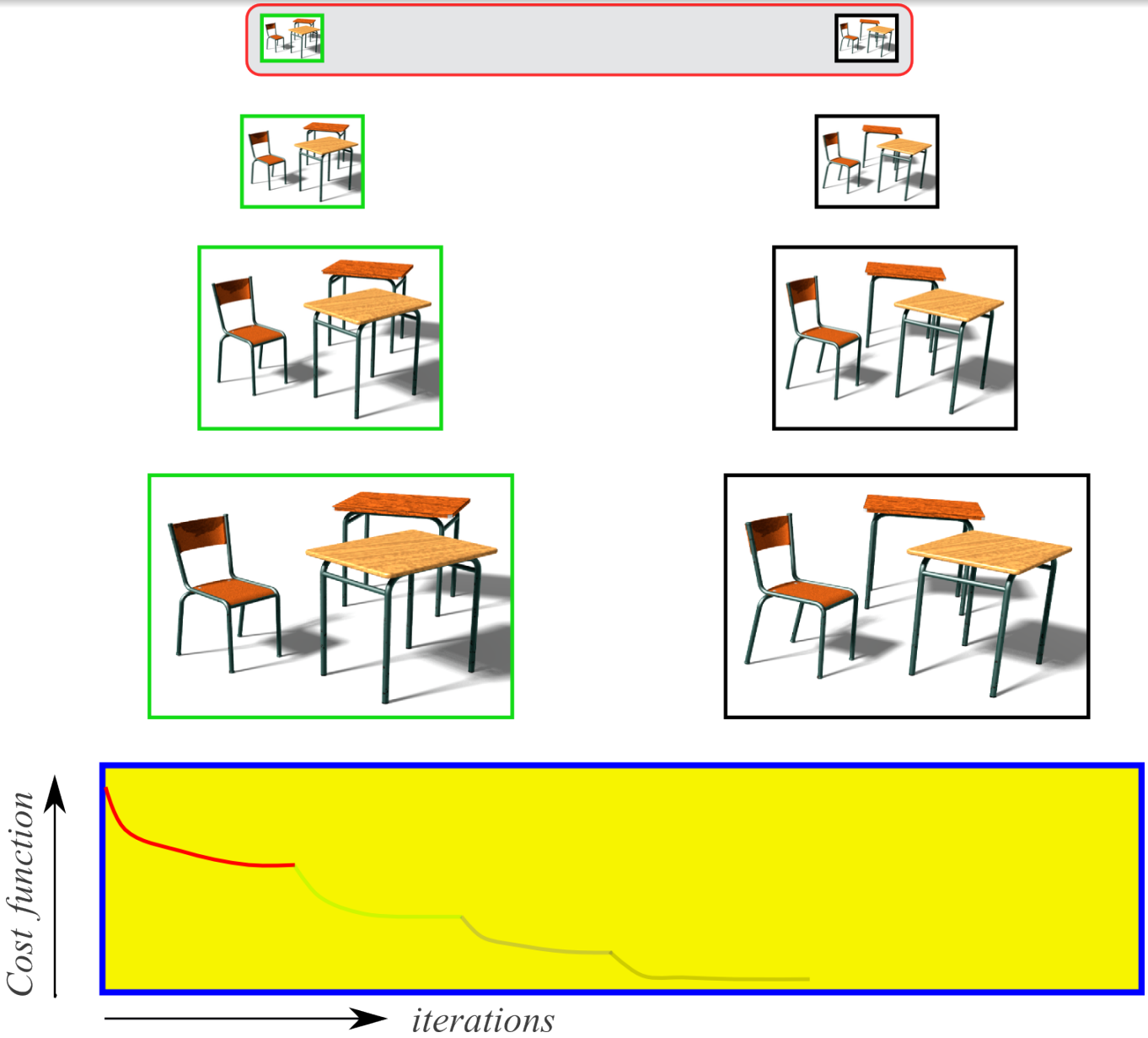
$I(x)$



$T(x)$: Predicted Image
$I(x)$: Current image
p: Camera params
$W(x; p)$: Warp

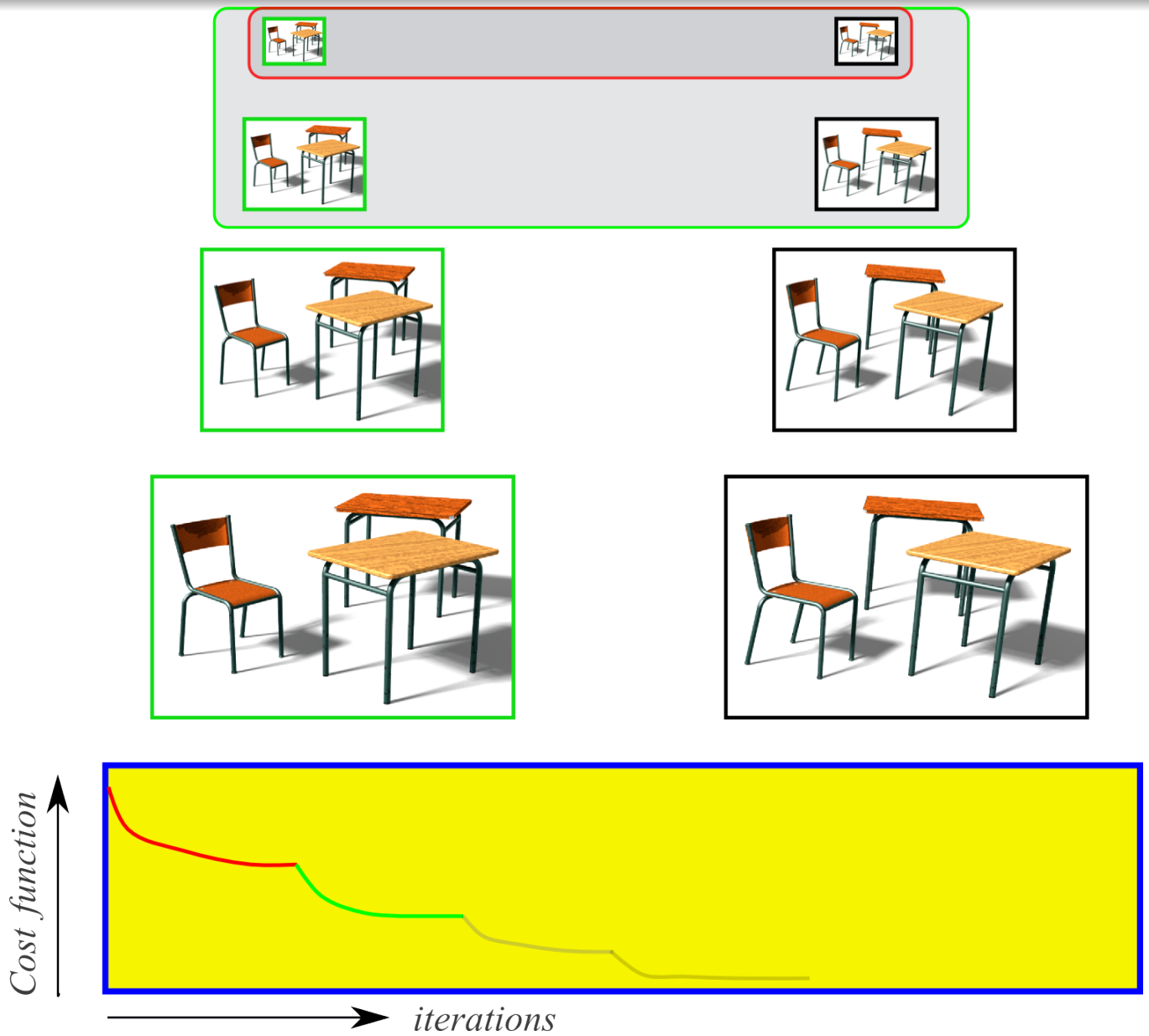
Dense 3D image alignment: Step 4

Dense 3D tracking: Background



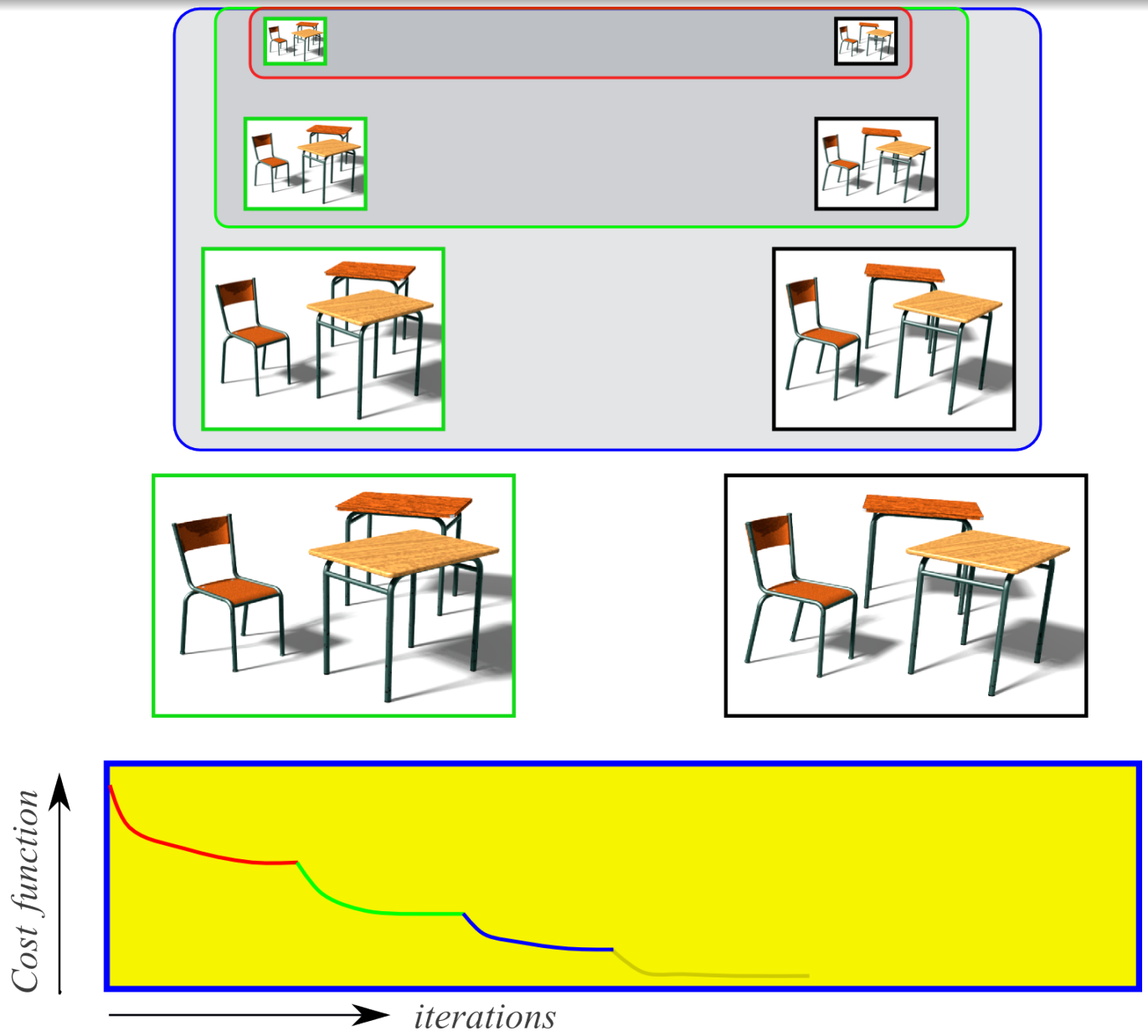
Dense 3D image alignment: Pyramids

Dense 3D tracking: Background



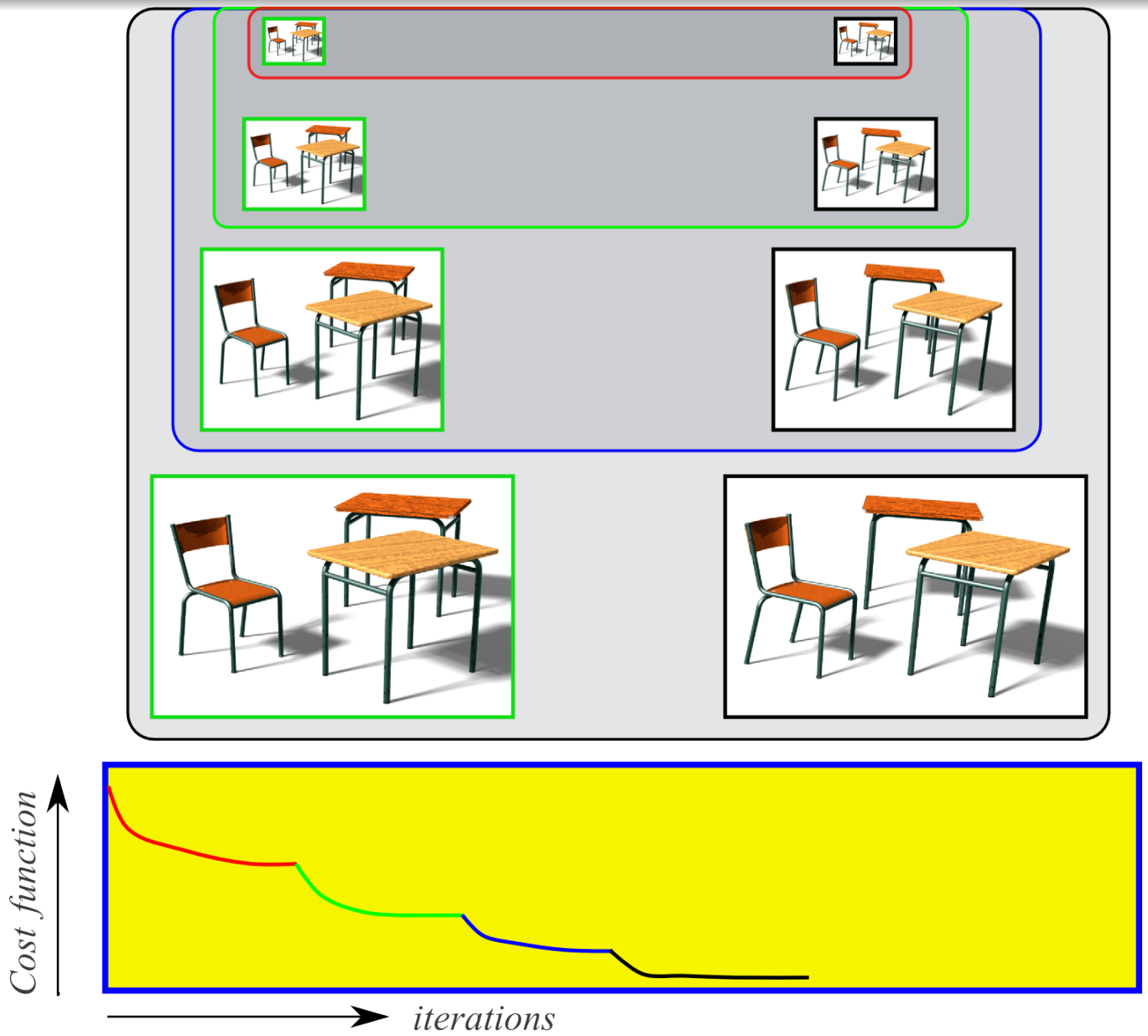
Dense 3D image alignment: Pyramids

Dense 3D tracking: Background



Dense 3D image alignment: Pyramids

Dense 3D tracking: Background



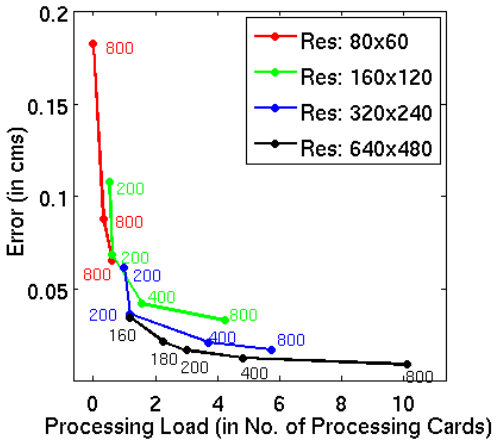
Dense 3D image alignment: Pyramids

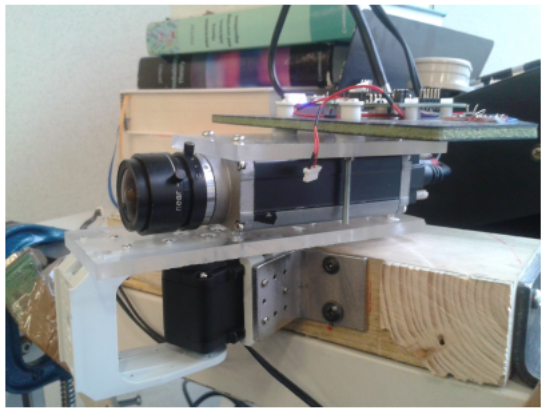
Synthetic Scenes with POVRay

Why synthetic data?

- We needed repeatable trajectories at different frame-rates.
- We wanted fixed light settings to carry out the experiments.
- Synthetic data allowed us to exercise full control on all parameters.

Experiments

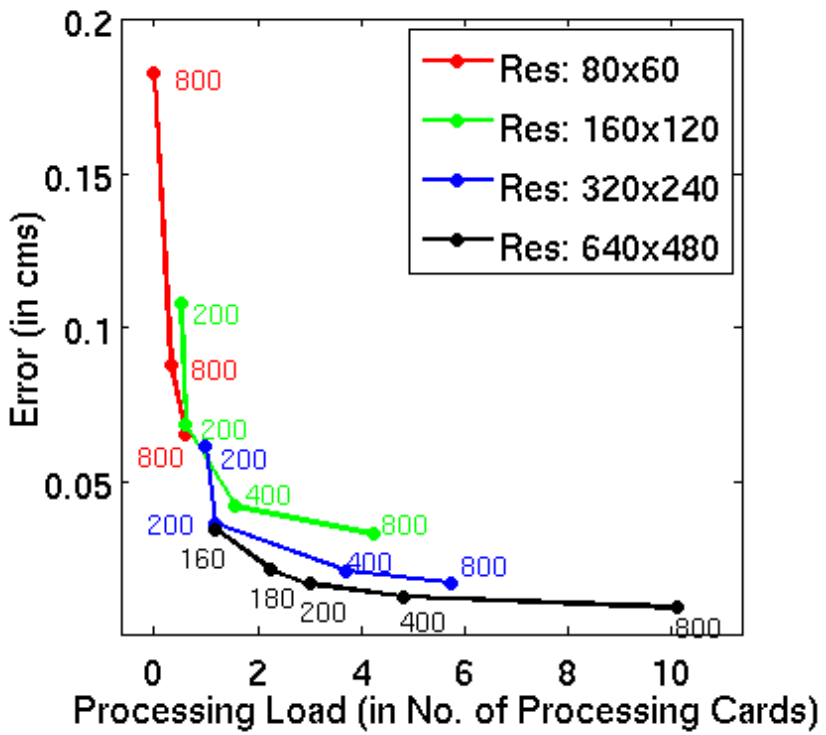




Interpretations

- We ran our dense tracker on trajectories taken at different frame-rates.
- Error is mean absolute distance between predicted and ground truth pose.
- We will get Pareto Fronts showing the optima choice of frame-rate as a function of compute power.
- Verified with real world experiments, *PhD Thesis, 2013*.

Experiment assuming perfect lighting



Pareto front for mininum error/minimum processing load performance, highlighting with numbers the frame-rates that are optimal for each available budget.

Interpretations

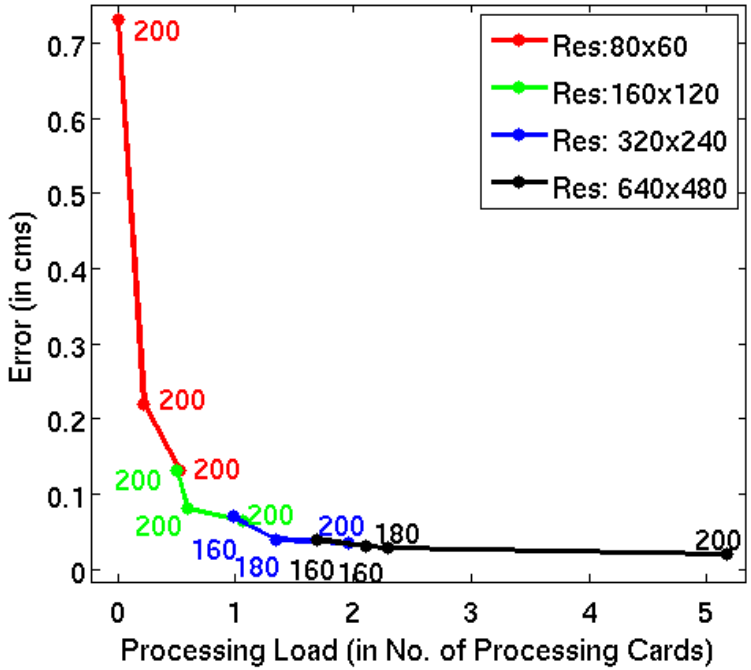
- No noise and no blur — perfect lighting conditions.
- For very low budget few iterations on higher frame-rates (800Hz) are sufficient because baseline is already small to achieve the accuracy.
- Crossovers as the budget changes.
- A combination of high frame-rate and high resolution works best as budget increases.

Realistic lighting conditions

Characterisations

- Real lighting conditions mean there is noise and blur in images.
- The degree of noise depends on the shutter-time as well as strength of scene lighting.
- The strength of scene lighting is quantified by a parameter $\alpha \in \{1, 10, 40\}$.
- Motion blur is generated by averaging the irradiance and not image brightness — Debevec et al., SIGGRAPH 1997.
- Matching of blurry image is against a blurry predicted image obtained from the model.
- Our tracking results will be again represented in the form of a Pareto-Front.

Bright lighting



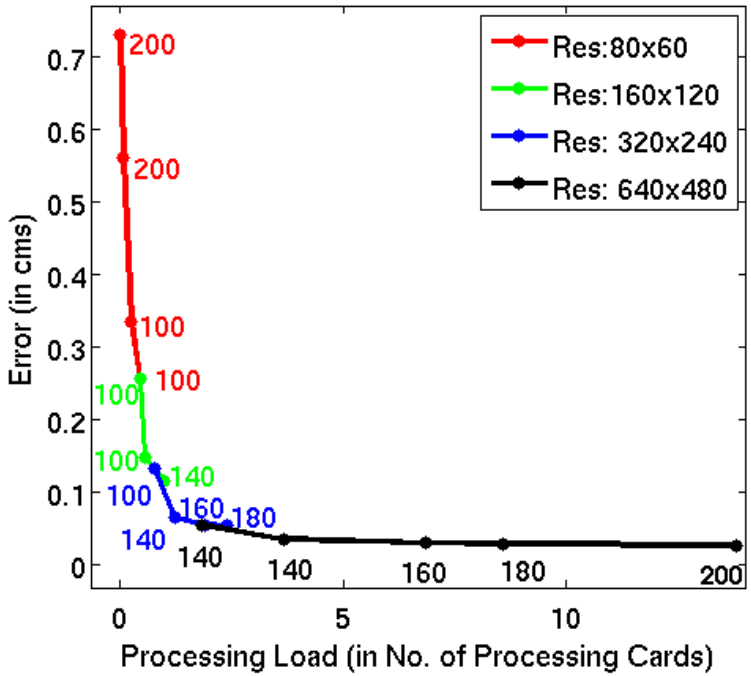
α=40

Pareto Fronts for lighting level $\alpha = 40$. Numbers on the curves show the frame-rates that can be used to achieve the desired error with a given computational budget.

Interpretations for α=40, bright lighting

- Images at high frame-rate are darker but still good enough SNR. Very similar to perfect lighting conditions.
- Very low-budget allows 200Hz due to short baseline.
- 160Hz best choice for 320x240 resolution. This is where cross-overs occur.
- Budget increase prefers high frame-rate.

Moderate lighting



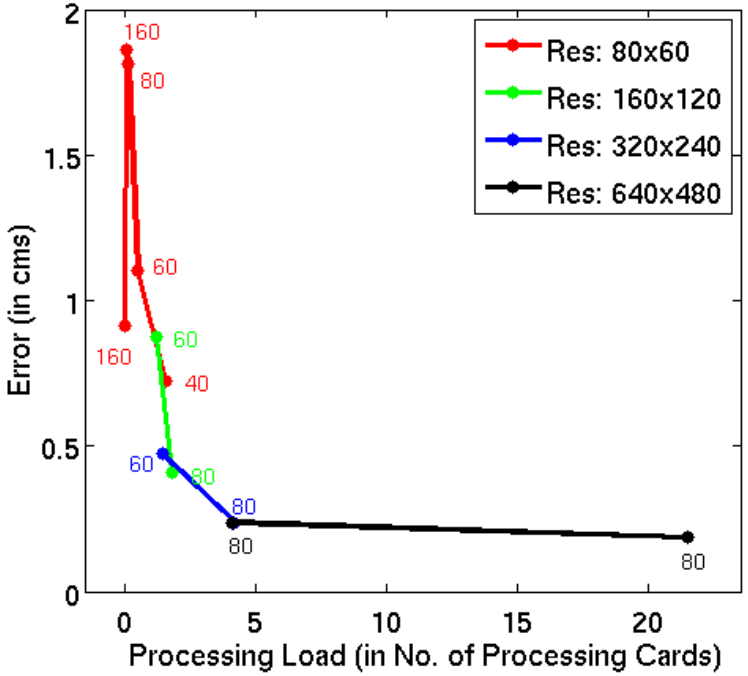
α=10

Pareto Fronts for lighting level $\alpha = 10$.

Interpretations for α=10, moderate lighting

- 200Hz is best choice for very very low budget because prediction is strong and few iterations are sufficient.
- A slight increases sees 100Hz as the best choice — contrast to bright as well as perfect lighting conditions.
- Best choice of frame-rates shift to slightly lower values compared to bright lighting.

Low lighting



α=1

Pareto Fronts for lighting level $\alpha = 1$. Vision in very dark scenes.

Interpretations for α=1, low lighting

- High frame-rates pitch black and no gradient information.
- Pareto-Front does not feature high frame-rates like 200Hz at all.
- 80Hz best choice for low budget.
- Overall tracking quality is much lower.
- Also takes more time to converge.

Discussion

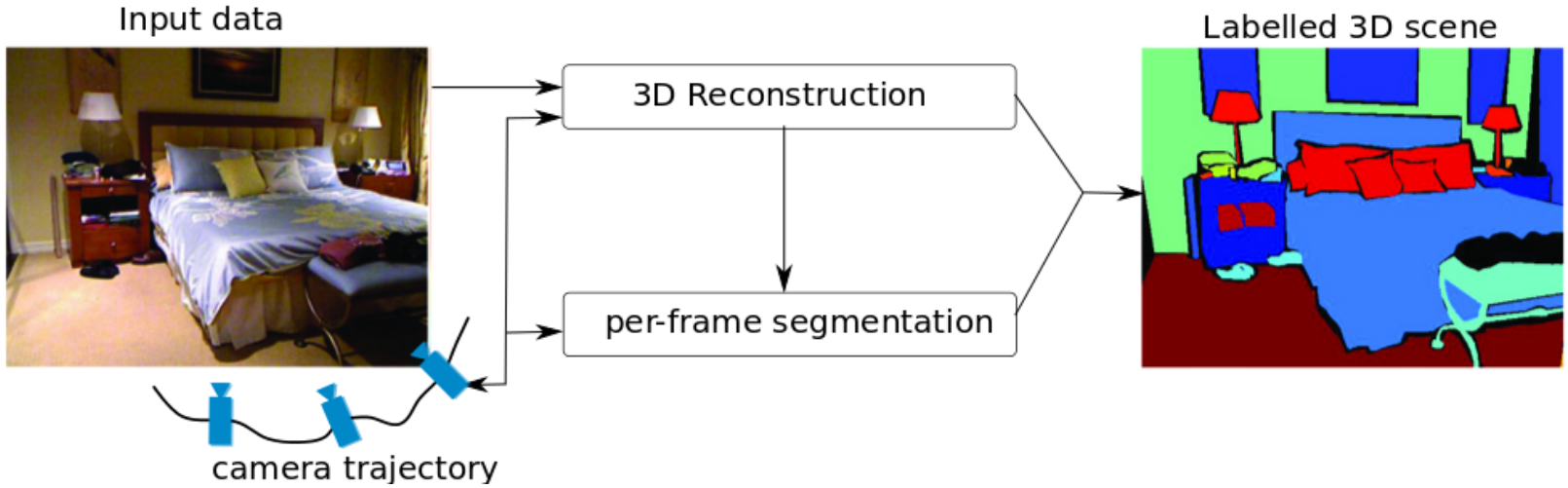
- Given perfect lighting condition and therefore a high signal-to-noise ratio (SNR), the highest accuracy is achieved using a combination of high frame-rate and high resolution.
- Using a realistic camera model, there is an optimal frame-rate for given lighting conditions due to trade-off between SNR and motion blur.
- Realistic conditions mean that there is a combination of optimal frame-rate and high resolution for accuracy.
- We cannot separate resolution from frame-rate given the nature of camera tracking algorithm.

More synthetic scenes with POVRay

Where can large scale synthetic dataset be useful?

- Perfect instance segmentation and semantic segmentation ground truth.
- Getting perfect ground truth poses, and depth and 3D.
- Ground truth for transparent objects and surfaces.
- Physics?

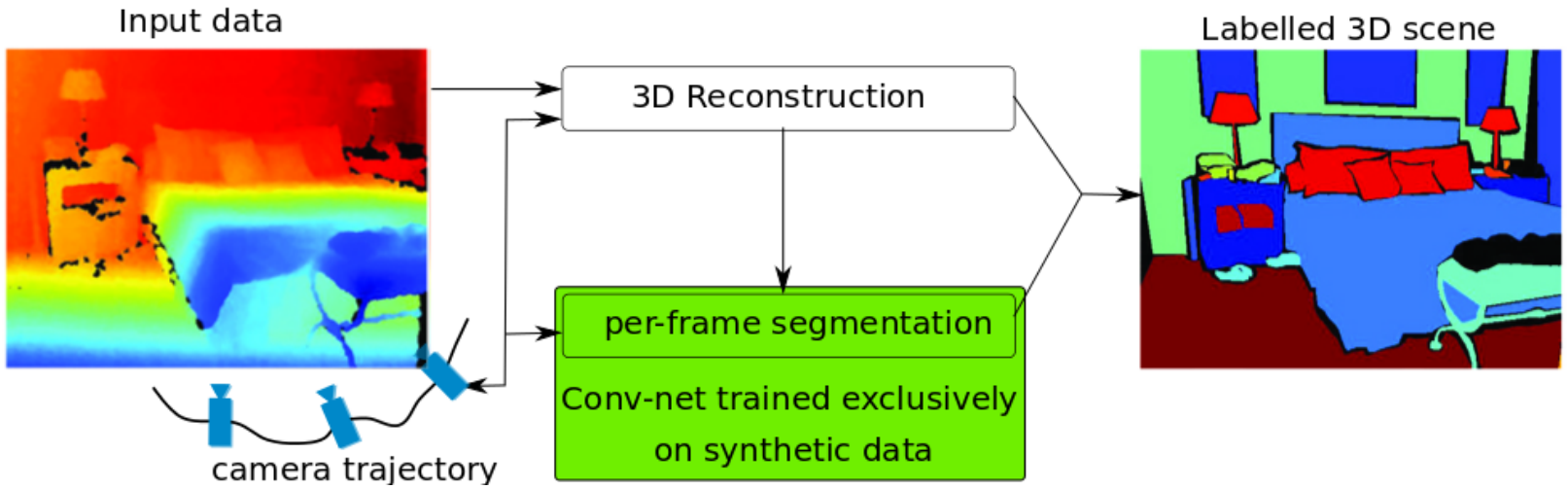
Semantic segmentation of Indoor scenes



What is the goal?

- Given enough training data we would like to segment 3D scenes in real-time.
- Overlay the per-pixel segmentations onto the 3D map.
- Previous scenes we looked at took a long time to render — RGB rendering is very time consuming.
- We decided on depth based semantic segmentation here mainly to understand the role of geometry. We will look into RGB later.

Semantic segmentation of Indoor scenes



How do we get enough training data?

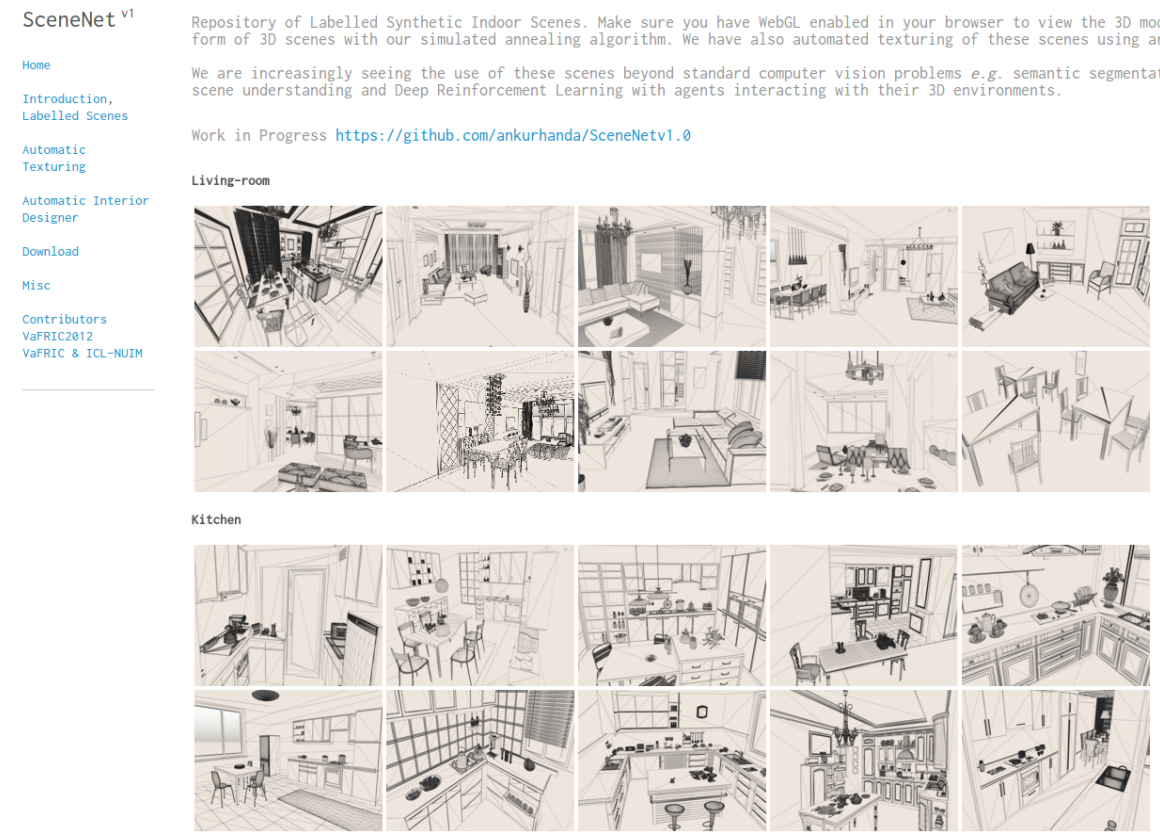
- Real world datasets are limited in size *e.g.* NYUv2 and SUN RGB-D have 795 and 5K images for training respectively.
- We can leverage computer graphics to generate the desired data.
- There are lots of CAD repositories of objects available but none contains scenes.
- We put together a repository of labelled 3D scenes called SceneNet and train per-pixel segmentation on synthetic data.

Outline

- Background
- SceneNet: Repository of Labelled Synthetic Indoor Scenes
- Results on Semantic Segmentation on Real World Data
- gvnn: Neural Network Library for Geometric Vision
- Future Ideas

SceneNet

Repository of labelled 3D synthetic scenes.



Hosted at http://robotvault.bitbucket.org

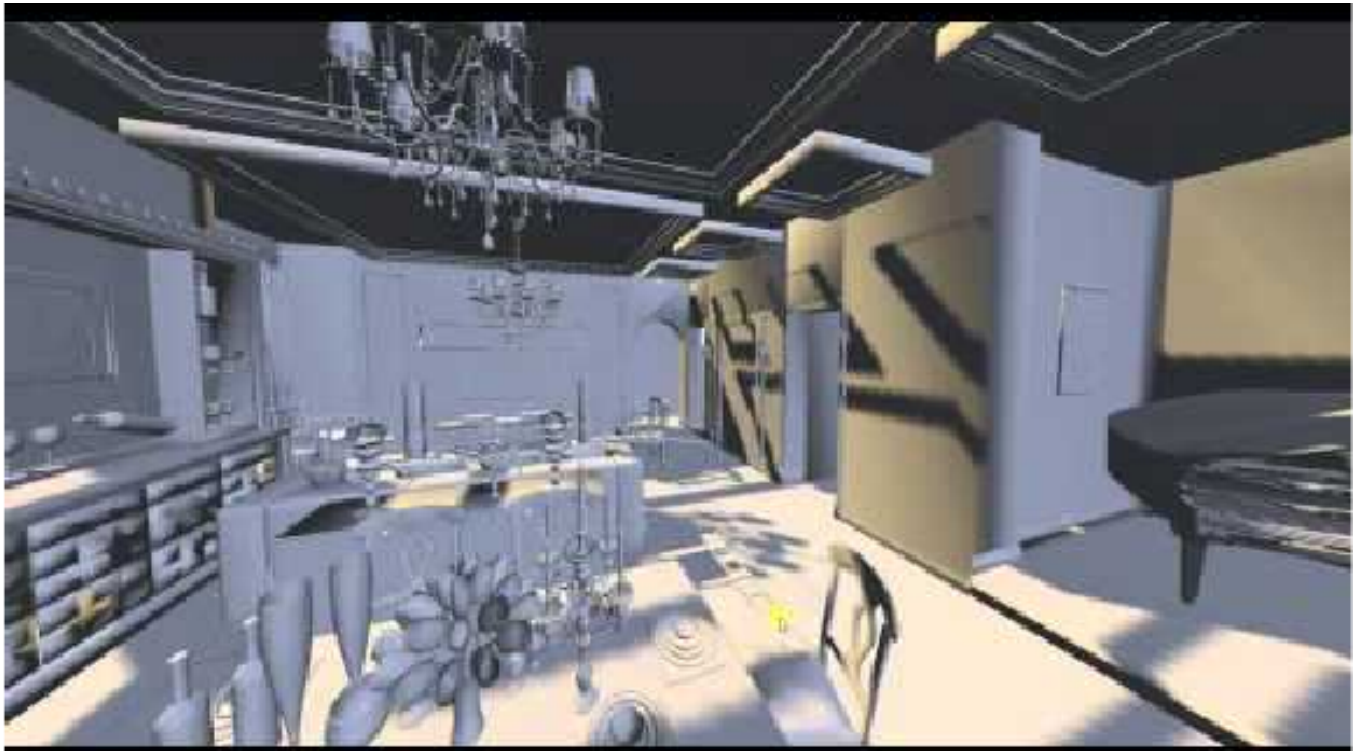
SceneNet

SceneNet Basis Scenes

Room Type	Number of Scenes
Living Room	10
Bedroom	11
Office	15
Kitchens	11
Bathrooms	10

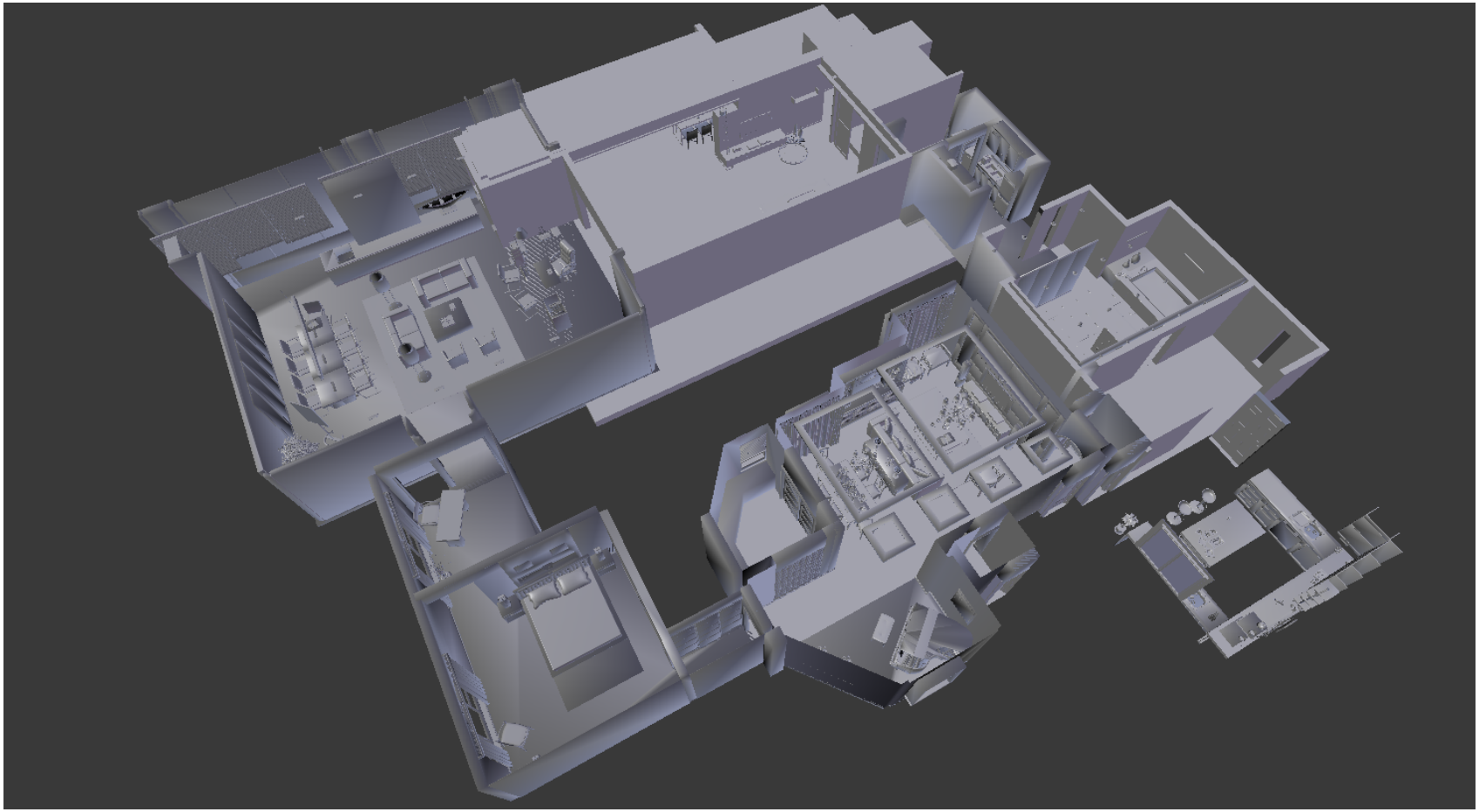
In total we have 57 very detailed scenes with about 3700 objects. We build upon these scenes to create unlimited number of scenes later for large scale training data.

SceneNet



A very detailed living room. Each object in this scene is labelled directly in 3D.

SceneNet



dimensions: 27x25x2.7m^3

Generating Unlimited Number of Synthetic Scenes

Generating indoor scenes as energy minimisation problem

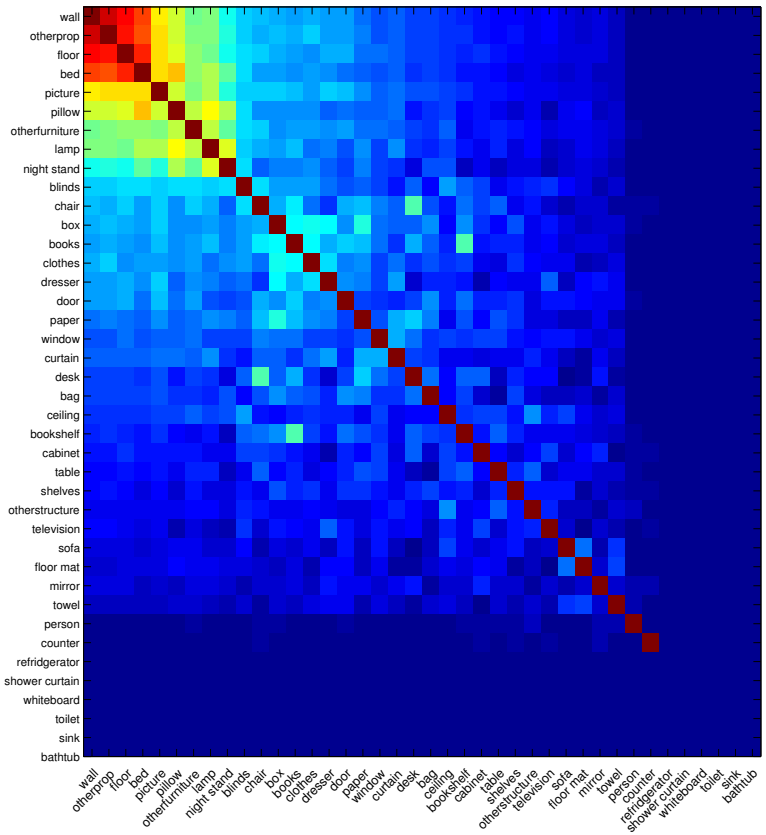
- SceneNet is still very limited sized dataset with 57 scenes.
- We would like to have potentially unlimited scenes with lots of variety and shapes.
- However.... we could use layouts of SceneNet and the object co-occurrence statistics, *i.e.* which objects likely co-occur together.
- We can then sample objects from object repositories like **ModelNet** and arrange them according to these co-occurrence statistics to create more scenes.
- We empirically scale the **ModelNet** CAD models to metric space *i.e.* chairs should be $\sim$ 0.8-1m in height.
- Placing objects in the scene could be turned into an energy minimisation problem.

Generating Unlimited Number of Synthetic Scenes

Generating indoor scenes as energy minimisation problem

- Bounding box constraint.
  - Each object must maintain a safe distance from the other.
- Pairwise Constraint.
  - Objects that co-occur should not be more than a fixed distance from each other, *e.g.* beds and nightstands
- Visibility Constraint.
  - Objects with visibility constraint must not have anything joining their line of sight joining their centers, *e.g.* sofa,table and TV must not having anything in between them.
- Distance to Wall Constraint.
  - Some objects are more likely to occur next to walls *e.g.* cupboards, sofa etc.
- Use simulated annealing to solve this energy minimisation problem.

Generating Unlimited Number of Synthetic Scenes

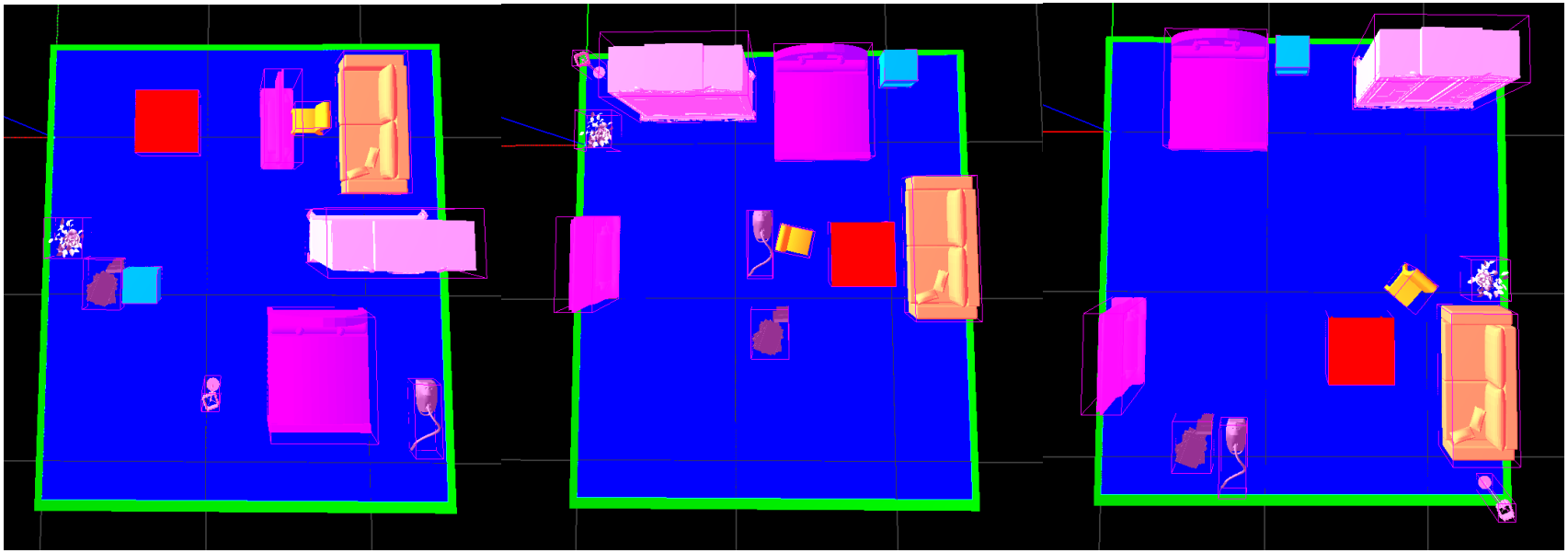


Co-occurrence statistics of objects from NYUv2 bedrooms.

Interpretations

- Heat-map of object co-occurrence.
- Training sets in NYUv2 and SceneNet basis scenes are used to obtain these statistics.
- Shows that bed, picture, pillow and nightstand co-occur together.
- We sample objects from **ModelNet** and arrange them using these object co-occurrence statistics.

Generating Unlimited Number of Synthetic Scenes

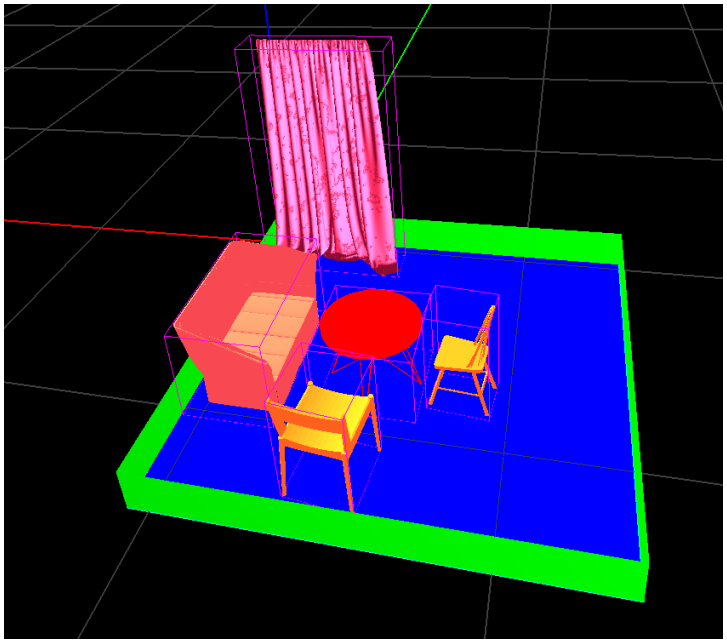


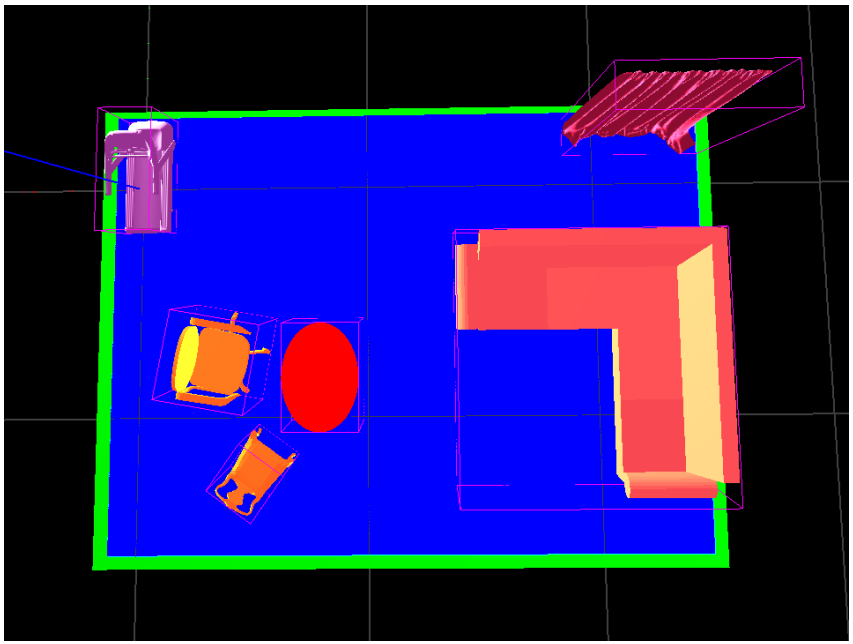
Random placement | With Pairwise only | All constraints

Constraints

- Objects that co-occur in real world should be placed together, *e.g.* beds and nightstands.
- Visibility constraint - nothing in the line of sight between sofa-table and TV.

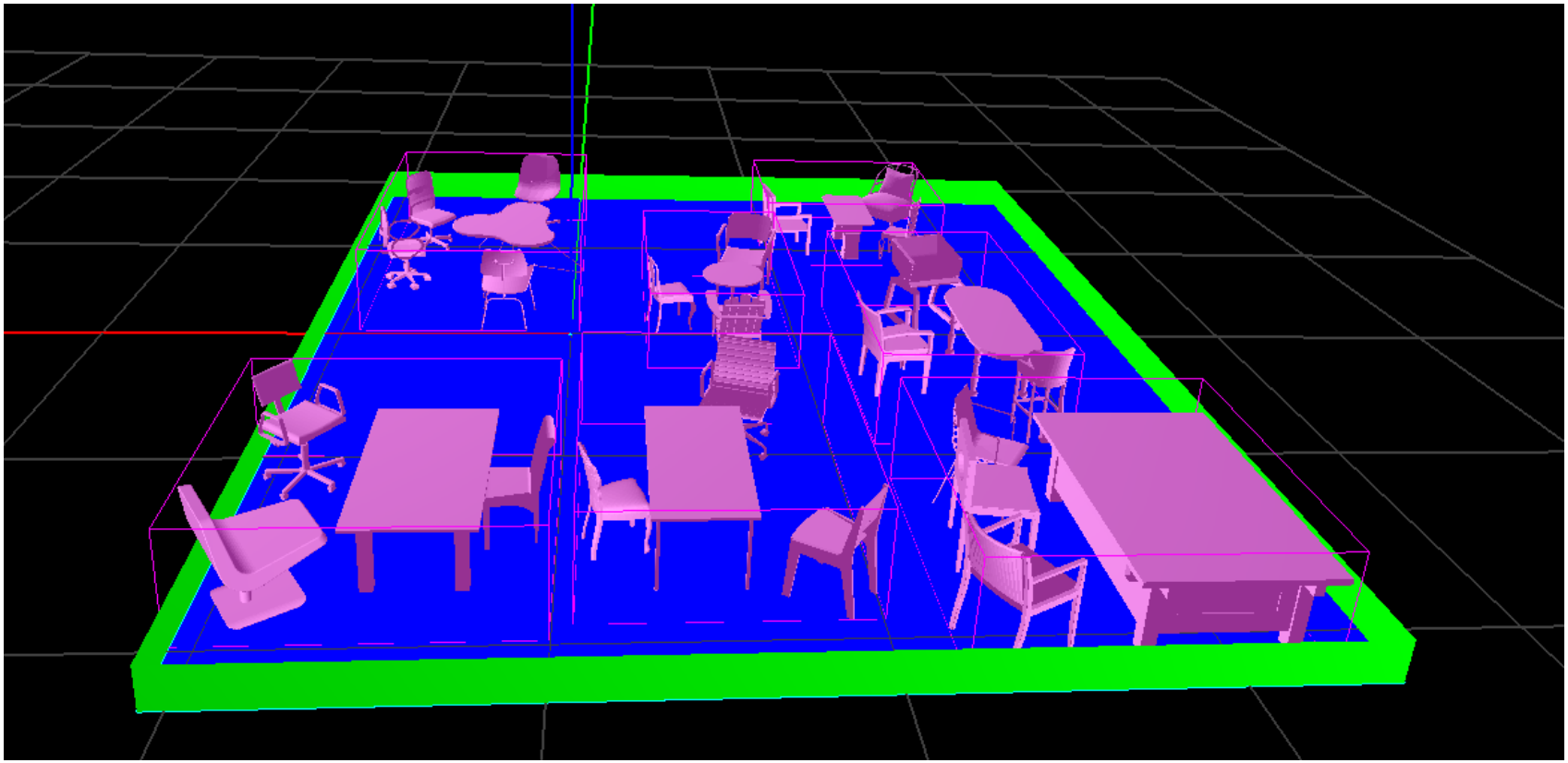
Generating Unlimited Number of Synthetic Scenes





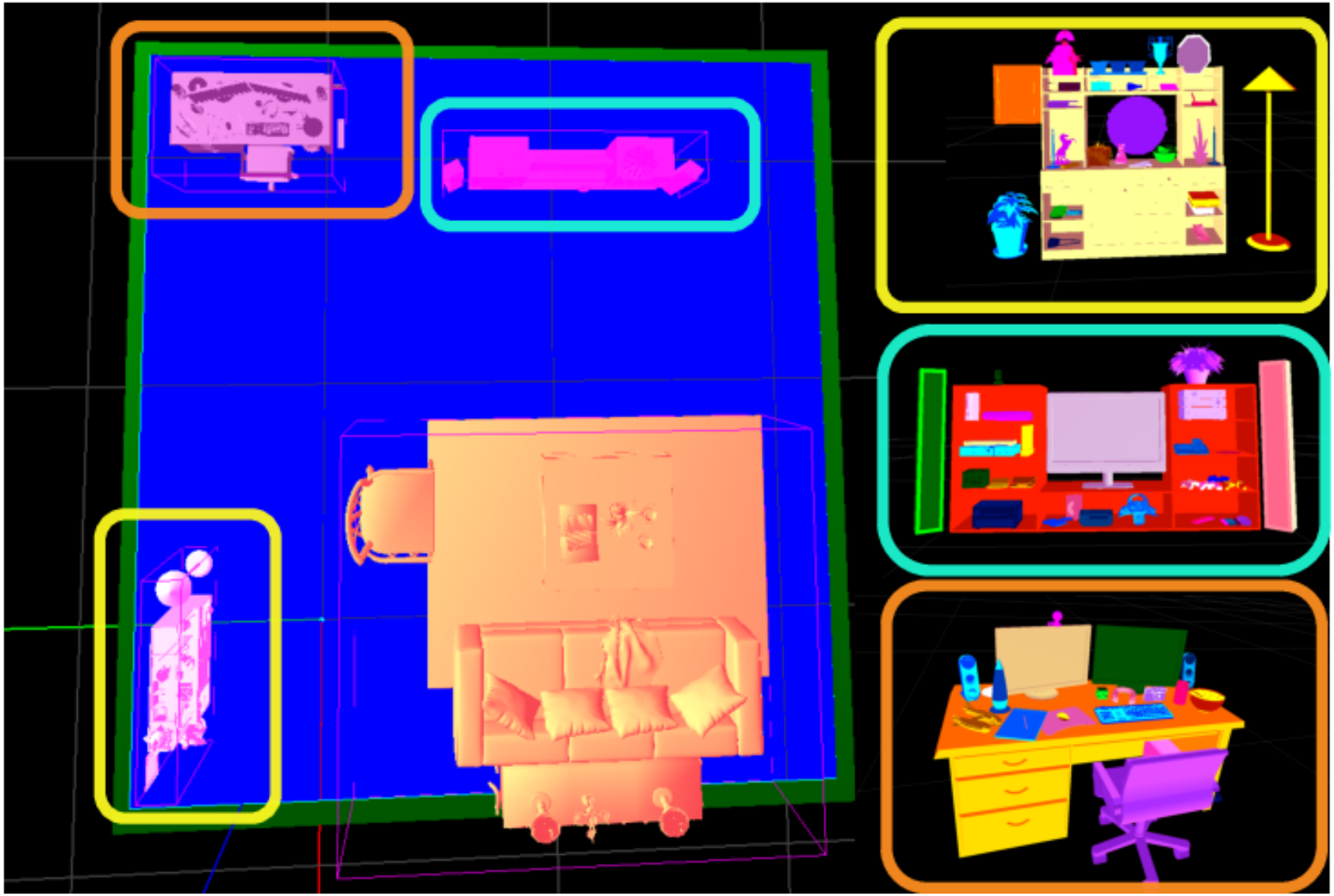
Very simple scenes with curtains.

Generating Unlimited Number of Synthetic Scenes



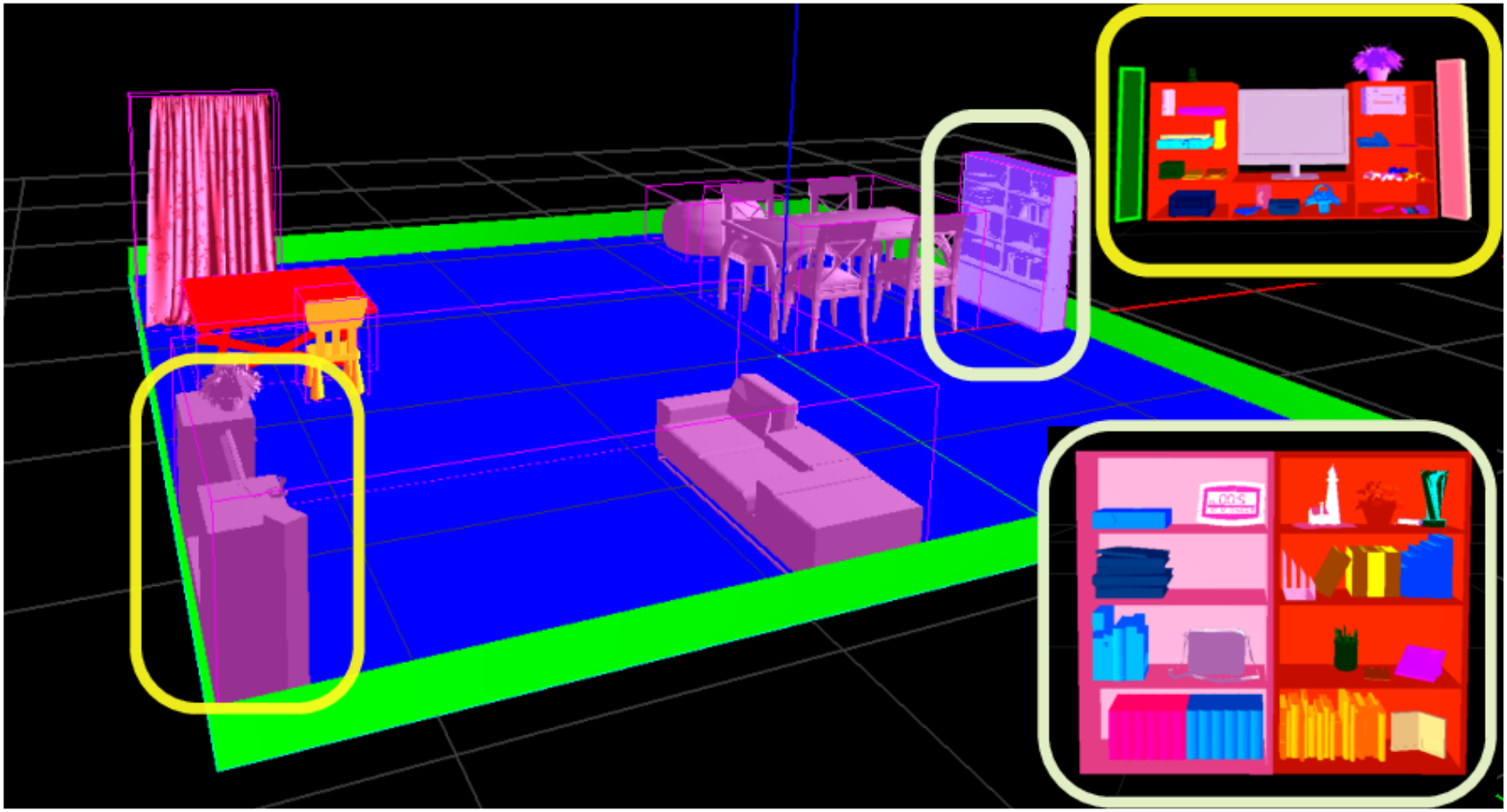
Common room obtained from hierarchically optimising chairs and table combination.

Generating Unlimited Number of Synthetic Scenes



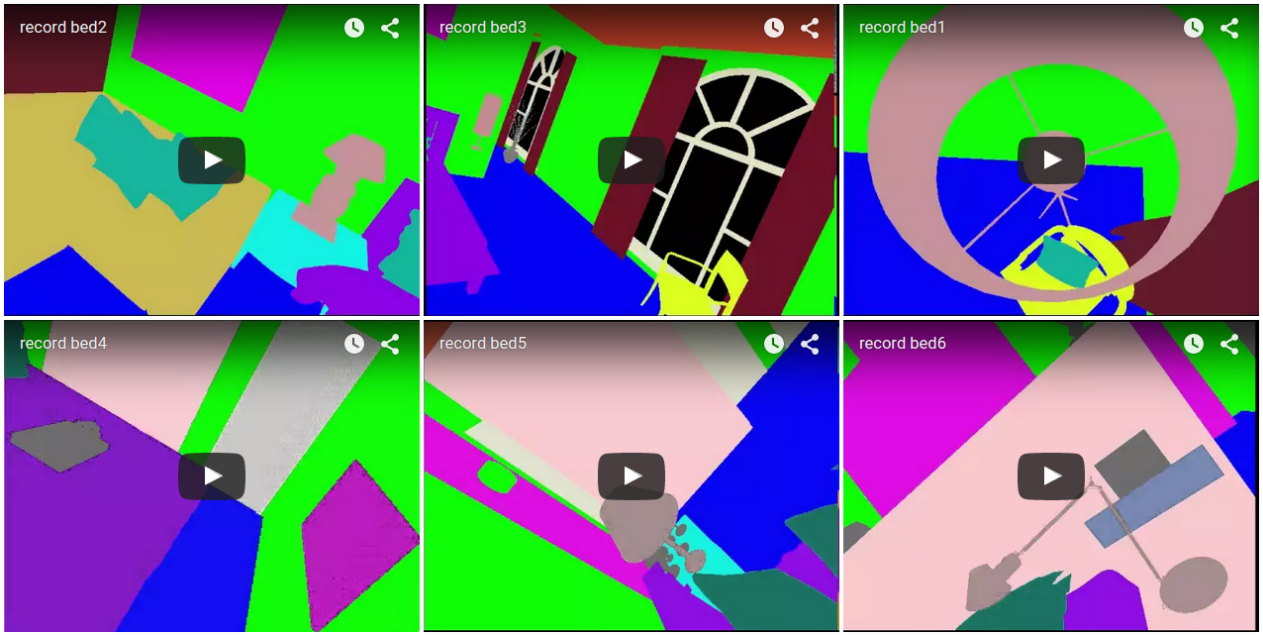
Living Room. Note that the inset objects are from Stanford Scenes and have been grouped together already.

Generating Unlimited Number of Synthetic Scenes



Living Room. Grouped objects from Stanford Scenes.

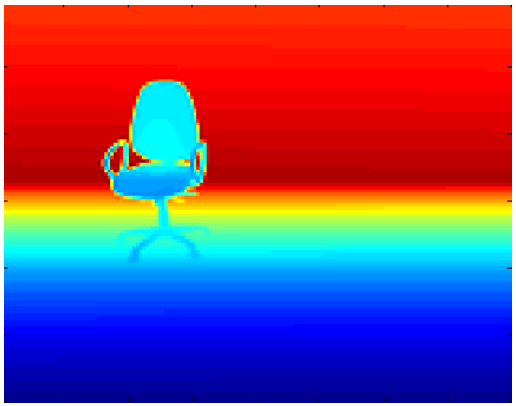
Generating Unlimited Number of Synthetic Scenes



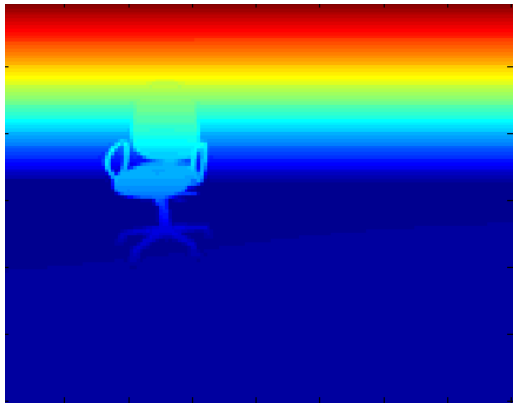
Data Collection

- Collecting trajectories with joy-stick (though we have randomised them now).
- Using OpenGL glReadPixels to obtain the depth as well as ground truth labels.

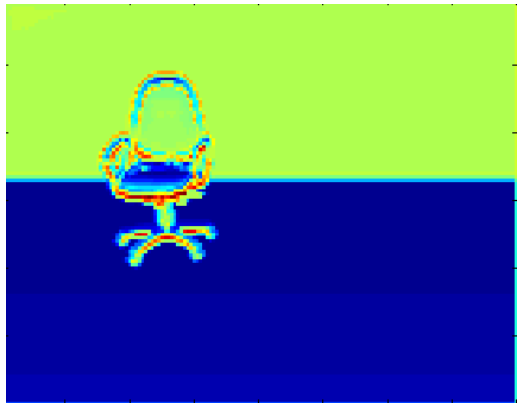
Generating Unlimited Number of Synthetic Scenes



Depth



Height

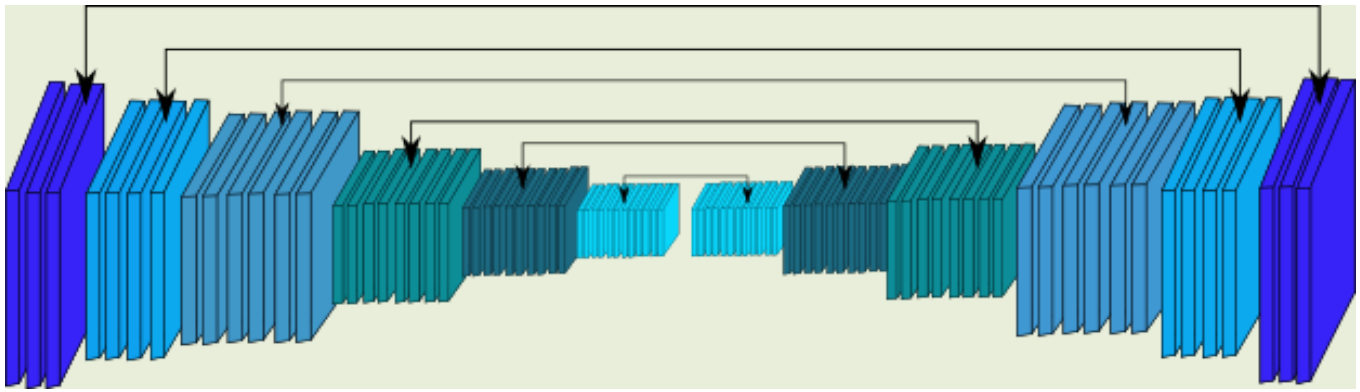


Angle with gravity

Assumptions

- Here we only study the segmentation from just depth-data in the form of DHA images *i.e.* depth, height from ground plane and angle with gravity vector.
- Allows us to study the effects of geometry in isolation. Texturing takes time and needs careful photo-realistic synthesis. But we have added a custom raytracer to study RGB-D based segmentation now.

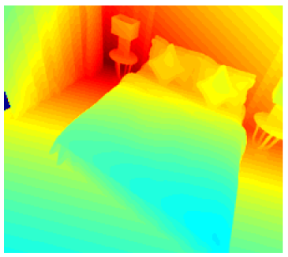
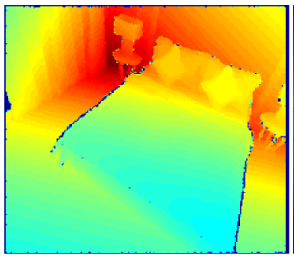
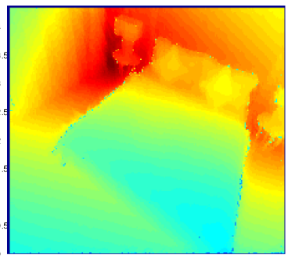
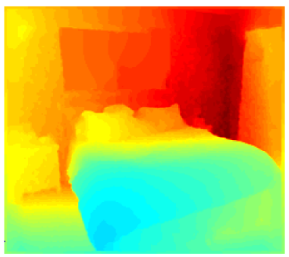
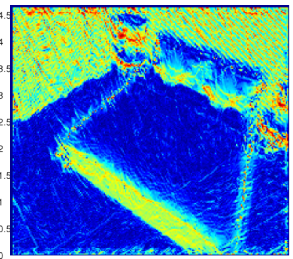
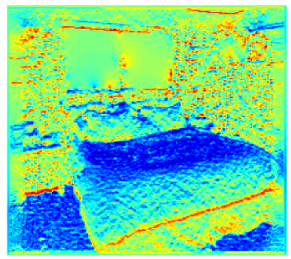
Network Architecture



SegNet: [Badrinarayanan, Handa, and Cipolla, arXiv 2015]

- Saves pooling indices explicitly in the encoder and passes on to the decoder for upsampling.
- Inspired by Ranzato *et al.*, CVPR 2007.
- Similar ideas emerged in DeconvNet from Hyeonwooh Noh *et al.* ICCV2015 and Zhao *et al.* Stacked What-Where Auto-encoders, arxiv2015.

Carefully synthesising realistic depth-maps

Adding noise to depth-map | Denoising depth-map | Angle with gravity maps

Adapting synthetic data to match real world data

- Synthetic depth-maps are injected with appropriate noise models.
- We then denoise these depth-maps to ensure that input images look similar to the NYUv2 dataset.
- Side by side comparison of synthetic bedroom with one of the NYUv2 bedrooms.

Outline

- Background
- SceneNet: Repository of Labelled Synthetic Indoor Scenes
- Results on Semantic Segmentation on Real World Data
- gvnn: Neural Network Library for Geometric Vision
- Future Ideas

Results on Semantic Segmentation on Real World Data

NYUv2 test images

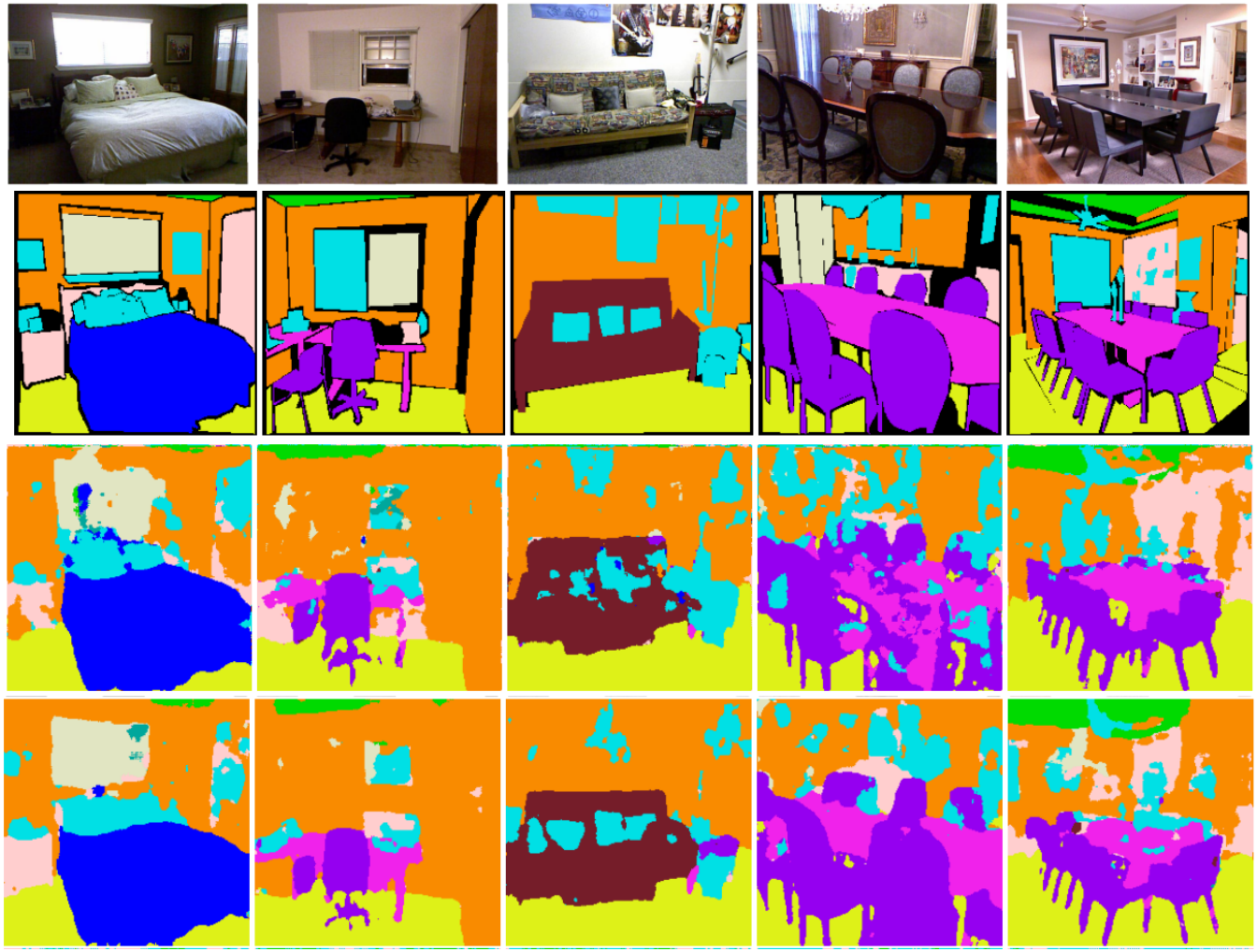
Results on Semantic Segmentation on Real World Data

Ground truth segmentation

Results on Semantic Segmentation on Real World Data

Training on NYUv2 training set only

Results on Semantic Segmentation on Real World Data



Training on SceneNet and fine-tuning on NYUv2

Results on Semantic Segmentation on Real World Data

Training on NYUv2 only

Training on SceneNet and fine-tuning on NYUv2

What we learnt? [ICRA 2016, CVPR 2016]

- Computer graphics is a great source for collecting data.
- Depth based semantic segmentation allowed us to understand the role of geometry.
- We only used 10K synthetic frames in this experiment mainly because of limited compute power.

Results on Semantic Segmentation on Real World Data

11 class semantic segmentation: NYUv2

Training	global acc.	class acc.
SceneNet-DHA	54.4	42.6
NYU-DHA	63.8	53.7
SceneNet-FT-NYU-DHA	67.4	59.1
SceneNet-DO-DHA	54.6	43.2
NYU-DO-DHA	65.7	56.9
SceneNet-FT-NYU-DO-DHA	68.0	59.9
Eigen *et al.*(rgbd+normals) [12]	**69.5**	**63.0**

11 class semantic segmentation: NYUv2

Training	bed	ceil.	chair	floor	furn	objs.	sofa	table	tv	wall	window
NYU-DHA	64.5	68.2	51.0	95.0	51.0	48.2	49.7	41.9	12.8	84.2	24.5
SceneNet-DHA	60.8	43.4	68.5	90.0	26.5	24.3	21.2	42.1	0	**92.1**	0.3
SceneNet-FT-NYU-DHA	70.3	75.9	59.8	96.0	**60.7**	49.7	59.9	**49.7**	24.3	84.8	27.9
NYU-DO-DHA	69.0	74.6	54.0	95.6	57.1	48.7	55.7	42.5	18.5	84.7	25.5
SceneNet-DO-DHA	67.7	40.9	67.5	87.8	38.6	22.6	15.8	44.2	0	89.0	0.8
SceneNet-FT-NYU-DO-DHA	**72.5**	74.1	61.0	96.2	60.4	50.0	**62.8**	43.8	19.4	85.3	30.0
Eigen *et al.* (rgbd+normals) [12]	61.1	78.3	**72.1**	**96.5**	55.1	**52.1**	45.8	45.0	**41.9**	88.7	**57.7**

What we learnt? [ICRA 2016, CVPR 2016]

- We perform better than state-of-the-art on functional categories of objects
- Suggesting shape is a strong cue for segmentation.
- We fall behind on objects that explicitly need RGB data.

Results on Semantic Segmentation on Real World Data

13 class semantic segmentation: NYUv2

Training	global acc.	class acc.
Hermans *et al.*(rgbd+crf) [22]	54.2	48.0
SceneNet-DHA	54.4	37.1
NYU-DHA	63.6	47.2
SceneNet-FT-NYU-DHA	66.5	51.7
SceneNet-DO-DHA	55.3	37.6
NYU-DO-DHA	65.0	48.6
SceneNet-FT-NYU-DO-DHA	67.2	52.5
Eigen *et al.*(rgbd+normals) [12]	**68.0**	**60.8**

13 class semantic segmentation: NYUv2

Training	bed	books	ceil.	chair	floor	furn	objs.	paint.	sofa	table	tv	wall	window
NYU-DHA	67.7	6.5	69.9	47.9	**96.2**	53.8	46.5	11.3	50.7	41.6	10.8	85.0	25.8
SceneNet-DHA	60.8	2.0	44.2	68.3	90.2	26.4	27.6	6.3	21.1	42.2	0	**92.0**	0.0
SceneNet-FT-NYU-DHA	70.8	5.3	75.0	58.9	95.9	63.3	48.4	15.2	58.0	43.6	22.3	85.1	29.9
NYU-DO-DHA	69.6	3.1	69.3	53.2	95.9	60.0	49.0	11.6	52.7	40.2	17.3	85.0	27.1
SceneNet-DO-DHA	67.9	4.7	41.2	67.7	87.9	38.4	25.6	6.3	16.3	43.8	0	88.6	1.0
SceneNet-FT-NYU-DO-DHA	**70.8**	5.5	76.2	59.6	95.9	**62.3**	**50.0**	18.0	**61.3**	42.2	22.2	86.1	32.1
Eigen *et al.* (rgbd+normals) [12]	61.1	**49.7**	78.3	**72.1**	96.0	55.1	40.7	**58.7**	45.8	**44.9**	**41.9**	88.7	**57.7**
Hermans *et al.*(rgbd+crf)[22]	68.4	N/A	**83.4**	41.9	91.5	37.1	8.6	N/A	28.5	27.7	38.4	71.8	46.1

What we learnt? [ICRA 2016, CVPR 2016]

- We perform better than state-of-the-art on functional categories of objects
- Suggesting shape is a strong cue for segmentation.
- We fall behind on objects that explicitly need RGB data.

Results on Semantic Segmentation on Real World Data

13 class semantic segmentation: SUNRGBD

Training	global acc.	class acc.
SceneNet-DHA	56.9	30.2
SUNRGBD-DHA	73.7	49.2
SceneNet-FT-SUNRGBD-DHA	74.7	52.7
SceneNet-DO-DHA	54.7	31.6
SUNRGBD-DO-DHA	74.2	52.2
SceneNet-FT-SUNRGBD-DO-DHA	**75.0**	**53.1**

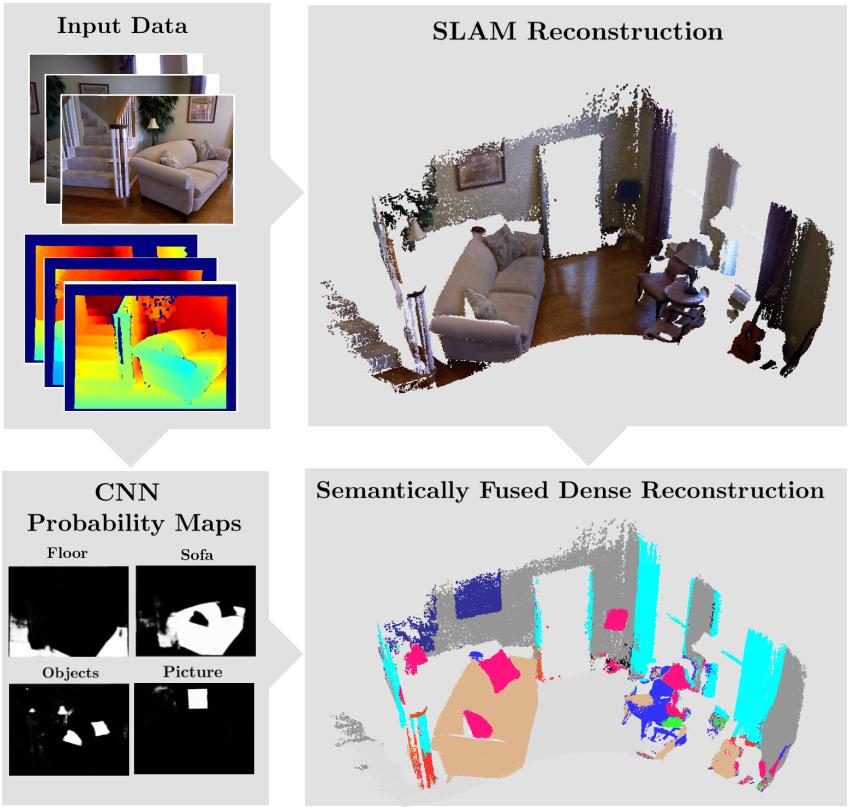
13 class semantic segmentation: SUNRGBD

Training	bed	books	ceil.	chair	floor	furn	objs.	paint.	sofa	table	tv	wall	window
SceneNet-DHA	33.2	2.5	40.6	54.0	71.1	26.2	22.1	9.5	15.0	29.2	0	89.2	0.0
SceneNet-DO-DHA	46.1	5.2	43.6	54.8	63.1	37.4	23.2	10.7	12.2	29.8	0	83.6	1.0
SUNRGBD-DHA	70.4	11.2	64.7	69.2	**94.0**	48.4	35.3	13.7	48.2	63.0	3.5	**89.7**	27.9
SUNRGBD-DO-DHA	73.6	16.6	**71.6**	70.1	93.5	47.9	38.7	**17.2**	**58.5**	61.8	6.8	88.7	33.9
SceneNet-FT-SUNRGBD-DHA	69.0	**20.0**	70.3	70.7	93.7	49.7	35.5	15.7	57.8	**65.9**	**14.1**	89.0	33.8
SceneNet-FT-SUNRGBD-DO-DHA	**75.6**	13.5	69.2	**73.6**	93.8	**52.0**	**37.1**	16.8	57.2	62.7	9.5	88.8	**36.5**

What we learnt? [ICRA 2016, CVPR 2016]

- We perform better than state-of-the-art on functional categories of objects
- Suggesting shape is a strong cue for segmentation.
- We fall behind on objects that explicitly need RGB data.

SemanticFusion Mccormac *et al. 2016*



Notes

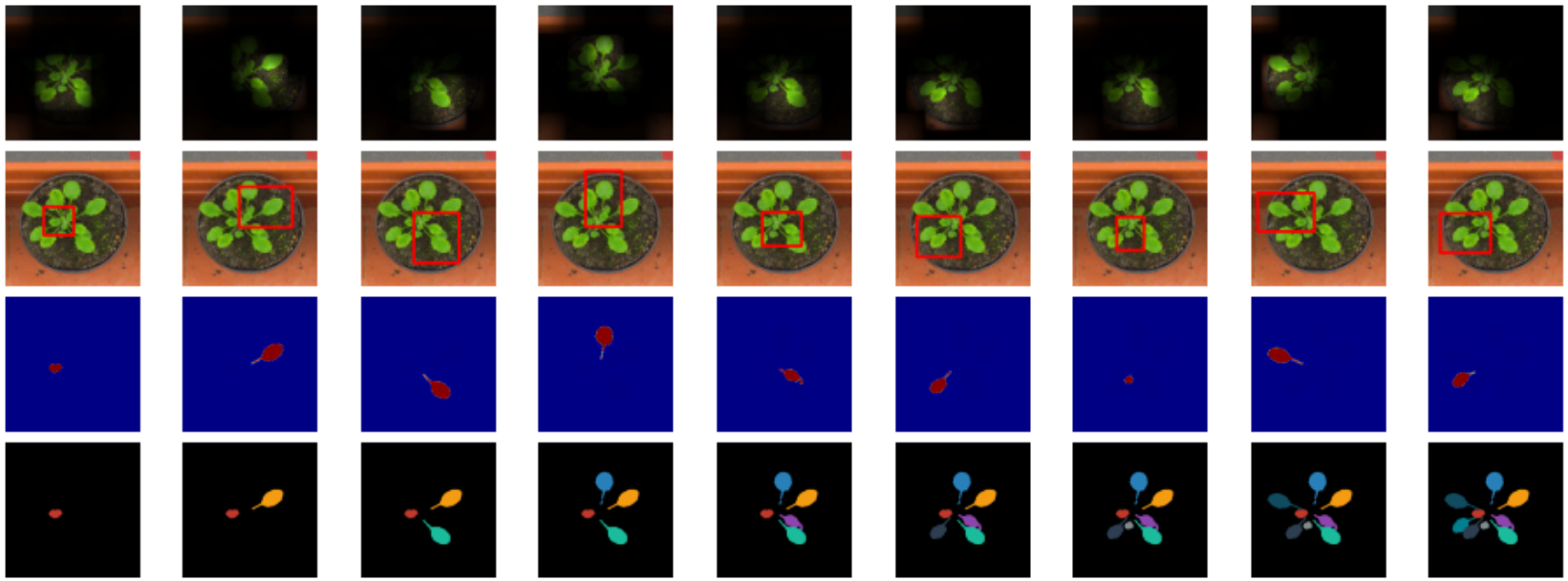
- Semantic mapping at the level of objects
- Combines real-time 3D mapping system with a per-frame CNN semantic predictions in a recursive bayesian loop.
- Able to map large scale areas.

Limitations

What we learnt? [ICRA 2016, CVPR 2016]

- The cost functions are still primitive - per-pixel discrepancy summed up assumes that pixels are independent. Maybe better cost functions that take global context into account ala discriminator in GANs can help.
- Actually, we don't need to segment the whole image at once - attention can help?
- Still hard to get sharp boundaries with segmentation algorithms today.
- Extracting thin structures is still very hard.
- Benchmarking could be misleading...

Attention based scene understanding

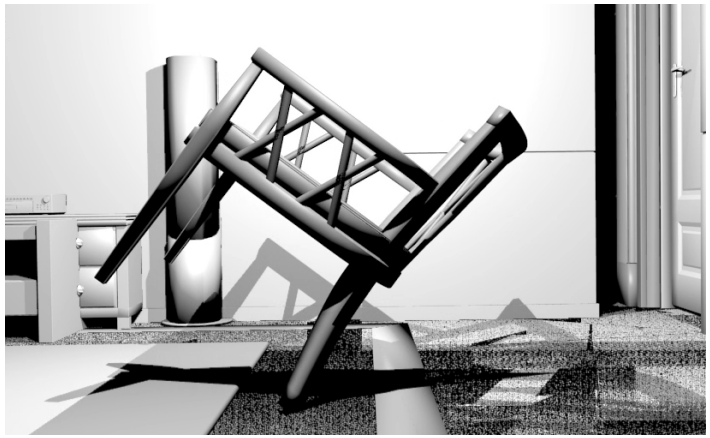


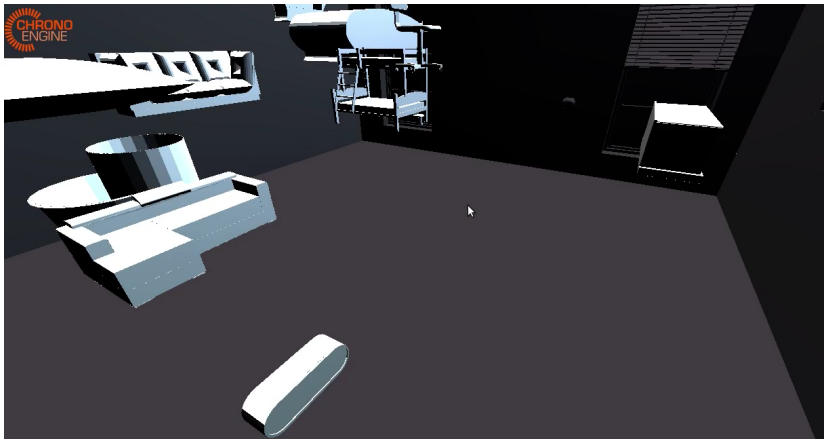
Ren and Zemel, arXiv May 2016.

Attention

- Attention (and glimpses) based scene understanding.
- We don't need to segment the whole image at once.

SceneNet 2.0 and Physics





Adding Physics

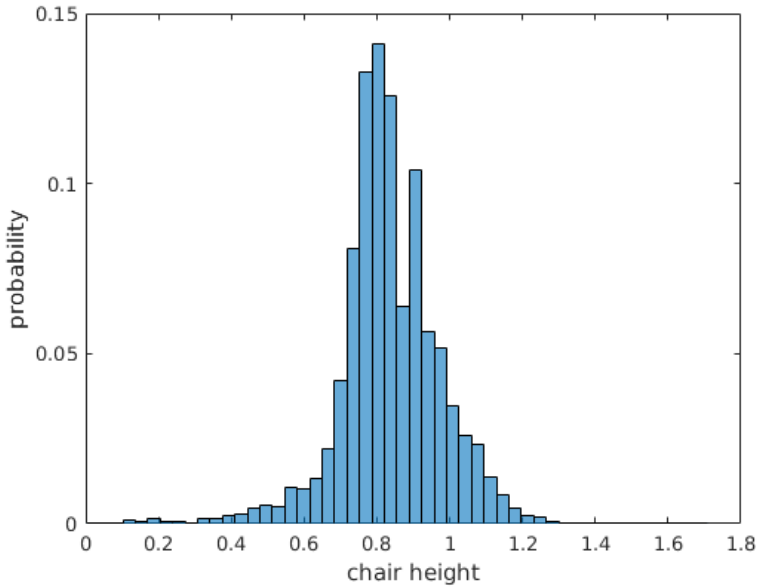
- Since then we have extended SceneNet quite a lot.
- Adding physics allows us to create scenes with arbitrary clutter.
- Placing pens and other small objects on tables is relatively easy with physics.
- We don't want chairs always in upright positions. Physics allows us to explore the space of poses of objects relatively easy.

SceneNet 2.0 and Raytracing

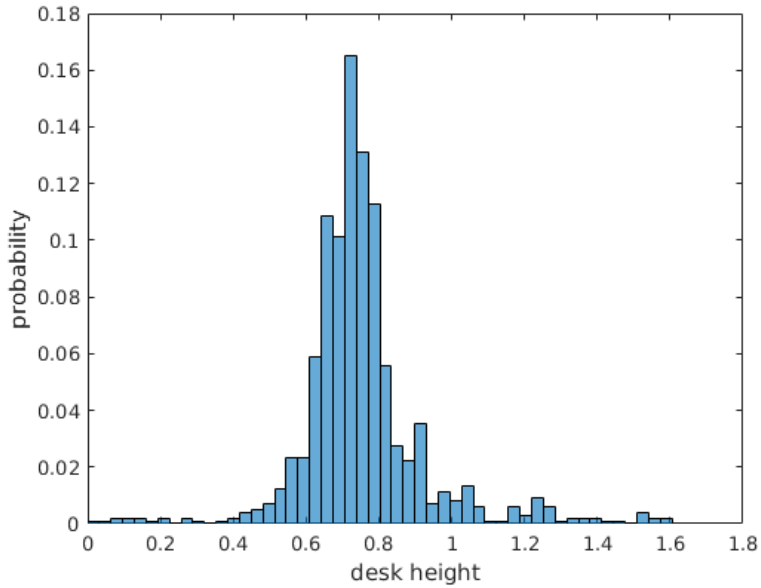
Adding Raytracing

- ModelNet used in our previous experiment was not a big repository.
- Now, we use a much bigger repository, ShapeNets, to sample objects.
- Most CAD repositories do not have models in metric scales.
- Use 3D bounding boxes in SUN RGB-D to get the metric scales of objects.
- Use physics to add clutter in the scenes.
- Render RGB using a customised raytracer on GPU.

SceneNet 2.0 and Raytracing



Probability of chair height

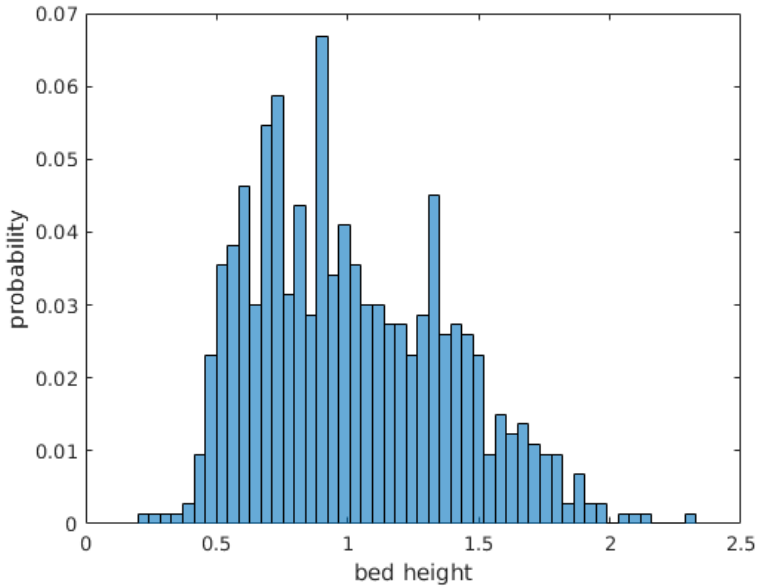


Probability of desk height

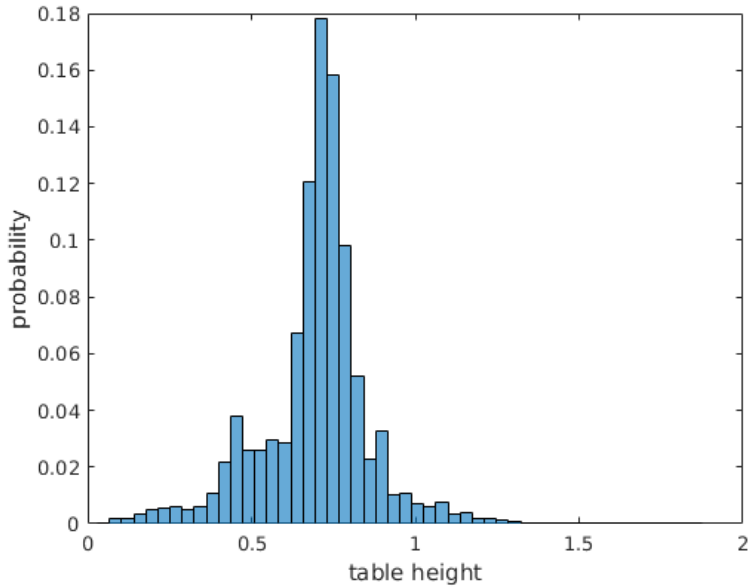
Adding Raytracing

- Use a much bigger repository, ShapeNets, to sample objects.
- Most CAD repositories do not have models in metric scales.
- Use 3D bounding boxes in SUN RGB-D to get the metric scales of objects.
- Use physics to add clutter in the scenes and render using a customised raytracer on GPU.

SceneNet 2.0 and Raytracing



Probability of bed height

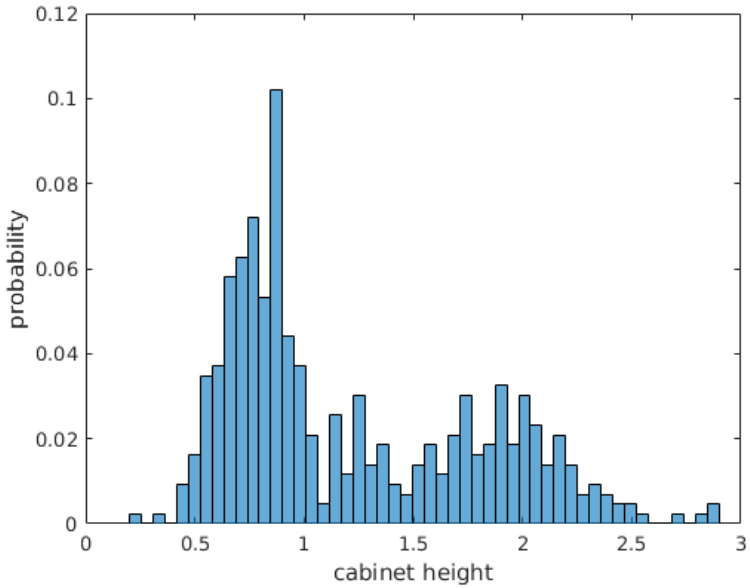


Probability of table height

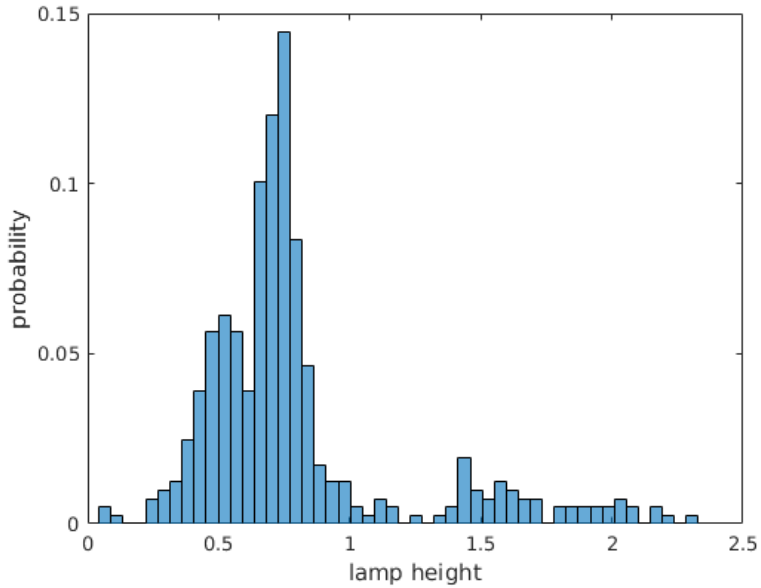
Adding Raytracing

- Use a much bigger repository, ShapeNets, to sample objects.
- Most CAD repositories do not have models in metric scales.
- Use 3D bounding boxes in SUN RGB-D to get the metric scales of objects.
- Use physics to add clutter in the scenes and render using a customised raytracer on GPU.

SceneNet 2.0 and Raytracing



Probability of cabinet height



Probability of lamp height

Adding Raytracing

- Use a much bigger repository, ShapeNets, to sample objects.
- Most CAD repositories do not have models in metric scales.
- Use 3D bounding boxes in SUN RGB-D to get the metric scales of objects.
- Use physics to add clutter in the scenes and render using a customised raytracer on GPU.

SceneNet 2.0 and Raytracing

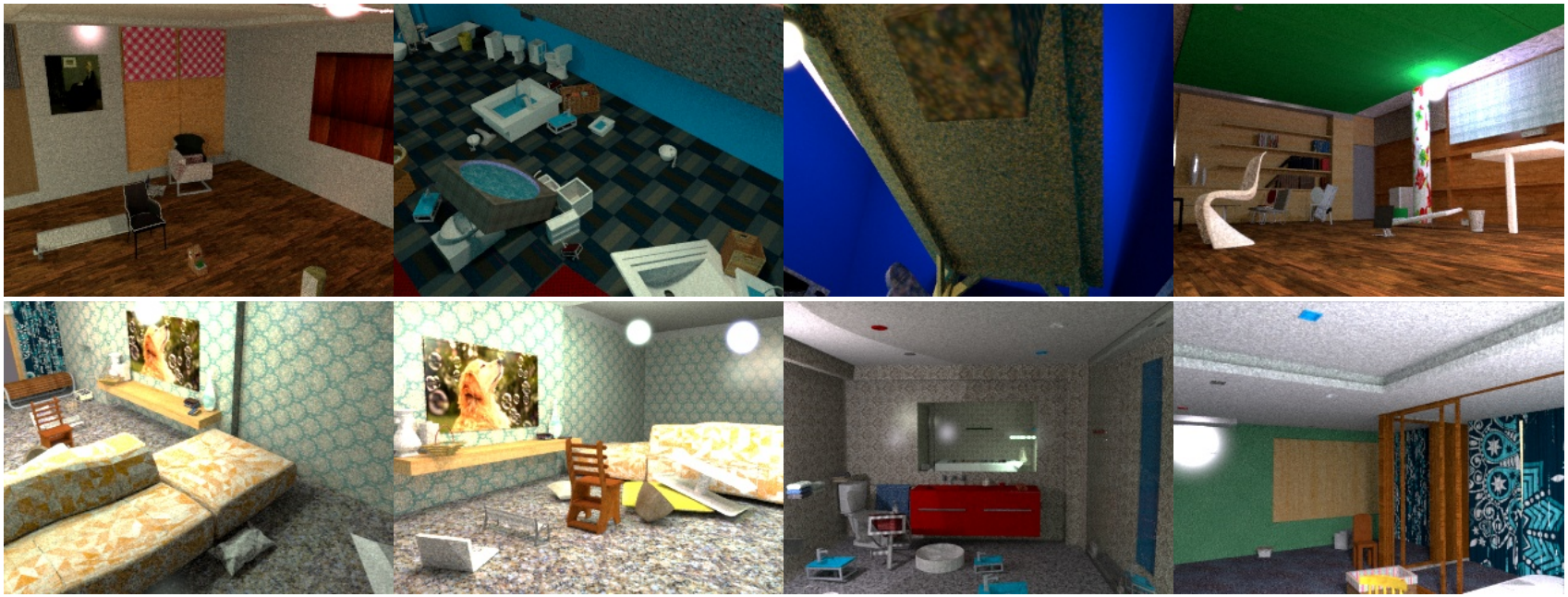
Low-res on laptop

High quality

Adding Raytracing

- Sufficient Photorealism is desirable.
- Depth sensors have limited range.
- RGB is universal and allows us to model various different cameras.
- Need good approximation of shadows, lighting, and various other global lighting artefacts.

SceneNet 2.0 and Raytracing



Randomness

- Textures on the layouts are randomised.
- Object textures are from ShapeNets.
- Random lighting.
- Random positions of objects and random camera trajectories.
- We have 2.5 million rendered images now and rendering.

SceneNet 2.0 and Raytracing

Randomness

- Textures on the layouts are randomised.
- Object textures are from ShapeNets.
- Random lighting.
- Random positions of objects and random camera trajectories.
- We have 2.5 million rendered images now and rendering.

SceneNet 2.0 and Raytracing

Randomness

- Textures on the layouts are randomised.
- Object textures are from ShapeNets.
- Random lighting.
- Random positions of objects and random camera trajectories.
- We have 2.5 million rendered images now and rendering.

SceneNet 2.0 and Raytracing: High quality renders

Randomness

- Textures on the layouts are randomised.
- Object textures are from ShapeNets.
- Random lighting.
- Random positions of objects and random camera trajectories.
- We have 2.5 million rendered images now and rendering.

SceneNet 2.0 and Raytracing: High quality renders

Where can large scale synthetic dataset be useful?

- Getting perfect ground truth poses, and depth and 3D and physics?
- Instance and semantic segmentation ground truth.
- Optical flow ground truth for non-rigid scene understanding.
- Ground truth for transparent objects and surfaces.
- We are working on these problems now.

Outline

- Background
- SceneNet: Repository of Labelled Synthetic Indoor Scenes
- Results on Semantic Segmentation on Real World Data
- gvnn: Neural Network Library for Geometric Vision
- Future Ideas

gvnn: Geometric Vision with Neural Networks

What is gvnn?

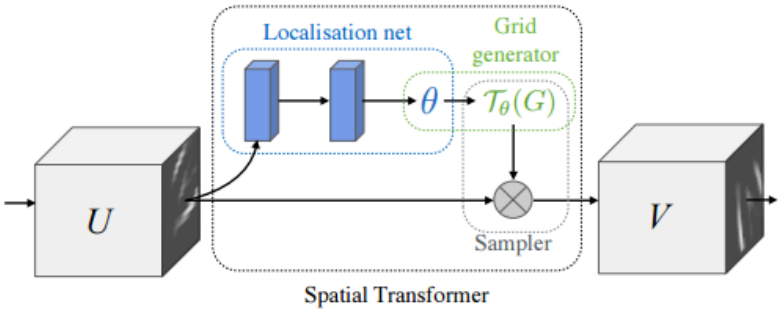
- A new library inspired by Spatial Transformer Networks (STN).
- Implemented in torch.
- Includes various different transformations often used in geometric computer vision.
- These transformations are implemented as new layers that allow backpropagation as in STN.
- Brings together the domain knowledge in geometry within the neural network.

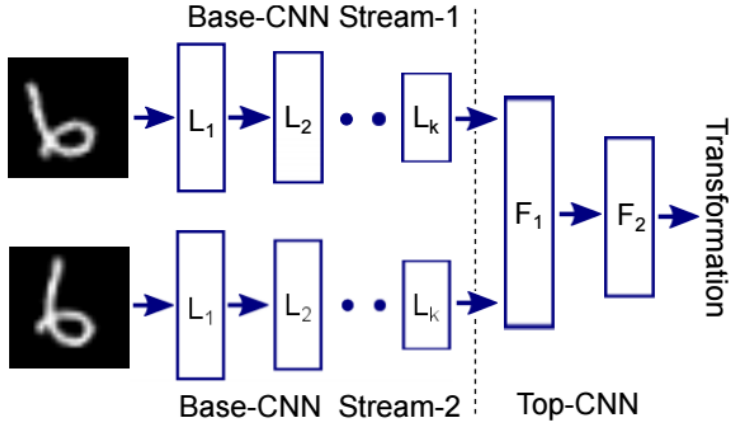
Different new layers in gvnn

What is gvnn?

- Original Spatial Transformer (NIPS 2015) had 2D transformations mainly.
- Added SO3, SE3 and Sim3 layers for global transformation on the image.
- Optical flow, over-parameterised Optical flow, Slanted plane disparity.
- Pin-Hole Camera Projection layer.
- Per-pixel SO3, SE3 and Sim3 for non-rigid registration.
- Different robust M-estimators.
- Very useful for 3D alignment, unsupervised warping with optic flow or disparity and geometric invariance for place recognition.

Inspiration





Notes

- Spatial transformer modules allow to add domain knowledge into the network, *Spatial Transformer Networks, NIPS15*.
- Self-supervised learning can provide general feature representations *Learning to see by moving, ICCV15*.

Global Transformations in gvnn

SO3 Layer (SE3 and Sim3 follow easily from here) for 3D Rotations

$$\frac{\partial \mathcal{C}}{\partial \mathbf{v}} = \frac{\partial \mathcal{C}}{\partial \mathrm{R}(\mathbf{v})} \cdot \frac{\partial \mathrm{R}(\mathbf{v})}{\partial \mathbf{v}} \tag{1}$$

$$\frac{\partial \mathrm{R}(\mathbf{v})}{\partial v_i} = \frac{v_i[\mathbf{v}]_\times + [\mathbf{v} \times (\mathrm{I} - \mathrm{R})e_i]_\times}{||\mathbf{v}||^2}\mathrm{R} \tag{2}$$

Notes

- $\mathcal{C}$ is the cost function being minimised.
- $\mathbf{v} = (v_1, v_2, v_3)$ is the SO3 vector. Note the derivative is not at $v_i \approx 0$ and e_i is the i^{th} column of the Identity matrix.

Global Transformations in gvnn

```
require 'nn'
require 'stn'

concat = nn.ConcatTable()

height = 240
width  = 320
u0     = 160
v0     = 120

fx = 240
fy = 240

-- first branch is there to transpose inputs to BHWD, for the bilinear sampler
tranet=nn.Sequential()
tranet:add(nn.SelectTable(1))
tranet:add(nn.Identity())
tranet:add(nn.Transpose({2,3},{3,4}))

rotation_net = nn.Sequential()
rotation_net:add(nn.SelectTable(2))
rotation_net:add(nn.TransformationRotationSO3())
rotation_net:add(nn.Transform3DPoints_R(height, width, fx, fy, u0, v0))
rotation_net:add(nn.PinHoleCameraProjectionBHWD(height, width, fx, fy, u0, v0))
rotation_net:add(nn.ReverseXYOrder())

concat:add(tranet)
concat:add(rotation_net)

warping_net = nn.Sequential()
warping_net:add(concat)
warping_net:add(nn.BilinearSamplerBHWD())
warping_net:add(nn.Transpose({3,4},{2,3}))
~
```

Notes

- $\mathcal{C}$ is the cost function being minimised.
- $\mathbf{v} = (v_1, v_2, v_3)$ is the SO3 vector. Note the derivative is not at $v_i \approx 0$ and e_i is the i^{th} column of the Identity matrix.

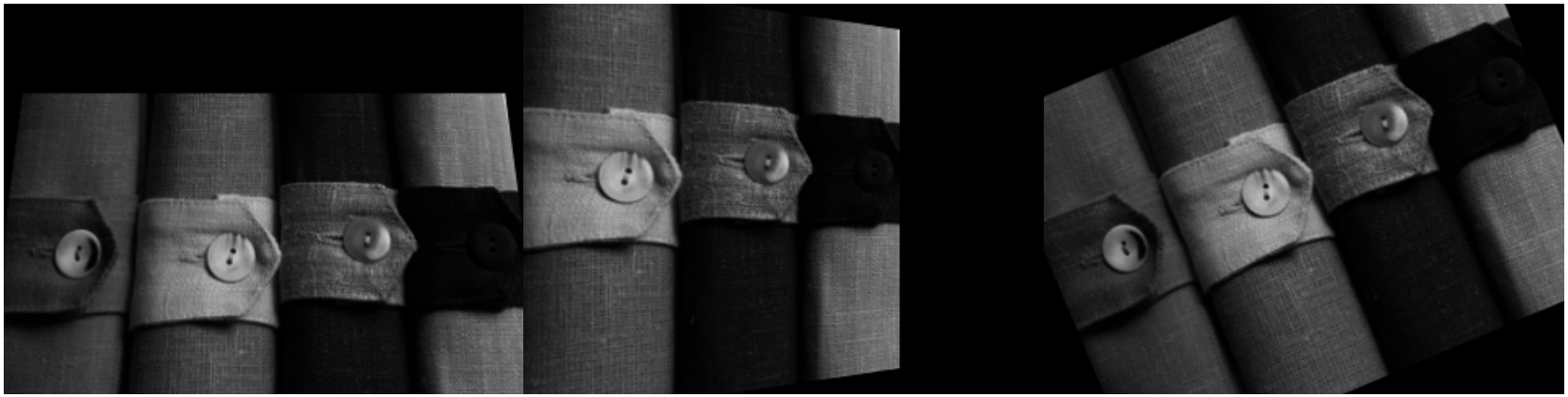
Global Transformations in gvnn

```
require 'image'
require 'nn'
require 'torch'

dofile('imagewarping.lua')

x = image.loadPNG('linen1.png')
input = torch.Tensor(1,1,240,320)
input[1] = x

r = torch.Tensor(1,3):zero()
r[1][1] = 0.2

t = {input, r}

out_w = warping_net:forward(t)
w = out_w[1]

image.display(w)
image.save('warped.png', w)
~
```

Notes

- $\mathcal{C}$ is the cost function being minimised.
- $\mathbf{v} = (v_1, v_2, v_3)$ is the SO3 vector. Note the derivative is not at $v_i \approx 0$ and e_i is the i^{th} column of the Identity matrix.

Global Transformations in gvnn



(a) $\mathbf{v} = (0.2, 0.0, 0.0)$ (b) $\mathbf{v} = (0.0, 0.3, 0.0)$ (c) $\mathbf{v} = (0.0, 0.0, 0.4)$

Notes

- $\mathcal{C}$ is the cost function being minimised.
- $\mathbf{v} = (v_1, v_2, v_3)$ is the SO3 vector. Note the derivative is not at $v_i \approx 0$ and e_i is the i^{th} column of the Identity matrix.

Per-pixel 2D Transformations in gvnn

Optical flow and Over-parameterised Optical Flow

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \end{pmatrix} \tag{3}$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_0 & a_1 & a_2 \\ & & \\ a_3 & a_4 & a_5 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{4}$$

Notes

- Over-parameterised optical flow also needs extra regularisation.
- 2D and 6D transformations per-pixel. Useful for warping images (and unsupervised learning), Patraucean, *et al.* ICLRW2016.

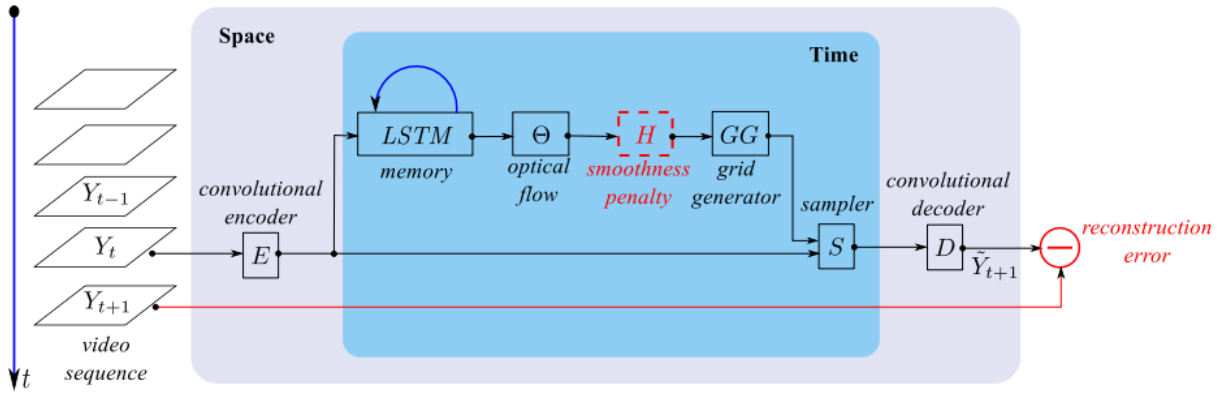
Per-pixel 2D Transformations in gvnn

Optical flow warping with gvnn

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \end{pmatrix} \tag{5}$$

- Useful for warping one image on top of other - learning optical flow with a CNN.
- Self-supervised learning, Patraucean, *et al.* ICLRW2016.

Per-pixel 2D Transformations in gvnn



Optical flow warping with gvnn

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \end{pmatrix} \tag{6}$$

- Useful for warping one image on top of other - learning optical flow with a CNN.
- Self-supervised learning, Patraucean, *et al.* ICLRW2016.

Per-pixel 2D Transformations in gvnn

Disparity and Slated Plane disparity

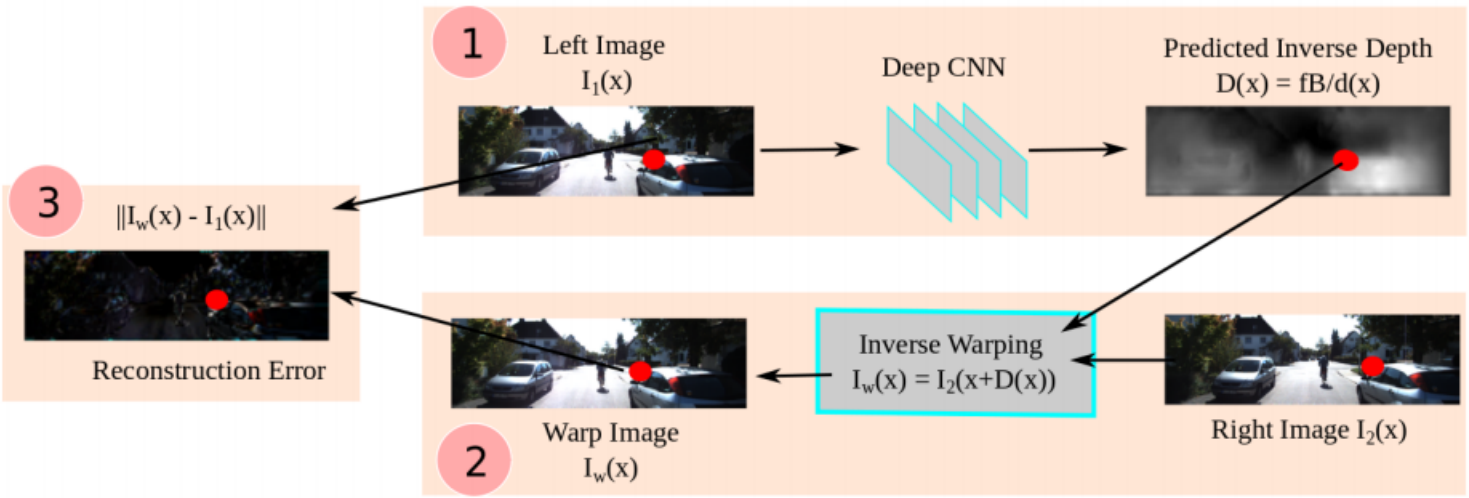
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x + d \\ y \end{pmatrix} \tag{7}$$

$$d = ax + by + c \tag{8}$$

Notes

- Fitting slanted planes at each pixel.
- Again, very useful for warping images (and unsupervised learning), Garg *et al.* ECCV2016.

Per-pixel 2D Transformations in gvnn



Notes

- Fitting slanted planes at each pixel.
- Again, very useful for warping images (and unsupervised learning), Garg *et al.* ECCV2016.

Camera Projection Layer in gvnn

Pin Hole Camera Projection Layer

$$\pi \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} f_x \frac{u}{w} + p_x \\ f_y \frac{v}{w} + p_y \end{pmatrix} \tag{9}$$

$$\frac{\partial \mathcal{C}}{\partial \mathbf{p}} = \frac{\partial \mathcal{C}}{\partial \pi(\mathbf{p})} \cdot \frac{\partial \pi(\mathbf{p})}{\partial \mathbf{p}} \tag{10}$$

$$\frac{\partial \pi \begin{pmatrix} u \\ v \\ w \end{pmatrix}}{\partial \begin{pmatrix} u \\ v \\ w \end{pmatrix}} = \begin{pmatrix} f_x \frac{1}{w} & 0 & -f_x \frac{u}{w^2} \\ 0 & f_y \frac{1}{w} & -f_y \frac{v}{w^2} \end{pmatrix} \tag{11}$$

- Useful in differentiable renderer ala OpenDR.
- Reasoning from 3D to 2D.

Camera Projection Layer in gvnn

Rezende *et al.*, arXiv 2016

Notes

- Useful in differentiable renderer ala OpenDR.
- Reasoning from 3D to 2D.

Non-rigid per-pixel transformation

Non-rigid registration for point clouds: 6DoF and 10DoF

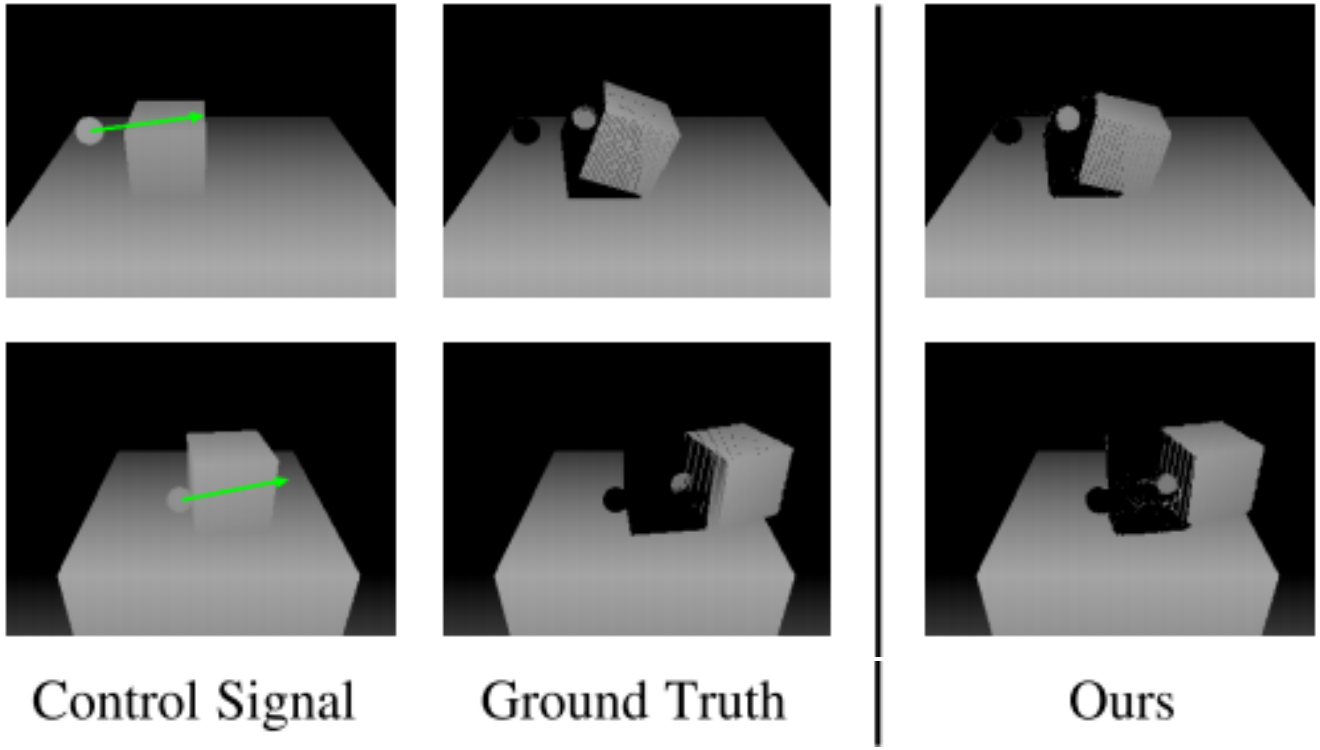
Per-pixel 6DoF and 10DoF transformations.

$$\begin{pmatrix} x_i' \\ y_i' \\ z_i' \end{pmatrix} = \mathsf{T}_i \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} \tag{12}$$

$$\mathbf{x}_i' = s_i(\mathsf{R}_i(\mathbf{x}_i - \mathbf{p}_i) + \mathbf{p}_i) + \mathbf{t}_i \tag{13}$$

Also useful for volumetric spatial transformers when using voxel grid representation.

Non-rigid per-pixel transformation

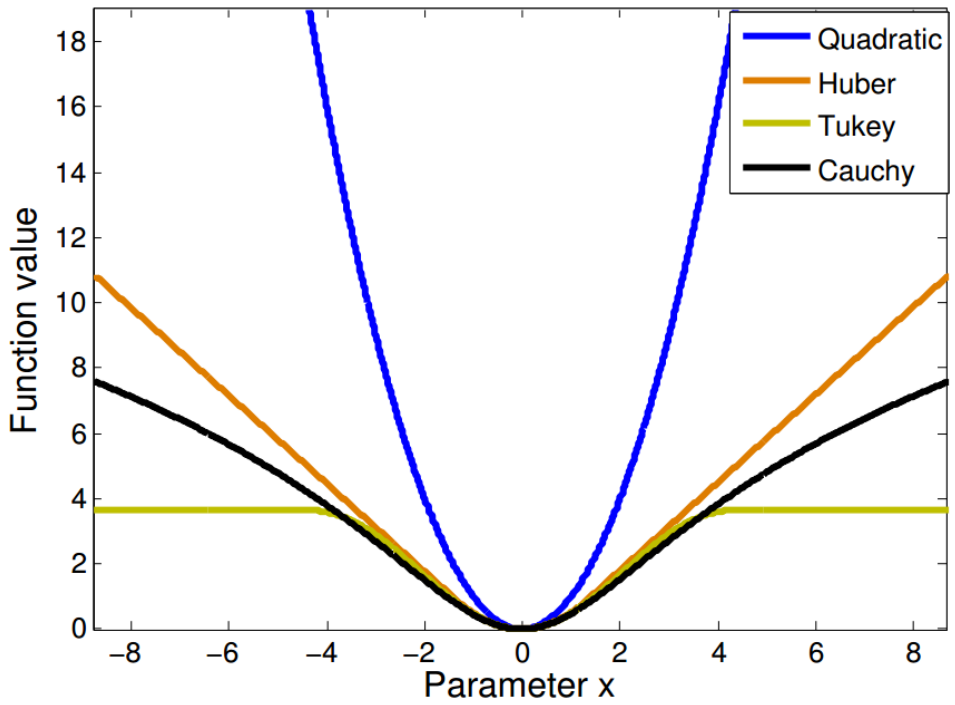


SE3-Nets Byravan and Fox, arXiv 2016

Non-rigid prediction

- Useful for predicting the dynamics of the objects.

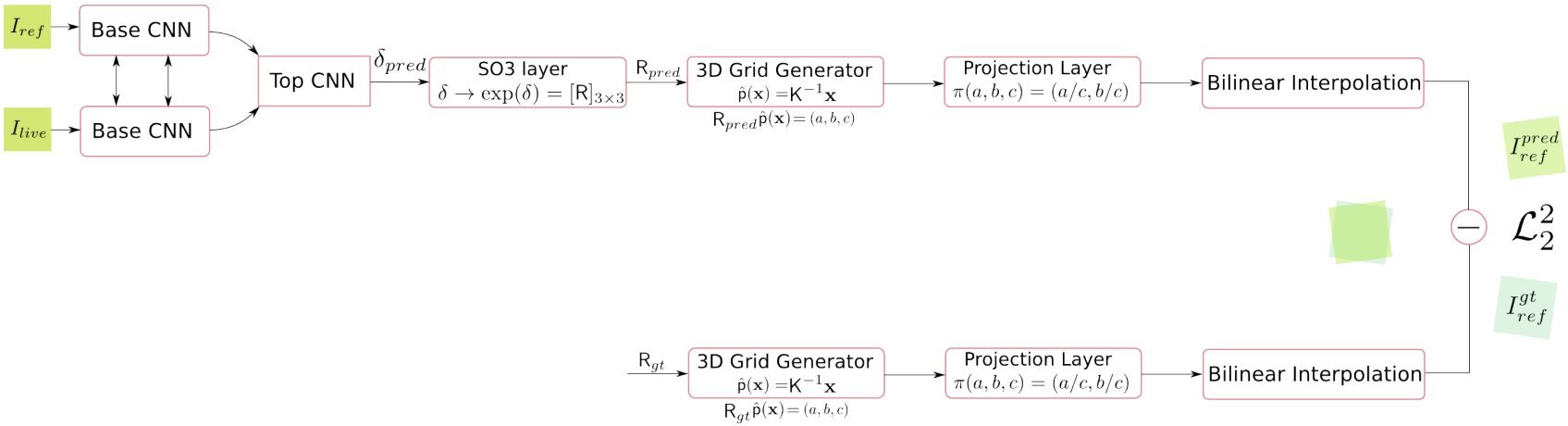
M-estimators



Notes

- Different M-estimators for robust outlier rejection.
- Very useful when doing regression.
- Implemented as layers in gvnn.

Sanity Checks on End-to-End Image Registration (SO3)



Notes

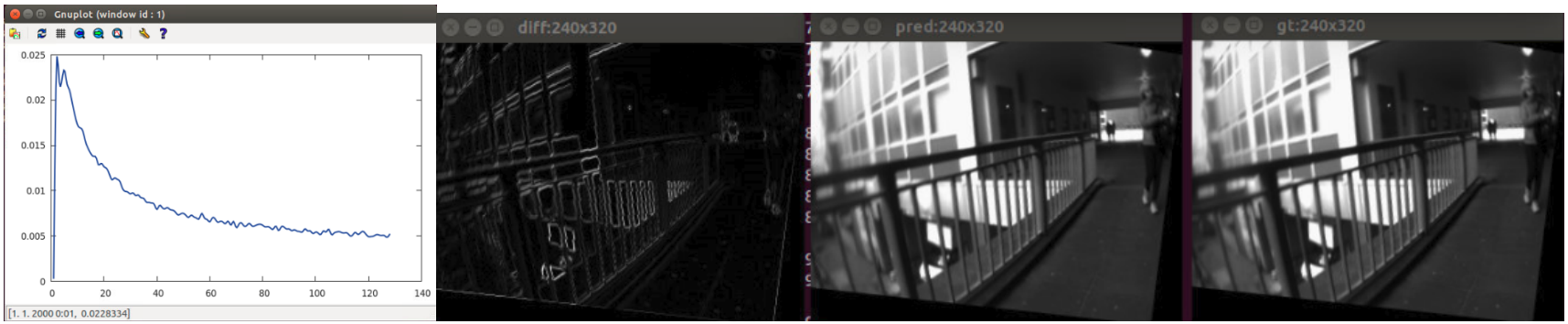
- General dense image registration with global transformation.
- Can use optical flow/disparity for dense per-pixel image registration either with a CNN or RNN.
- Initial experiments with supervised learning.

Sanity Checks on End-to-End Image Registration (SO3)

Notes

- General dense image registration with global transformation.
- Can use optical flow/disparity for dense per-pixel image registration either with a CNN or RNN.

Sanity Checks on End-to-End Image Registration (SO3)



Notes

- General dense image registration with global transformation.
- Can use optical flow/disparity for dense per-pixel image registration either with a CNN or RNN.

Sanity Checks on End-to-End Image Registration (SO3)

Notes

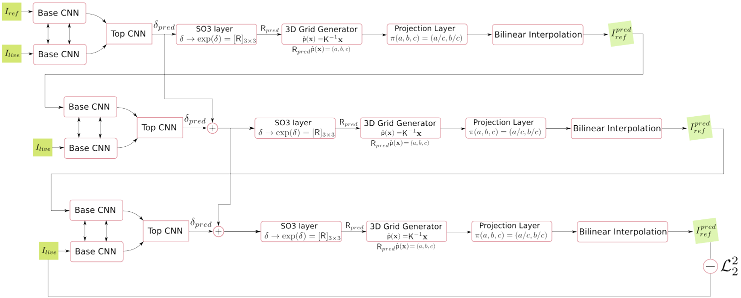
- General dense image registration with global transformation.
- Can use optical flow/disparity for dense per-pixel image registration either with a CNN or RNN.

Sanity Checks on End-to-End Image Registration (SO3)

Notes

- General dense image registration with global transformation.
- Can use optical flow/disparity for dense per-pixel image registration either with a CNN or RNN.
- Aim to train on large scale data from SceneNet and RGB videos for unsupervised learning.

Sanity Checks on End-to-End Image Registration (SO3)



$$\delta_{update} = f_{siamese}(\mathcal{I}_t^k, \mathcal{I}_{t+1}) \quad (14)$$

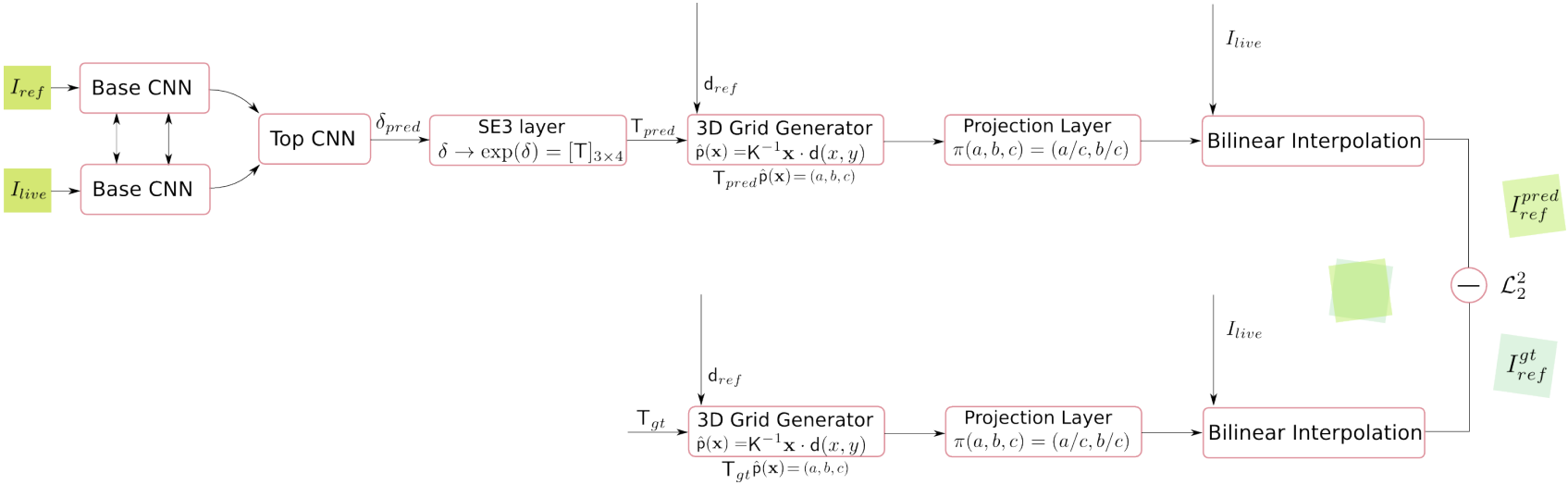
$$\delta_k = \delta_{k-1} + \delta_{update} \quad (15)$$

$$\hat{\mathcal{I}}_t^k = f_{warping}(\mathcal{I}_t, \delta_k) \quad (16)$$

$$\hat{\mathcal{I}}_t^0 = \mathcal{I}_t \quad (17)$$

$$\delta_0 = \mathbf{0} \quad (18)$$

Sanity Checks on End-to-End Image Registration (SE3)



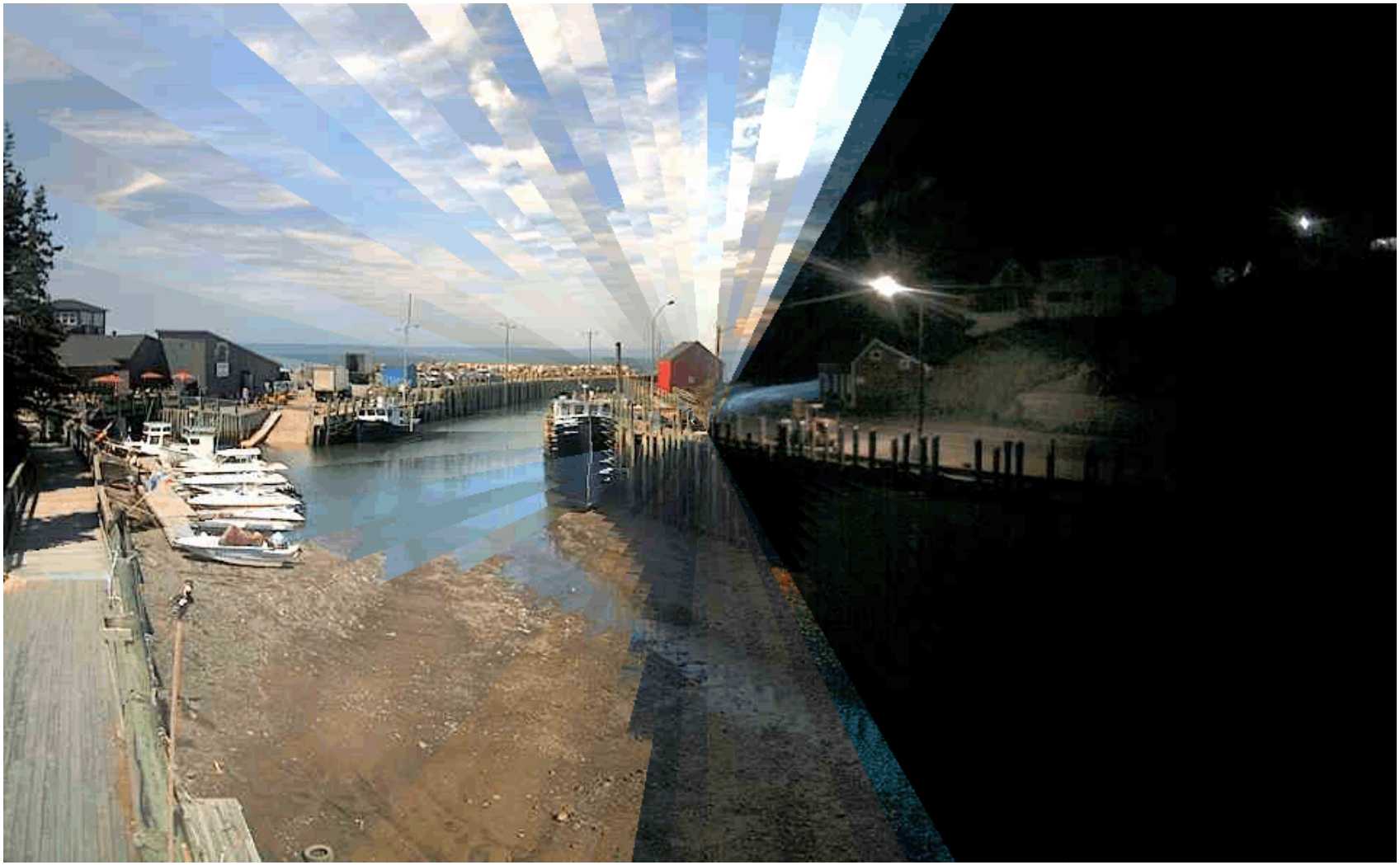
Notes

- Needs explicitly the depth to do warping.
- This can either come from a sensor or an extra CNN/RNN module that learns depth.

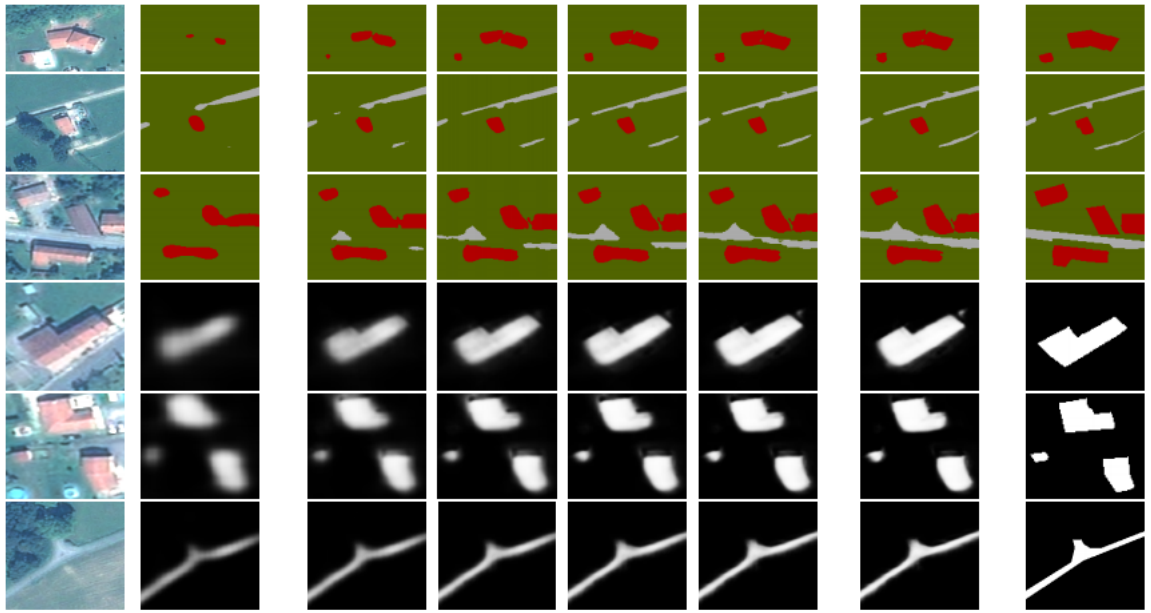
Sanity Checks on End-to-End Image Registration (SE3)

(a) Prediction (b) Ground Truth (c) Residual (difference)

Aligning images taken at different times of the day



New libraries: gvrnn



Maggiori *et al.*, arXiv 2016

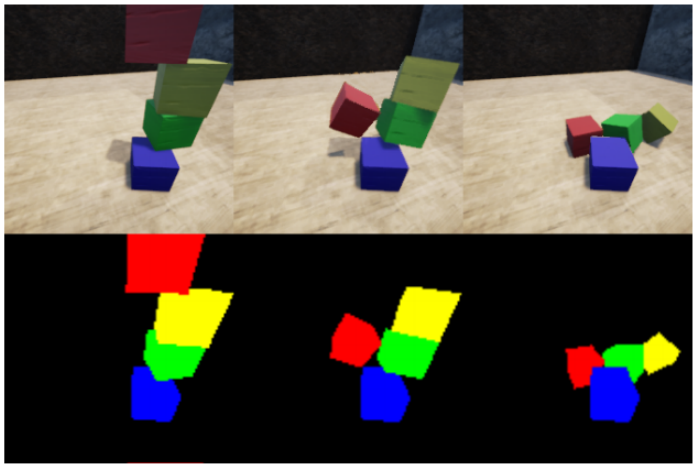
gvrnn

- CNNs are good at localisation but not good with accuracy.
- Implements various PDE solvers as layers in torch.
- Primal-dual optimisations (Pock et al. 2011).
- Iterative image alignment.

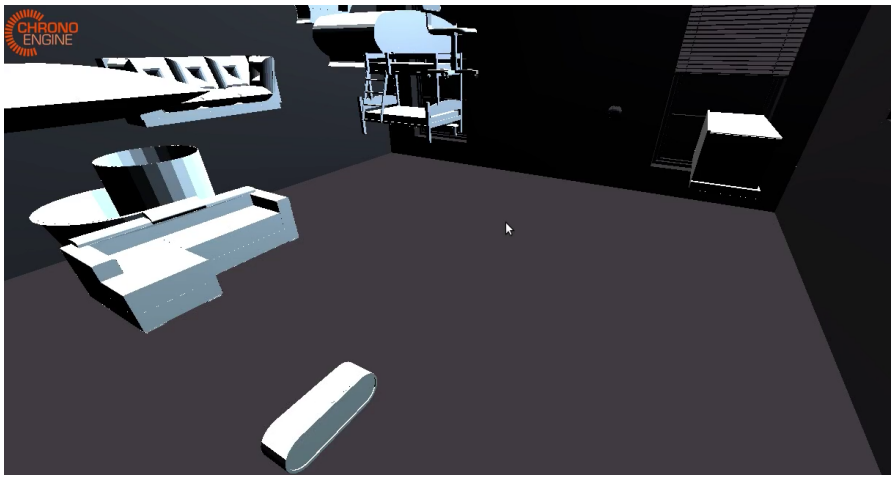
Outline

- Background
- SceneNet: Repository of Labelled Synthetic Indoor Scenes
- Results on Semantic Segmentation on Real World Data
- gvnn: Neural Network Library for Geometric Vision
- Future Ideas

Future Ideas



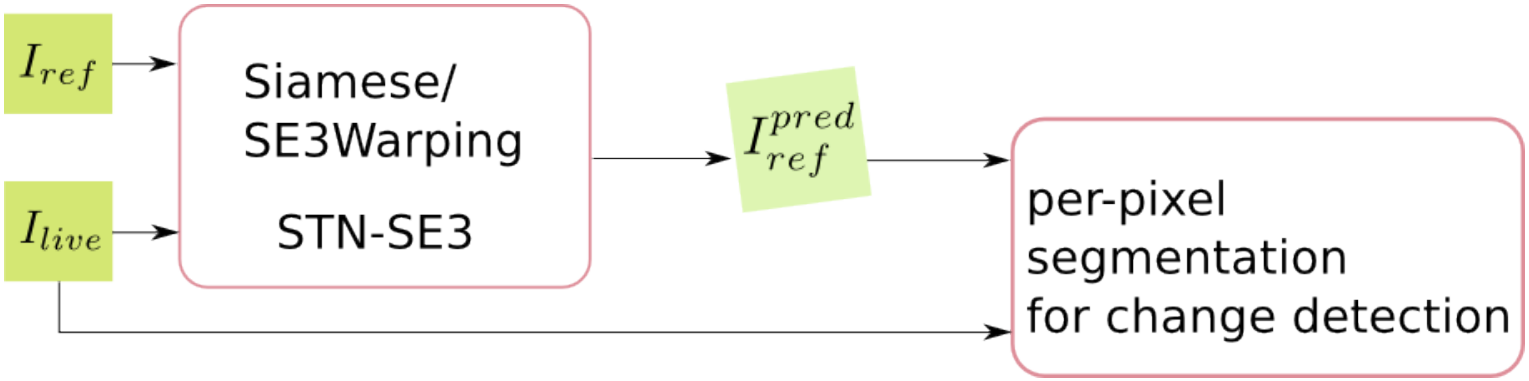
PhysNet, Lerer *et al.* ICML 2016



SceneNet2.0

- Physical scene understanding with SceneNet to reason about how the dynamics of the scene is going to evolve over time.

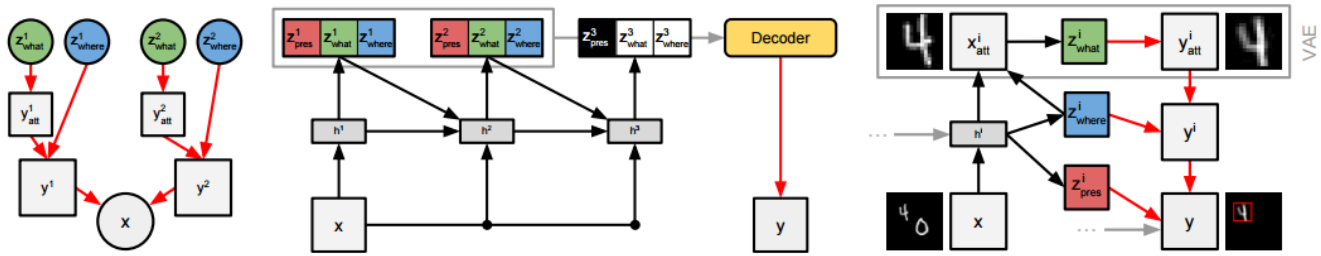
Future Ideas



Deep learning with geometry: Example

- Understanding dynamic scenes.
- CNNs provide relatively stable features for images of same scene taken across different lighting conditions.
- These images cannot be aligned with geometry based per-pixel dense image alignment methods.
- We can put this together with change detection segmentation to also reason about dynamic scenes.
- Easy to collect data with SceneNet.

Future Ideas



- Attend-Infer-Repeat (Eslami *et al.* NIPS 2016) style 3D scene understanding in the form of instances of objects and their 6DoF poses even for cluttered scenes, with recurrent neural networks.
- Solving 3D jigsaws as a means to unsupervised learning. Computer vision architecture and computer graphics architecture put together in the spirit of Variational auto-encoder. Computer vision gives the poses and identities of objects and computer graphics is a differentiable renderer .

Acknowledgements

Thank you

- Vijay Badrinarayanan, Viorica Patraucean, Simon Stent, and Roberto Cipolla, University of Cambridge
- Zoubin Ghahramani, University of Cambridge
- John McCormac and Andrew Davison, Imperial College London
- Daniel Cremers, TUM Munich