Single shot calibration with a known object

Purdue University - CS 635 – Final project

Mathieu Gaillard – 05/02/2020

Motivation

- With a single RGB image and a known object in this image
- We want to find a 7 parameters camera calibration
  - Focal length
  - 3 translations
  - 3 rotations
  - (no distortions)

Problem definition

- Based on a single image, and with a know object, how precisely can we calibrate a camera?

- Related problem: 6D pose estimation of a known object

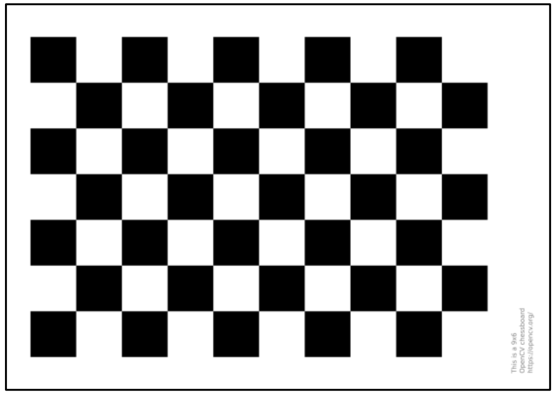
Objects

- A standard checkerboard calibration pattern (on a A4 sheet)
- A 3D model of a cat and a phone

- More generally: any 3D object
  - Lambertian surface
  - No transparency
  - No symmetries
  - Known 3D model

- Constraints: Single OBJ file with a single texture

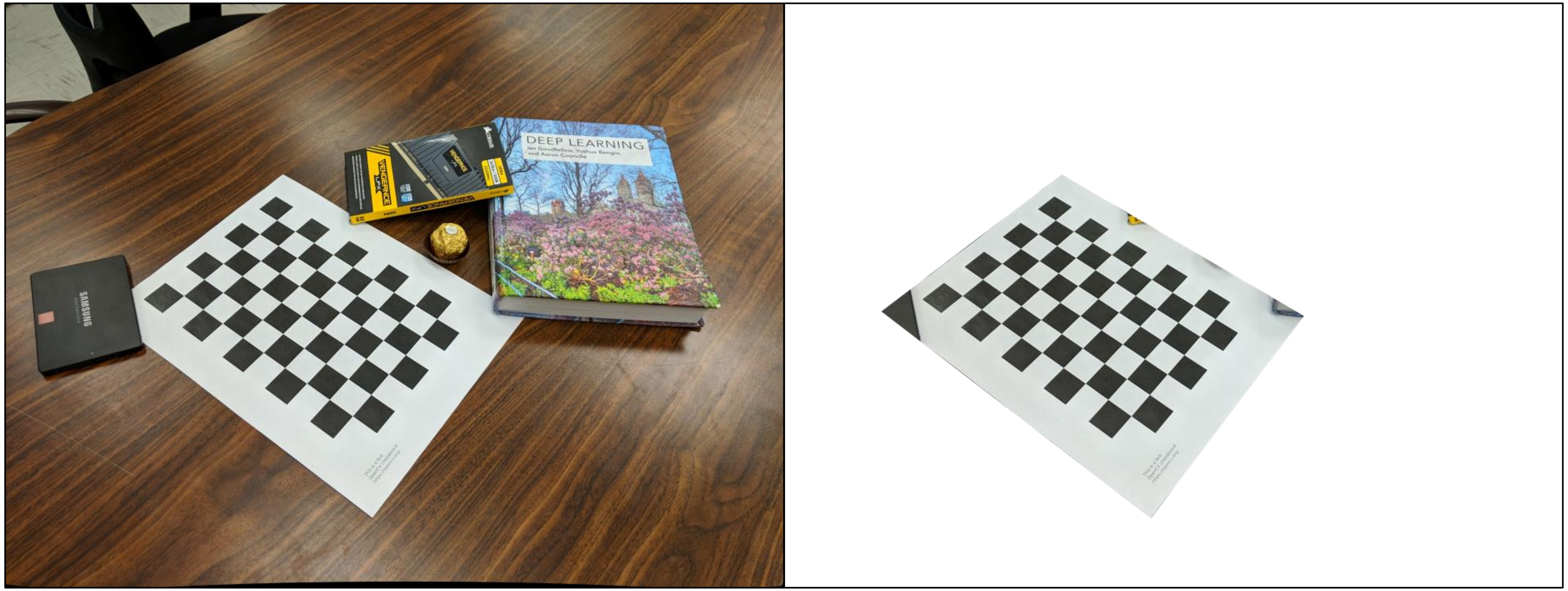
Our objects



Pipeline

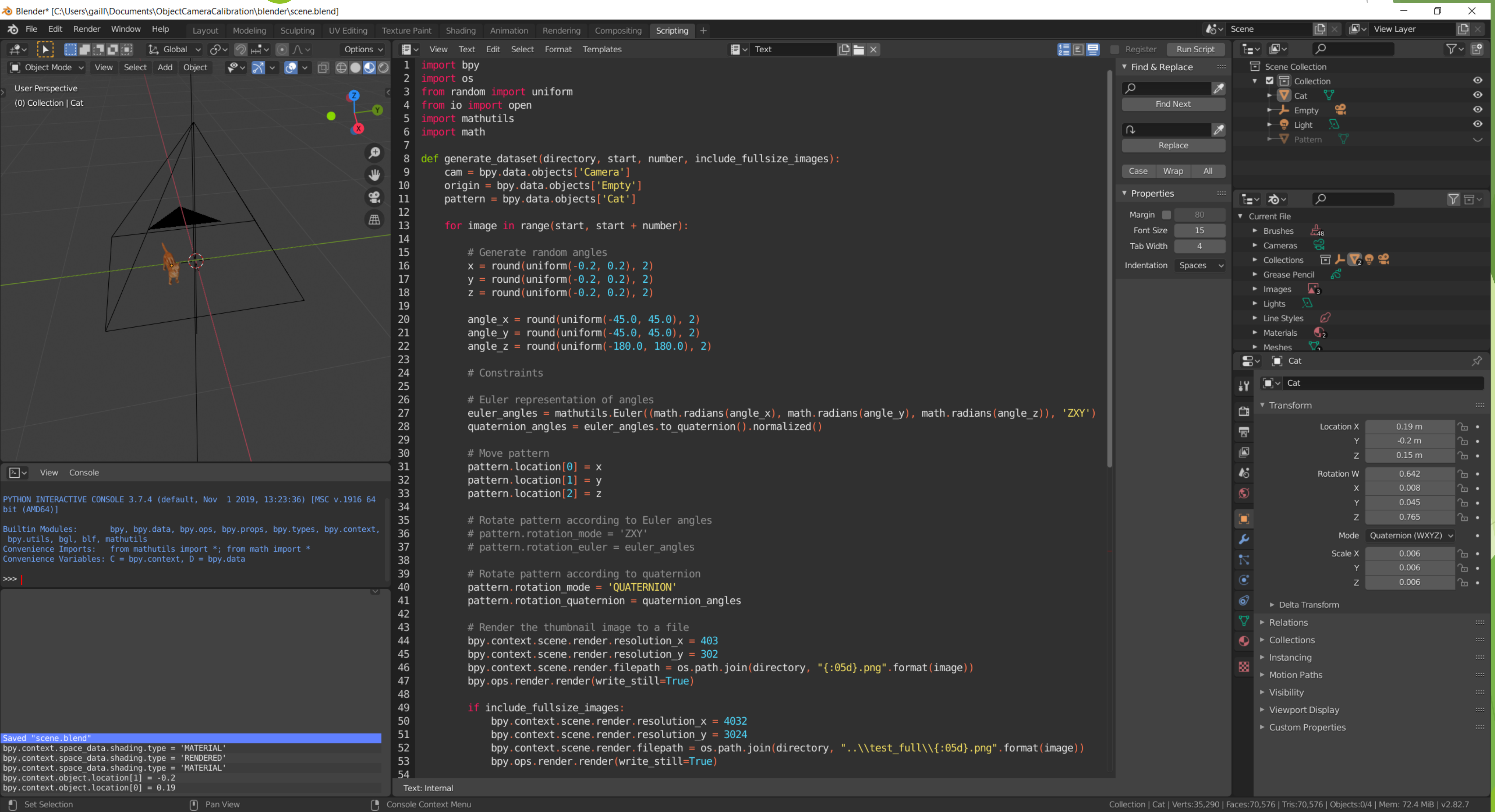
- Take a picture of an object whose 3D model is known
- Manual segmentation of the object in the picture (could be automated)
- Train a CNN to estimate the 6D pose with synthetic data
- Refinement of the pose using an image similarity measure

Picture



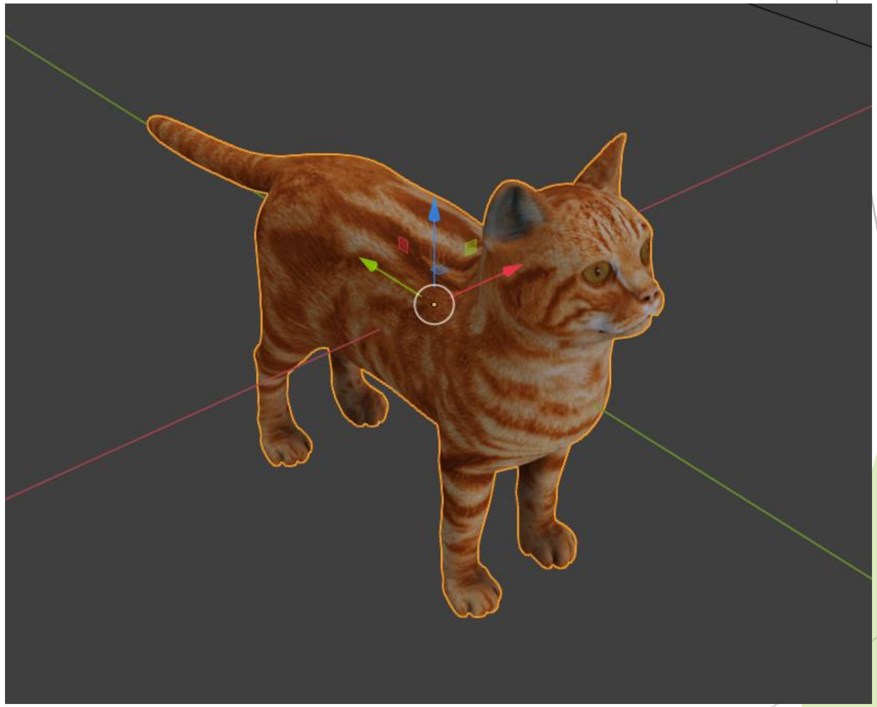
Taken with a Google Pixel 3
Undistorted using OpenCV
Segmented with Photoshop

Data generation

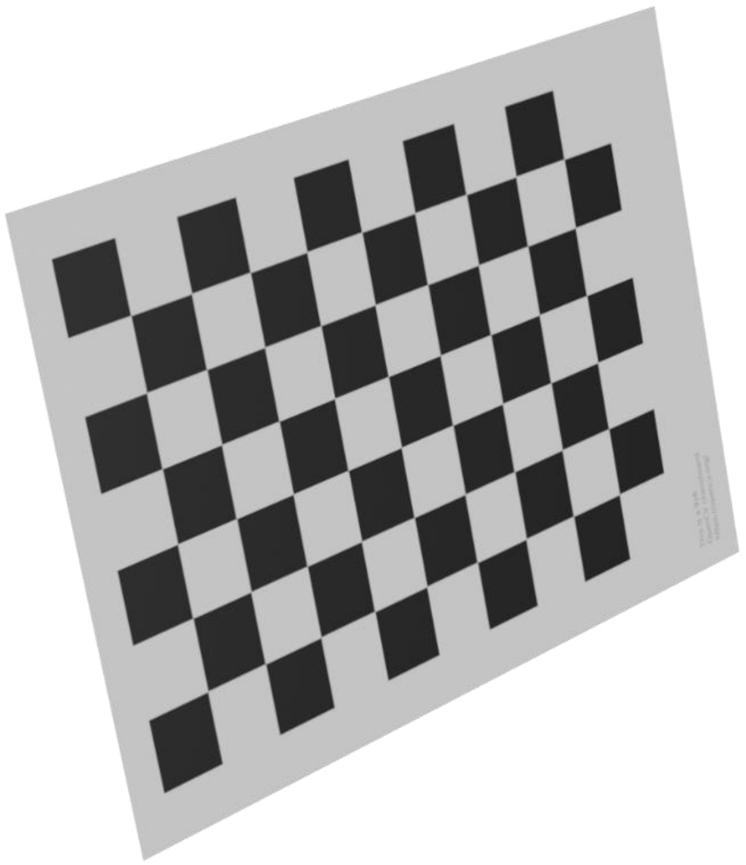


Data representation

- Representation has a big influence on the ability of the network to learn
- Frame of reference aligned to the camera
- Translations: **perspective corrected**
  - $X = \frac{x}{z}\ ;\ Y = \frac{y}{z}\ ;\ z = depth$
- Rotation: **Unit quaternions**
- Normalization between -1 and 1



Synthetic data example

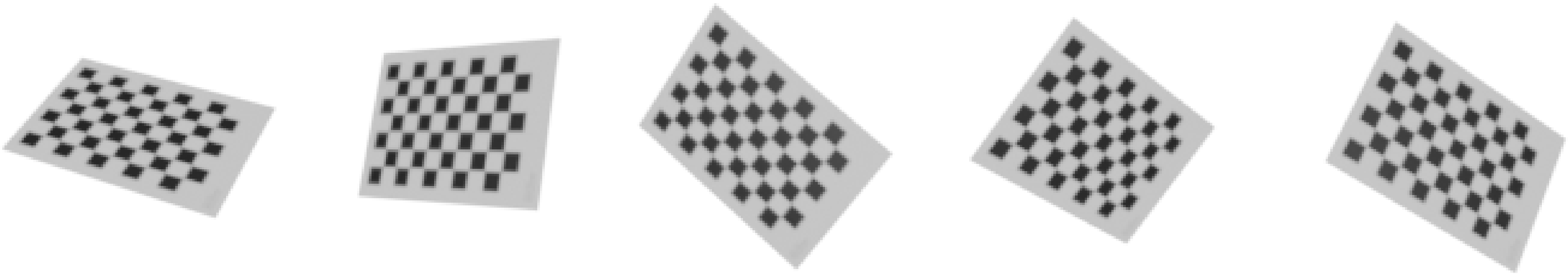


Translations (according to origin):
- -0.03 m
- 0.11 m
- 0.17 m

Euler angles (ZXY):
- 40.39°
- -18.88°
- -22.79°

Unit Quaternion:
- 0.919
- 0.364
- -0.084
- -0.127

More data examples



Estimation neural network

- Training on 403 x 302 images
- 40000 train + 8000 test
- Batch size: 64

- 8 convolutional layers
- 2 dense layers
- 3,859,287 parameters

- 40 epochs
- Adam optimizer with MSE loss
- 1h on Nvidia RTX 2080

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 302, 403, 4)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 302, 403, 4)       148
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 302, 403, 4)       148
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 151, 201, 4)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 151, 201, 4)       148
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 151, 201, 4)       148
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 75, 100, 4)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 75, 100, 8)        296
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 75, 100, 8)        584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 37, 50, 8)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 37, 50, 8)         584
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 37, 50, 8)         584
_________________________________________________________________
flatten (Flatten)            (None, 14800)             0
_________________________________________________________________
dense (Dense)                (None, 256)               3789056
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 1799
=================================================================
Total params: 3,859,287
Trainable params: 3,859,287
Non-trainable params: 0
_________________________________________________________________
```

Meaningful metrics while training

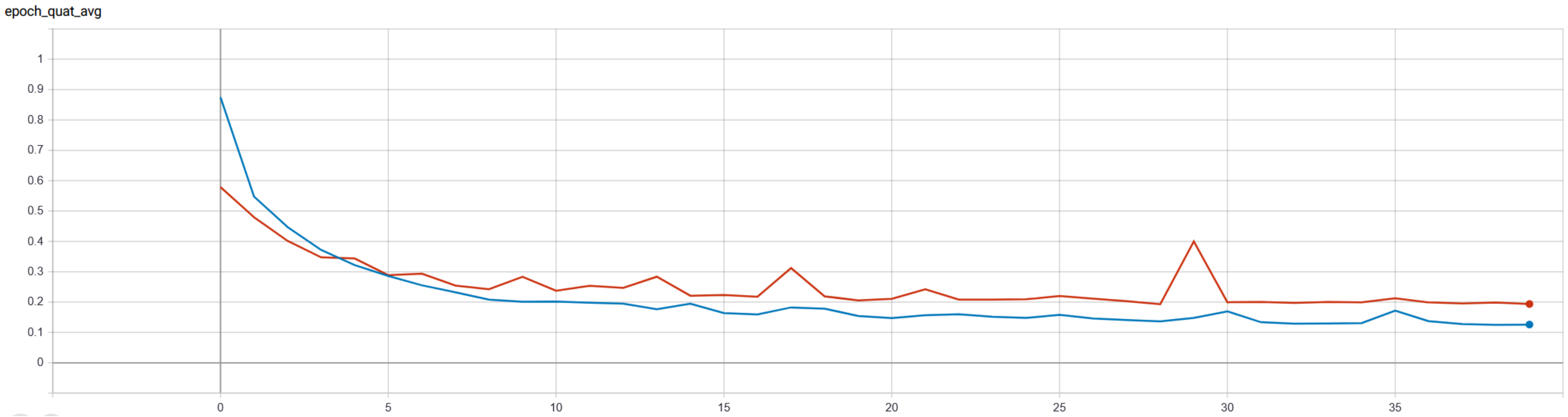
- MSE loss is not meaningful
- Other metrics give a better evaluation of the training accuracy

- Translation:
  - Average and maximum distance between prediction and ground truth (in meters)

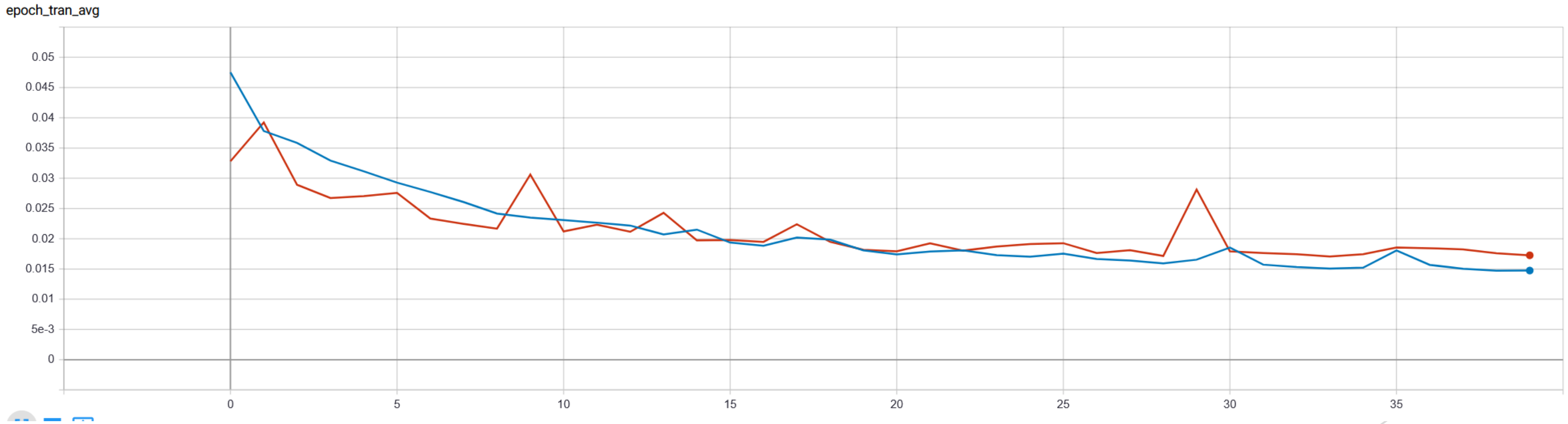
- Rotation:
  - Average and maximum smallest angle between prediction and ground truth (in radians)

Meaningful metrics while training

Average quaternion distance (in radians) : 0.19 rad (10°) after training



Average translation distance (in meters) : 17 mm after training



Refinement

- Rendering of full resolution (4032 x 3024) images in OpenGL
- Image similarity computation in a compute shader
- Optimize the silhouette overlapping with the Dice coefficient
- Optimize the similarity of color in the overlapping area with MSE
- Weighted sum of the 2 measures (90% silhouette + 10% colors)

Implementation details

- Switch to another representation easier to optimize
  - Euler angles (ZXY) in degrees
  - Translation from origin in meters
  - Normalized between -1 and 1

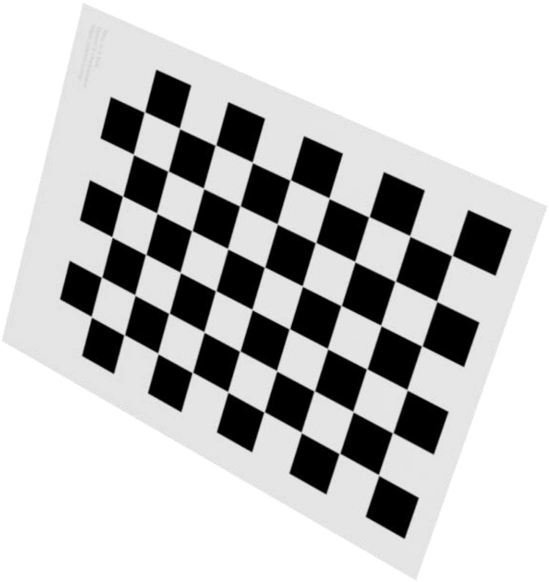
- Unconstrainted optimization:
  - Second order method
  - Approximate derivatives
  - Maximization of the similarity function

Results: Pattern

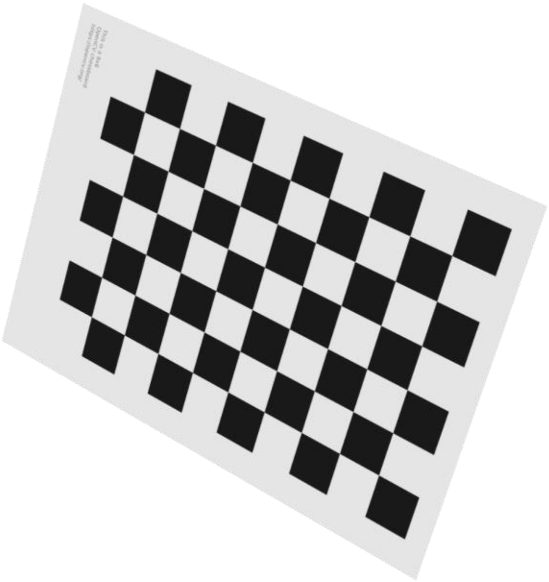
- Number of images: 100
- Optimization time: about 20 minutes

	After CNN	After refinement
Average translation	0.033 m	0.006 m
Average angle	0.270 rad (15.47°)	0.048 rad (2.75°)

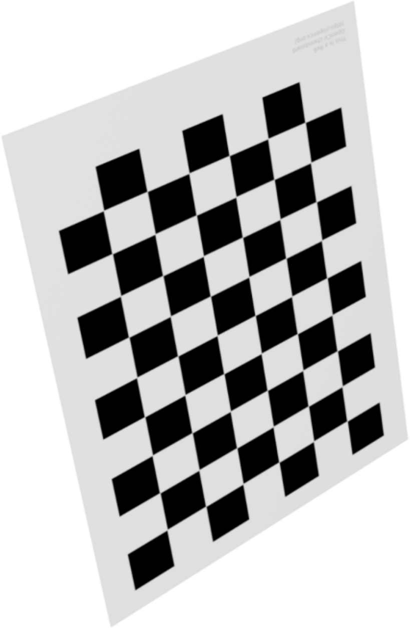
Pattern 1



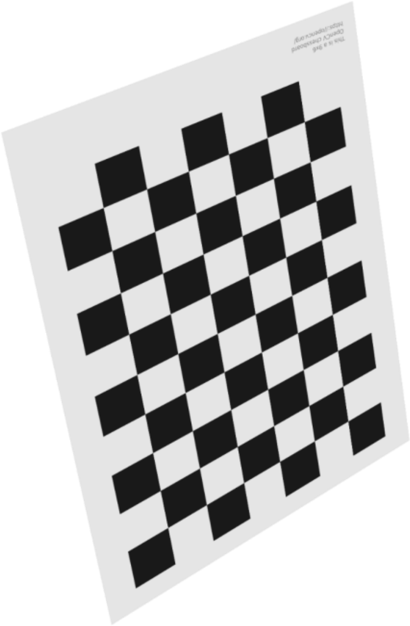
Pattern 1



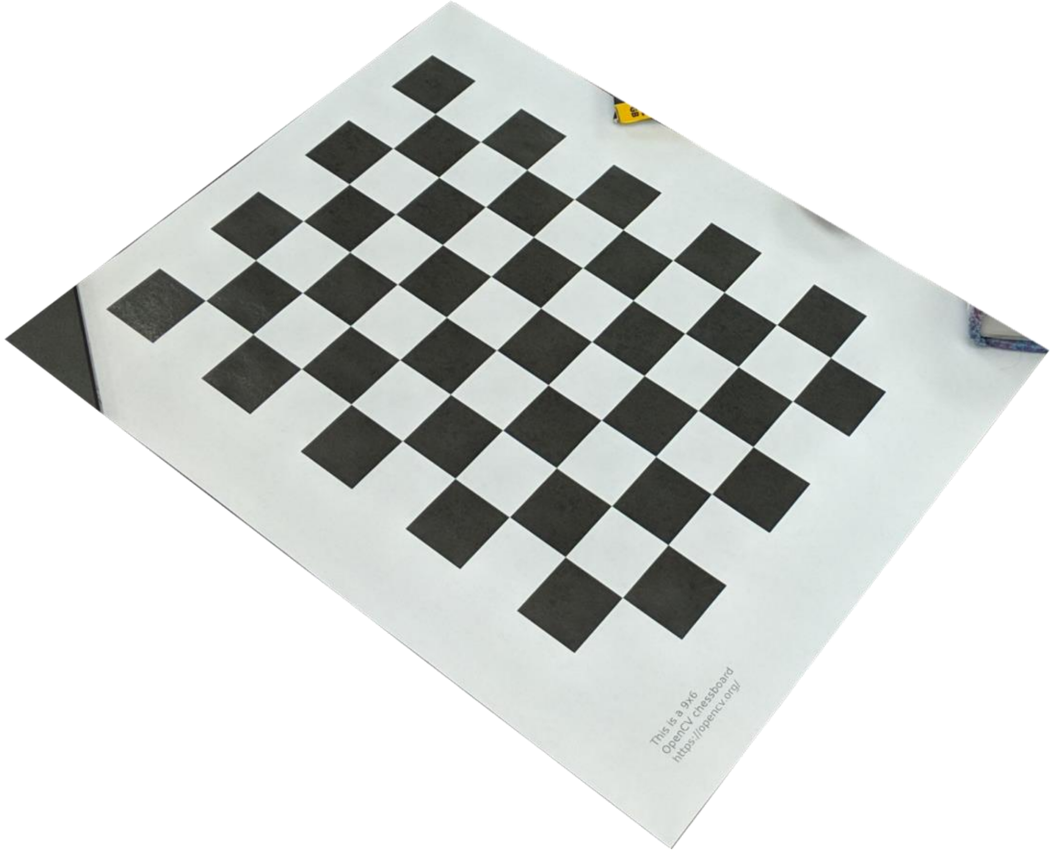
Pattern 2



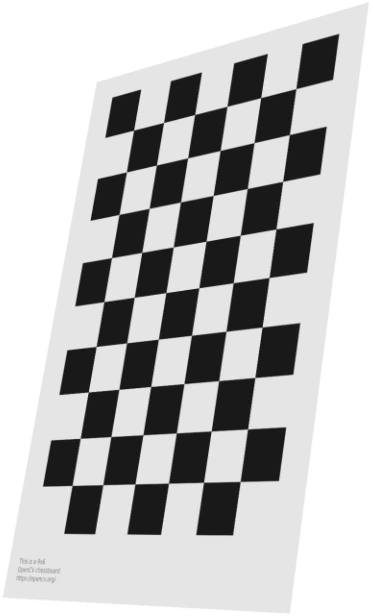
Pattern 2



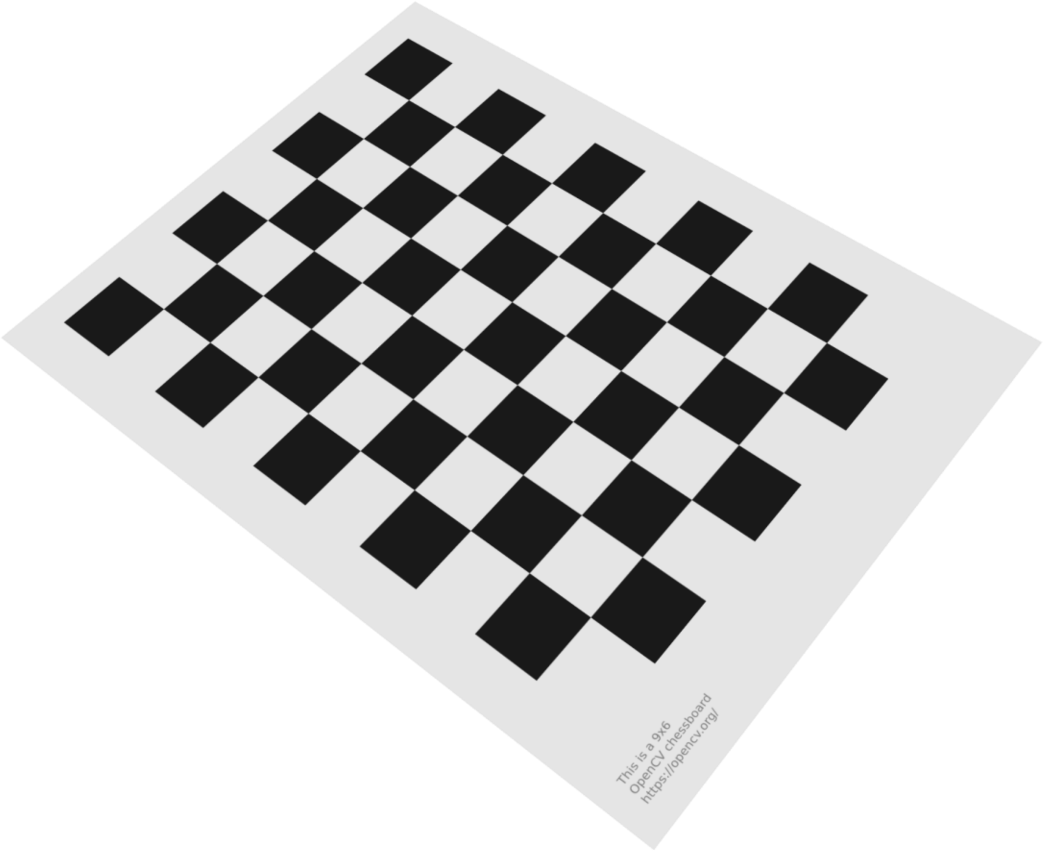
Real pattern



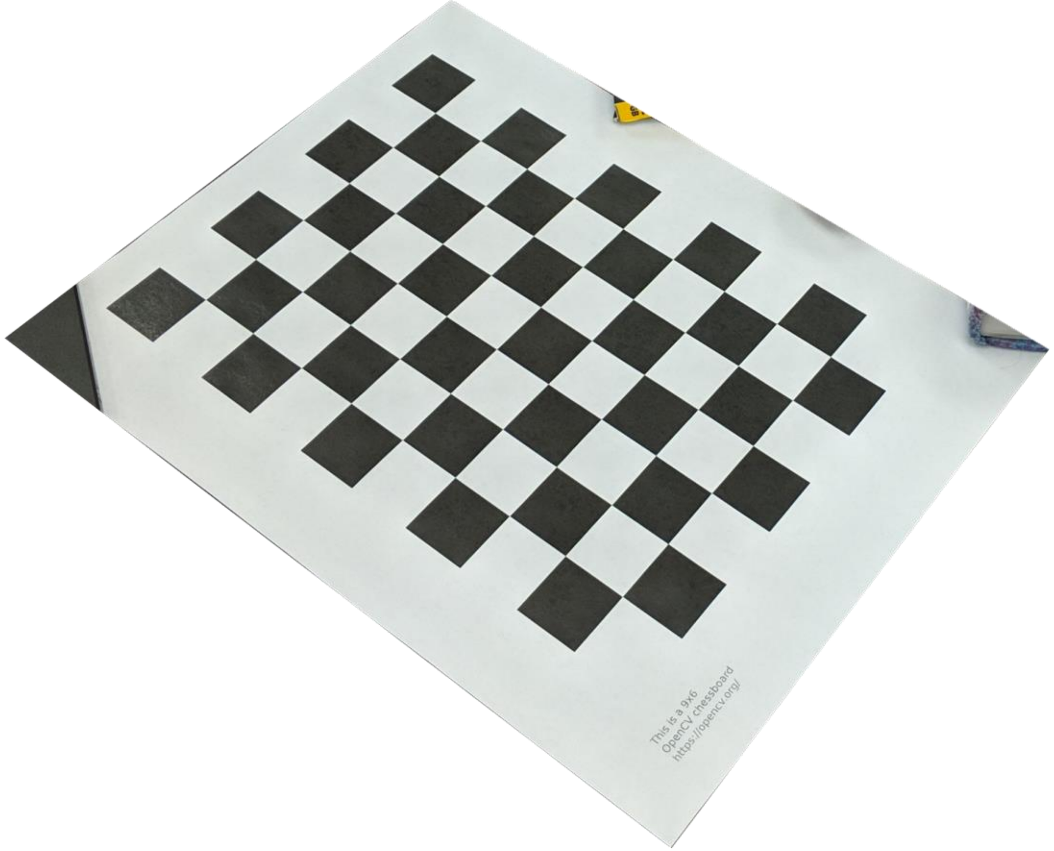
Real pattern: after CNN



Real pattern: after refinement



Real pattern



Results: Cat

- Number of images: 100
- Optimization time: about 20 minutes

	After CNN	After refinement
Average translation	0.037 m	0.015 m
Average angle	0.162 rad (9.28°)	0.085 rad (4.87°)

Image 1 : reference

Image 1: prediction

Image 2: reference

Image 2: prediction

Image 3: reference

Image 3: reference

Results: Phone

- Number of images: 100
- Optimization time: about 20 minutes

	After CNN	After refinement
Average translation	0.033 m	0.002 m
Average angle	0.214 rad (12.26°)	0.043 rad (2.46°)

Image 1: reference

Image 1: prediction

Image 2: reference

Image 2: prediction

Image 3: reference

Image 3: prediction

Discussion on robustness

- In the case, of a non synthetic dataset, the ground-truth pose would be provided but with imprecision. We try to inject noise while training and see how it affects the end result.

- 10% random (normal) change in translation: (± 2 cm)
- 10% random (normal) change in rotation: (± 10°)

	Without noise	With noise	After refinement
Average translation	0.033 m	0.038 m	0.005 m
Average angle	0.270 rad (15.47°)	0.170 rad (9.74°)	0.015 rad (0.86°)

Discussion on transferability

- Try to estimate the pose of the cat with the network trained on phones

Network -> Dataset	Phone -> Phone	Phone -> Cat	Cat -> Cat
Average translation	0.033 m	0.181 m	0.037 m
Average angle	0.214 rad (12.26°)	1.819 rad (104.2°)	0.162 rad (9.28°)

- It doesn’t work at all!

Future work

- Estimation
  - Improve the network architecture
  - Try a loss based on the reprojection of points on the mesh
  - Try to predict multiple objects at the same time
- Try to improve the image similarity measure
  - Based on luminance for the pattern and the phone
  - Based on hue for the cat
- Output the camera matrices for calibration
- Adding the focal parameter for camera calibration

Conclusion

- We can estimate the 6D pose of a known object in a picture
- There are some constraints:
  - Simple and compact shape
  - Textured surface
  - Not transparent Lambertian surface
  - No symmetries
- Relatively precise but not real time

References

- Xiang, Y., Schmidt, T., Narayanan, V., & Fox, D. (2017). **Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes.** *arXiv preprint arXiv:1711.00199*.

- Li, Y., Wang, G., Ji, X., Xiang, Y., & Fox, D. (2018). **Deepim: Deep iterative matching for 6d pose estimation.** In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 683-698).