Fast and Precise Black and White Ball Detection for RoboCup Soccer

Jacob Menashe, Josh Kelle, Katie Genter, Josiah Hanna,
Elad Liebman, Sanmit Narvekar, Ruohan Zhang, and Peter Stone

{jmenashe,jkelle,katie,jphanna,eladlieb,sanmit,zharucs,pstone}@cs.utexas.edu

The Learning Agents Research Group
The University of Texas at Austin
Austin, Texas

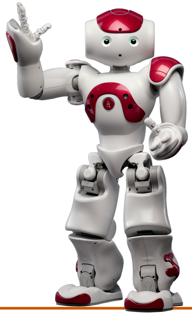
The RoboCup Symposium, 2017

UT Austin Villa

- Soccer SPL got 2nd place last year
- 3D Sim got 1st place 6 times in the last 7 years
- @Home DSPL got 3rd place this year

Nao Robot

- Soccer SPL uses the Nao robot
- 2 non-stereoscopic cameras at 30Hz
- Maximum resolution of 1280x960



Black and White Soccer Balls

- Bright orange through 2015
- 2016: Standard black and white pattern
- Far more difficult given hardware constraints

How to Detect an Orange Ball

1. Read a subsampled image and apply color table
2. Make a list of orange pixels
3. Scan at high resolution around each orange pixel
4. Calculate statistics (roundness, green below, etc)
5. Choose the best candidate

50% recall at 8m

How to Detect a Black and White Ball

1. Read a subsampled image and apply color table
2. ~~Make a list of orange pixels~~
3. ~~Scan at high resolution around each orange pixel~~
4. Calculate statistics (roundness, green below, etc)
5. Choose the best candidate

How to Detect a Black and White Ball

1. Read a subsampled image and apply color table
2. Identify triangular collections of black pentagons
3. Calculate statistics (roundness, green below, etc)
4. Choose the best candidate

How to Detect a Black and White Ball

1. Read a subsampled image and apply color table
2. Identify triangular collections of black pentagons
3. Calculate statistics (roundness, green below, etc)
4. Choose the best candidate
5. Refine position estimate

How to Detect a Black and White Ball

1. Read a subsampled image and apply color table
2. Identify triangular collections of black pentagons
3. Calculate statistics (roundness, green below, etc)
4. Choose the best candidate
5. Refine position estimate
6. Filter with binary classifier

How to Detect a Black and White Ball

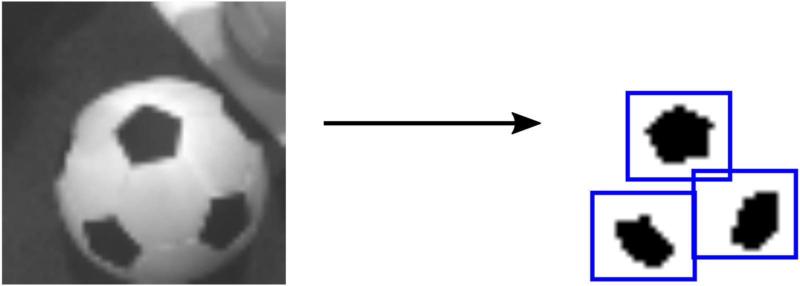
1. Read a subsampled image and apply color table
2. Identify triangular collections of black pentagons
3. Calculate statistics (roundness, green below, etc)
4. Choose the best candidate
5. Refine position estimate
6. Filter with binary classifier

50% recall at 4 meters

Identifying Potential Pentagons

Adaptive Thresholding[1]

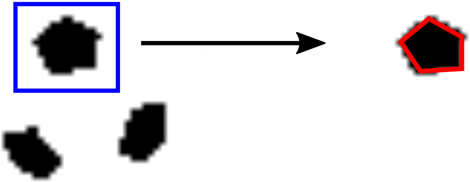
We identify areas of high contrast by using adaptive thresholding. These regions of interest (ROIs) are registered for further evaluation.



Pentagon Formation

Contiguous Blob Reconstruction

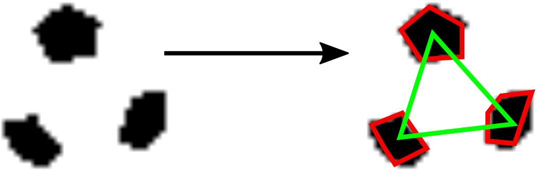
Candidate ball pentagons are found by examining each ROI and reconstructing its bounds by connecting contiguous pixels in black/white pixel space. Bad candidates (too large, bad proportions) are thrown out.



Triangle Construction

Triplet Enumeration and Comparison

The best N black blobs in the image (with respect to pentagon similarity) are arranged into all possible triplets. Each triplet's induced triangle is then used to generate a ball candidate.



Candidate Scoring

Parallel Statistic Evaluation

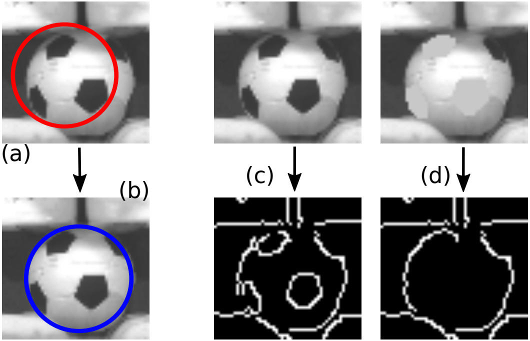
We gather metrics to score the quality of each candidate based on color, shape, size, and distance. All metrics are evaluated in parallel to produce a final candidate score [3].

- Green Below Ball
- Ball Green Percent
- Height
- Width/Projection Discrepancy
- Distance from Field
- Velocity

Position Refinement

Hough Circle Detection

We use a Hough circle detector to improve our estimate of the ball's location to account for variations in rotation and triangle positioning.



Learned Candidate Filters

Filtering with Binary Classifiers

New detections are filtered with a trained binary classifier. We use a neural network trained on a large training set of positive and negative examples gathered semi-autonomously.

Positive

Negative

NN Architectures

We compared four NN architectures[1]:

- Conv-1 - Coarse Convolutional NN
- Conv-2 - Fine Convolutional NN
- Fc-1 - One Fully Connected Layer
- Fc-2 - Two Fully Connected Layers

[1] All NNs implemented with Caffe[2]

NN Architecture Comparison Results

	Time	#Params	Precision	Recall	Accuracy
Conv-1	320s	**1,106**	.9797	.9746	.9907
Conv-2	213s	5,314	**.9948**	**.9941**	**.9977**
Fc-1	**12s**	6,146	.9251	.9341	.9712
Fc-2	116s	1,574,402	.9914	.9772	.9936

Table : Classification results of neural network classifiers.

NN Transfer Results: RoboCup $\rightarrow$ US Open

	RoboCup16	$\rightarrow$	RoboCup16
	Precision	Recall	Accuracy
Conv-1	.9820	.9776	.9928
Conv-2	**1.000**	**.9910**	**.9984**
Fc-1	.9623	.9731	.9884
Fc-2	.9977	.9843	.9968

	RoboCup16	$\rightarrow$	USopen16
	Precision	Recall	Accuracy
Conv-1	.6754	.2576	.8109
Conv-2	**.9890**	.3925	.8664
Fc-1	.8402	.5926	.8865
Fc-2	.9199	**.6064**	**.9026**

Table : Transferability using RoboCup 2016 dataset as source task and USopen 2016 dataset as target task.

NN Transfer Results: US Open → RoboCup

	USopen16	→	USopen16
	Precision	Recall	Accuracy
Conv-1	.9972	**1.000**	.9994
Conv-2	**.9991**	.9991	**.9996**
Fc-1	.9346	.9468	.9746
Fc-2	.9944	.9944	.9976
	USopen16	**→**	**RoboCup16**
	Precision	Recall	Accuracy
Conv-1	.1226	.1440	.6726
Conv-2	.4163	.8748	.7654
Fc-1	**.7349**	.8596	**.9218**
Fc-2	.7214	**.8923**	.9215

Table : Transferability results using USopen 2016 dataset as source task and RoboCup 2016 dataset as target task.

SVM Comparison Results

SVM Kernel	Accuracy	AUC	Precision	Recall
Linear	0.883	0.869	0.833	0.543
Polynomial	**0.970**	**0.990**	**0.972**	**0.881**
RBF	0.961	0.989	0.989	0.824

Table : Classification results of SVM classifiers.

SVM vs. NN

Learning Algorithm Comparison

We compared Support Vector Machines (SVM) and convolutional Neural Networks (NN) for binary classification of new detections.

- NNs train much faster on large datasets.
- NNs require more data to train effectively.
- SVMs classify faster.
- The best SVMs are comparable to the worst NNs.
- NN is preferable overall.

Conclusion

- Black-and-white ball detection can be fast and precise
- Better hardware is needed to go much further
- Future Work: handle natural lighting
- Future Work: try different network architectures

Questions?

Any questions?

References I

[1] John Bernsen. Dynamic thresholding of grey-level images. In *International conference on pattern recognition*, volume 2, pages 1251–1255, 1986.

[2] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[3] Jacob Menashe, Samuel Barrett, Katie Genter, and Peter Stone. Ut austin villa 2013: Advances in vision, kinematics, and strategy. In *The Eighth Workshop on Humanoid Soccer Robots at Humanoids 2013*, 2013.