

# 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)

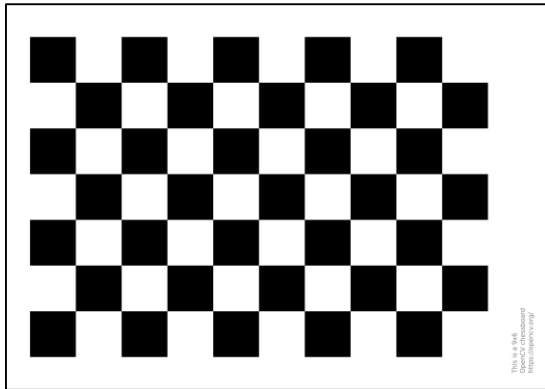
# Problem definition

- ▶ Based on a single image, and with a known 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

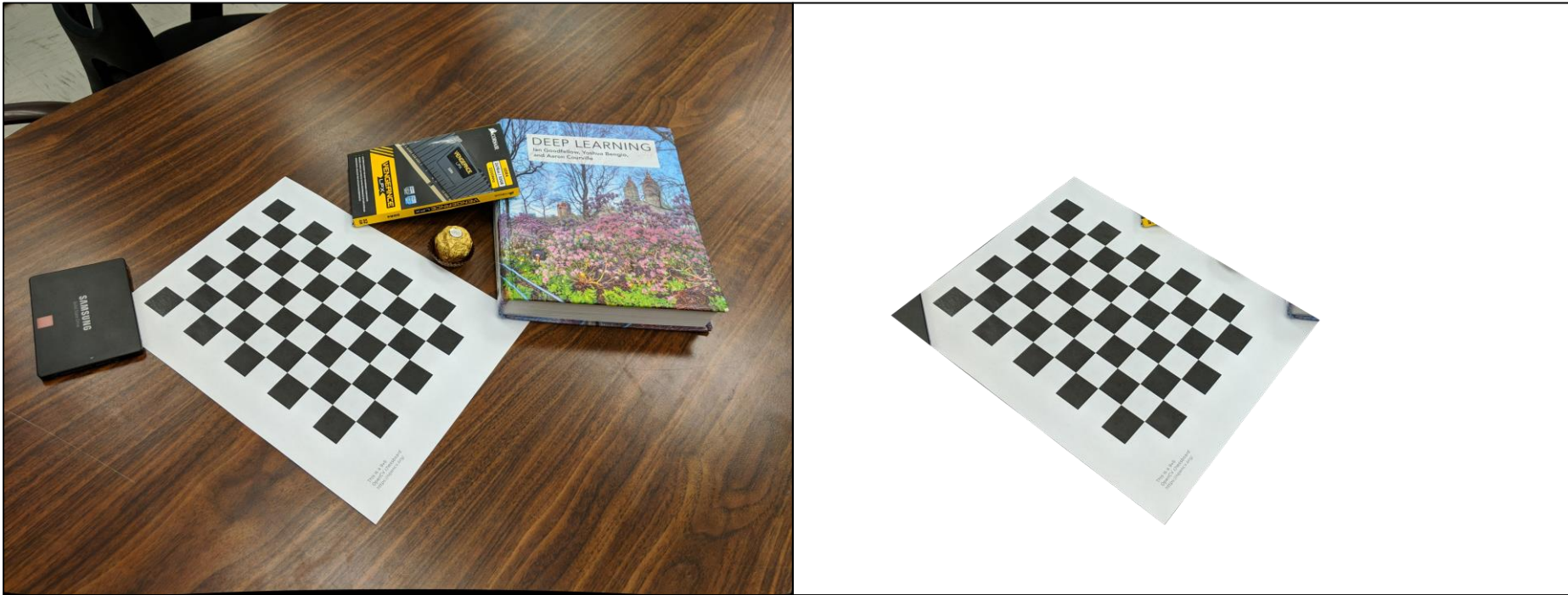
# Our objects



# 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

Blender\* [C:\Users\gaill\Documents\ObjectCameraCalibration\blender\scene.blend]

File Edit Render Window Help Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Scripting +

Object Mode View Select Add Object

User Perspective (0) Collection | Cat

```
1 import bpy
2 import os
3 from random import uniform
4 from io import open
5 import mathutils
6 import math
7
8 def generate_dataset(directory, start, number, include_fullsize_images):
9     cam = bpy.data.objects['Camera']
10    origin = bpy.data.objects['Empty']
11    pattern = bpy.data.objects['Cat']
12
13    for image in range(start, start + number):
14
15        # Generate random angles
16        x = round(uniform(-0.2, 0.2), 2)
17        y = round(uniform(-0.2, 0.2), 2)
18        z = round(uniform(-0.2, 0.2), 2)
19
20        angle_x = round(uniform(-45.0, 45.0), 2)
21        angle_y = round(uniform(-45.0, 45.0), 2)
22        angle_z = round(uniform(-180.0, 180.0), 2)
23
24        # Constraints
25
26        # Euler representation of angles
27        euler_angles = mathutils.Euler((math.radians(angle_x), math.radians(angle_y), math.radians(angle_z)), 'ZXY')
28        quaternion_angles = euler_angles.to_quaternion().normalized()
29
30        # Move pattern
31        pattern.location[0] = x
32        pattern.location[1] = y
33        pattern.location[2] = z
34
35        # Rotate pattern according to Euler angles
36        # pattern.rotation_mode = 'ZXY'
37        # pattern.rotation_euler = euler_angles
38
39        # Rotate pattern according to quaternion
40        pattern.rotation_mode = 'QUATERNION'
41        pattern.rotation_quaternion = quaternion_angles
42
43        # Render the thumbnail image to a file
44        bpy.context.scene.render.resolution_x = 403
45        bpy.context.scene.render.resolution_y = 302
46        bpy.context.scene.render.filepath = os.path.join(directory, "{:05d}.png".format(image))
47        bpy.ops.render.render(write_still=True)
48
49        if include_fullsize_images:
50            bpy.context.scene.render.resolution_x = 4032
51            bpy.context.scene.render.resolution_y = 3024
52            bpy.context.scene.render.filepath = os.path.join(directory, "..\\test_full\\{:05d}.png".format(image))
53            bpy.ops.render.render(write_still=True)
54
```

PYTHON INTERACTIVE CONSOLE 3.7.4 (default, Nov 1 2019, 13:23:36) [MSC v.1916 64 bit (AMD64)]

Builtin Modules: bpy, bpy.data, bpy.ops, bpy.props, bpy.types, bpy.context, bpy.utils, bgl, blf, mathutils  
Convenience Imports: from mathutils import \*; from math import \*  
Convenience Variables: C = bpy.context, D = bpy.data

>>> |

View Console

Scene

Find & Replace

Find Next

Replace

Case Wrap All

Properties

Margin 80

Font Size 15

Tab Width 4

Indentation Spaces

Scene Collection

- Collection
- Cat
- Empty
- Light
- Pattern

Current File

- Brushes 48
- Cameras
- Collections
- Grease Pencil
- Images 3
- Lights
- Line Styles
- Materials 2
- Meshes

Cat

Transform

Location X	0.19 m	
Y	-0.2 m	
Z	0.15 m	
Rotation W	0.642	
X	0.008	
Y	0.045	
Z	0.765	
Mode	Quaternion (WXYZ)	
Scale X	0.006	
Y	0.006	
Z	0.006	

Delta Transform

Relations

Collections

Instancing

Motion Paths

Visibility

Viewport Display

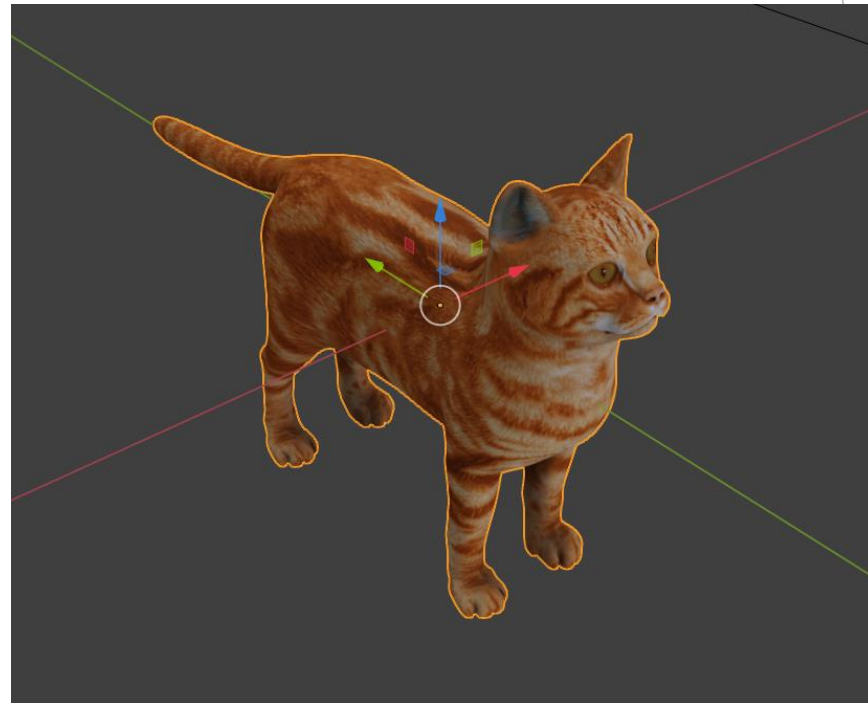
Custom Properties

Collection | Cat | Verts:35,290 | Faces:70,576 | Tris:70,576 | Objects:0/4 | Mem: 72.4 MiB | v2.82.7

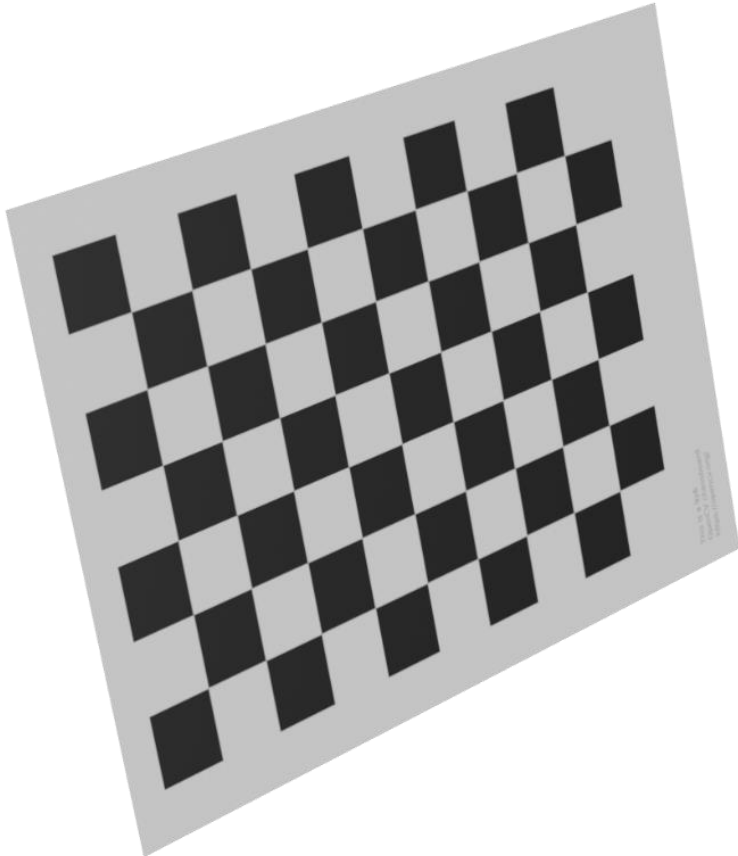


# Data representation

- ▶ 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

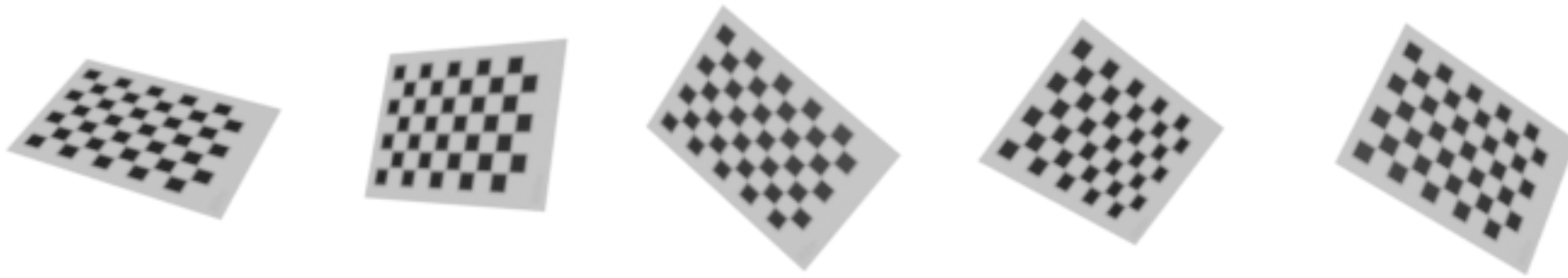
Euler angles (ZXY):

- $40.39^\circ$
- $-18.88^\circ$
- $-22.79^\circ$

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

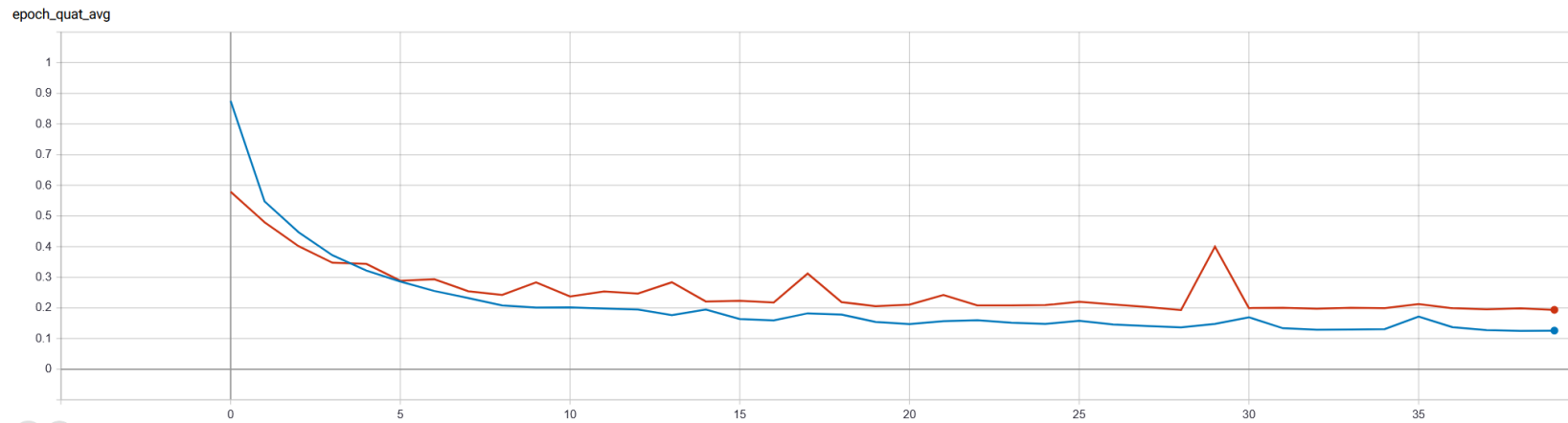
Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 302, 403, 4)]	0
conv2d (Conv2D)	(None, 302, 403, 4)	148
conv2d_1 (Conv2D)	(None, 302, 403, 4)	148
max_pooling2d (MaxPooling2D)	(None, 151, 201, 4)	0
conv2d_2 (Conv2D)	(None, 151, 201, 4)	148
conv2d_3 (Conv2D)	(None, 151, 201, 4)	148
max_pooling2d_1 (MaxPooling2D)	(None, 75, 100, 4)	0
conv2d_4 (Conv2D)	(None, 75, 100, 8)	296
conv2d_5 (Conv2D)	(None, 75, 100, 8)	584
max_pooling2d_2 (MaxPooling2D)	(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

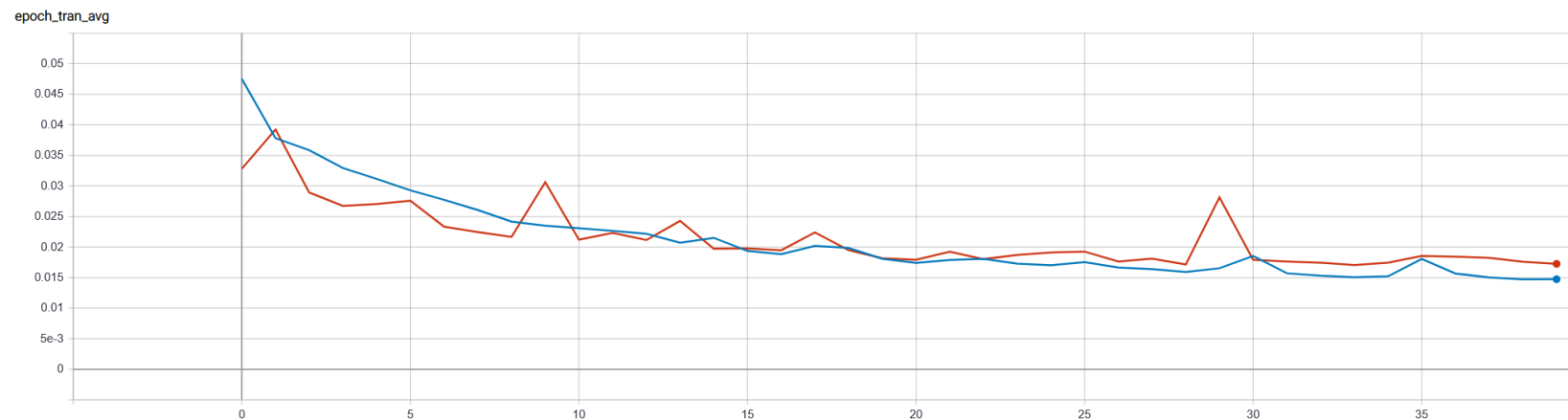
- ▶ MSE loss is not meaningful
- ▶ Other metrics give a better evaluation of the training accuracy
- ▶ Translation:
  - ▶ Average and maximum distance between prediction and ground truth (in meters)
- ▶ Rotation:
  - ▶ Average and maximum smallest angle between prediction and ground truth (in radians)

# Meaningful metrics while training

Average quaternion distance (in radians) : 0.19 rad ( $10^\circ$ ) 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
- ▶ Unconstrained optimization:
  - ▶ Second order method
  - ▶ Approximate derivatives
  - ▶ Maximization of the similarity function

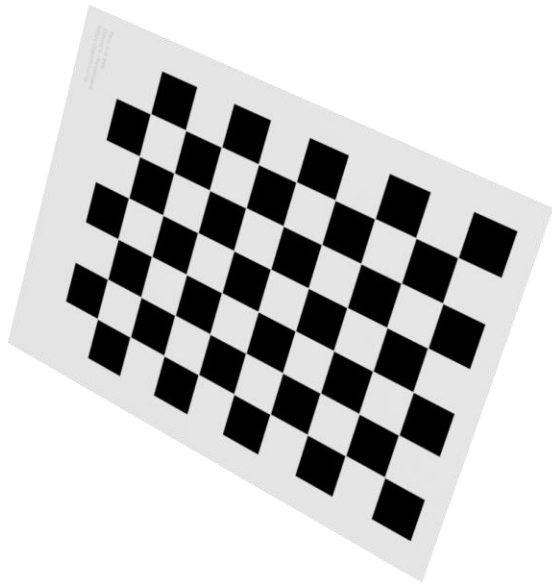


# Results: Pattern

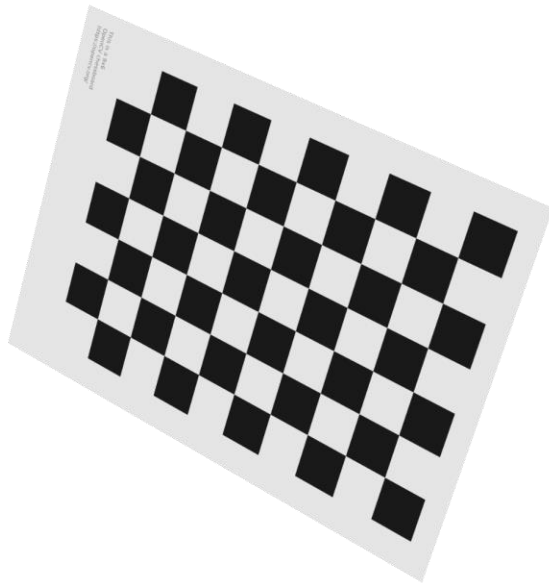
- ▶ Number of images: 100
- ▶ Optimization time: about 20 minutes

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

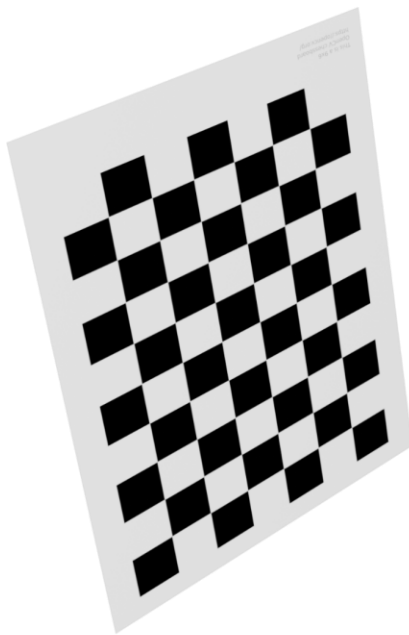
# Pattern 1



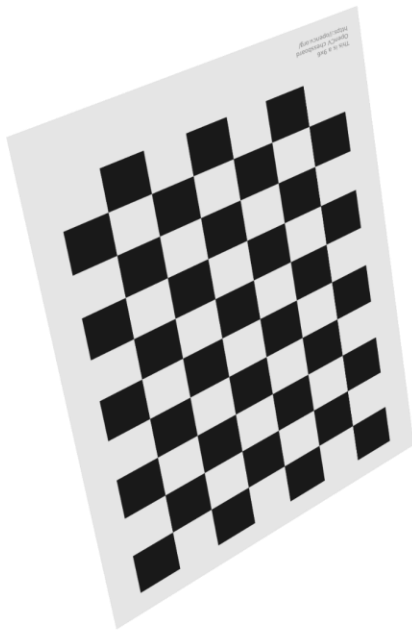
# Pattern 1



# Pattern 2



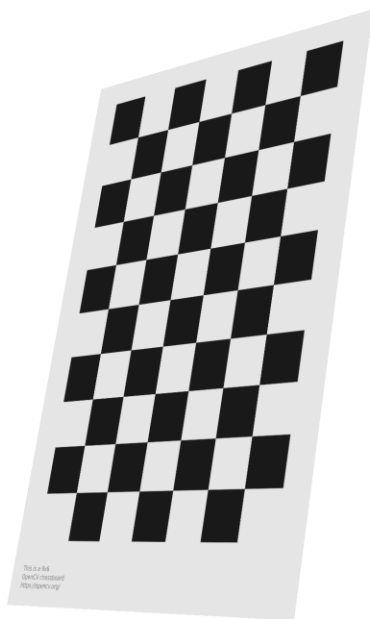
# Pattern 2



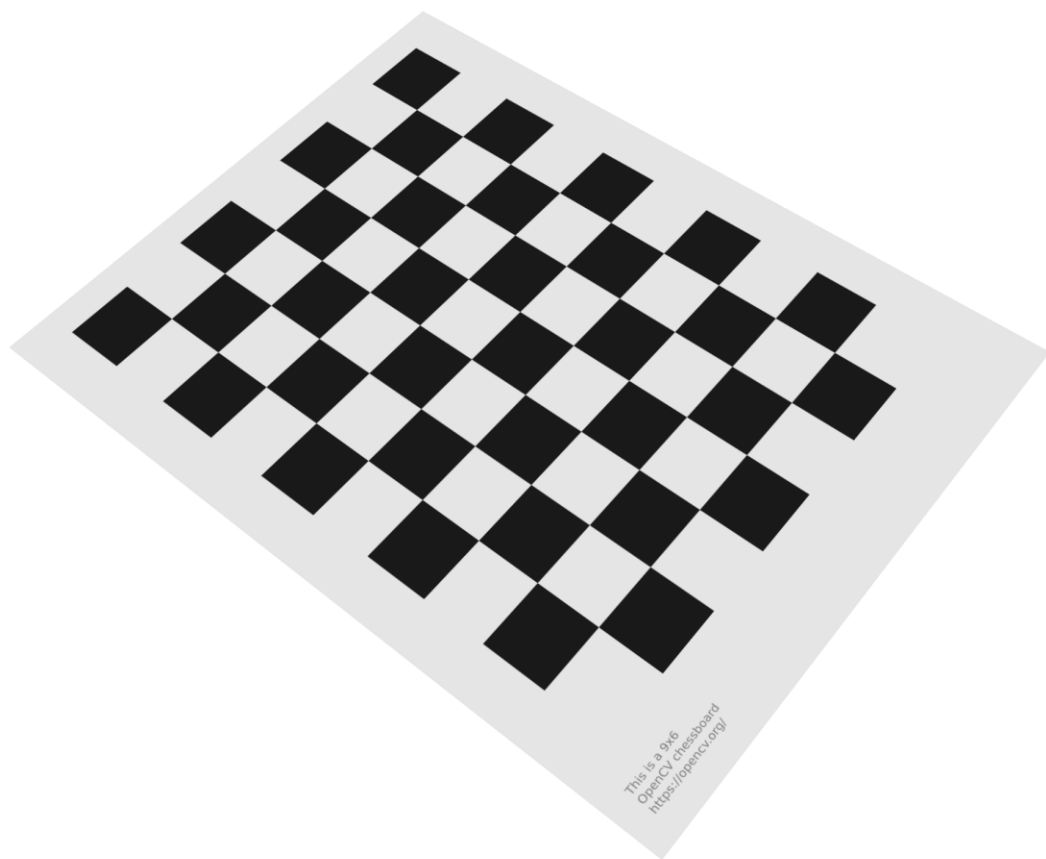
# Real pattern



# 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: ( $\pm 2$  cm)
- ▶ 10% random (normal) change in rotation: ( $\pm 10^\circ$ )

	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).