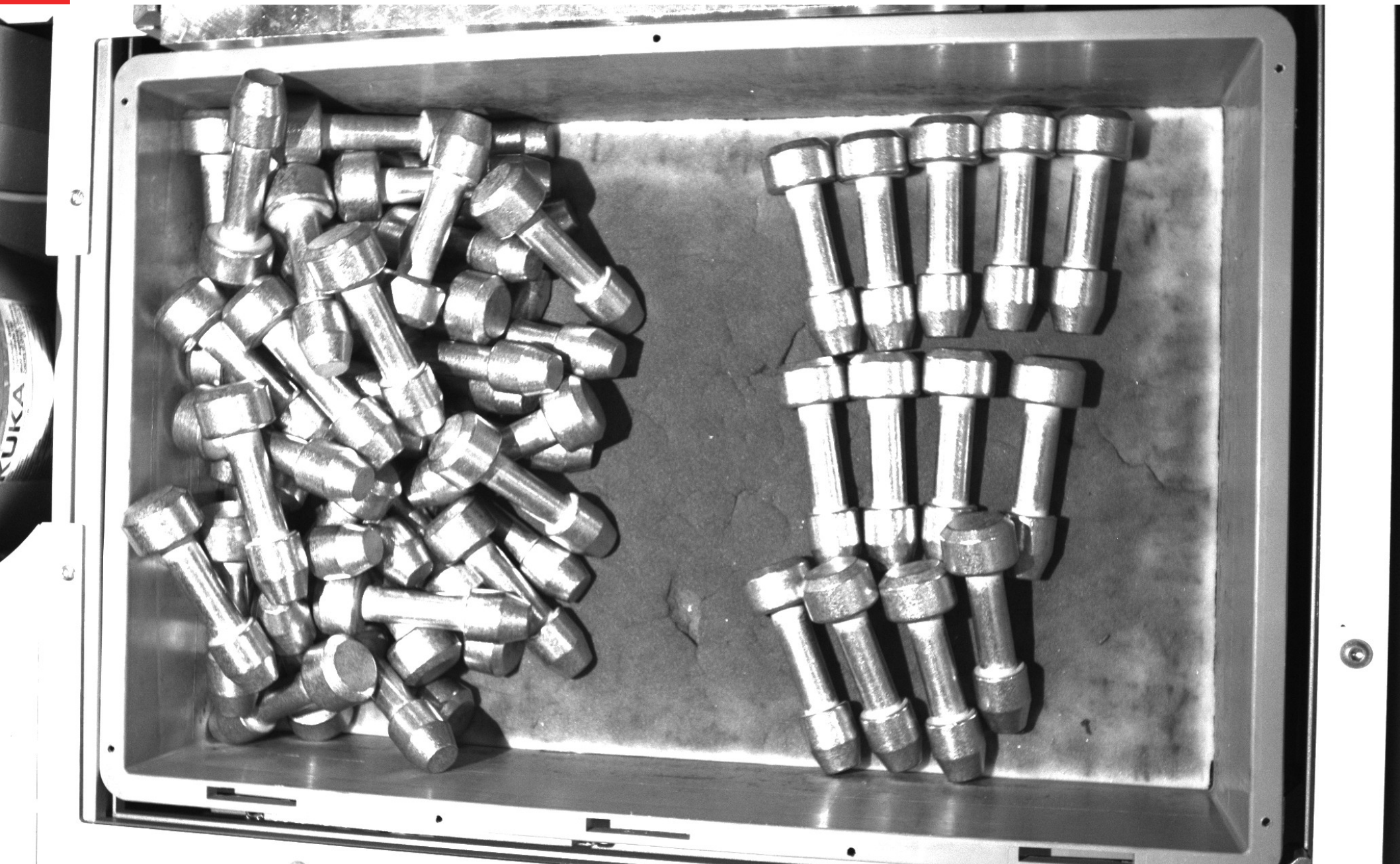
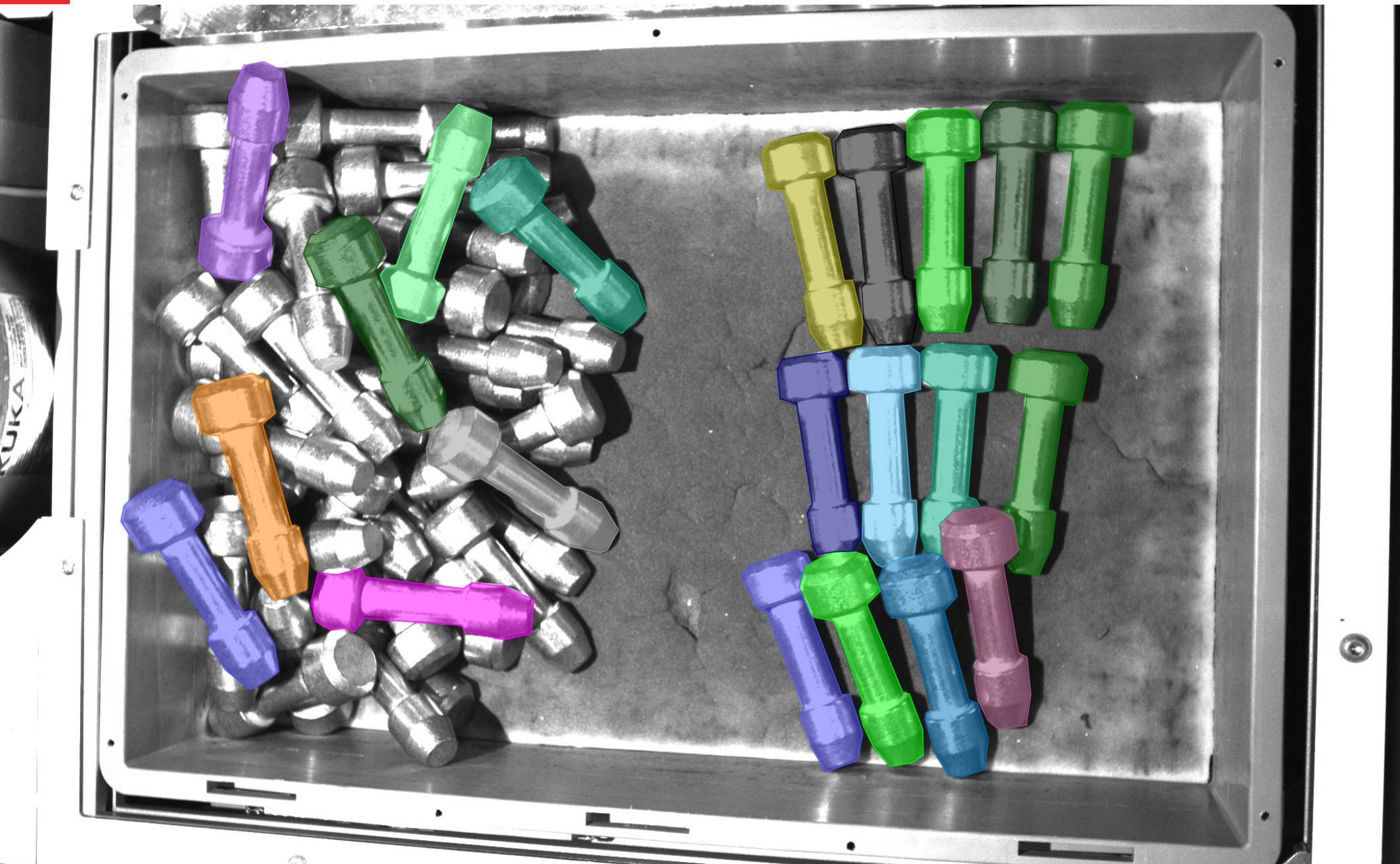
Object Detector Trained on Synthetic Data

Author: Miroslav Psota
Supervisor: Michal Malý

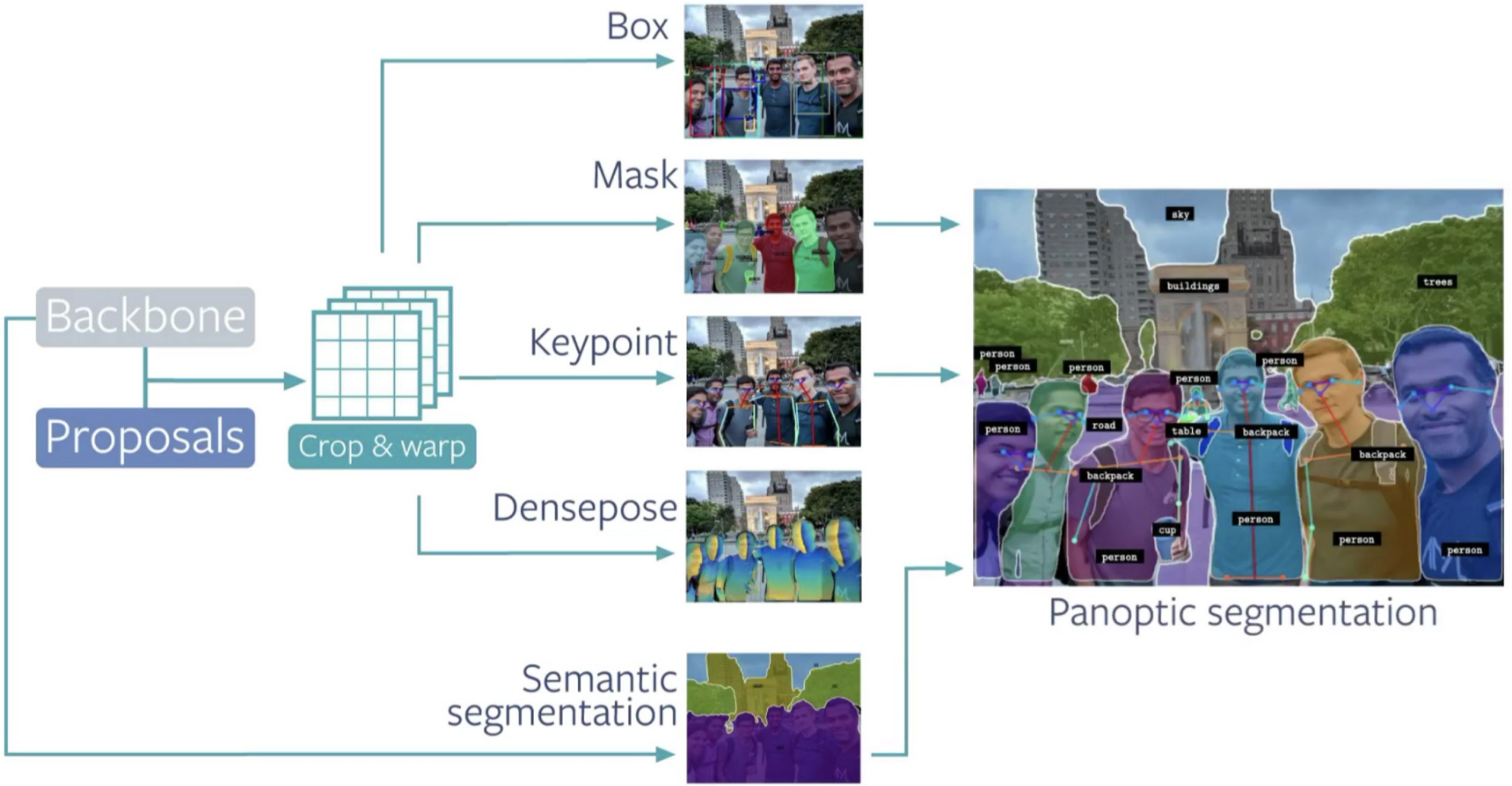
Bin picking



As instance segmentation



RCNN (meta) architecture



Framework selection

- Performance
- Easy of use
- Extensibility
- Production readiness
- Clear winner - Detectron2

Real data

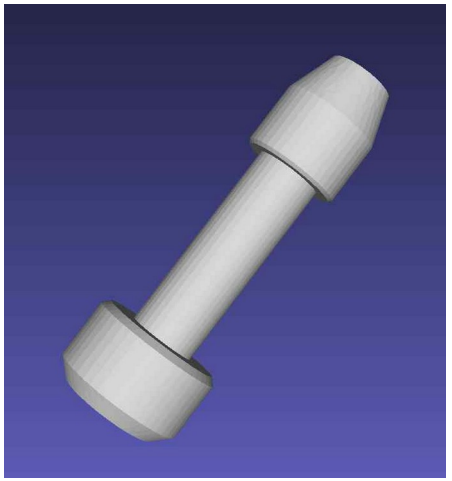
- Expensive and hard to get
- Prone to errors
- Less flexible in experimenting

- It just works

Synthetic data

- Rapid and cheap dataset generation
- Flexible in experimenting

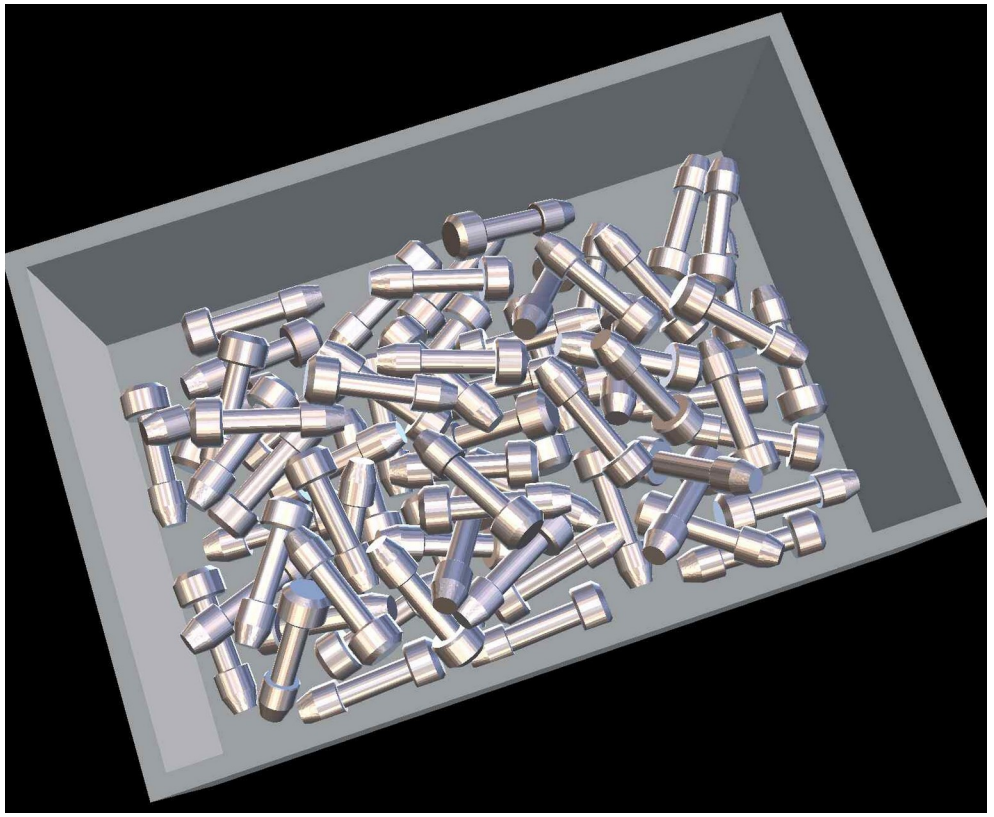
- Hard to bridge “reality gap”
- Expertise in computer graphics

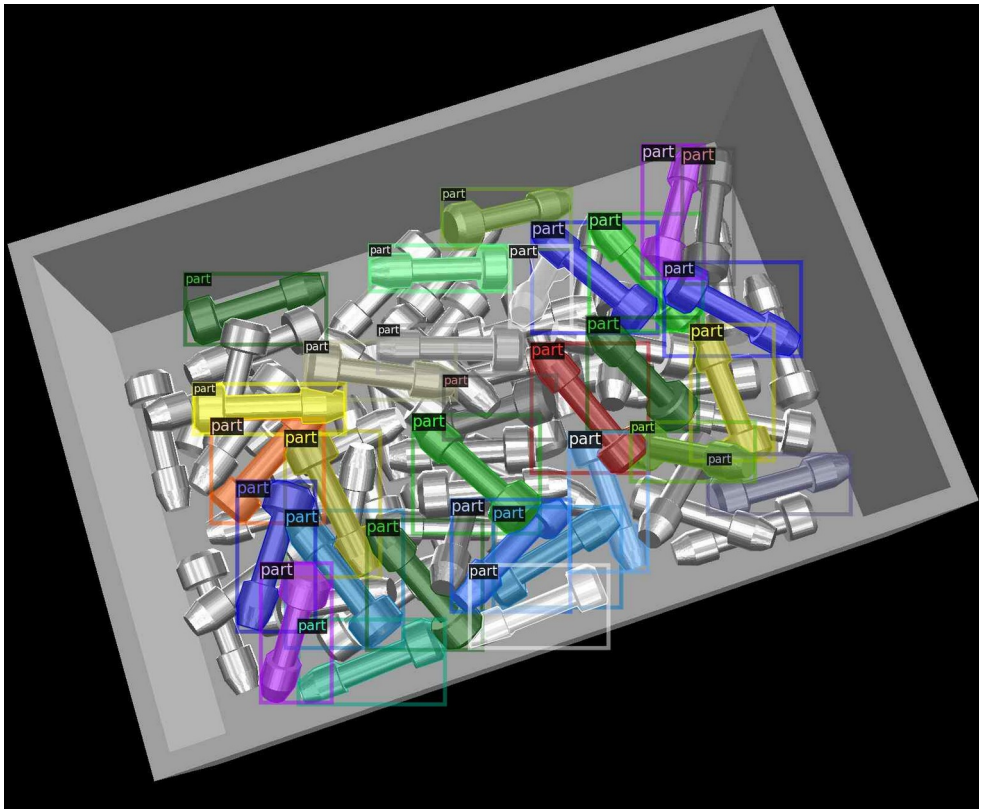


Simulator

- Using existing tool which simulates falling of object into bin
- More accurate scenario
- Slow simulation

Simulator

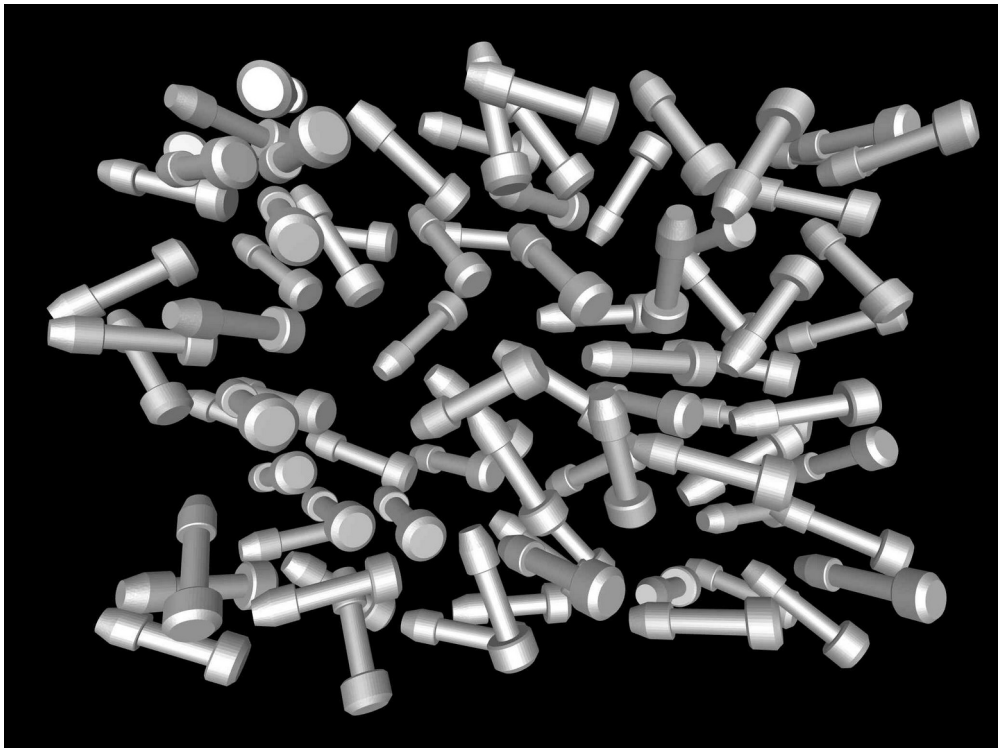


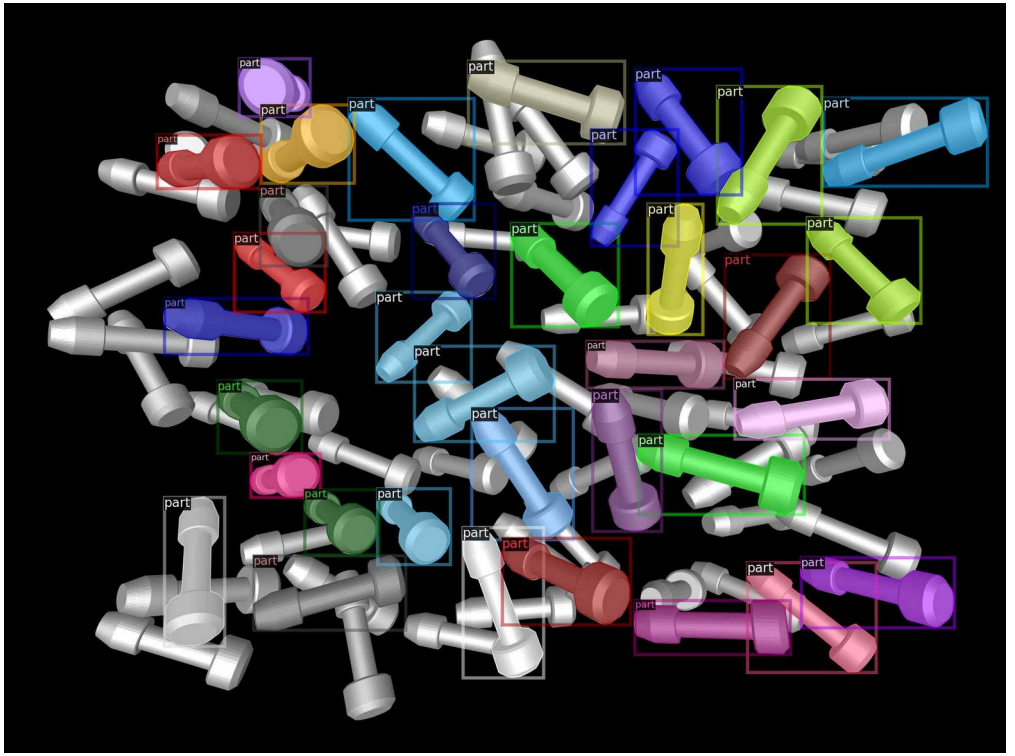


Simple generator

- Generate not colliding objects in space
- Fast generation
- Not as realistic

Simple generator





Training

- Mask RCNN, ResNet 50 backbone with slight architecture modifications
- Pretrained weights on COCO dataset
- More advanced data loading pipeline
- Grayscale input
- (Not) Freezing backbone

Training variants

- Per dataset
  - Simulator
  - Simple generator
- Frozen/unfrozen ResNet 50 backbone

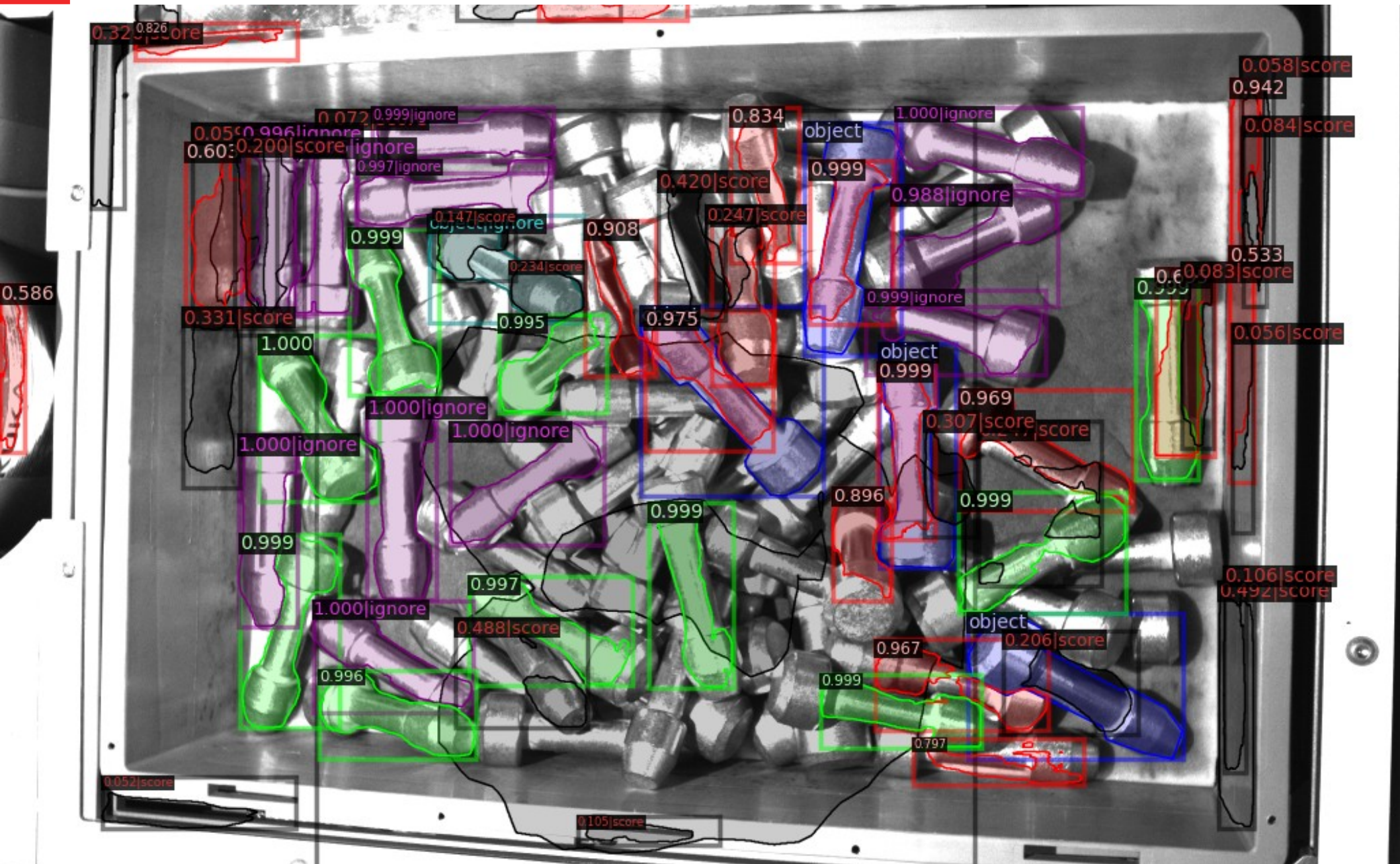
Evaluation on real data

- Modified standard COCO evaluation tools
- Some object marked as “ignore”
- Our main metrics are:
  - Precision@IoU=0.5, score>0.5
  - Recall@IoU=0.5, score>0.5
  - mAP@IoU=0.5

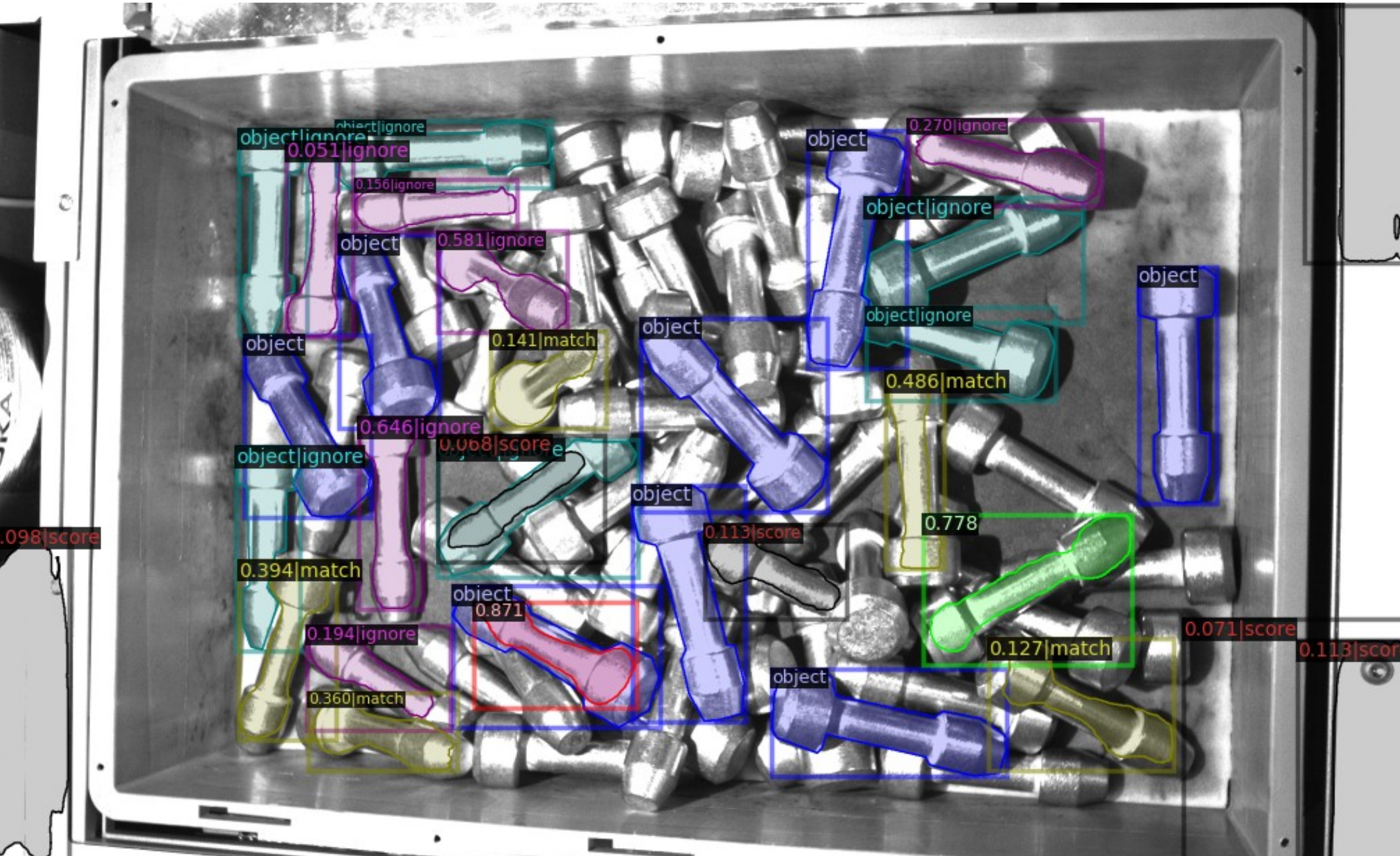
Evaluation on real data

	Precision	Recall	mAP
GEN	0.931	0.220	0.3309
GEN_F	0.625	0.163	0.2026
SIM	0.273	0.593	0.4386
SIM_F	0.240	0.537	0.3268

Example result - Sim



Example result - Gen



Conclusion

- Freezing weights is not helpful
- Data as main bottleneck
  - Reality gap still large
  - Training set almost perfectly learned

Future directions

- More realistic renderer
  - Textures, materials
  - Ambient lightning
  - Reflections on the sides of the bin
- Real time simulator
- Combine datasets (including real data)
- Use 3D data
- 6D pose estimation

Training details

- Model from Detectron2
  - We used our go-to model from other projects
- Custom input pipeline
  - imgaug as augmentation library
  - Integrated with COCO annotator
- Custom training script
- Training pipeline

Evaluation details

- COCOeval API calculates metrics
  - Added precision and recall at score API
  - Fixed bugs
- Visualizations of detections
  - Match/non-matched, ...
- Debug images and metrics

Synthetic dataset

- Simple generator
- Simulator
  - Image + png labels postprocessing

Remaining questions

- Most challenging in terms of thought and why
  - Practically oriented computer vision thesis, so more laborious and creative than a difficult topic
  - Understanding of RCNN architecture
- Comparison with other works
  - Not aware of exact same setup
  - Most similar works are better in terms of raw numbers