SOCCER ROBOT PERCEPTION

PRATIKA KOCHAR

AGENDA

1. MOTIVATION
2. PROBLEM STATEMENT
3. DATASETS
4. PRELIMINARIES
5. IMPLEMENTATION
6. RESULTS AND MODEL EVALUATION

MOTIVATION

Humanoid League contributes to the goal of beating the human soccer world champion by 2050 by gradually training the robot all FIFA rules (making the game rules more)

New constraint: Field dimensions updated to 14×9 m (from 9x6 m), with 2 players (robots)

Problems:

- **Perception** (further away balls and goalposts to be detected)
- **Localization** (robust line detection and state estimation)

Today's talk is based on:

RoboCup 2019 AdultSize Winner NimbRo: winner of all competitions in the AdultSize class for the RoboCup 2019 Humanoids League in Sydney

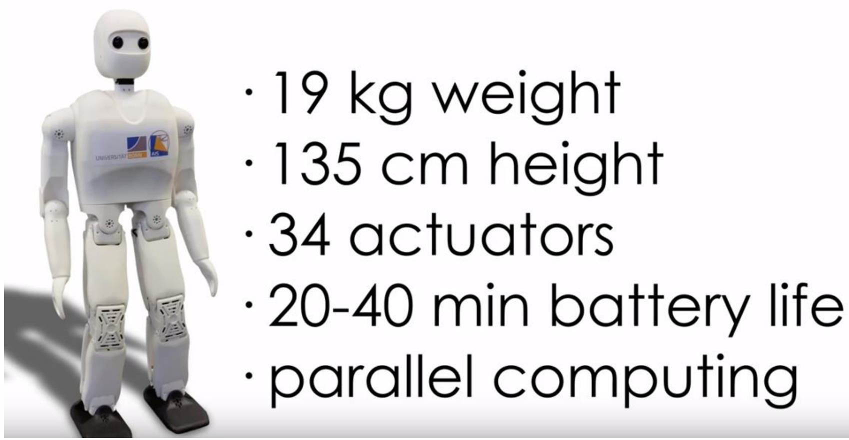
MOTIVATION


RoboCup2019
UNIVERSITÄT BONN
AIS
Humanoid AdultSize Team NimbRo

NIMBRO SPECS

NimbRo robot

- visual perception: Logitech C905 USB camera with wide-angle lens
- Robust to very bright and very dark conditions
- Detection range 10 m



NimbRo

Logitech C905 USB camera

PROBLEM STATEMENT

Reproduction of existing visual perception system for soccer-related objects

- Detecting ball, goalposts, and robots
- Classification (segmentation) of field and lines

through the usage of texture, shape, brightness, and color information

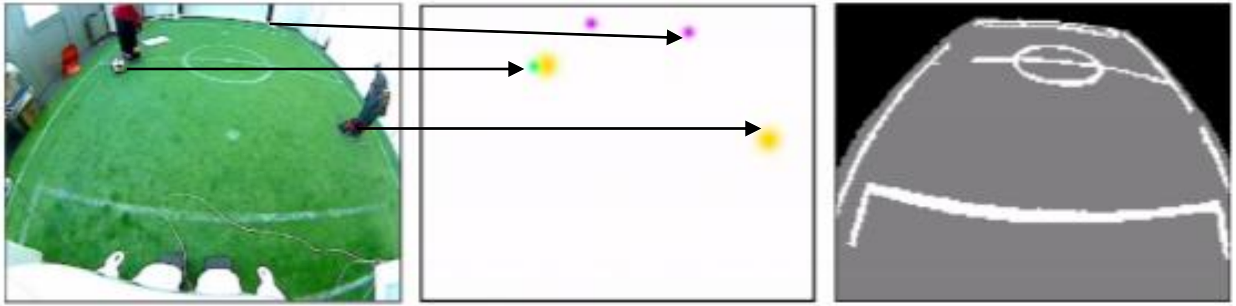


Image captured by robot

Expected output of the network indicating balls (cyan), goal posts (magenta), and robots (yellow).

Expected output of the segmentation branch showing lines (white), field (gray), and background (black)

DATASETS

Two datasets:

- **Object Detection** for ball, goalposts, and robots

Input RGB image, along with annotations containing landmark points for each relevant object

- **Segmentation** for background, field and lines

Input RGB image, along with the ground truth (mask)

DATA SET FOR OBJECT DETECTION

Training Data: 1598 images **Test Data:** 150 images **Validation Data:** 150 images


```xml
<object>
  <name>robot</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>
    <xmin>455</xmin>
    <ymin>281</ymin>
    <xmax>527</xmax>
    <ymax>363</ymax>
  </bndbox>
</object>
<object>
  <name>ball</name>
  <pose>Unspecified</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>
    <xmin>299</xmin>
    <ymin>220</ymin>
    <xmax>354</xmax>
    <ymax>274</ymax>
  </bndbox>
</object>
```

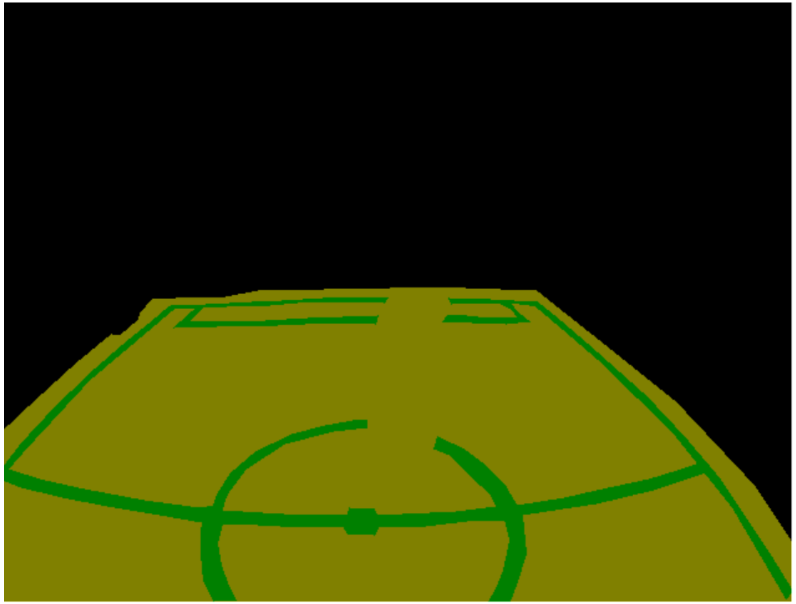

XML containing locations for ball, robot, goalpost

Corresponding Input Image

DATA SET FOR SEGMENTATION

Training set: 886 images **Test set:** 111 images **Validation set:** 111 images

Input Image



Ground Truth

KEY FEATURE: LOCATION-DEPENDENT CONV. LAYER

Requirement?

- Recognition of location-dependent features

Why do need this information?

- Task of the video prediction, for instance, learning the location of static obstacles in the environment leads to better frame forecasting.

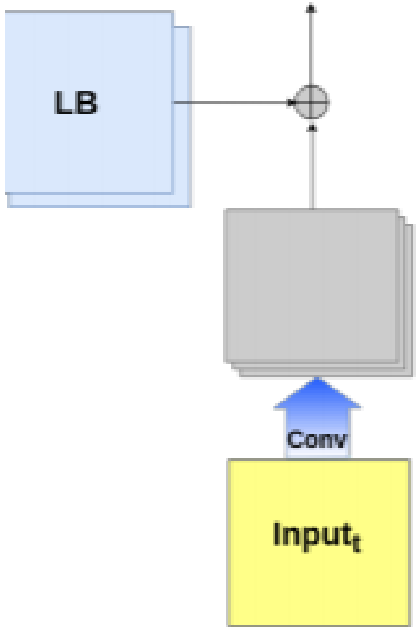
Problem?

- Convolutions are **location-invariant** => Convolutional deep learning architectures cannot recognize location-dependent features

Solution: Location-dependent convolutional layer

- Convolutional layers with learnable location-dependent biases

LOCATION-DEPENDENT CONVOLUTIONAL LAYER



Replace convolutional layer with location-dependent convolutional layer

$$LC(x,y) = A\left(\sum_{i,j}\Big(I(x+i,y+j) * W(i,j) + b\Big) + W_1'(x,y) + W_2'(x,y)\right)$$

- A is the activation function
- W and b are the weight and bias of the specified layer
- I(x, y) is the input vector at the Cartesian position (x, y)
- W's are location-dependent weights learned through the training procedure

Location Dependant Convolution

Source: www.ais.uni-bonn.de/~hfarazi/papers/LocDep.pdf

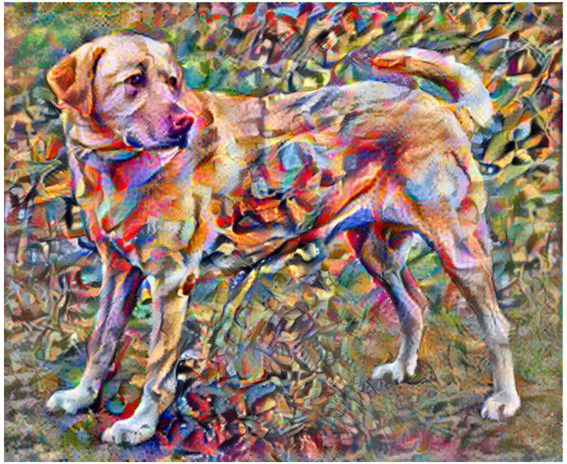
TOWARDS SMOOTH RESULTS

Total Variation Loss

Used to decrease lot of high frequency artifacts by using an explicit regularization term on the high frequency components of the image. In style transfer, this is often called the *total variation loss*

+

=



Without total variation loss

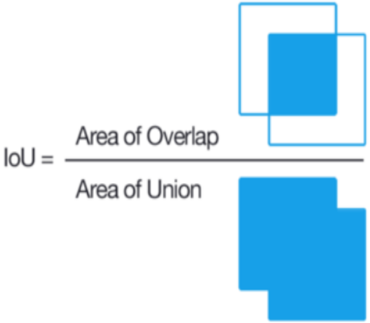
or

With total variation loss

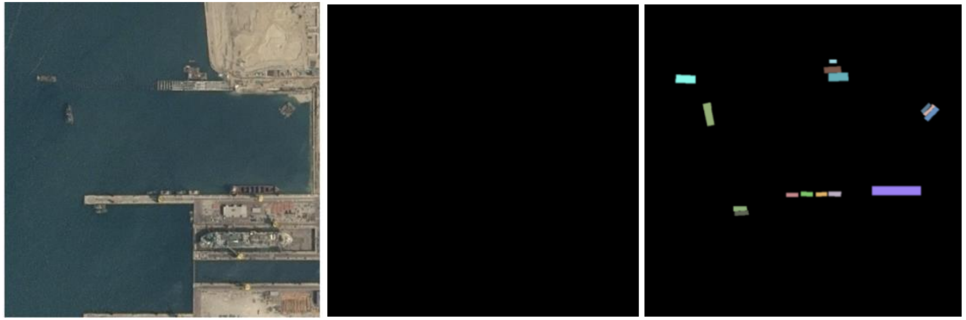
Source: www.tensorflow.org/tutorials/generative/style_transfer

INTERSECTION OVER UNION METRIC

- Metric ranges from 0–1 (0–100%) with 0 signifying no overlap (garbage) and 1 signifying perfectly overlapping segmentation



- **Better metric** as compared to pixel accuracy in case of class imbalance. For example:



Input Image Model Prediction Segmented Ground truth

Pixel accuracy* = 95%

IOU* = 47.5 %

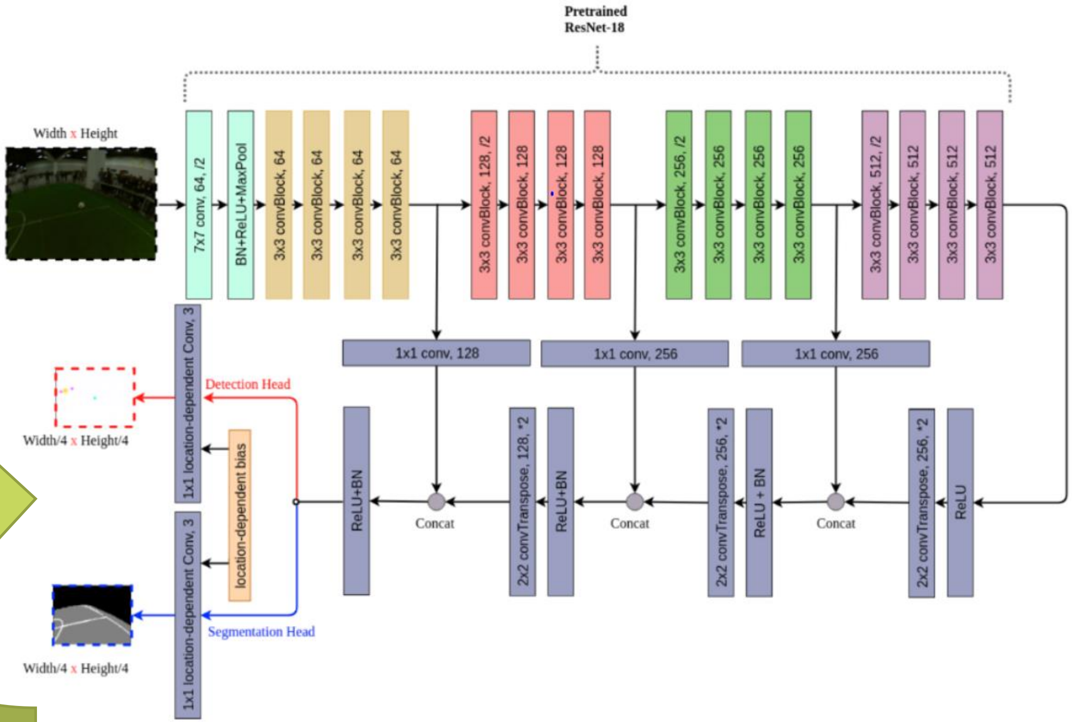
[Source: towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model](towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model)

MODEL ARCHITECTURE

Unified deep convolutional neural network to perform object detection and pixel-wise classification with one forward pass

- Encoder-decoder architecture
- Since ResNet is originally designed for recognition tasks, Global Average Pooling (GAP) and the fully connected layers in the model are removed
- Transpose-convolutional layers for up-sampling the representations

Source: [RoboCup 2019 AdultSize Winner NimbRo]()



Two output heads; one for object detection, and the other for pixel-wise segmentation

IMPLEMENTATION

Model Training

- Single model is trained jointly for both tasks
- A lower learning rate is employed for the encoder part, with the intuition that the pre-trained model needs less training time to converge
- Adam optimizer, which has an adaptable per-parameter learning rate

For Object Detection Head

- Target is constructed by Gaussian blobs around the ball center and bottom-middle points of the goalposts and robots
- Bigger radius for robots with the intuition that annotating a canonical center point is more difficult
- Loss: Mean squared error

IMPLEMENTATION: OBJECT DETECTION

Heatmap Generation

1. Define colors for each object

- cyan for ball: 0,255,255
- magenta for goalposts: 255,0,255
- yellow for robot: 255,255,0

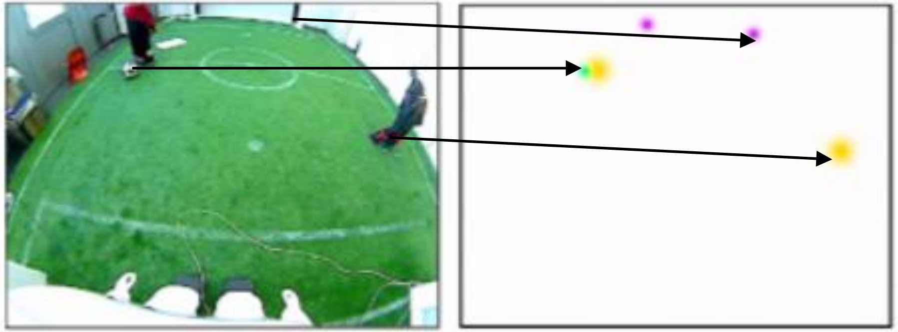
2. Define radius for each object

Radius = [11 , 15, 27]

3. Create gaussian kernel for each object

4. Set color depending on the object

5. Insert this kernel on a white image (dimensions as that of input image)



Input Image   Heatmap

IMPLEMENTATION: SEGMENTATION

For Segmentation Head

- Loss: Pixelwise Negative Log Likelihood

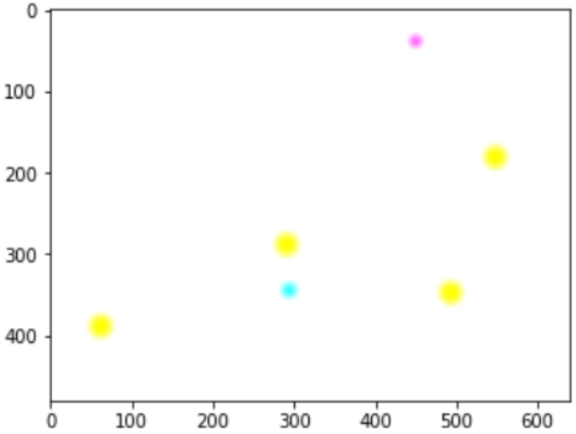
- Total Variation loss to the output of all result channels except the line segmentation channel

Total Variation loss encourages blob response => less false positives especially in field detection

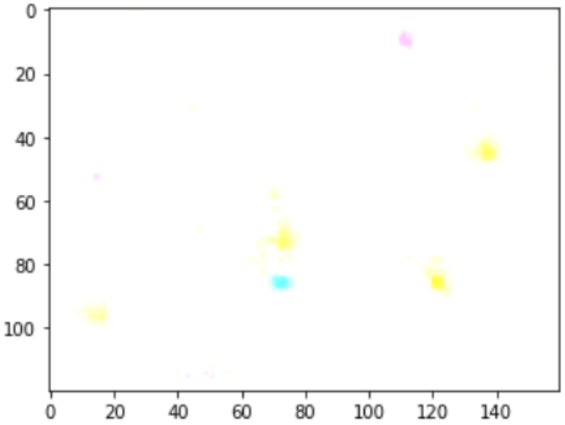
RESULTS

Object Detection Head

Input Image



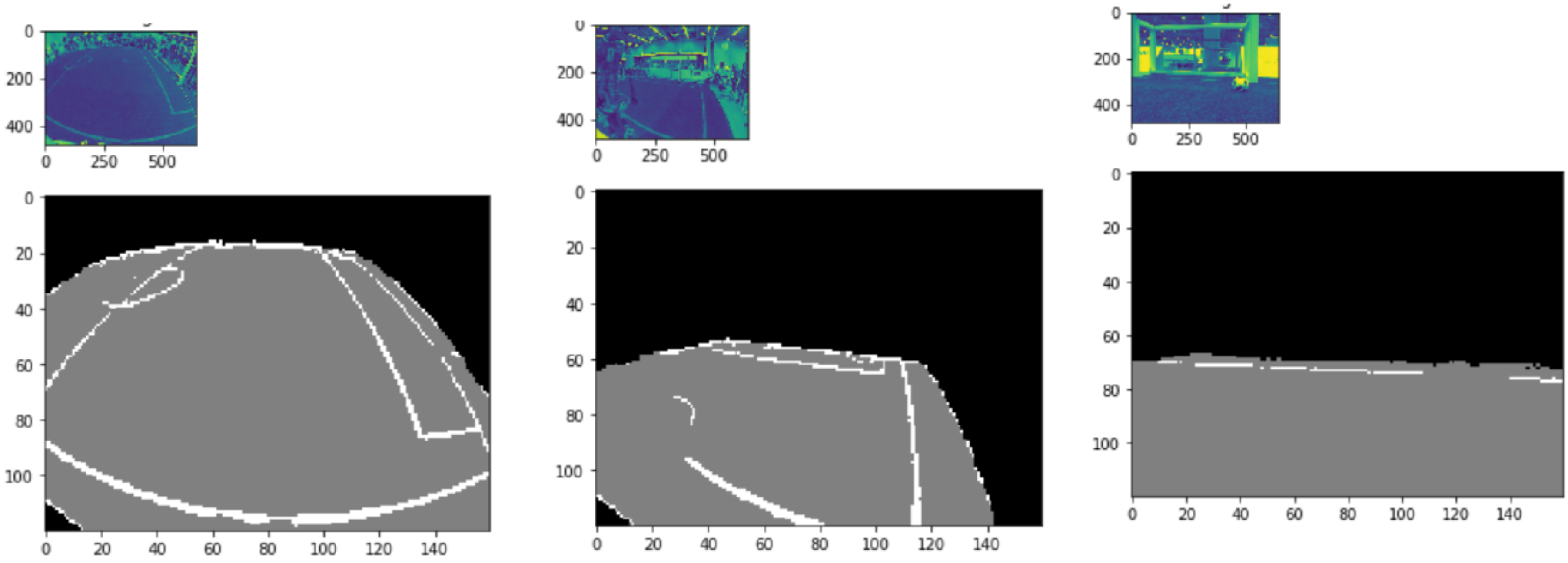
Ground Truth



Output: A heatmap corresponding to ball, goalpost and robot after thresholding

RESULTS

For Segmentation Head



First row corresponds to input images, second row segmented output: White for lines, gray for field, black for background

MODEL EVALUATION

Object Detection Head

Original implementation

Type	F1	Accuracy	Recall	Precision	FDR
Ball (NimbRoNet2)	**0.998**	**0.996**	0.996	**1.0**	**0.0**
Ball (NimbRoNet)	0.997	0.994	**1.0**	0.994	0.005
Ball (SweatyNet-1 [11])	0.985	0.973	0.988	0.983	0.016
Goal (NimbRoNet2)	**0.981**	**0.971**	0.973	**0.988**	**0.011**
Goal (NimbRoNet)	0.977	0.967	**0.988**	0.966	0.033
Goal (SweatyNet-1 [11])	0.963	0.946	0.966	0.960	0.039
Robot (NimbRoNet2)	**0.979**	**0.973**	**0.963**	**0.995**	**0.004**
Robot (NimbRoNet)	0.974	0.971	0.957	0.992	0.007
Robot (SweatyNet-1 [11])	0.940	0.932	0.957	0.924	0.075
Total (NimbRoNet2)	**0.986**	**0.986**	0.977	**0.994**	**0.005**
Total (NimbRoNet)	0.983	0.977	**0.982**	0.984	0.015
Total (SweatyNet-1 [11])	0.963	0.950	0.970	0.956	0.043

My implementation

Type	*F1*	*Accuracy*	*Recall*	*Precision*	*FDR*
Ball	0.989	0.98	1.0	0.98	0.019
Goal Post	0.979	0.96	0.979	0.97	0.02
Robot	0.959	0.933	0.95	0.97	0.024
Total	0.975	0.957	0.97	0.973	0.063

MODEL EVALUATION

Segmentation Head

Original implementation

Type	Accuracy	IOU
Field	0.986	0.975
Lines	0.881	0.784
Background	0.993	0.981
Total	0.953	0.913

My implementation

Type	*Accuracy*	*IOU*
Field	0.98	0.967
Lines	0.80	0.726
Background	0.989	0.979
Total	0.925	0.888

Similar results except for (field) lines

THANK YOU