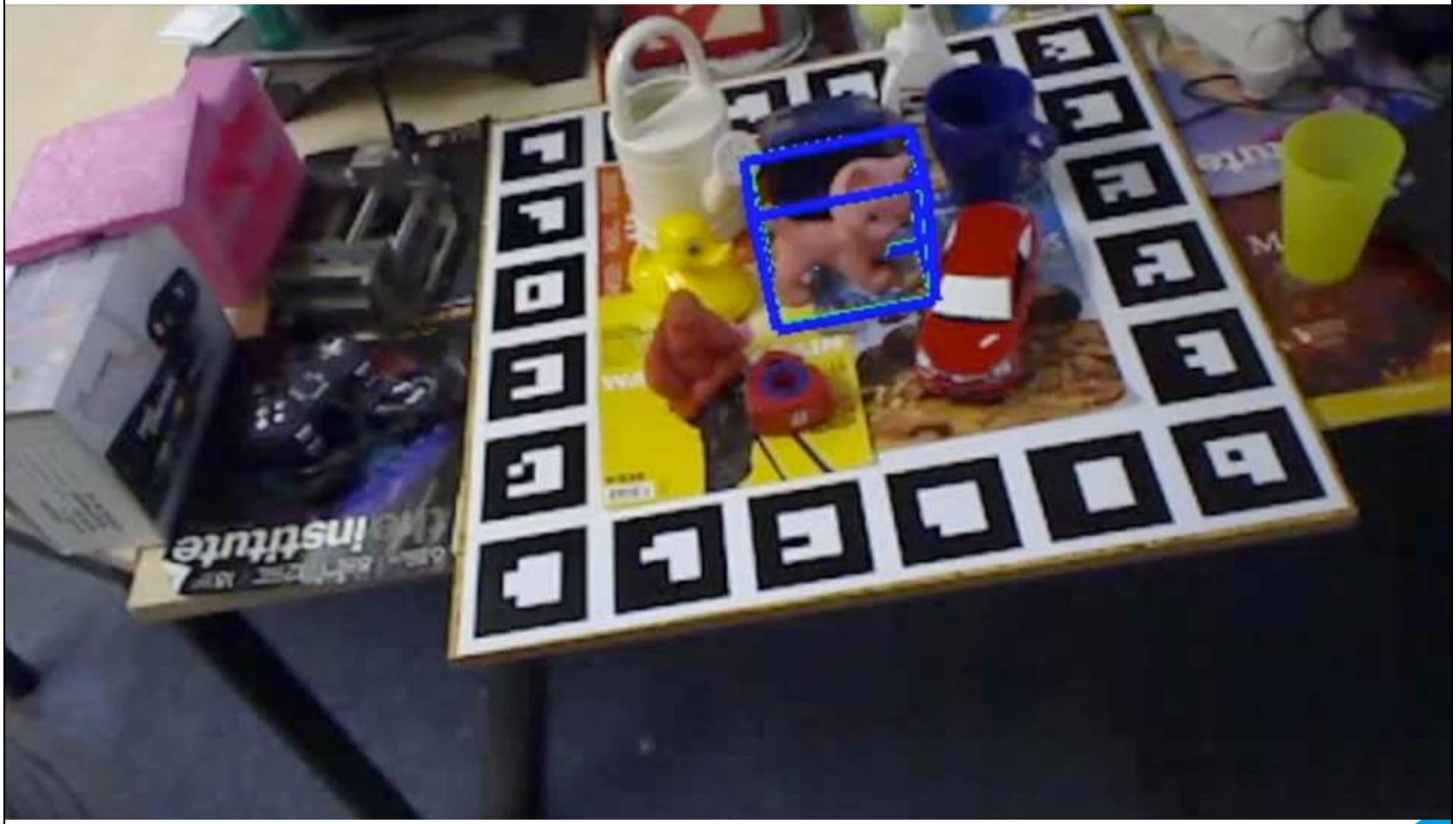
# Deep Learning for 3D Localization

Vincent Lepetit
University of Bordeaux, France & TU Graz, Austria

université de BORDEAUX

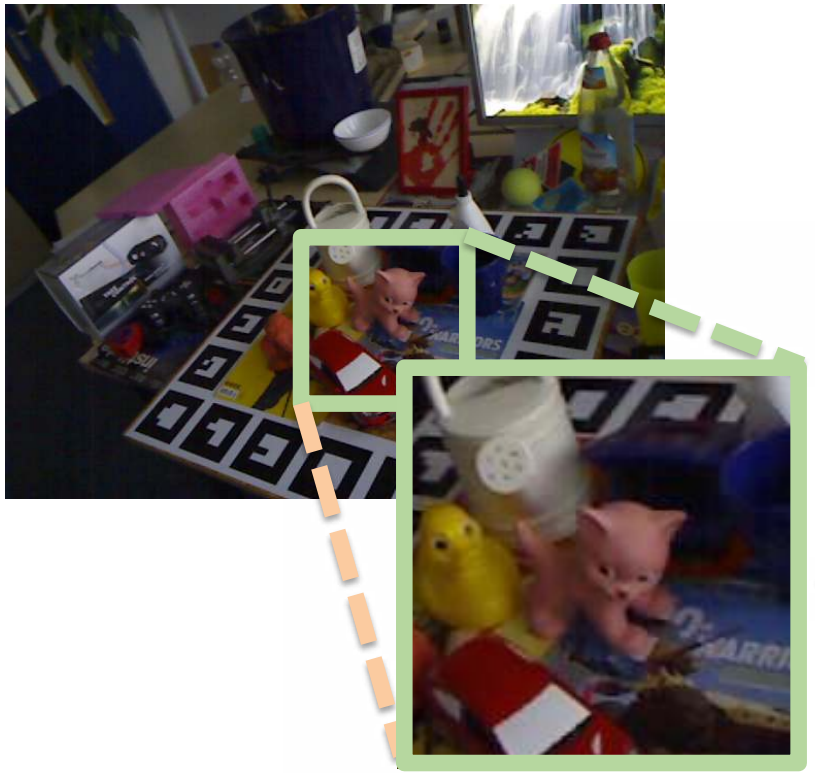
- 3D object detection from color images;
- Accurate geolocalization without registered images.

# BB8: 3D Pose Without Using Depth



BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth. Mahdi Rad, Vincent Lepetit, ICCV 2017.

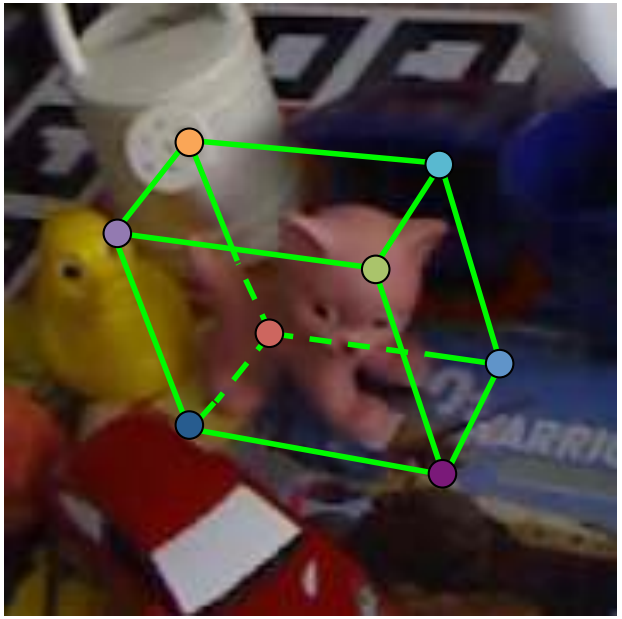
# Predicting the 3D Pose given a 2D Location of the object

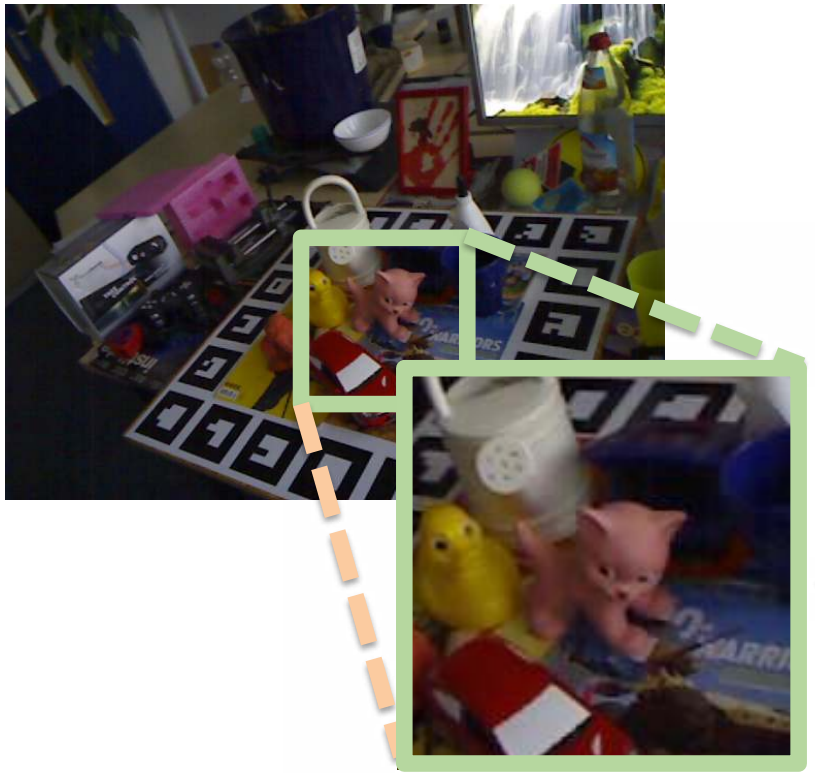


**Solution #1**: Directly predicting the pose

CNN → (Exponential Map / quaternion / rotation matrix, Translation)

# We Can Do Better



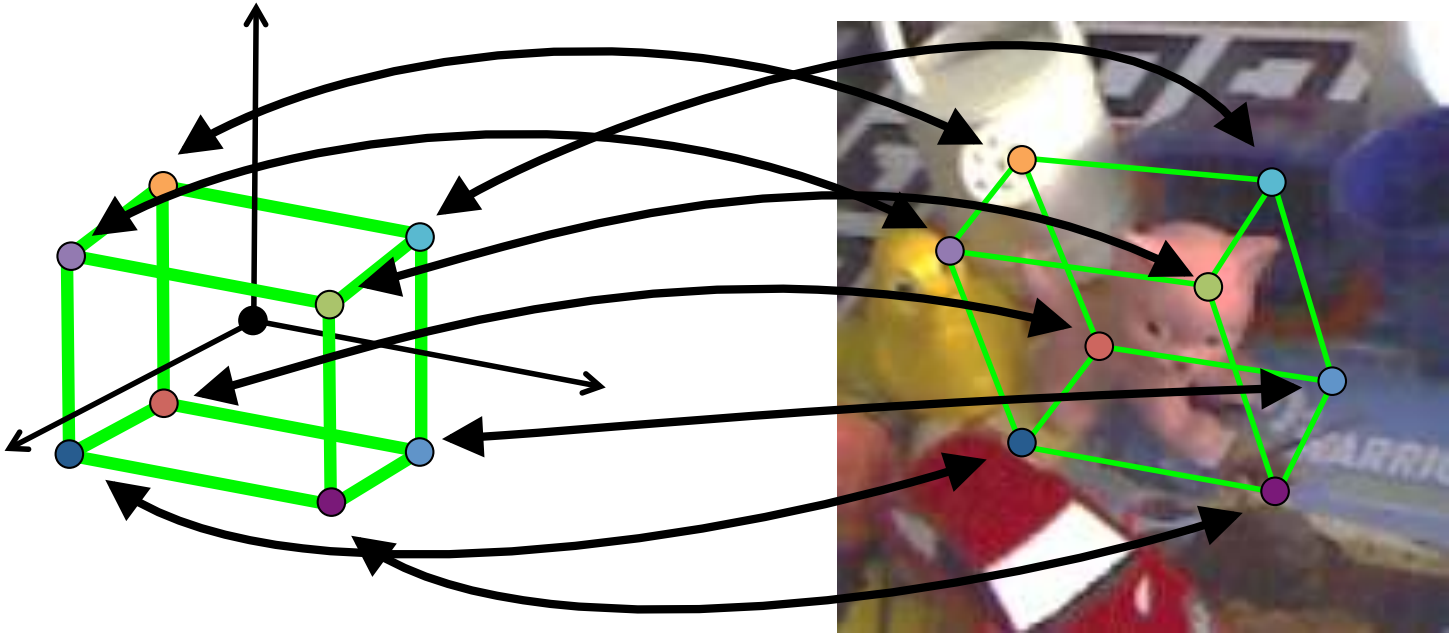


CNN

(the 2D projections of the 8 corners of the 3D bounding box)

Predicting 2D locations from an image is an easier regression task;
We can compute the 3D pose from these 2D locations.

# Getting the 3D Pose

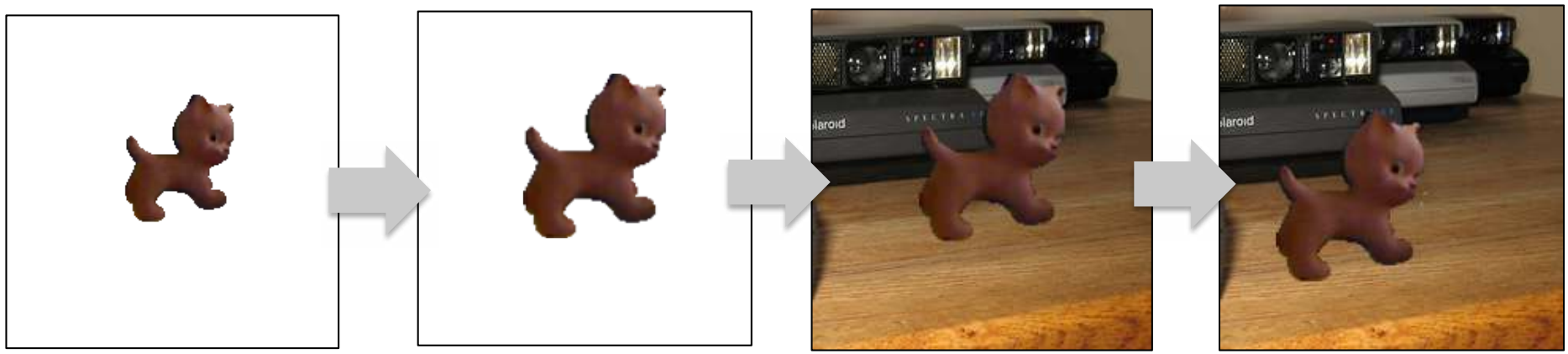


→ we can compute the 3D pose using a PnP algorithm.

# Training Set

Split the LINEMOD data: 15% of images for training, 85% for testing.

Augmentation (200,000 images in total):



extraction from real image

random scaling

random background

random translation

Other examples:

OR

VGG + retraining of the fully connected layers + fine-tuning of the last convolutional layers, can handle all 15 objects of the LINEMOD dataset.

# Direct Pose Estimation VS BB8

2D projection metric

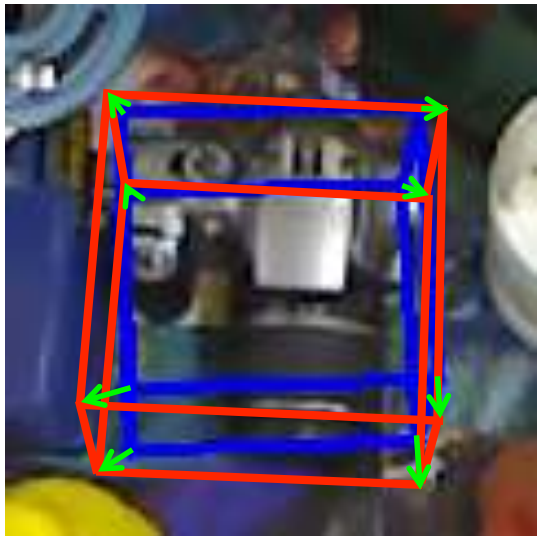
| | Direct Pose | BB8 |
|---|---|---|
| Average | 64.9 | **85.4** |
| Ape* | 91.2 | **96.2** |
| Bench Vise | 61.3 | **80.2** |
| Camera | 43.1 | **82.8** |
| Can | 62.5 | **85.8** |
| Cat* | 93.1 | **97.2** |
| Driller* | 46.5 | **77.6** |
| Duck | 67.9 | **84.6** |
| Egg Box | 68.2 | **90.1** |
| Glue | 69.3 | **93.5** |
| Hole Puncher | 78.2 | **91.7** |
| Iron | 64.5 | **79.0** |
| Lamp | 50.4 | **79.9** |
| Phone | 46.9 | **80.0** |

# Refining the Pose

input image

3D model rendered from the current pose estimate

$CNN_2$



(2D displacements improving the current estimates of the corners' projections)

# Refining the Pose

| | BB8 | BB8 + Refinement |
|---|---|---|
| Average | 85.4 | **91.7** |
| Ape | 96.2 | **97.6** |
| Bench Vise | 80.2 | **92.0** |
| Camera | 82.8 | **88.3** |
| Can | 85.8 | **93.7** |
| Cat | 97.2 | **98.7** |
| Driller | 77.6 | **83.4** |
| Duck | 84.6 | **94.1** |
| Egg Box | 90.1 | **93.4** |
| Glue | 93.5 | **96.0** |
| Hole Puncher | 91.7 | **97.4** |
| Iron | 79.0 | **85.2** |
| Lamp | 79.9 | **83.8** |
| Phone | 80.0 | **88.8** |

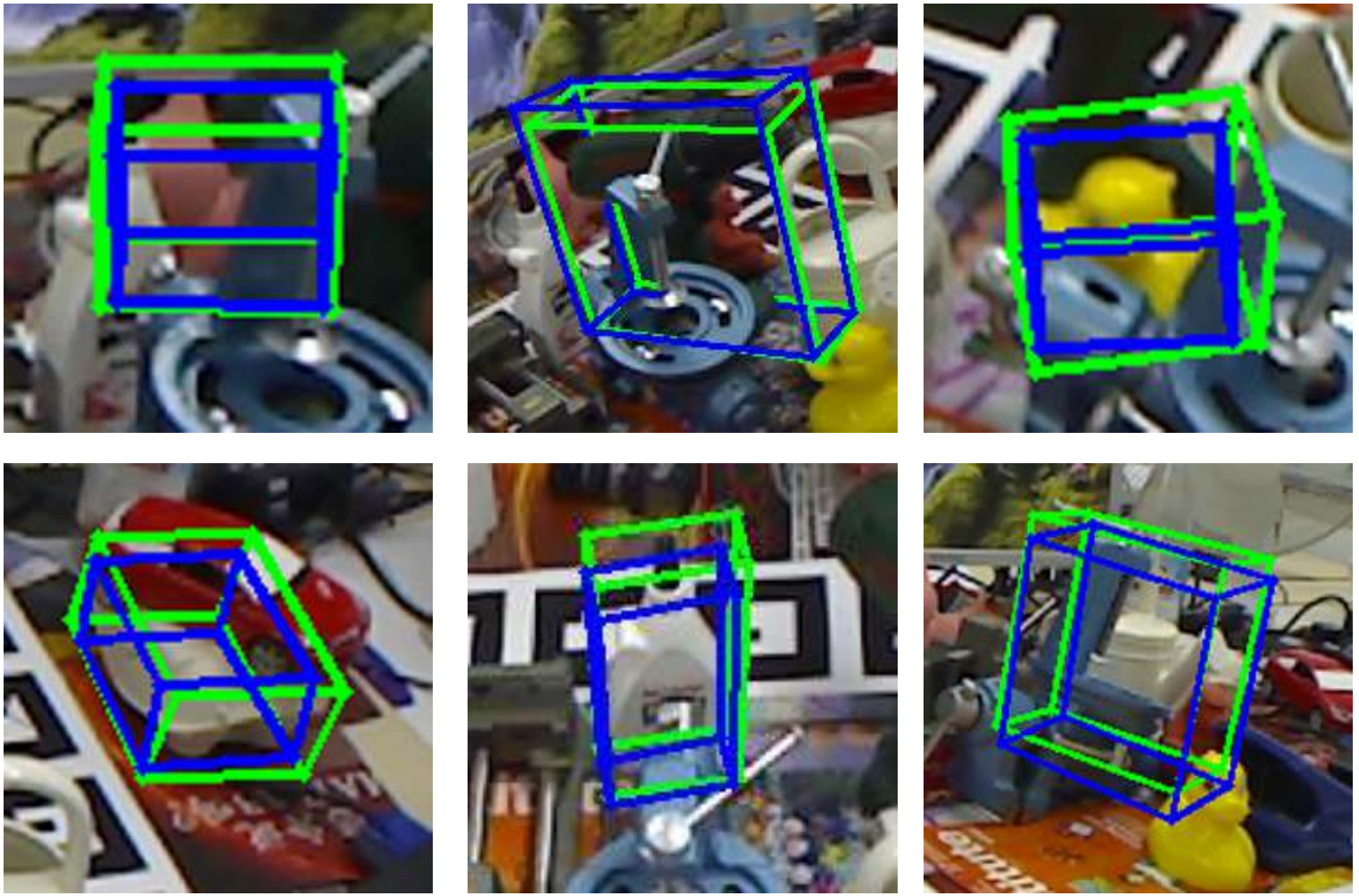
# Others [Brachmann16] VS Ours

| Metric | 2D Projection | | 6D Pose | | 5cm and 5° | |
|---|---|---|---|---|---|---|
| Sequence | others | ours | others | ours | others | ours |
| Average | 73.7 | **89.4** | 50.2 | **62.8** | 40.6 | **69.0** |
| Ape | 85.2 | **96.5** | 33.2 | **40.2** | 34.4 | **80.0** |
| Bench Vise | 67.9 | **91.0** | 64.8 | **92.0** | 40.6 | **81.8** |
| Camera | 58.7 | **86.2** | 38.4 | **56.2** | 30.5 | **60.3** |
| Can | 70.8 | **92.1** | 62.9 | **64.6** | 48.4 | **77.1** |
| Cat | 84.2 | **98.7** | 42.7 | **62.3** | 34.6 | **79.6** |
| Driller | 73.9 | **80.7** | 61.9 | **74.1** | 54.5 | **69.3** |
| Duck | 73.1 | **92.4** | 30.2 | **44.8** | 22.0 | **53.6** |
| Egg Box | 83.1 | **91.1** | 49.9 | **58.1** | 57.1 | **81.3** |
| Glue | 74.2 | **92.5** | 31.2 | **41.6** | 23.6 | **54.2** |
| Hole Puncher | 78.9 | **95.1** | 52.8 | **67.1** | 47.3 | **73.1** |
| Iron | 83.6 | **85.0** | 80.0 | **84.9** | 58.7 | **61.3** |
| Lamp | 64.0 | **75.5** | 67.0 | **76.3** | 49.3 | **67.4** |
| Phone | 60.6 | **85.1** | 38.1 | **53.9** | 26.8 | **58.4** |

# Robustness to Partial Occlusion

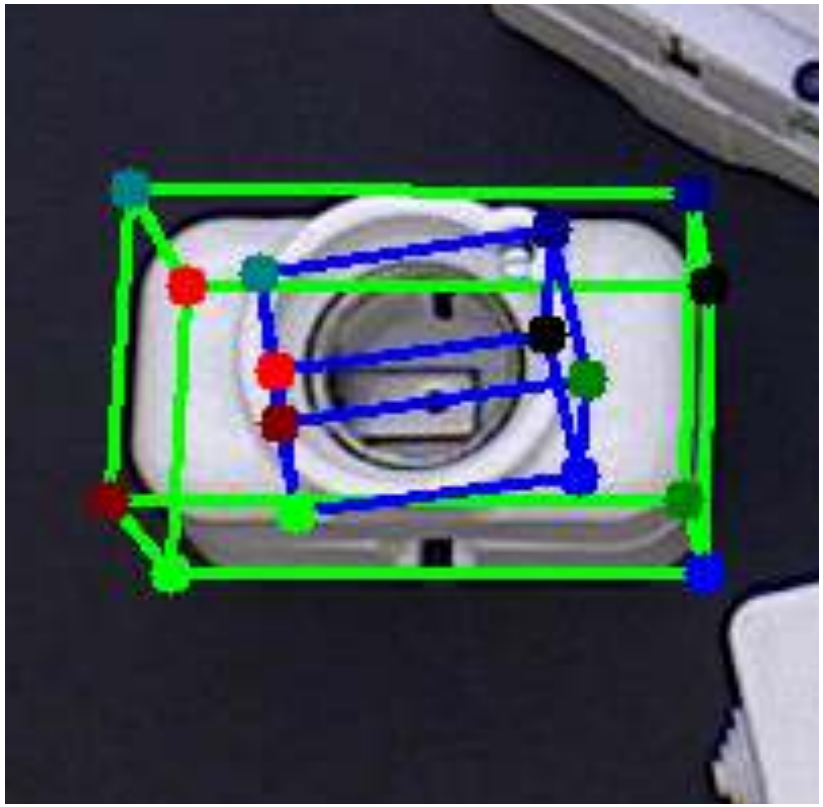
When generating training images, we randomly superimpose objects from other sequences to the target object to be robust to occlusion:

# Robustness to Partial Occlusion



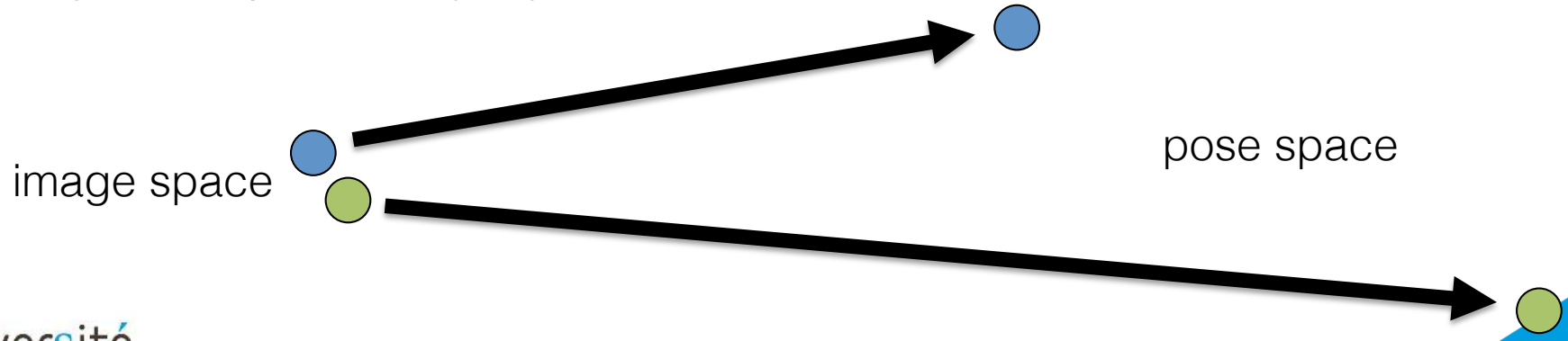
# (Almost) Symmetric Objects

# T-Less Dataset [Hodan et al]: (Almost) Symmetric Objects



# What Is the Problem?

The network is trained to predict very different poses for very similar images, or exactly the same images if the object is perfectly symmetrical.



# Solution (1)

Let β be the angle of symmetry of the object:

β = 180° in this object

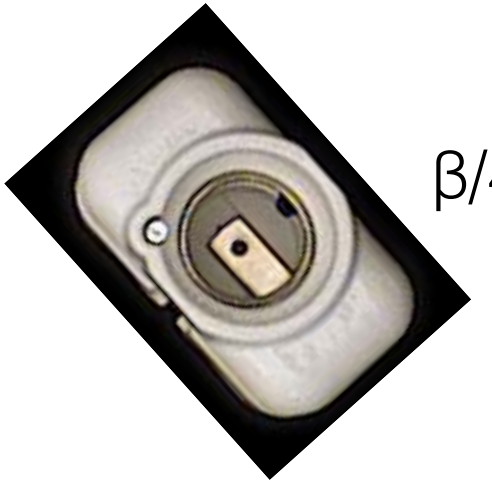
1. We train a regressor (Convolutional Neural Network in practice) to predict the projection of the 3D bounding box as in our previous method BUT ONLY on a restricted range: [0°, β/2].

0° …  β/4 …  β/2

→ no more ambiguities

# Solution (2)

2. To handle larger ranges, we train a classifier (also a Convolutional Neural Network) to tell if the pose is between [0°, β/2] or between [β/2, β]:

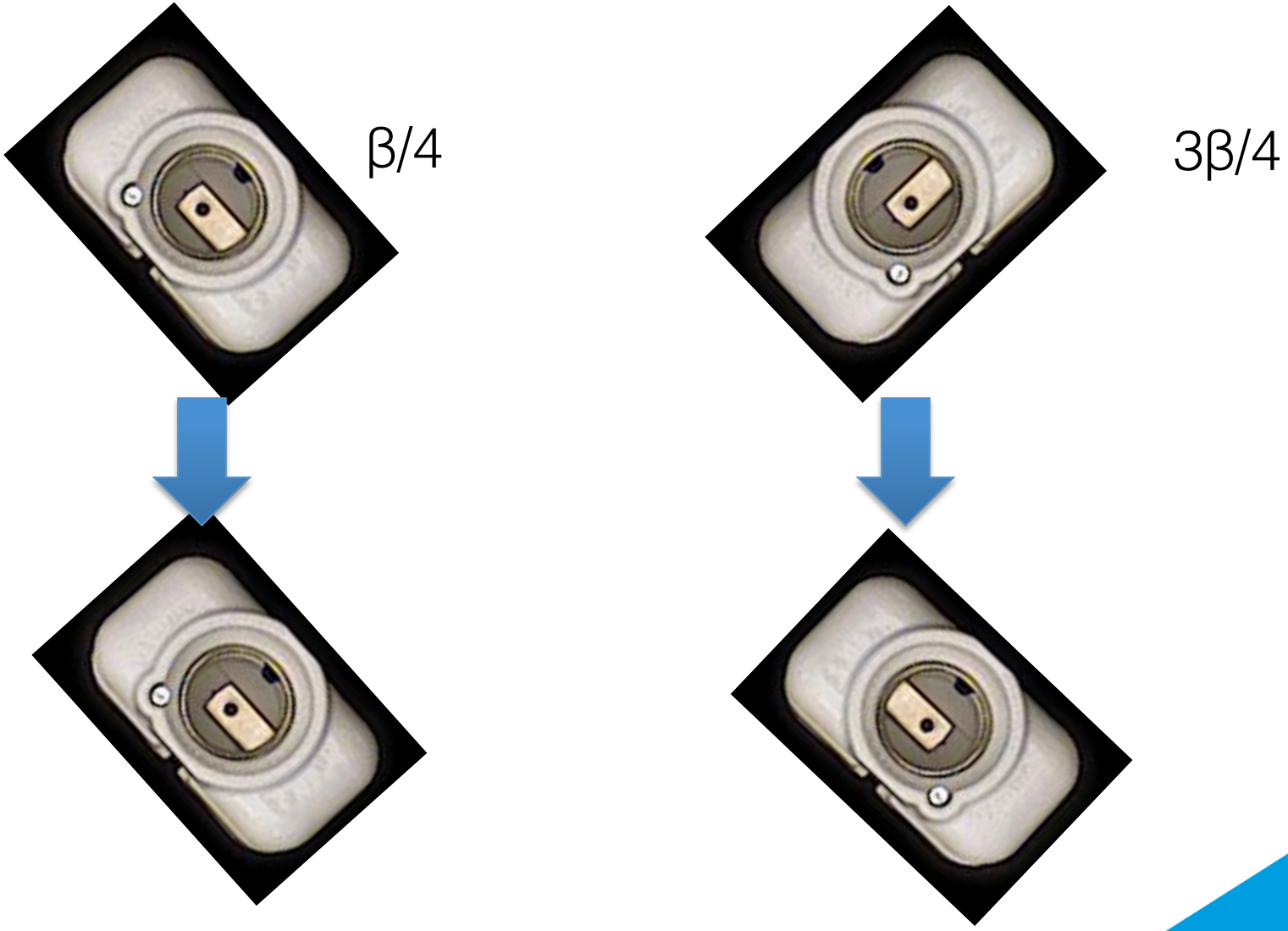


β/4

3β/4

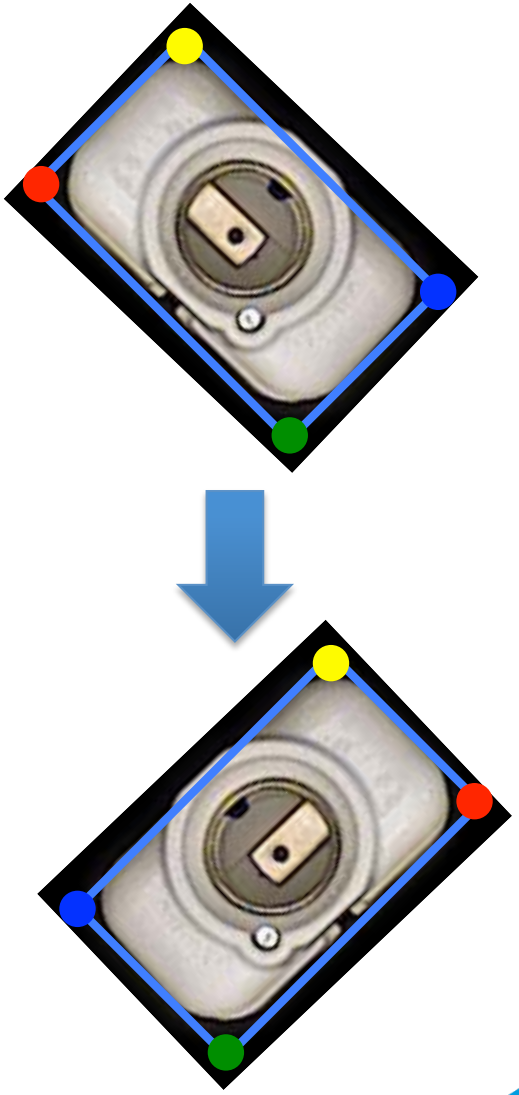
# Solution (3)

At run-time, if the pose is between [β/2, β] (as given by the classifier), we flip the image before applying the regressor:



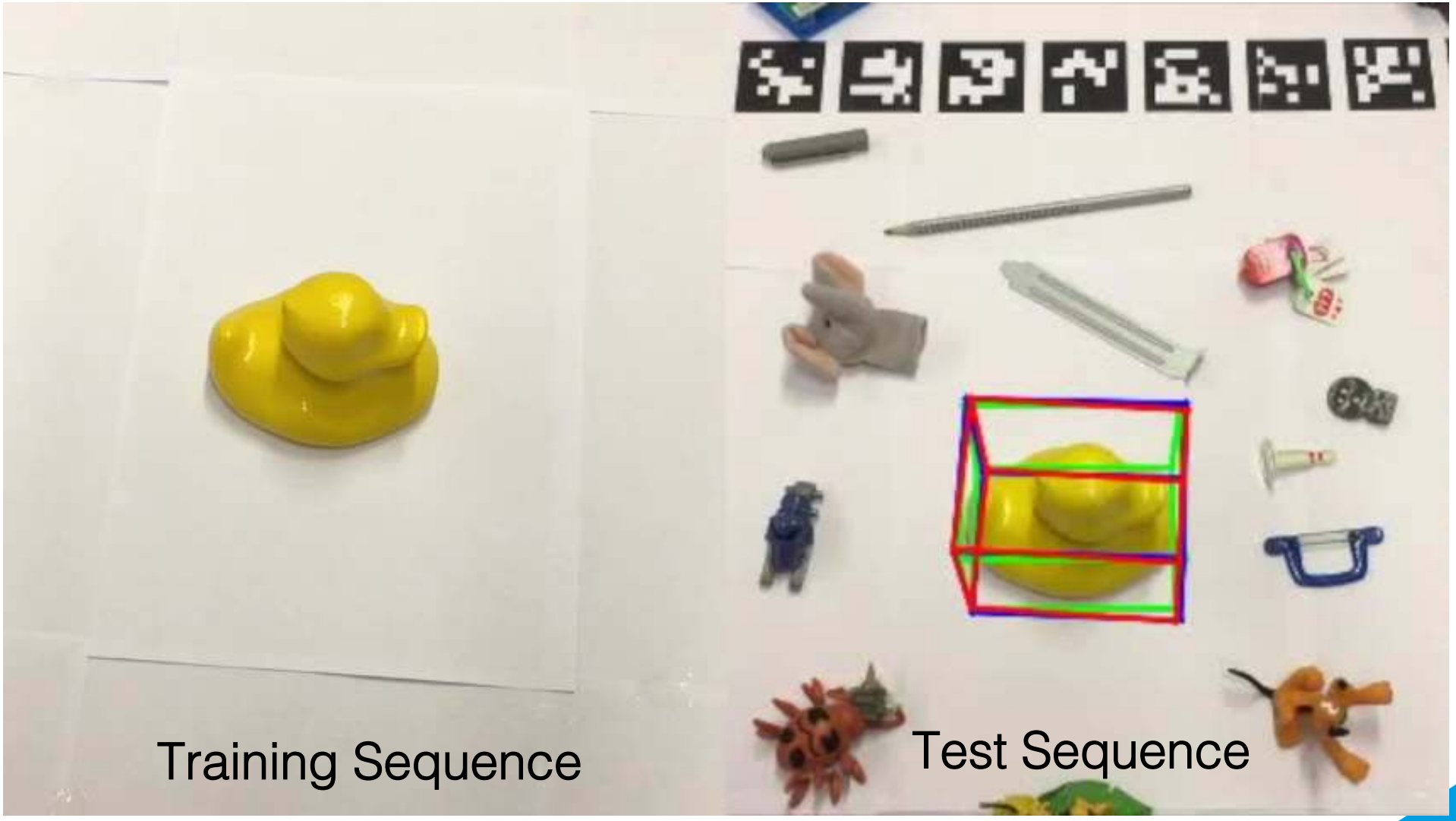
# Solution (4)

(still at run-time), if we flipped the image before applying the regressor, we flip the corners of the bounding box predicted by the regressor:



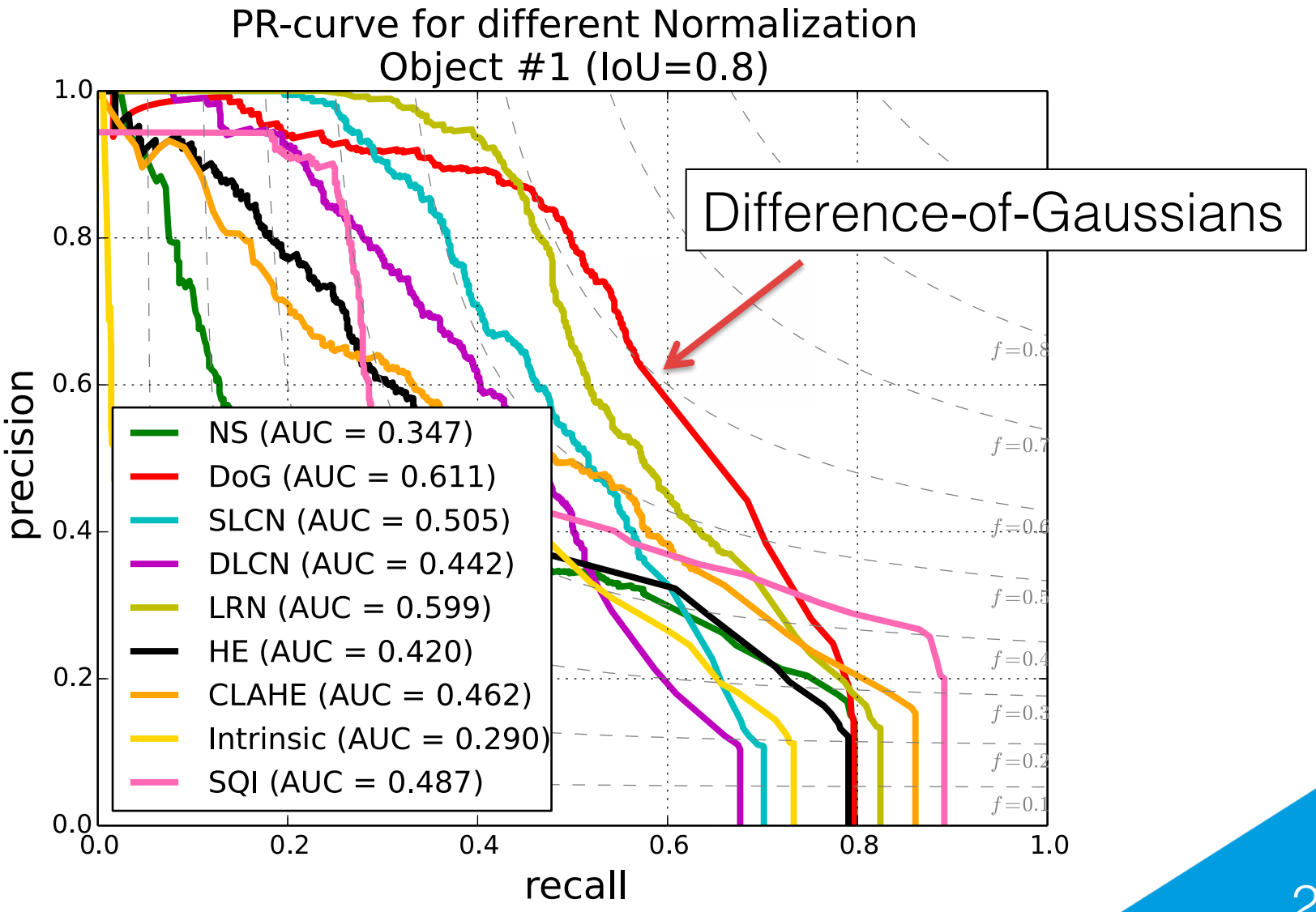

T-LESS

# Robustness to Light Changes: Adaptive Local Contrast Normalization (*ALCN*)




ALCN: Adaptive Local Contrast Normalization for Robust Object Detection and 3D Pose Estimation. Mahdi Rad, Peter Roth, Vincent Lepetit, BMVC 2017.

# Existing Illumination Normalization Methods



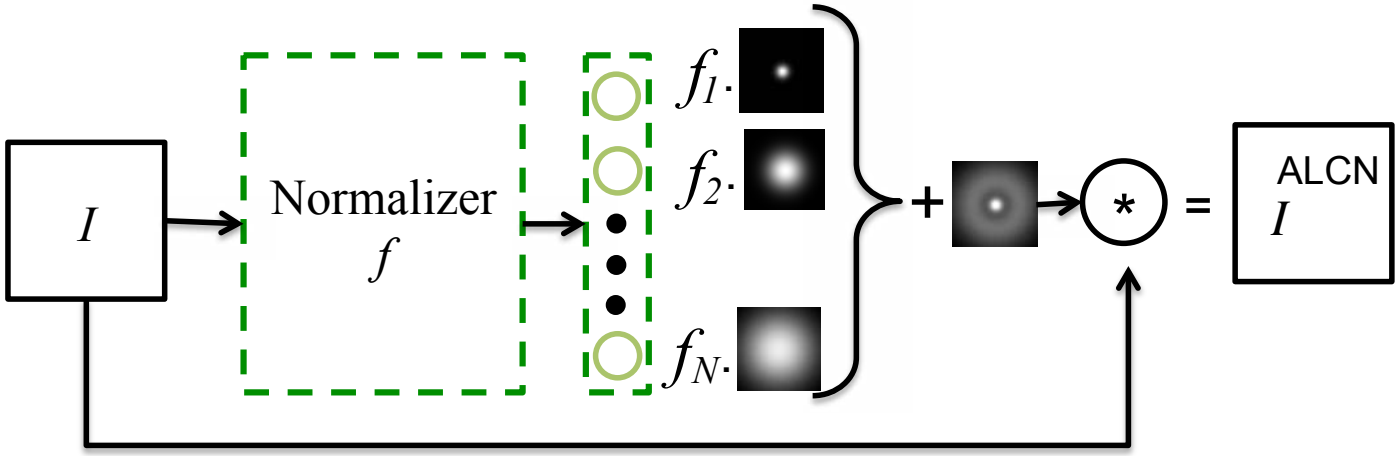
# Difference-of-Gaussians

$$I^{DoG} = k_1 G_{\sigma_1} * I - k_2 \, G_{\sigma_2} * I$$

Observation: The parameters should be carefully tuned for optimal performance

# Idea: The Parameters Should Adapt to the Input Image

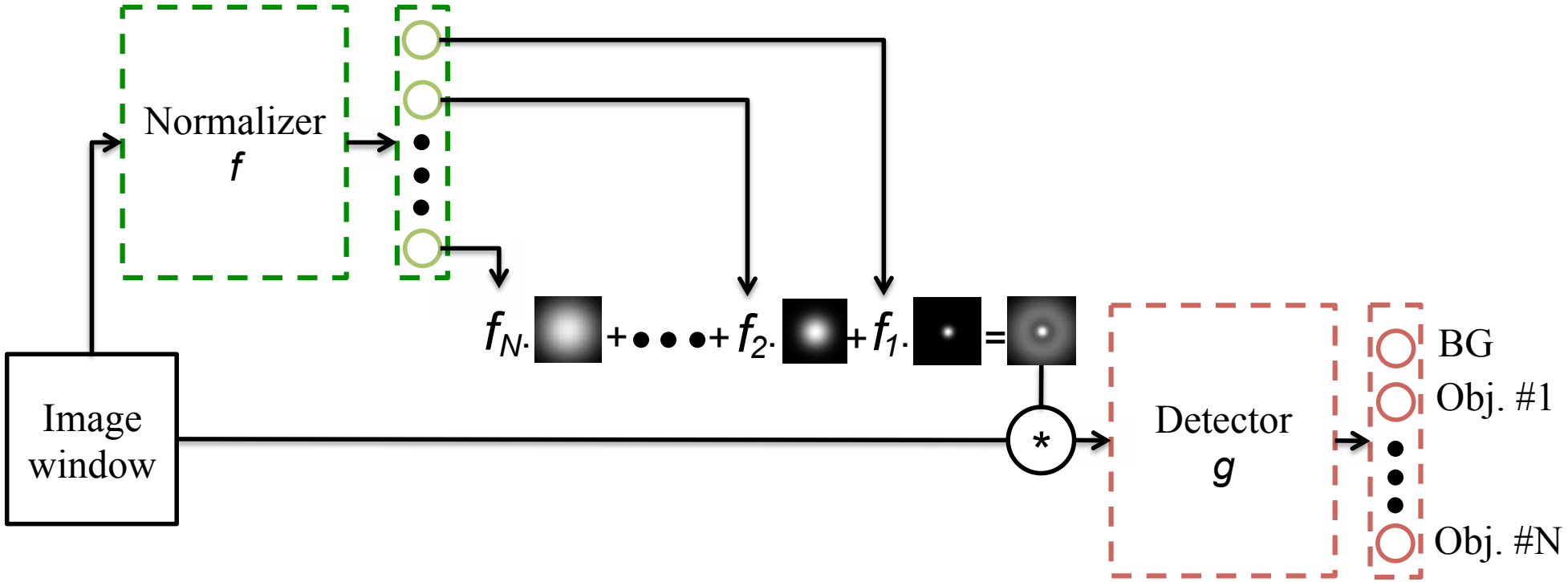
$$I^{ALCN} = \left( \sum_{i=1}^{N} f_i(I) . G_{\sigma_i^{ALCN}} \right) * I$$




BUT we cannot train this CNN in a standard supervised manner.

# Training

**Solution**: train jointly in a supervised way together with another CNN

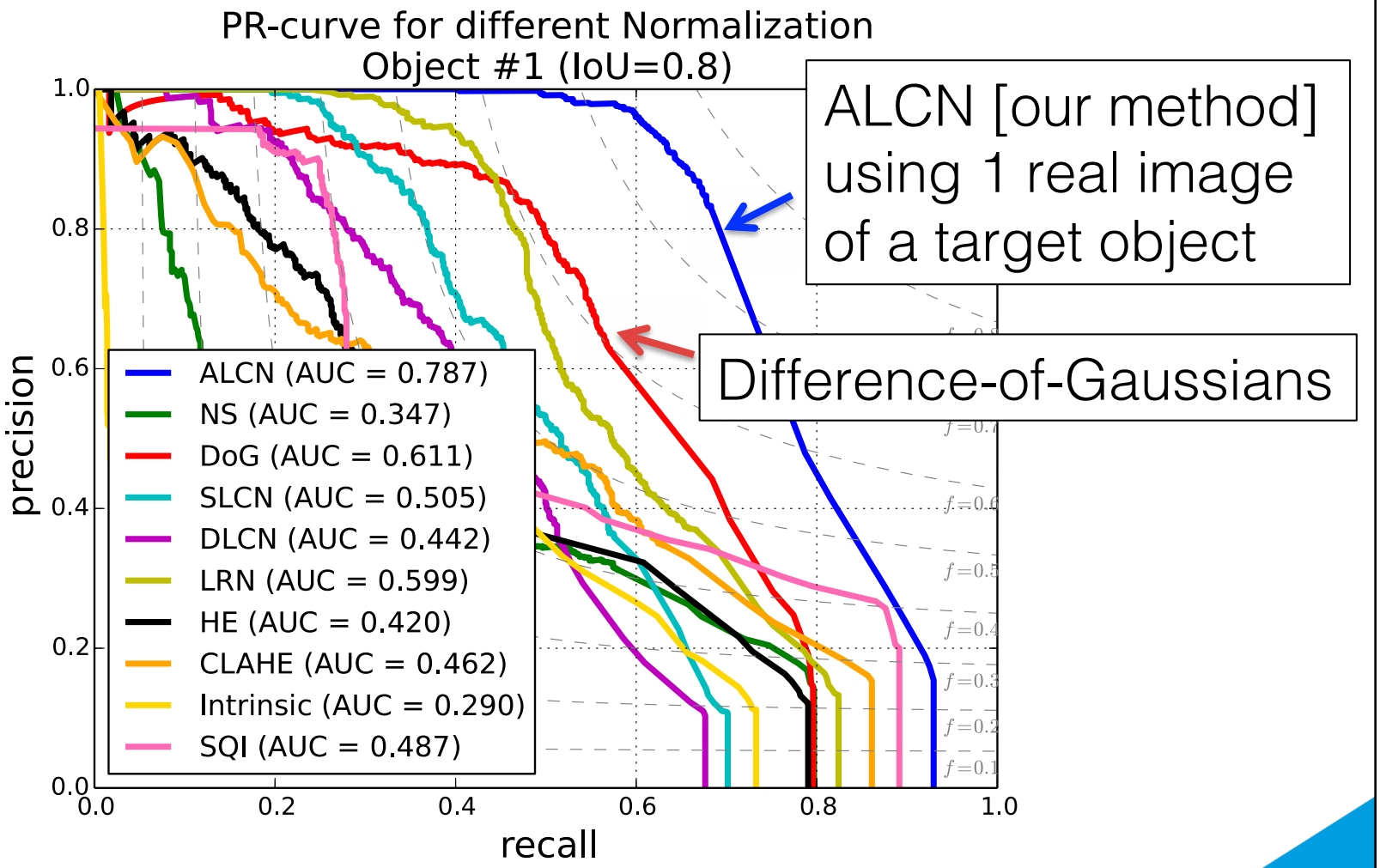


# Training

We augment the Phos dataset:

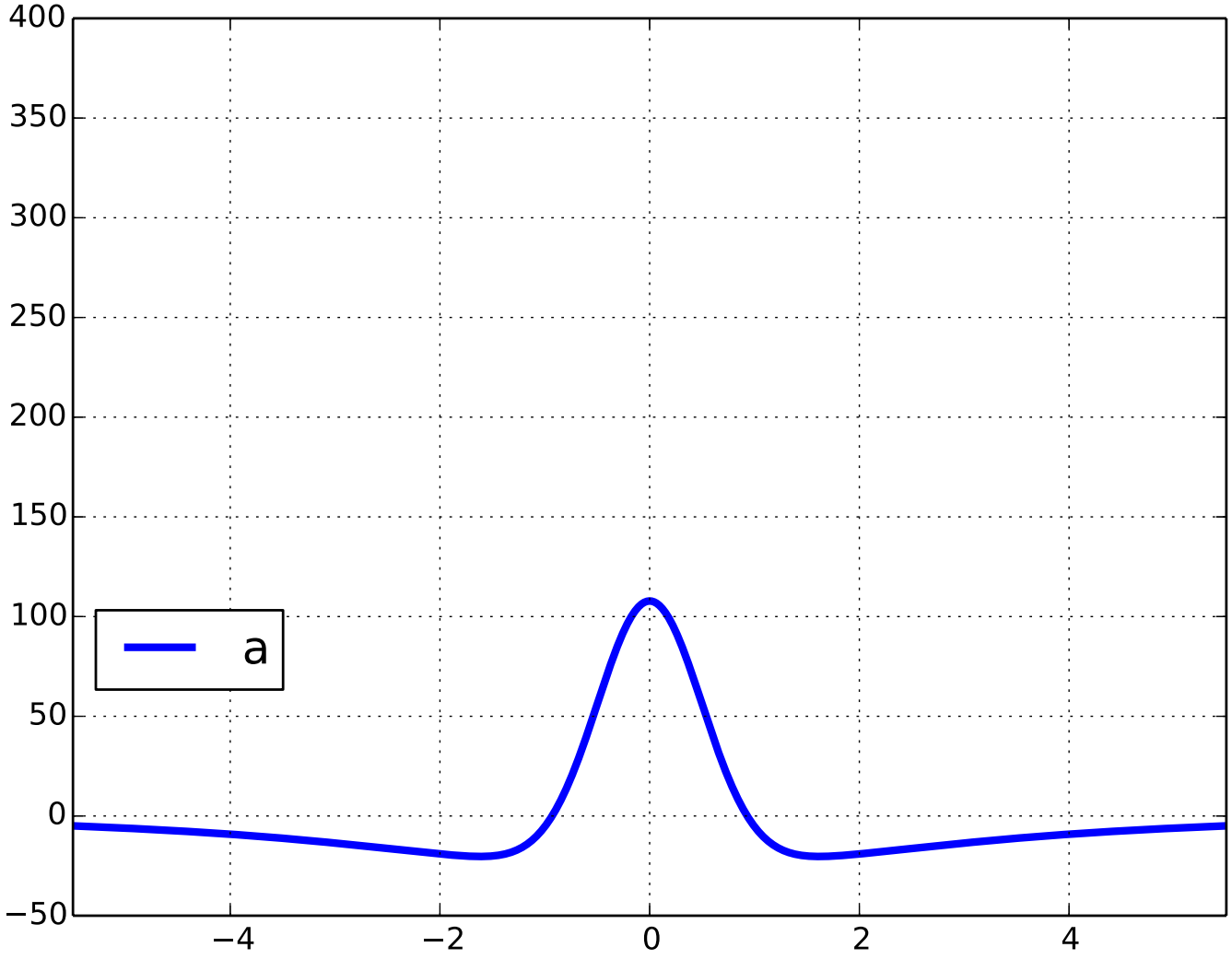
with synthetic images:

# Comparing with Existing Methods



# Predicted Filters for Different Input

(a)

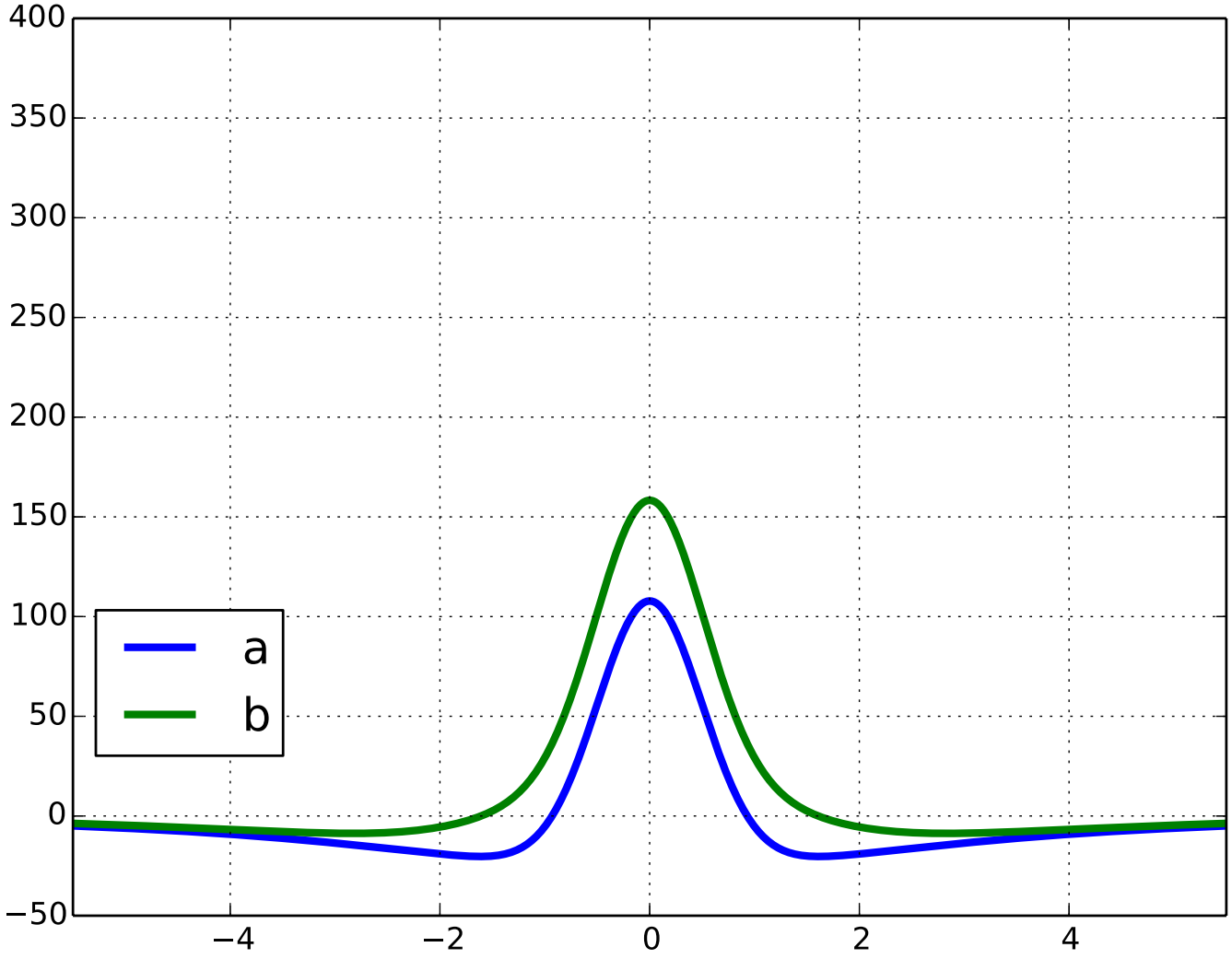





université de BORDEAUX

# Predicted Filters for different input

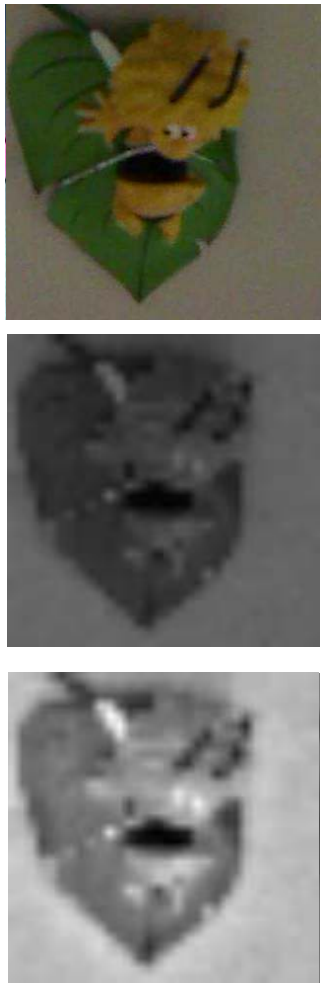
(b)



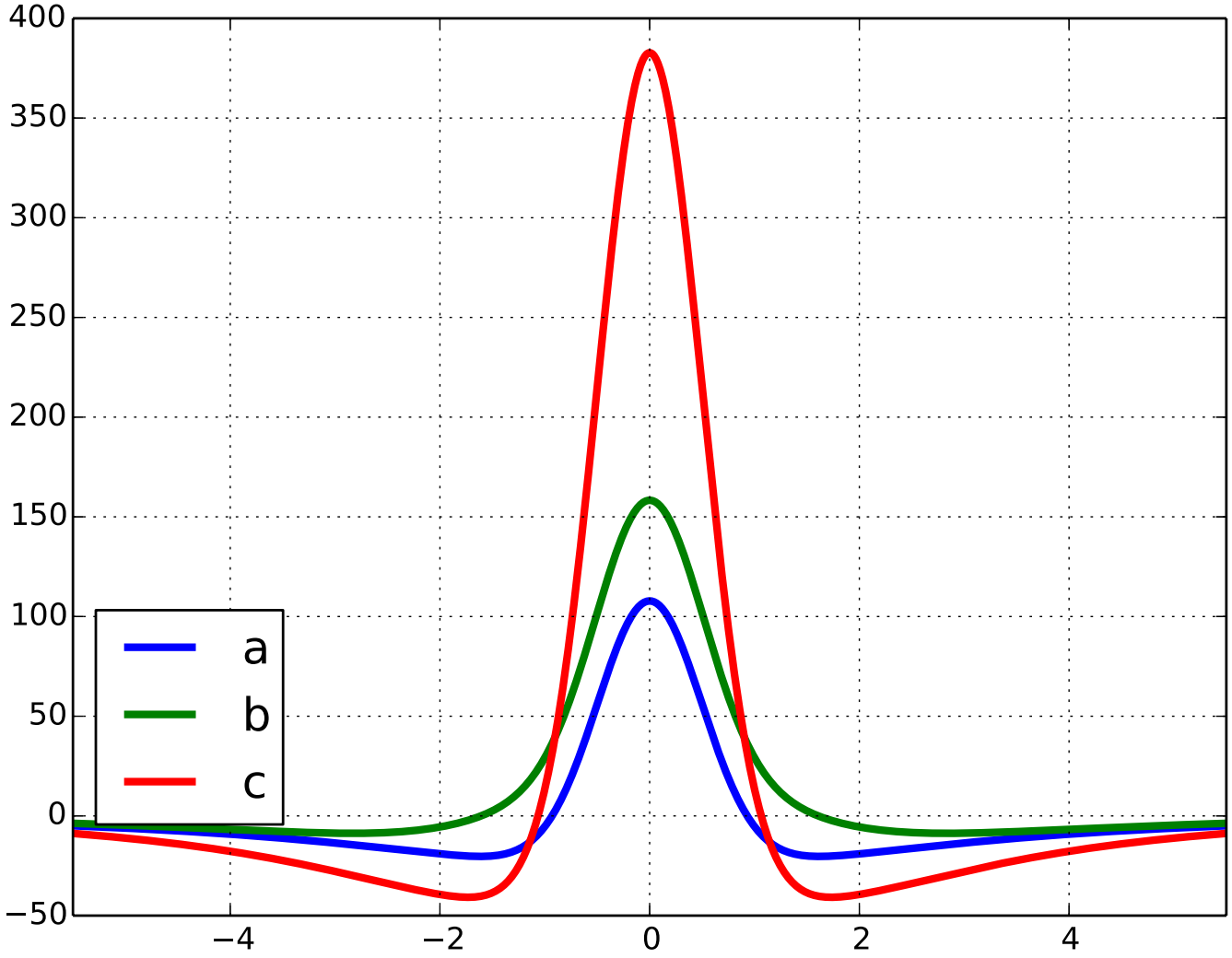



université de BORDEAUX

# Predicted Filters for different input



(c)

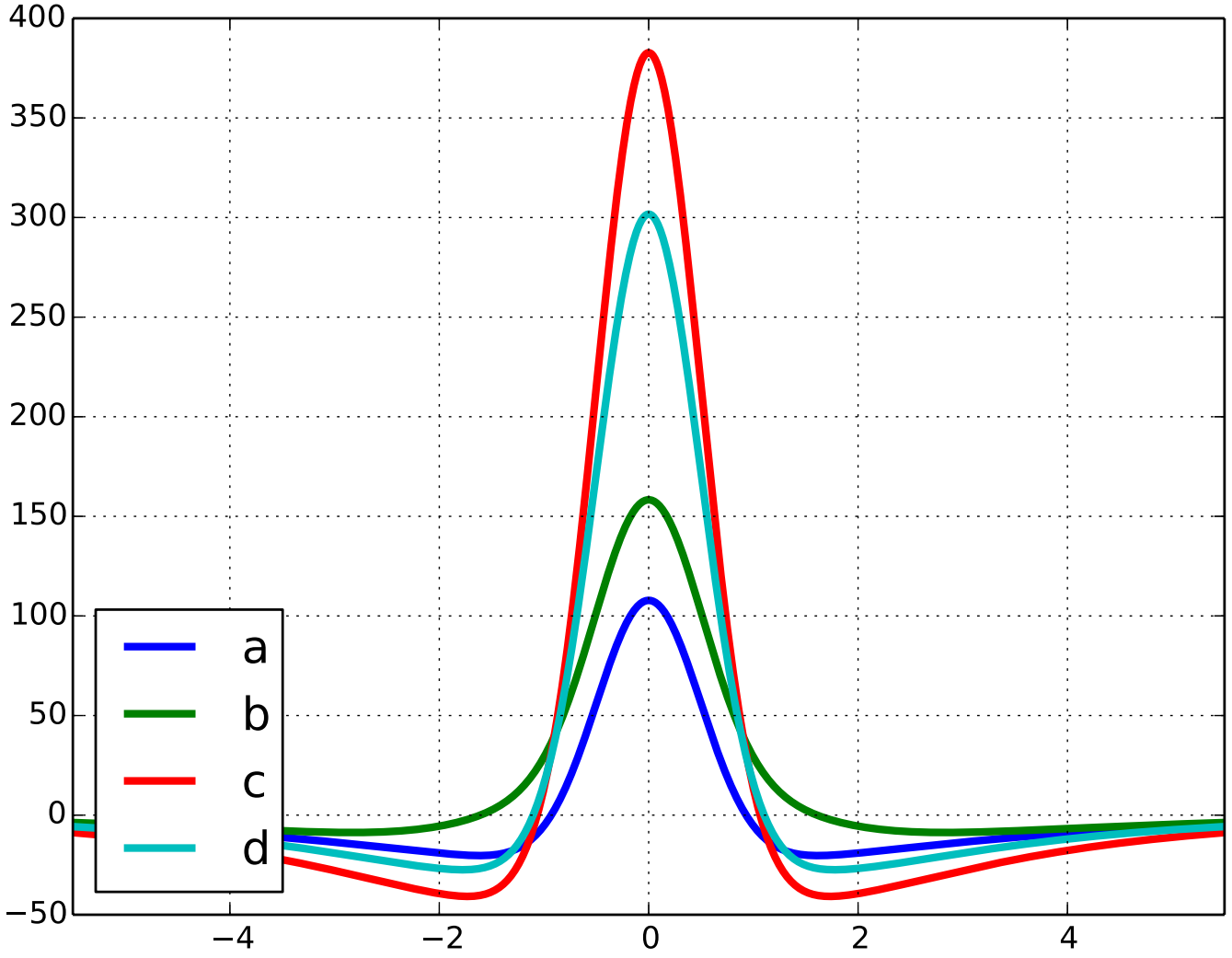





université de BORDEAUX

# Predicted Filters for different input

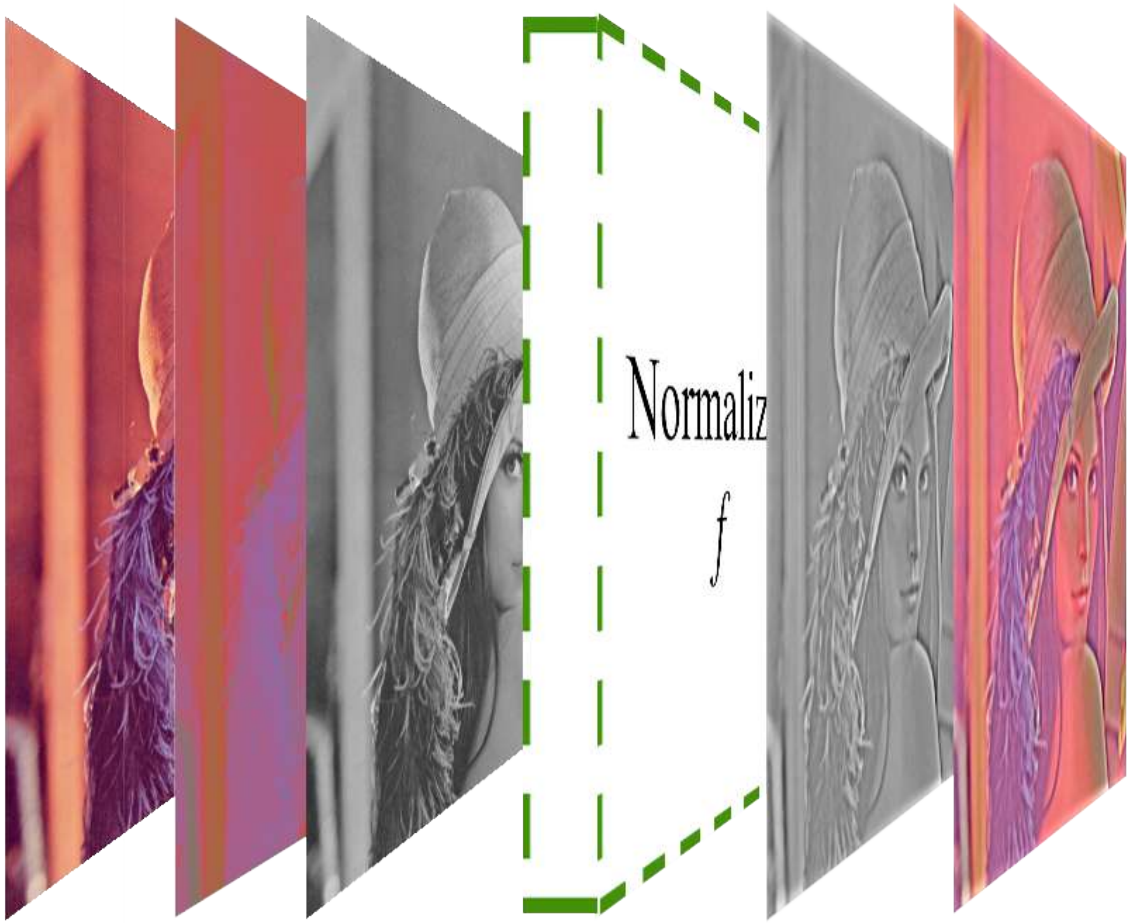
(d)






université de BORDEAUX

# Color Image Normalization

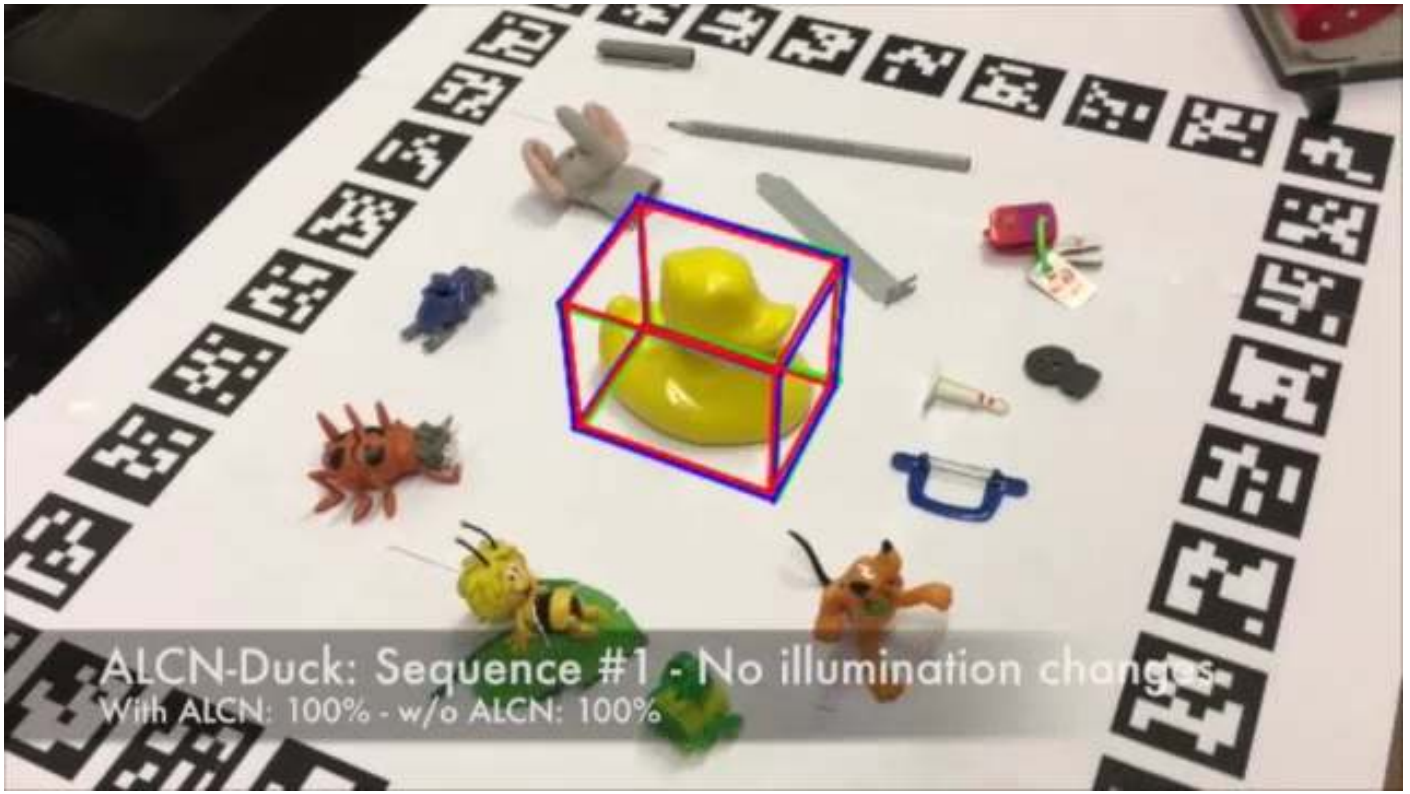


# Explicit Normalization vs. Illumination Robustness with Deep Learning

| | AUC |
|---|---|
| Shallow | 0.401 |
| ALCN + Shallow | 0.787 |
| VGG | 0.606 |
| ResNet (20 Layers) | 0.456 |
| ResNet (32 Layers) | 0.498 |
| ResNet (44 Layers) | 0.518 |
| ResNet (56 Layers) | 0.589 |
| ResNet (110 Layers) | 0.565 |

# Evaluation

**Green**: Ground Truth
**Red**: BB8
**Blue**: BB8 + ALCN




| | w/o | with illumination changes | | | | | | |
|---|---|---|---|---|---|---|---|---|
| sequence | #1 | #2 | #3 | #4 | #5 | #6 | #7 | #8 |
| BB8 | 100 | 47.3 | 18.3 | 32.7 | 0.00 | 0.00 | 0.00 | 0.00 |
| BB8 + ALCN | 100 | 77.8 | 60.7 | 70.7 | 64.1 | 51.4 | 76.2 | 50.1 |

# Accurate Localization from Images

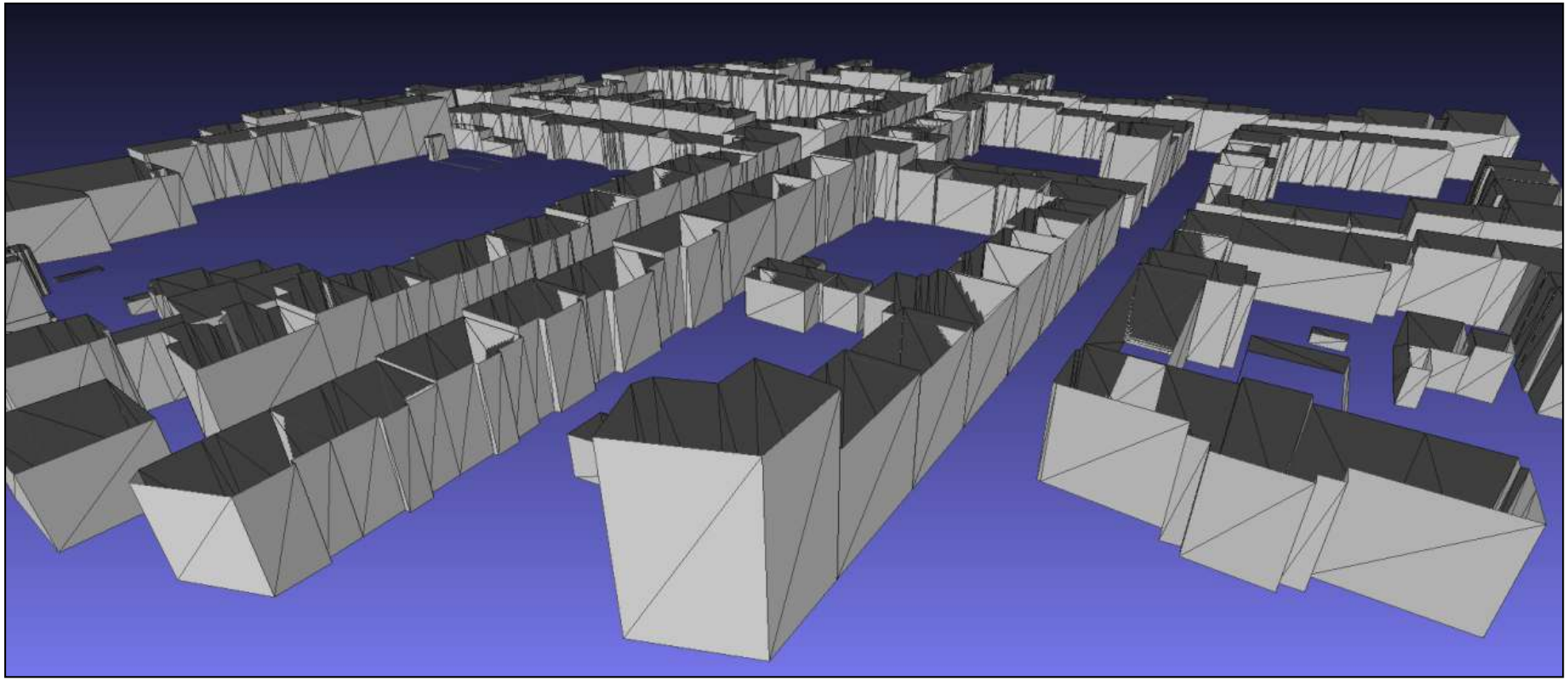
Accurate Camera Registration in Urban Environments Using High-Level Feature Matching, Anil Armagan, Martin Hirzer, Peter Roth, Vincent Lepetit, BMVC 2017.

reprojection of the 2D map using the pose from the sensors

*Use registered images?* Very cumbersome

Idea: Use 2.5D maps from OpenStreetMap

# We Want to Use 2.5D Maps



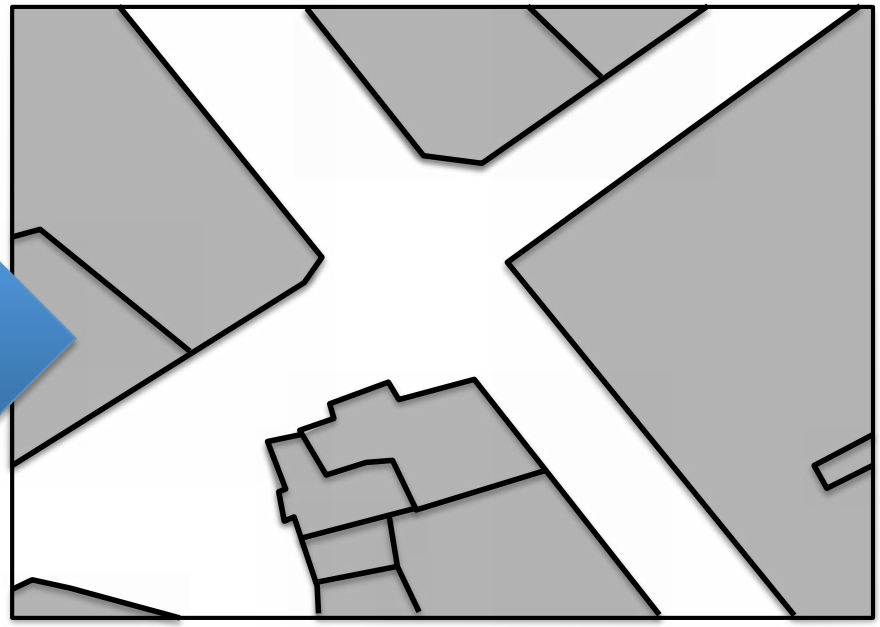
In 2.5D maps, buildings are modeled as 2D polygons + height

- sensors give accurate angles wrt gravity:
→ we can work on rectified images.



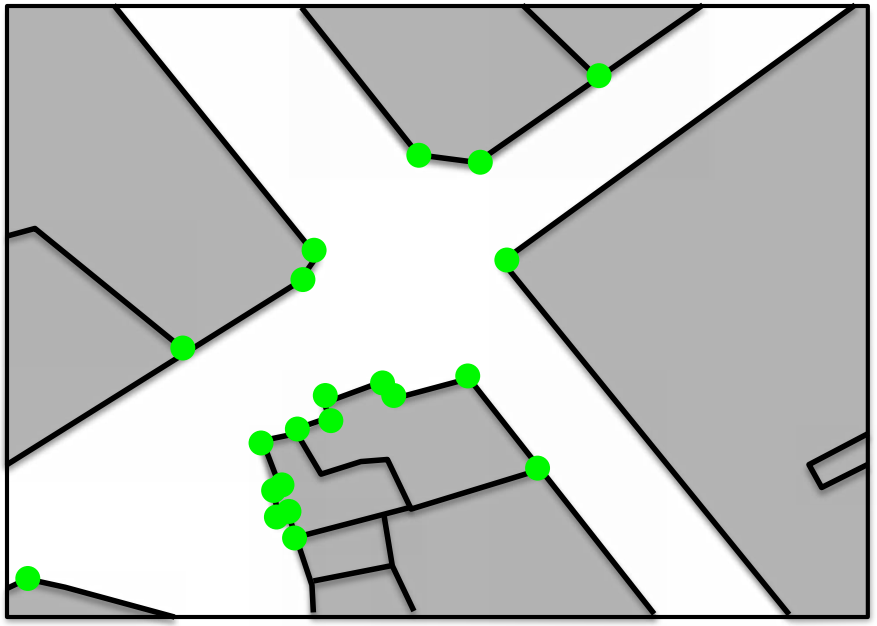
- we assume we know the altitude of the camera;

Remain 3 degrees-of-freedom:
2D translation + rotation along the ground plane

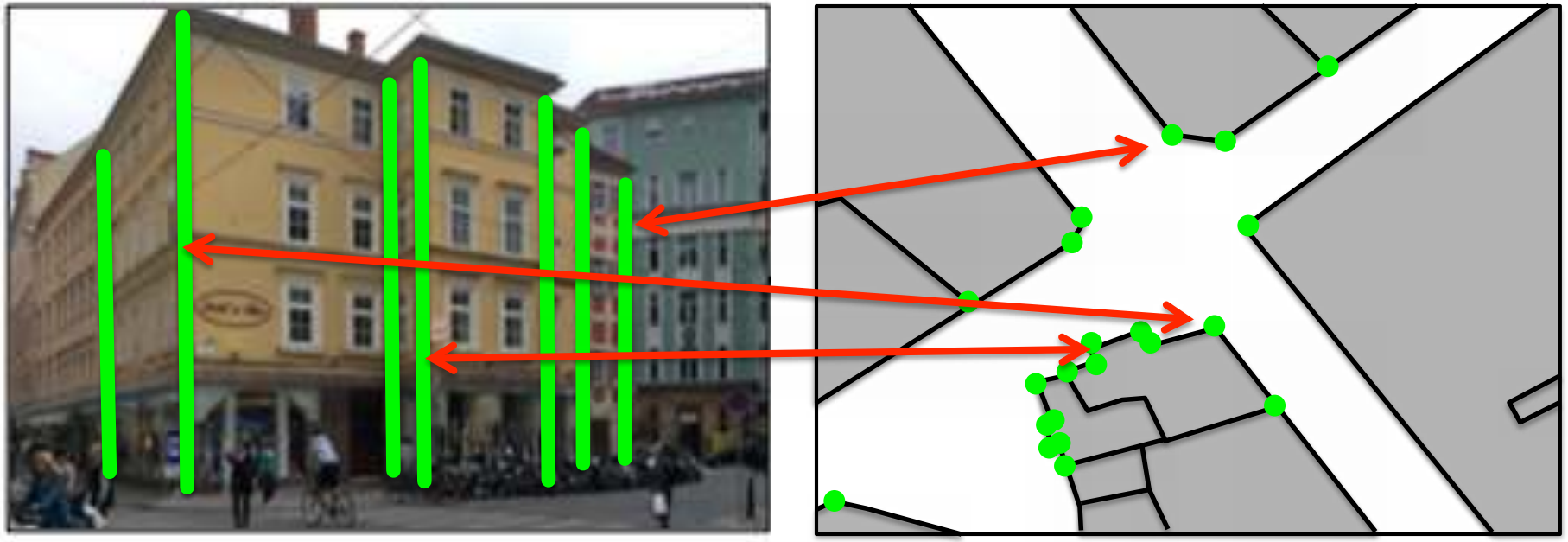



2.5D map around the GPS location

# Matching High-Level Features

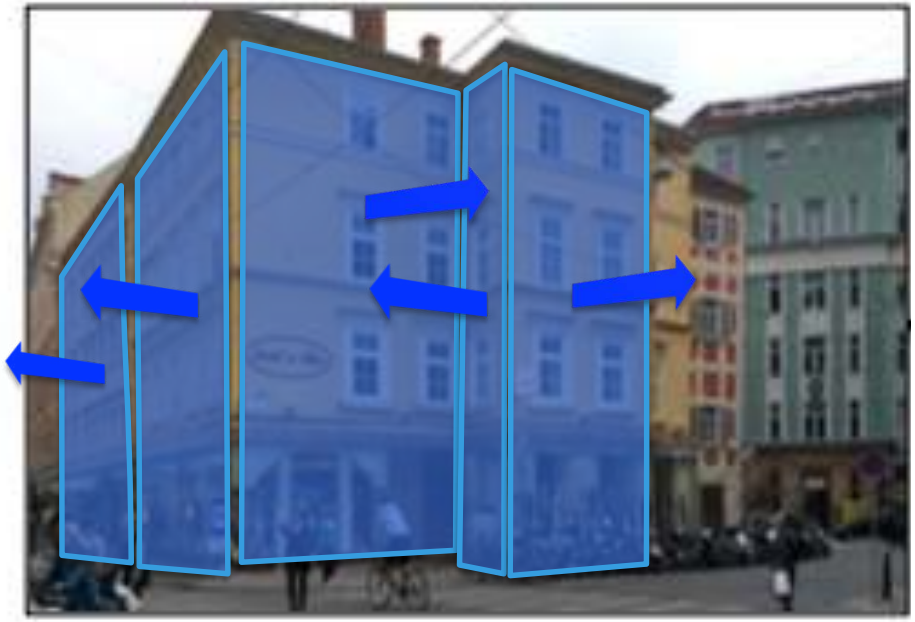


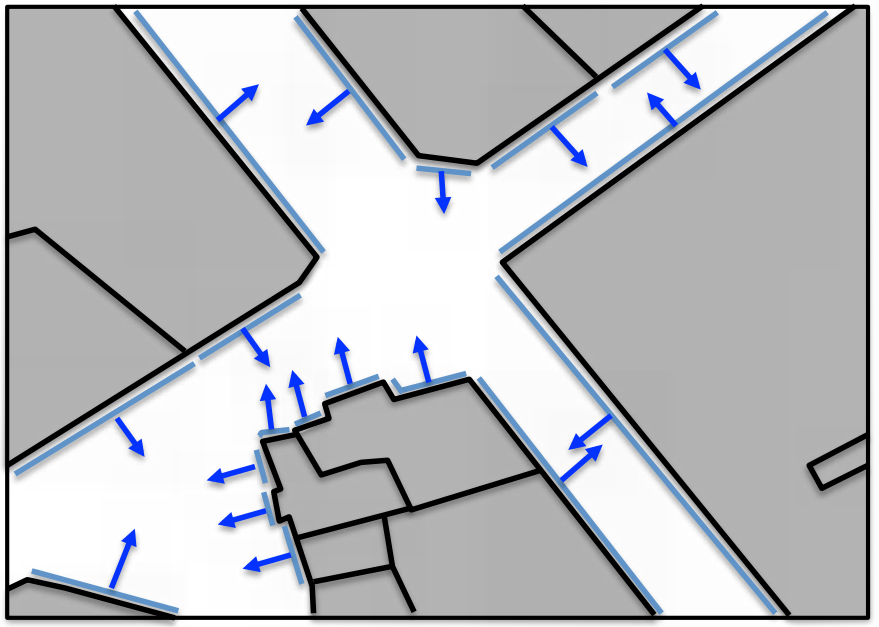
# Matching High-Level Features



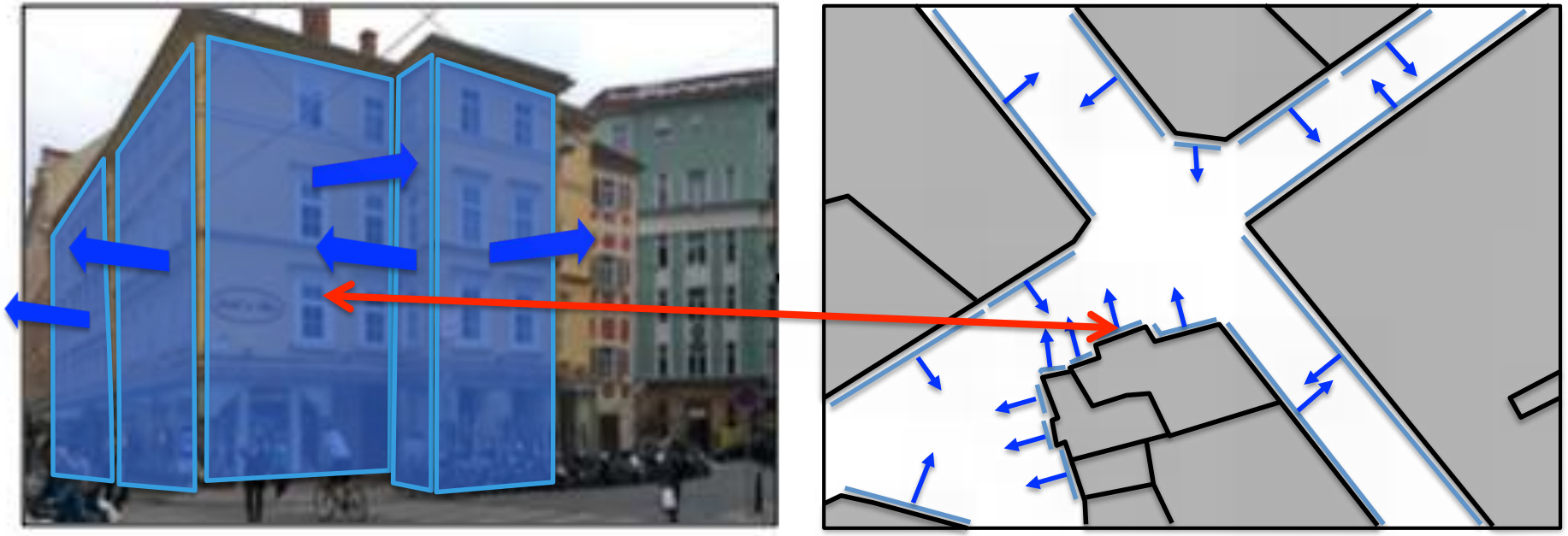
3 correspondences between building corners in the image and in the 2D map → pose (2d translation + angle)
+ RANSAC

# Matching High-Level Features





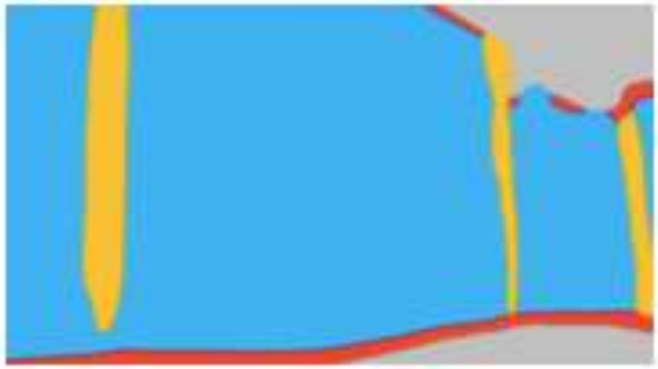
# Matching High-Level Features



1 correspondence between a façade in the image and a façade in the 2D map → pose (2d translation + angle)
+ RANSAC

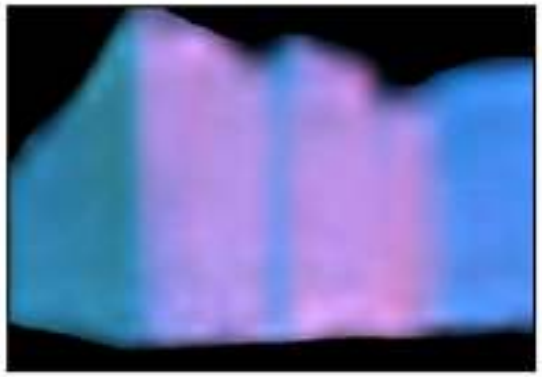
# Semantic Segmentation and Façades' Normals Estimation

$CNN_1$



Semantic Segmentation

$CNN_2$



Façades' Normal Estimation

# Creating Training Data

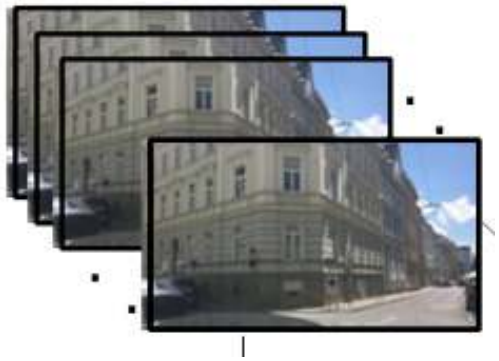
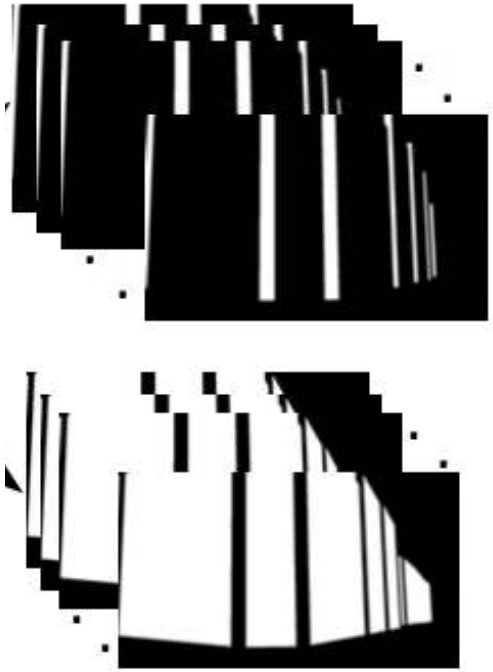


Image Sequences

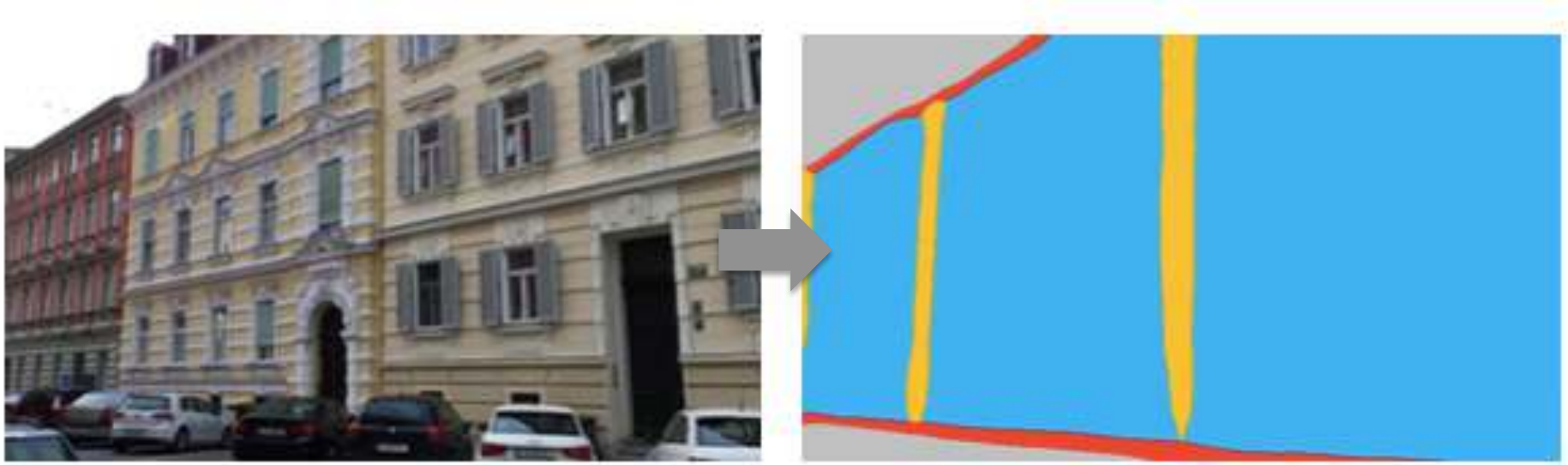
3D Tracker
with human supervision

3D Model from
2.5D Map



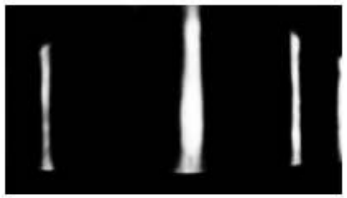
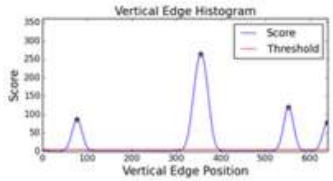
Façade, Corners, and
Normals Labels

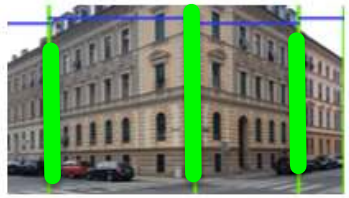
# Segmentation Results



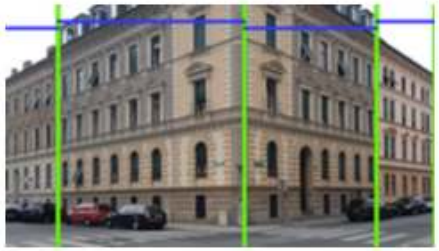
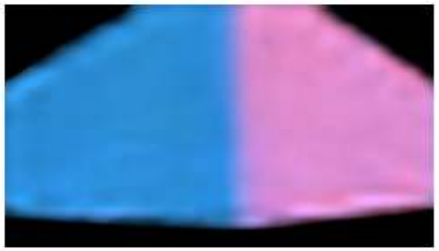
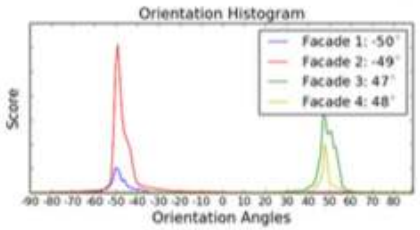
The segmentation is robust to (limited) occlusions.

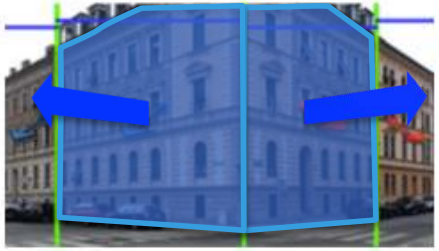
# Locating the Buildings' Corners and Façades

x-coordinates for the building corners
min and max x-coordinates for the façades

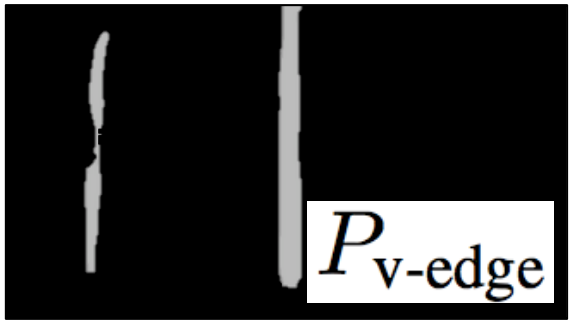




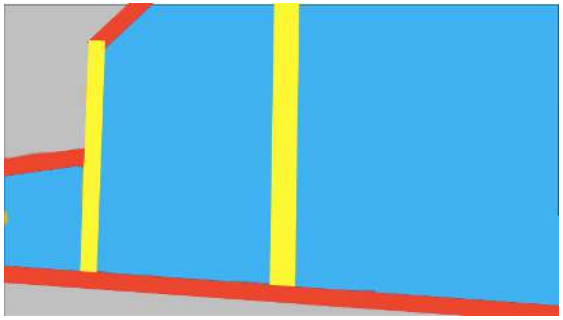


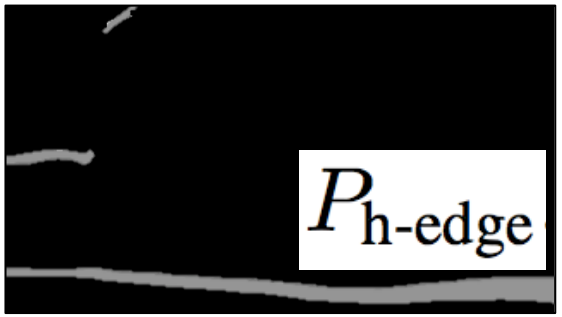
orientation (1 angle) for the façades

# Selecting the Best Hypothesis: Maximum Likelihood

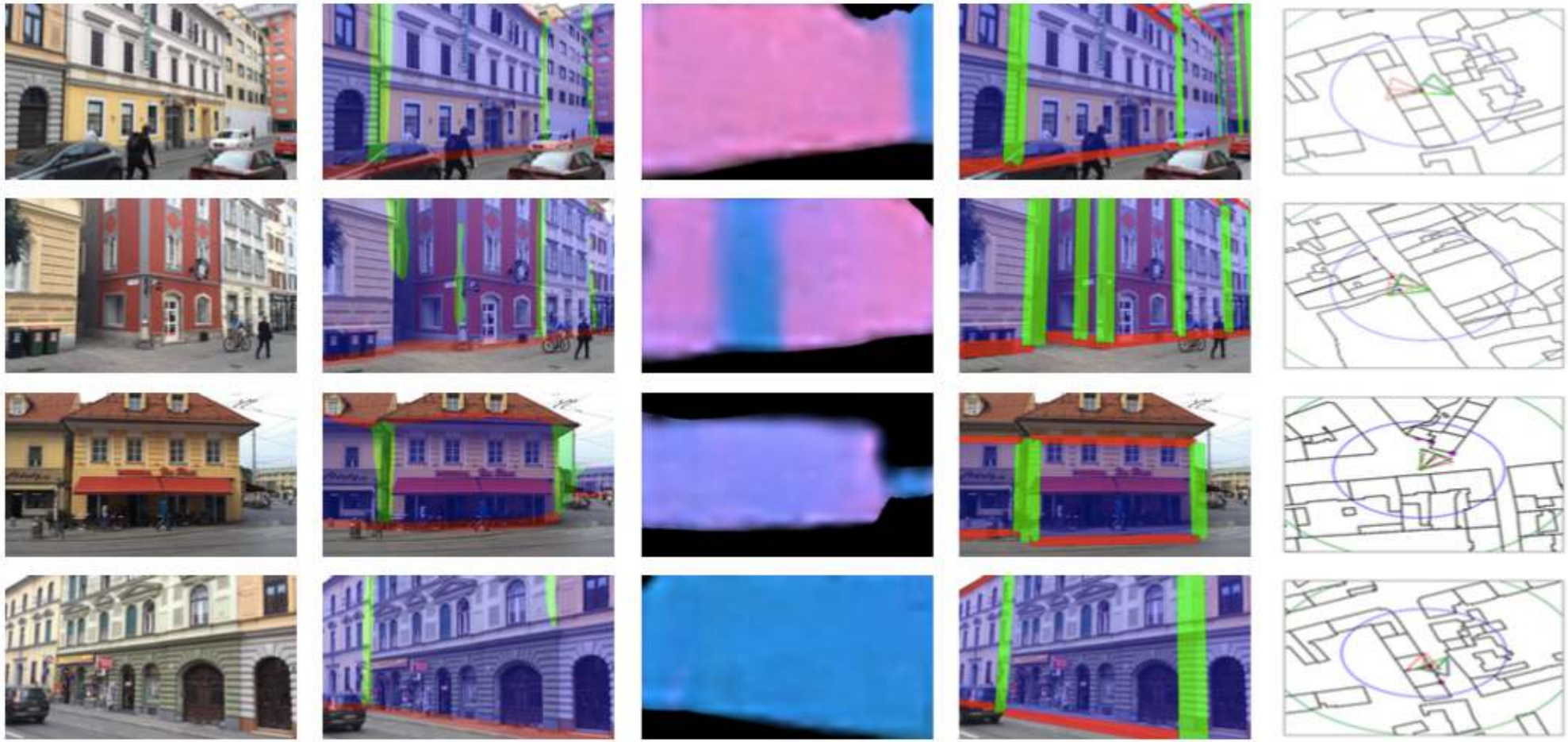
$$\max_{\text{pose}} \sum_{\mathbf{x}} \log P_{\text{class}(\text{Render}(\text{pose}),\mathbf{x})}(\mathbf{x})$$

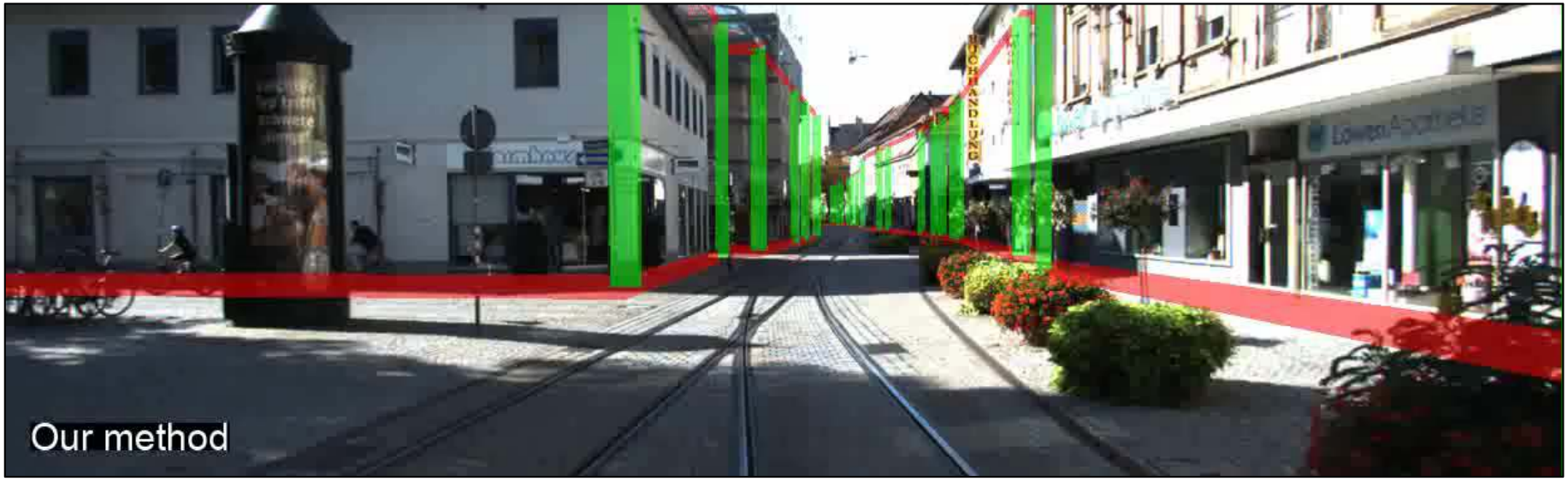






# Some Results



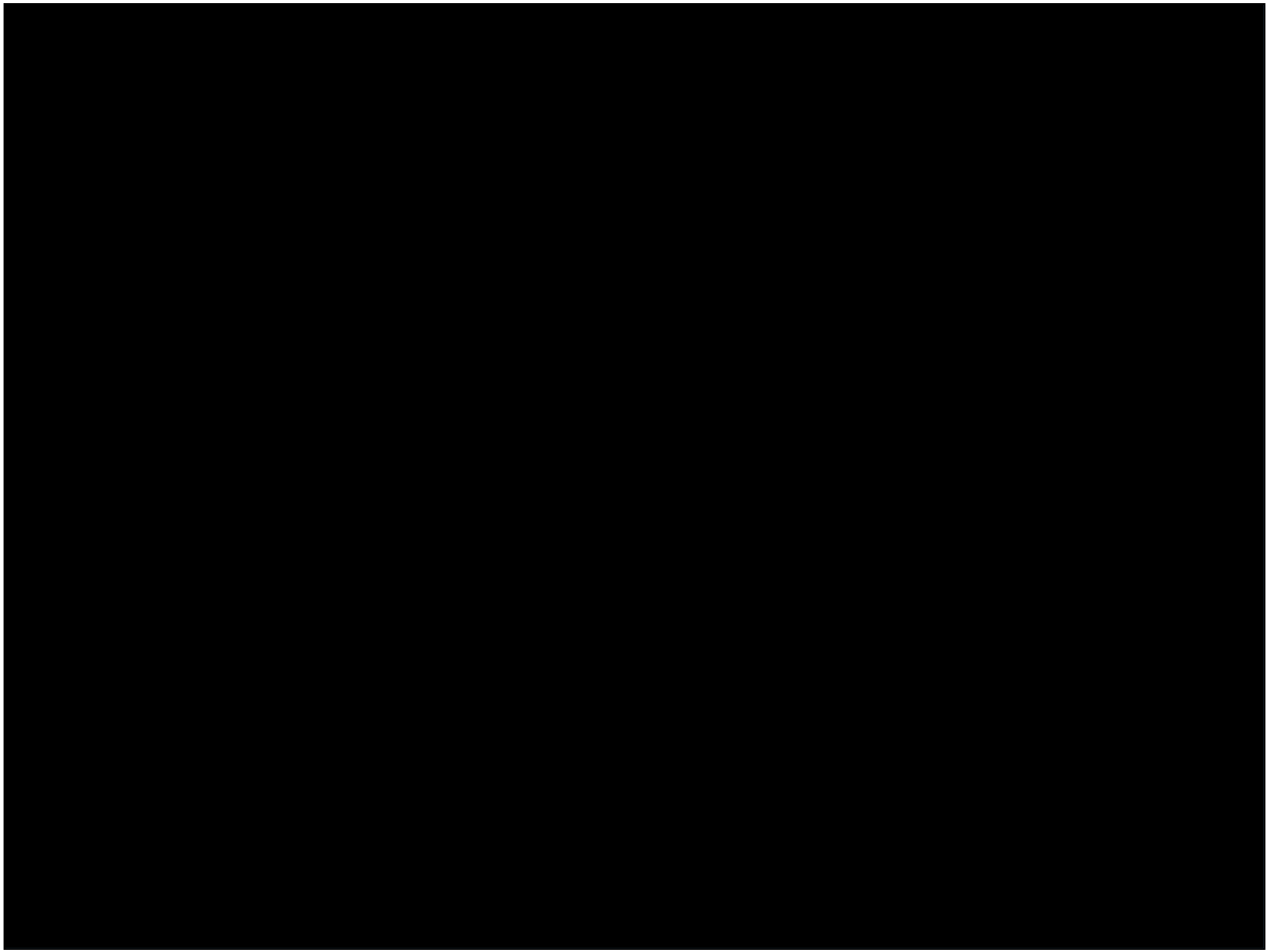
# Some More Results


Efficient 3D Tracking in Urban Environments with Semantic Segmentation, Martin Hirzer, Clemens Arth, Peter Roth, Vincent Lepetit, BMVC 2017.

Thanks for listening!

Questions?



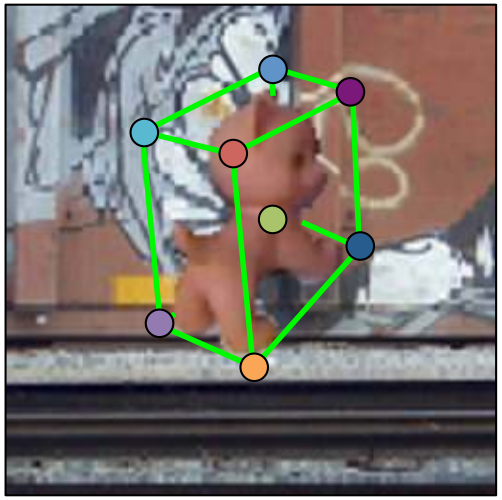
# Training Set

Split the LINEMOD data: 15% of images for training, 85% for testing.

Augment the training set:

1. Extract the object from a training image;
2. Scale the segmented object;
3. Change the background with a random image from ImageNet;
4. Shift the object by some pixels.

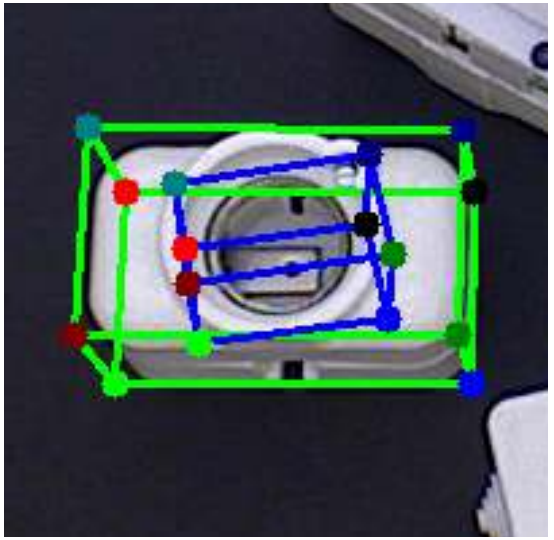
We generate 200,000 training images

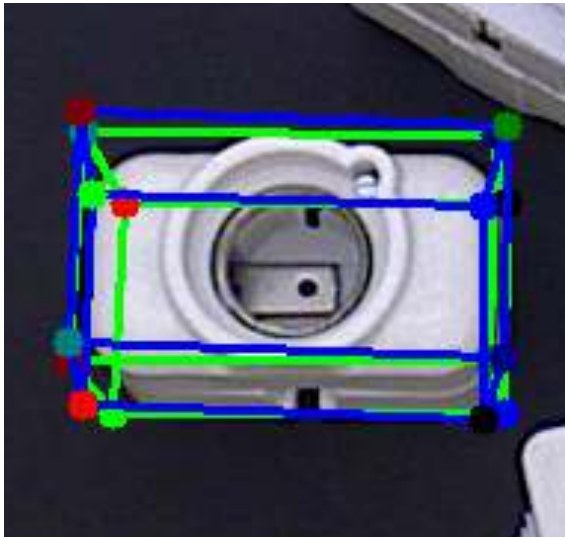

# Results

- Rectangular cuboid:

Previous method



Ours

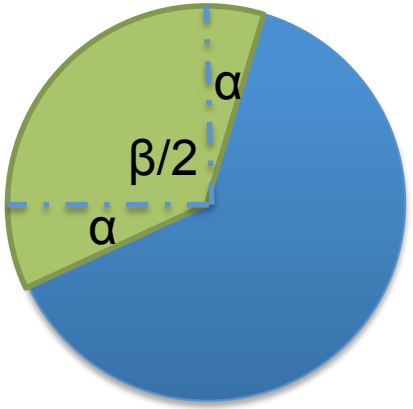


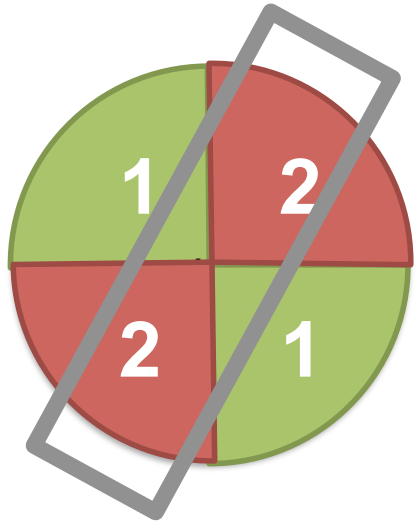
Green: Ground truth - Blue: Estimated Pose

# Implementation

Train two CNNs:

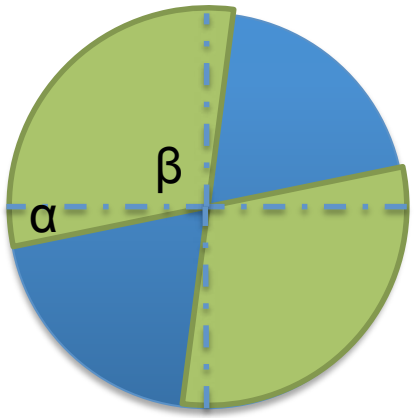
1. Train a regressor using training images between range of [0° - α, β/2 + α]
   - β is the symmetric angle (e.g. β = 180° for rectangular cuboid.
   - α << β
   - α helps to have more accurate results for 0° and β.
2. Train a classifier to predict if the angle is between 0° and β/2 (class 1), or between β/2 and β (class 2).
   - If it is detected as class 2, we rotate the image by β degree, then apply the regressor to predict BB.



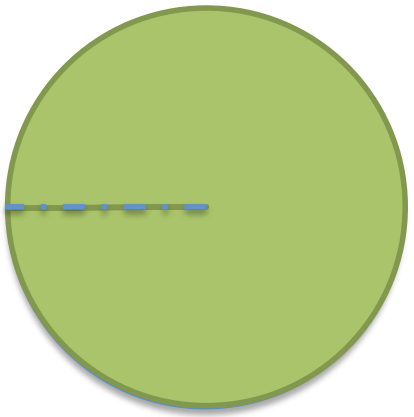



# Implementation

Examples:



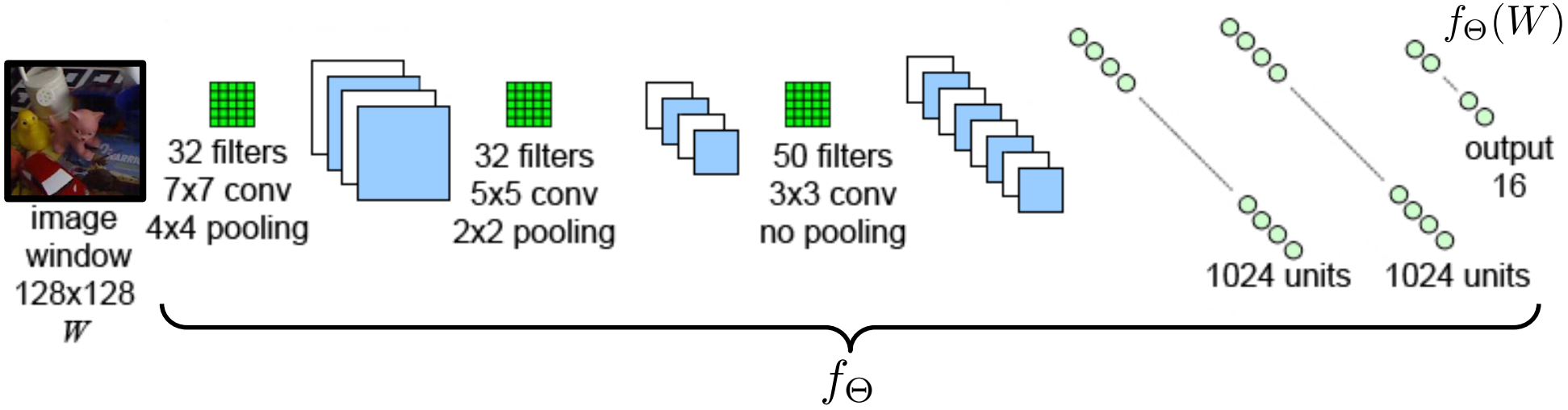

Rectangular cuboid: β = 180°



Cylindrical: β = 0°, α = 0°

# CNN

(the 2D projections of the 8 corners of the 3D bounding box)



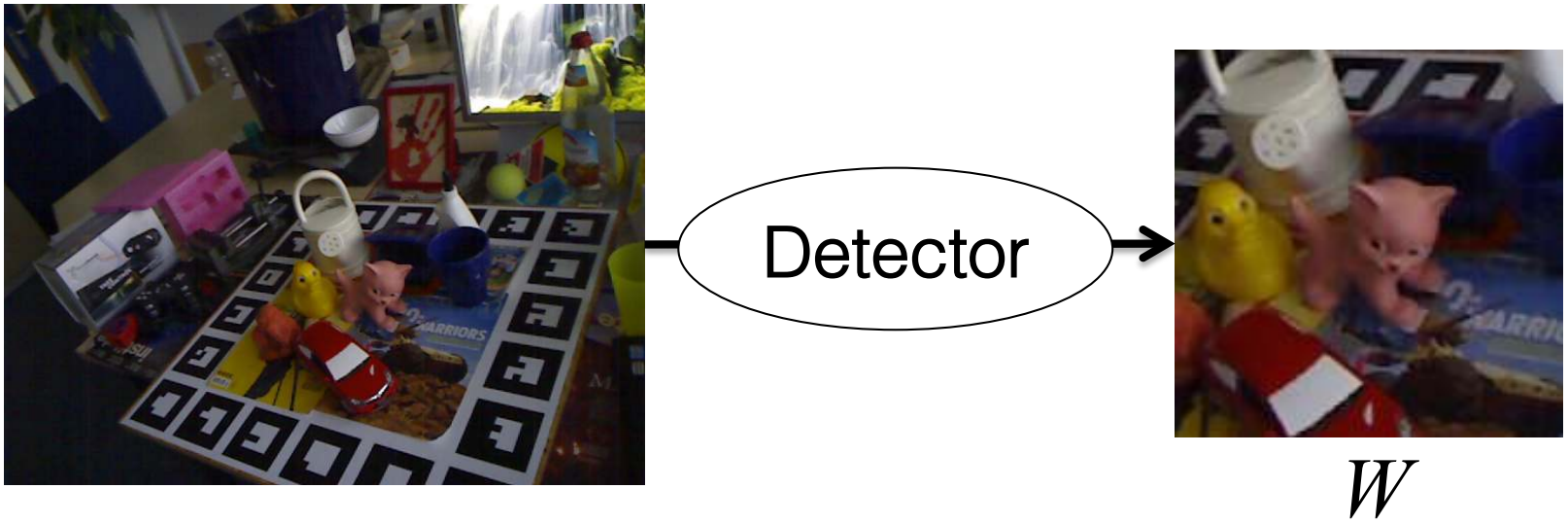

$$\hat{\Theta} = \arg\min \sum_{(W,\mathbf{p}) \in \text{TrainingSet}} \sum_{i=1}^{8} \| f_\Theta(W)[i] - \mathbf{p}_i \|^2$$

network parameters

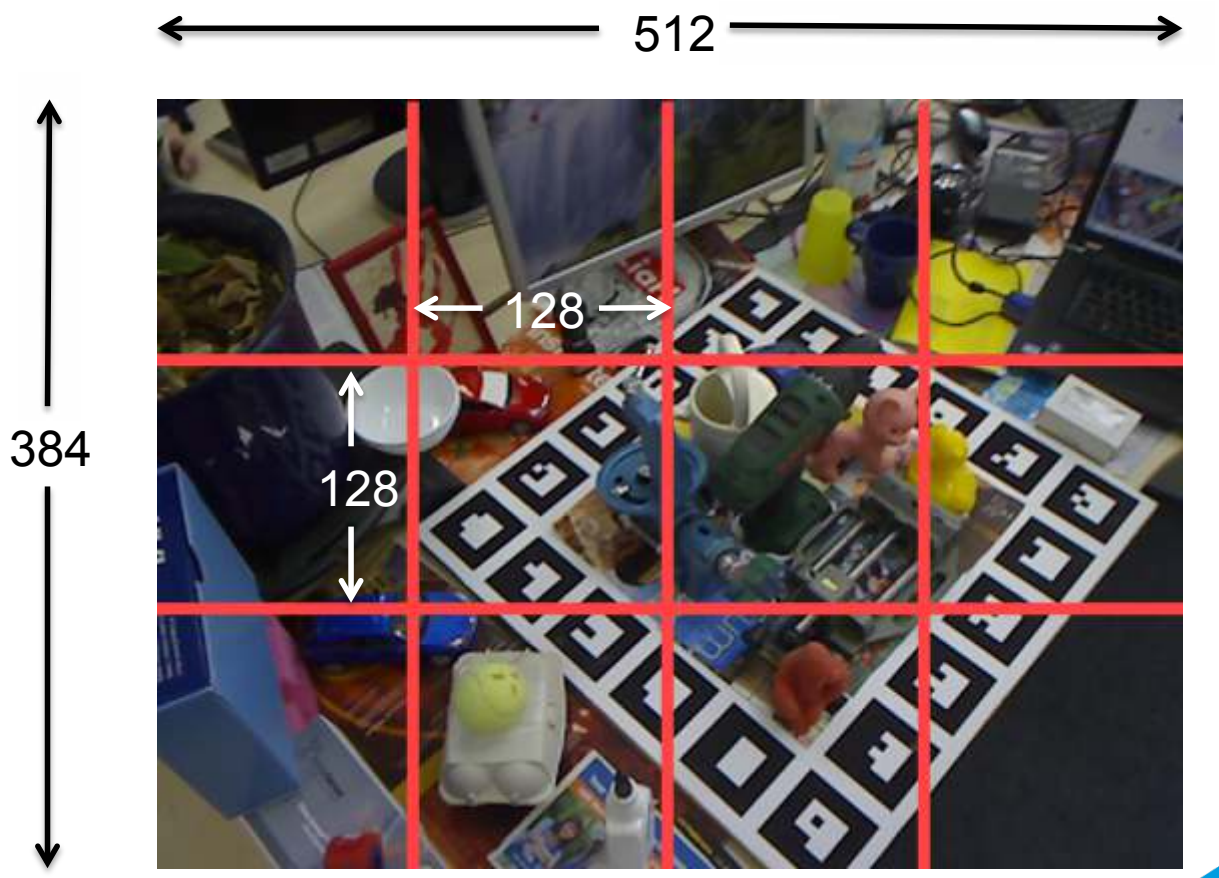
prediction for the *i*-th corner (2 values)

projection of the *i*-th corner

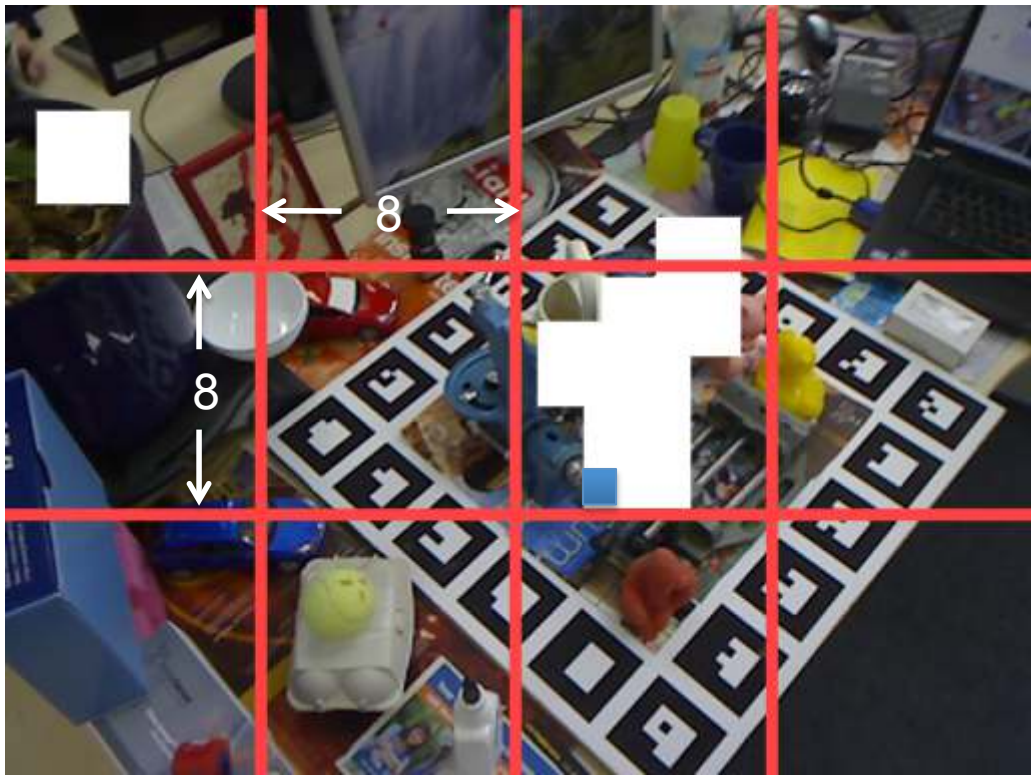
# But Before That, We Need to Find the Object in 2D



1. We split the input image into regions of size 128x128:

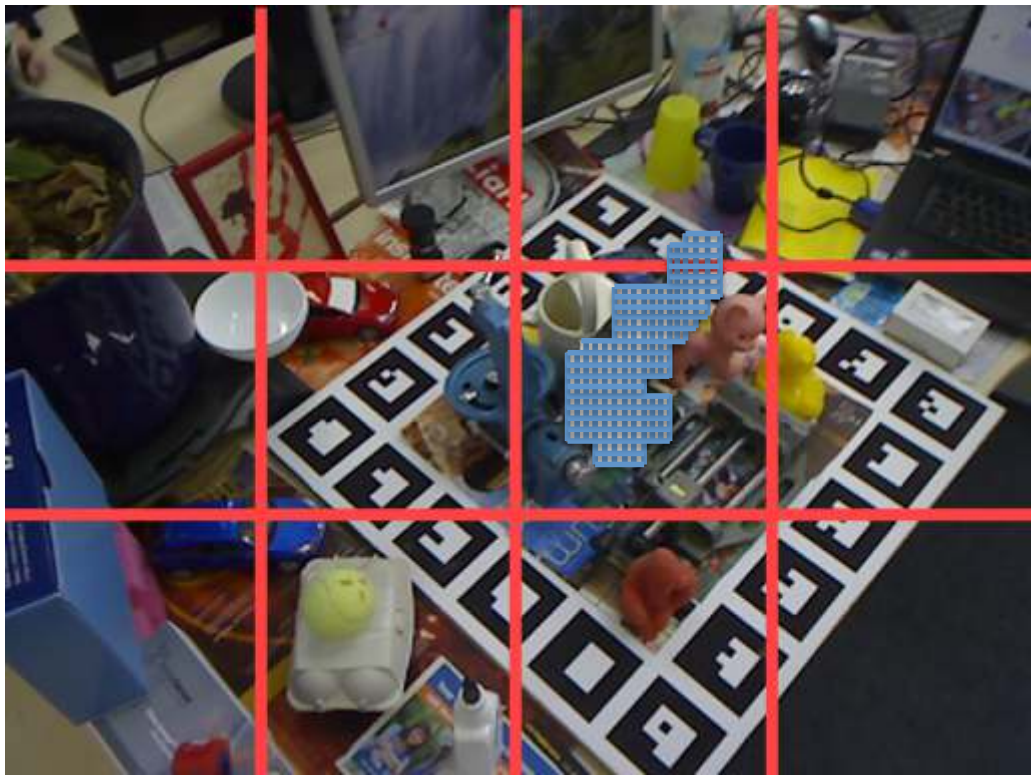


2. We segment each region as an 8x8 binary mask.
Each block of the masks corresponds to a 16x16 image window

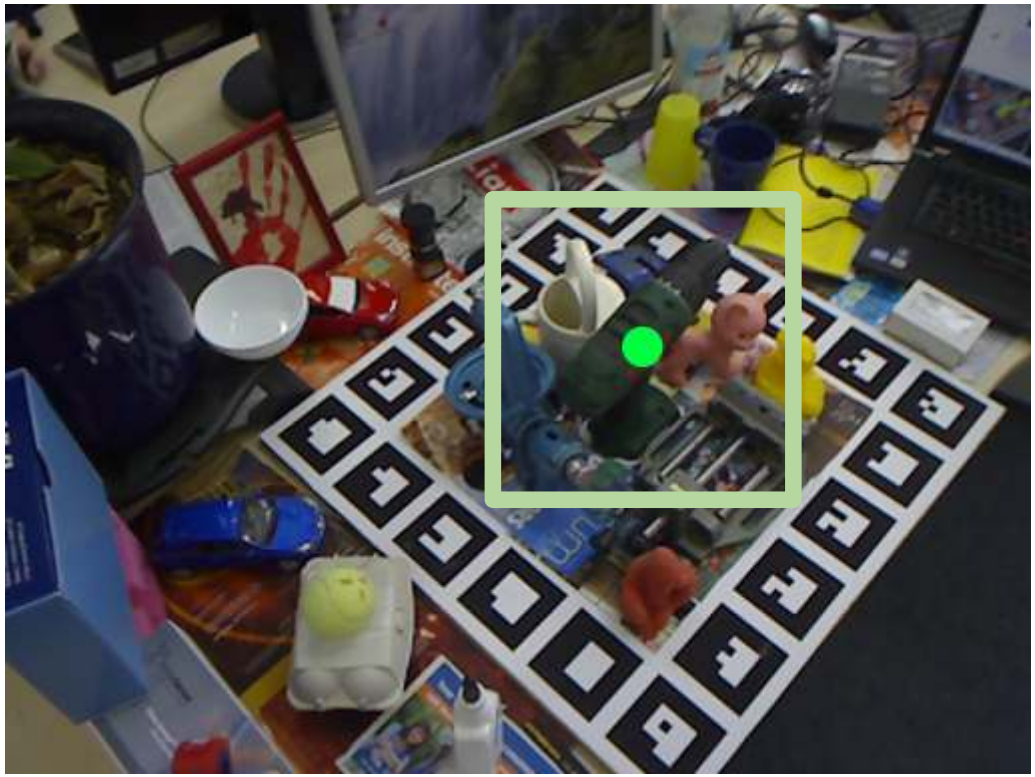


3. We only keep the largest component.

4. We segment each active block again.
Decreases the uncertainty from 16px to 4px

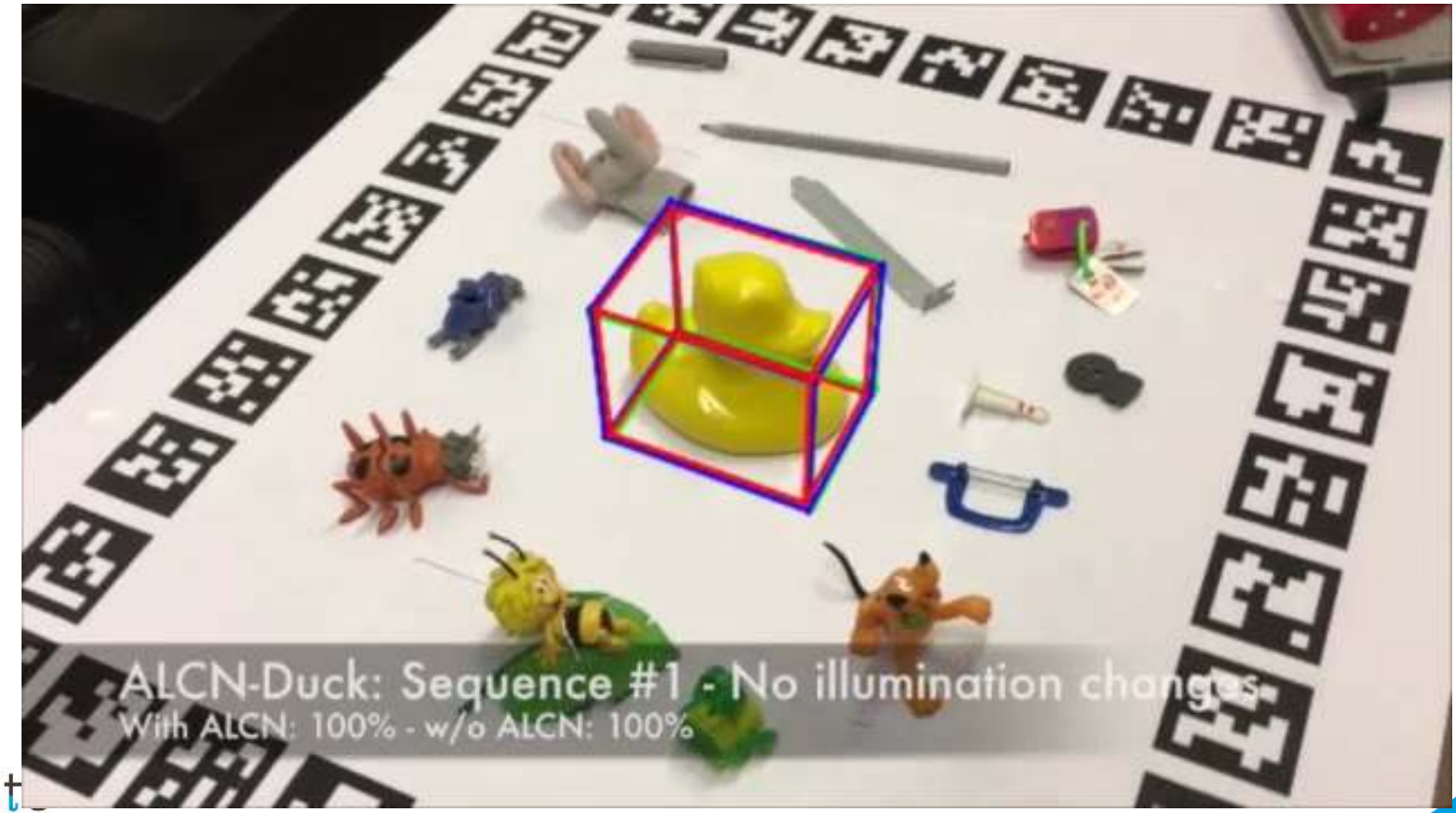


5. We use the centroid of the segmentation as the center of window $W$



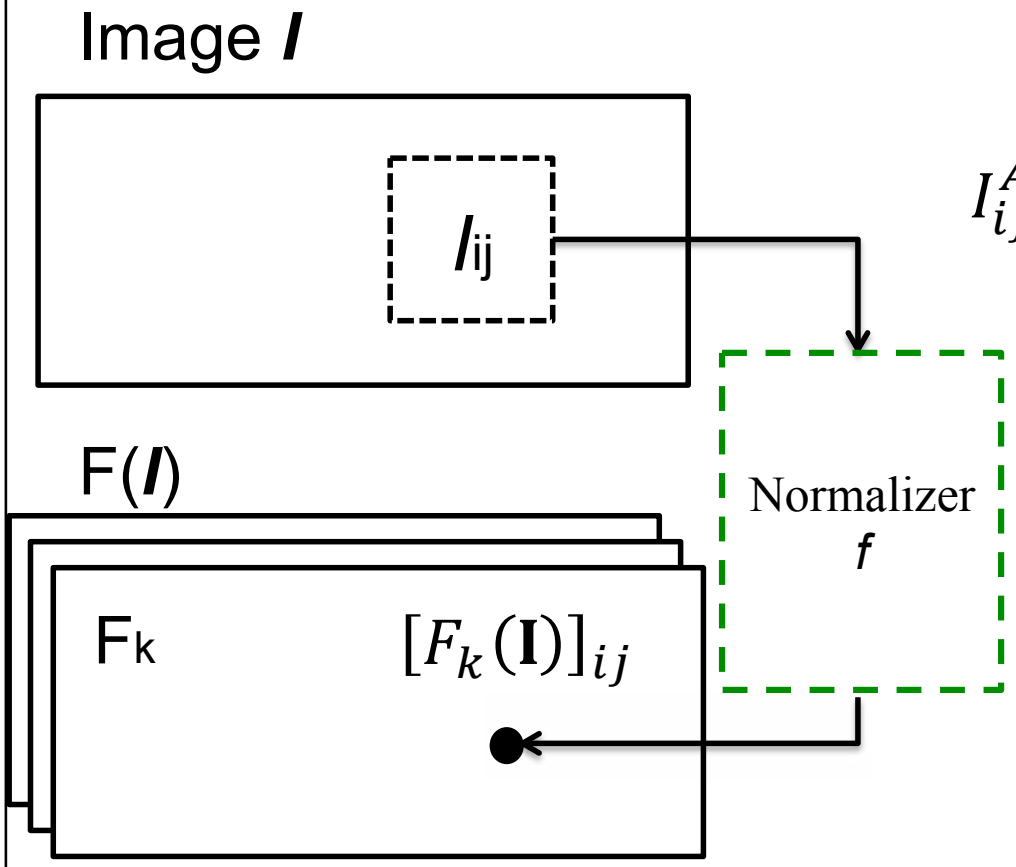
# Robustness to Light Changes: Adaptive Local Contrast Normalization

Normalization to illumination model: $\mathbf{ALCN}(I;\, w) = \left( \sum_{i=1}^{N} w_i \cdot G_{\sigma_i^{\mathrm{ALCN}}} \right) * I$



# Comparing with Existing Methods

# From Image Window Normalization to Whole Image Normalization

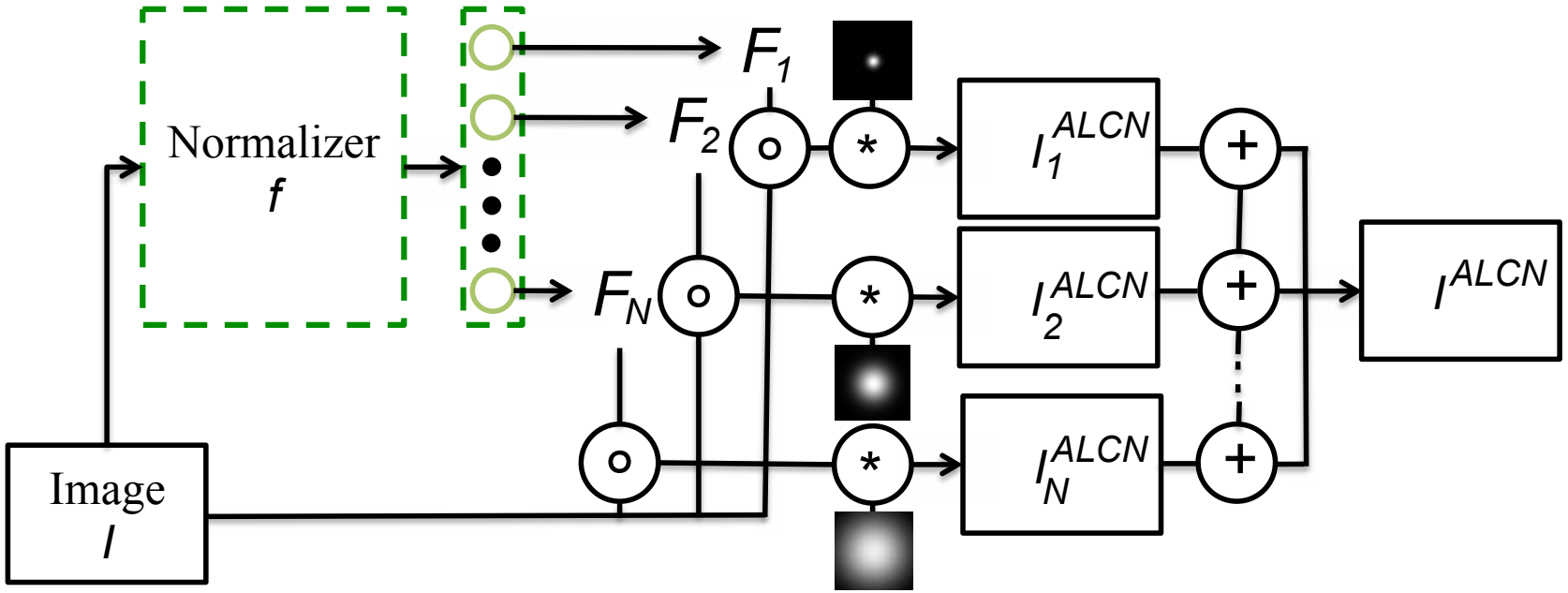



$$I_{ij}^{ALCN} = \left( \sum_{k=1}^{N} f_k(I_{ij}) . G_{\sigma_k^{ALCN}} \right) * I_{ij}$$

$$\boldsymbol{I}^{ALCN} = \sum_{k=1}^{N} G_{\sigma_k^{ALCN}} * (F(\boldsymbol{I}) \circ \boldsymbol{I})$$

$$[F_k(\mathbf{I})]_{ij} = f_k(I_{ij})$$

# From Image Window Normalization to Whole Image Normalization



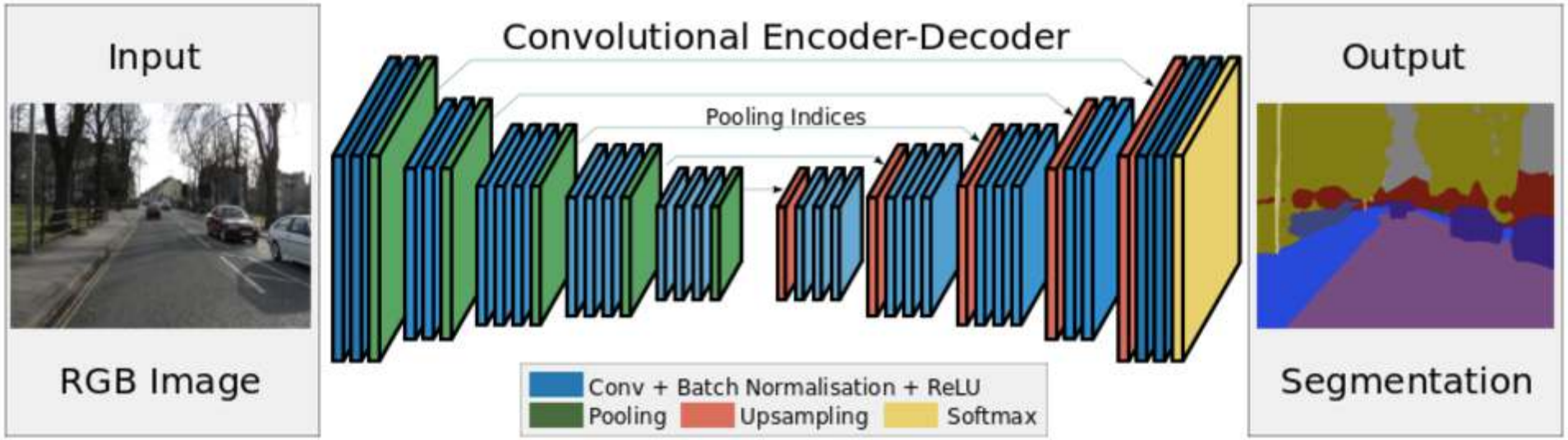
# Robustness to Light Changes: Adaptive Local Contrast Normalization

How we can detect the target object:

- Under challenging illumination conditions
- From very few training samples

Example:

# Semantic Segmentation



# SegNet Video

# Images with Various Illuminations After Filtering