Camera Calibration through Camera Projection Loss

Talha Hanif Butt – L181864
Supervisor: Dr. Asif Mahmood Gilani (FAST)
Co-Supervisor: Dr. Murtaza Taj (LUMS)

What is Camera Calibration?

Why do we need it?

```
def image_to_camera(self, u, v, disparity):

    xCam = (self.intrinsic.fx * self.extrinsic.baseline) / disparity
    yCam = - (xCam / self.intrinsic.fx) * (u - self.intrinsic.u0)
    zCam = (xCam / self.intrinsic.fy) * (self.intrinsic.v0 - v)

    return [xCam, yCam, zCam]


def image_to_world(self, u, v, disparity):

    [xCam, yCam, zCam] = self.image_to_camera(u, v, disparity)

    yWorld = yCam + self.extrinsic.y
    xWorld = xCam * math.cos(self.extrinsic.pitch) + zCam * \
      math.sin(self.extrinsic.pitch) + self.extrinsic.x
    zWorld = - xCam * math.sin(self.extrinsic.pitch) + zCam * \
      math.cos(self.extrinsic.pitch) + self.extrinsic.z

    return [xWorld, yWorld, zWorld]
```

Can mathematical equations help CNNs?

Paper	Input	Parameters
[24]	RGB Image	Tilt, Roll, Focal length, Radial distortion
[32]	RGB Image, Projected Radar Data	Tilt, Pan, Roll
[16]	RGB Image, Raw LiDAR point cloud	Rotation, Translation
[29]	Stereo Image pair	Fundamental Matrix
[5, 8, 12, 25]	RGB Image pair	Rotation, Translation
[2]	RGB Image	Focal length, Distortion
[1, 17, 19, 26, 40]	RGB Image	Rotation, Translation
[15]	RGB Image	Tilt, Roll, Focal length
[38]	RGB Image	Focal length
[33]	Head Detections, Focal length	Rotation, Translation
[21]	RGB Image	Focal length, Position, Orientation
[30]	Putative matches	Fundamental Matrix

Table 1: Overview of some recent configurations for different aspects of Camera Calibration

How to embed an equation in a CNN?

```python
def add_layer(tensor):
    return tensor[0] + tensor[1]

def mul_layer(tensor):
    return tensor[0] * tensor[1]

def div_layer(tensor):
    return tensor[0] / tensor[1]

def sub_layer(tensor):
    return tensor[0] - tensor[1]
```

Equation in Python

```python
xCam = (self.intrinsic.fx * self.extrinsic.baseline) / disparity
```

Equivalent Lambda layer representation

```python
mul_1 = Lambda(mul_layer)([pred_fx, pred_baseline])
xCam = Lambda(div_layer, name='xCam')([mul_1, pred_disparity])
```

Where to get data for training?

CARLA using Town 2 for training while Town 1 for testing having 24 episodes each.

Reason: 48 Camera Configurations without spending a penny on actual equipment.

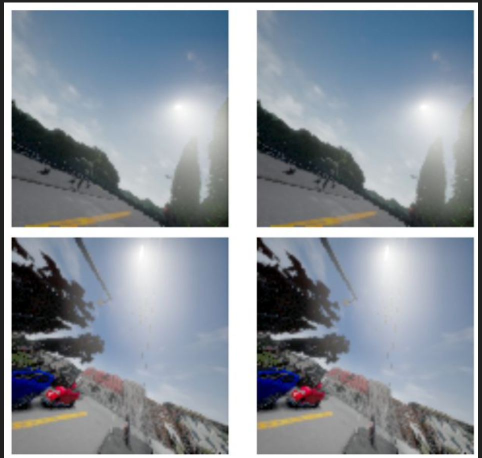

Table 1 Table showing MAE in predicted parameters on synthetic test set comprising of 23,796 images.

	f_x	f_y	u_0	v_0	b	d	t_x	t_y	t_z	θ_p
Average [1]	72.44	72.44	40.27	40.27	12.53	21.34	12.53	12.90	12.73	89.68
Deep-Homo [7]	28.51	28.52	**1.01**	**1.02**	1.51	**0.17**	1.51	1.32	1.23	22.48
MTL-Baseline (Ours)	20.90	23.98	14.63	13.95	1.06	1.35	0.89	1.01	1.01	20.02
MTL-CPL-U (Ours)	38.36	58.19	46.02	46.11	2.79	11.87	2.80	1.11	1.44	107.89
MTL-CPL-A (Ours)	4.79	**4.22**	4.12	3.97	0.65	0.25	2.42	0.62	2.42	5.69
MTL-CPL-U-TL (Ours)	**2.50**	382.20	35.70	3.91	**0.47**	20.89	0.18	**0.39**	0.19	9.75
MTL-CPL-A-TL (Ours)	21.92	128.92	185.29	31.95	0.65	2.14	**0.10**	1.96	**0.17**	**2.53**

Will the proposed approach work on real data without training?

Table 2 Table showing MAE in predicted parameters on Tsinghua-Daimler test set comprising of 2,914 images. For this experiment, we just did a forward pass without any transfer learning or training.

	f_x	f_y	u_0	v_0	b	d	t_x	t_y	t_z	θ_p
Deep-Homo [7]	2206.58	2205.52	986.60	474.45	2.39	**6.43**	**0.60**	3.35	0.81	64.66
MTL-Baseline (Ours)	1831.53	1803.43	**759.84**	**436.34**	19.48	35.79	12.34	16.27	14.77	498.59
MTL-CPL-U (Ours)	**1355.94**	**1790.74**	3680.99	3919.11	58.00	1223.16	15.54	**2.22**	**0.25**	3861.55
MTL-CPL-A (Ours)	2208.87	2206.74	987.81	475.70	3.01	6.44	3.07	3.14	0.97	51.65
MTL-CPL-U-TL (Ours)	2166.18	4160.66	896.35	470.40	**2.22**	27.04	2.12	3.45	1.08	30.88
MTL-CPL-A-TL (Ours)	3341.94	2215.48	985.91	474.32	2.74	27.81	1.26	4.58	2.13	**29.04**

Conclusion

Questions