Stereo Calibration for a Formula Student Driverless Race Car

Charles Vorbach
cvorbach@mit.edu
MIT
Cambridge, Boston

Max Tell
maxtell@mit.edu
MIT
Cambridge, Boston

Abstract

We present a stereo extrinsics monitoring and online calibration system designed for the MIT/Delft University of Technology Formula Student Driverless electric race car. Many autonomous vehicles rely on stereo cameras. Unfortunately, the extrinsic calibration of such cameras degrades over time - especially in high acceleration applications. Our project developed a method for monitoring the accuracy of extrinsic calibrations using stereo matching density and visual odometry. We also created pipeline for computing new calibrations online and present a neural network approach for learned keypoint descriptors, which improves the keypoints extracted by SIFT.

1. Introduction

Stereo vision is a powerful technique in computer vision with many and increasingly important applications. Stereo vision can both determine the 3D structure of a scene via stereopsis and provide rich color and texture information. Compared to its main competitor LIDAR, stereo vision is much lower-cost, functions at longer ranges, and is less sensitive to environmental conditions such as snow and rain. LIDAR also can only provide return intensities, which lack much of the rich detail provided by stereo vision. The rise of machine learning techniques, which frequently depend heavily on texture, has made stereo vision even more valuable.

Recent successful applications of stereo vision include Boston Dynamics's SpotMini robot and Tesla's self-driving vehicles. Both of these have had great success using stereo vision without complementary LIDAR sensors.

However, a practical obstacle to widespread deployment is stereopsis's great sensitivity to the calibration of its stereo camera pair. Precise calibrations techniques are required to obtain 10^{-2} to 10^{-3} degrees of accuracy in the camera's orientation [2]. Offline techniques to perform this calibration are well-known and widely used [14]. Unfortunately, camera mountings can not be made completely rigid and allow the stereo cameras' poses to slightly drift over time. This causes the cameras' extrinsic calibration to become increasingly inaccurate. Periodic recalibrations are required to maintain accuracy.

Figure 1: MIT/DUT 2018 at Formula Student Germany

The MIT/DUT 2018 driverless race car suffered greatly from drifting calibrations. The autonomous vehicle employs a 10 cm baseline stereo camera using two Basler daA1600 sensors. Stereo image pairs from these cameras are fed into a custom convolutional neural net based on YoloV3 architecture [9] for cone identification and into a commercial FPGA for disparity matching.

Unfortunately, due to the high accelerations experienced by the vehicle and the insufficient rigidity of its camera mounts, we found the stereo camera calibrations degraded rapidly. No more than a few hours of racing could be performed before matching performance was significantly affected and recalibration was necessary. Even worse, it was difficult to determine the accuracy of a given calibration, endangering test data and requiring the team to perform unnecessary calibrations. This was a significant waste of testing and engineer time.

This has pushed us to develop a reliable stereo extrinsic health check and online calibration system for the 2019 vehicle. We developed both utilities for monitoring the accuracy of a stereo calibration and created a multi-step system for online calibration.

1.1. Related Work

Our project has benefited from the significant academic interest in stereo vision online calibration. The work of Dang *et al.* [2] on continuous refinement using reduced order bundle adjustment with the epipolar and trilinear constraints was particularly illuminating for our project. We also contacted the authors of Rehder *et al.* [10] about their impressive incremental bundle adjustment.

While valuable, we ultimately decided that these works were overly general for our application and decided to produce a somewhat simpler calibration scheme.

The design and testing of our system relied on the 2015 KITTI data set [3]. Working with this data set was easier than working directly with rosbag recordings from the MIT/DUT 2018 vehicle because it includes both raw and rectified images as well as measured calibrations. We used the PyKITTI library to integrate KITTI data with OpenCV and Python.

The learned keypoint descriptor component of our work draws heavily from Balntas *et al.* [13]. They demonstrated that a compact CNN architure could outperform traditional keypoint descriptors such as SIFT in robustness [6]. Their work shows an improved accuracy in keypoint matching as a result of this greater robustness. They leverage a triplet loss formulation to embed patches into the desired descriptor space. With a relatively compact architecture and minimial triplet mining, they demonstrate superior accuracy in classifying patch pairs relative to SIFT and several other machine learning-based approaches.

Learned descriptors have also been attempted in several other instances. Guo *et al.* [8] utilizes the perceptive fields of Fully Convolutional Networks (FCNs) to extract the relevant feature maps at various depths within the CNN architecture. These low and middle-level feature patches are combined and pairwise similarities computed in order to establish matching keypoint pairs. The approach demonstrates $\sim 1\%$ improvement relative to SIFT. The idea of learned descriptors was also influenced by the work of Di Febbo *et al.* [7]. They demonstrated the efficiency of compact CNN architectures for computing patch descriptors.

This work is promising for high-speed applications such as driverless cars as any latency in the autonomous pipeline caused by online calibration reduces the look head of the entire vehicle. Their approach relies on formulating the problem of detecting keypoints as a regression problem given the input images and keypoint locations extracted by traditional handcrafted approaches such as SIFT. Although this inherently limits the model's ability to correctly detect keypoints to that of SIFT, the authors demonstrate that their model can achieve this accuracy with an impressive accompanying speed increase by leveraging the speed of matrix computations on GPU architectures. Their method is 5x faster than traditional keypoint extraction methods such as SIFT.

Given that the MIT/DUT 2019 vehicle includes an powerful NVIDIA T4 GPU this approach has the potential to provide a significant speed-up.

1.2. Approach

We approached this project iteratively, building up from a basic project to an increasingly complex and fully-featured system.

For the purposes of this project, each stereo camera is modeled as a pinhole camera mapping 3D world coordinates to 2D image coordinates as follows.

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \boldsymbol{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

We decompose the projection matrix into an extrinsic and intrinsic calibration.

$$\boldsymbol{P} = \boldsymbol{K}[\boldsymbol{R}, -\boldsymbol{R}\boldsymbol{T}]$$

$\boldsymbol{K}$ is the intrinsic calibration. Rotation matrix $\boldsymbol{R}$ and translation vector $\boldsymbol{T}$ from one stereo camera to the other represents the extrinsic calibration.

$$\boldsymbol{K} = \begin{bmatrix} f_x & \alpha & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Since the camera is not ideal, we have to also correct for nonlinear distortion before using this camera model. We consider radial and tangential distortion as follows.

$$x_{\text{corrected}} = x(1 + k_1 r^2 + k_2 r^r + k_3 r^6)$$
$$y_{\text{corrected}} = y(1 + k_1 r^2 + k_2 r^r + k_3 r^6)$$
$$x_{\text{corrected}} = x + [2p_1 xy + p_2(r^2 + 2x^2)]$$
$$y_{\text{corrected}} = y + [p_1(r^2 + 2y^2) + 2p_2 xy]$$

The coefficients of lens distortion and intrinsic calibration change very slowly if at all, so offline calibrations can be maintained indefinitely. Our system limits itself to monitoring and determining the extrinsic calibration $\boldsymbol{R}$ and $\boldsymbol{T}$.

1.2.1 Calibration Monitoring

Our first step was to develop utilities for checking the accuracy of stereo calibrations. Similarly to Rehder *et al.* [10], we used the epipolar constraint between image pairs. By finding the density of corresponding points along epipolar lines we could get an estimate of the accuracy of the fundamental matrix and therefore of the calibration. We implemented this using a block matching algorithm [5].

Unfortunately, this method is not sensitive to translations along the base length axis because these translations preserve epipolar lines and only change the scale factor in the projection matrix.

To detect inaccuracies along the base length calibration, we also implemented the second method described in Rehder *et al.* [10] by comparing visual odometry to odometry from the KITTI test vehicle's OXTS RT3003 GPS. By tracking the root-mean-squared error between the visual odometry and GPS odometry, we can evaluate the scale-sensitive calibration.

Using the known calibration and a sequence of stereo images, I_l^t, I_r^t, I_l^{t+1}, I_r^{t+1}, ..., we use the following method to find each $\boldsymbol{R}_{frame}^t$ and $\boldsymbol{T}_{frame}^t$.

1. Rectify each image using the calibration.

2. Use block matching to find disparities D^t between I_l^t, I_r^t and D^{t+1} between I_l^{t+1}, I_r^{t+1}.

3. Use SIFT to find matching points between I_l^t, I_l^{t+1} and between I_r^t, I_r^{t+1}.

4. Compute world coordinates W^t, W^{t+1} using D^t, and D^{t+1} and the calibration.

5. Throw out points which were not matched and perform heuristic inlier detection.

6. Perform Levenberg-Marquardt optimization to find $\boldsymbol{R}_{frame}^t$ and $\boldsymbol{T}_{frame}^t$ from remaining points.

1.2.2 Online Calibration

Now, with a system which can reliably detect inaccurate calibrations, we developed the following online optimization scheme to solve for extrinsic calibrations from a sequence of images I_l^t, I_r^t, I_l^{t-1}, I_r^{t-1}..., I_l^{t-N}, I_r^{t-N}.

1. Rectify each image using the current estimated calibration $\hat{\boldsymbol{R}}$, $\hat{\boldsymbol{T}}$.

2. Use SIFT to find feature descriptors.

3. Find corresponding points using approximate nearest neighbors algorithm from the FLANN library. Throw out points which fail ratio test [6].

4. Use vehicle odometry, to transform each image pair into the reference frame of the most recent pair I_l^t, I_r^t.

5. Solve for new extrinsic calibration minimizing epipolar projection error with the method from Hartley [4].

6. Repeat above using the new calibration.

A disadvantage to this approach is that it is scale insensitive because it relies solely on the epipolar constraint. Incorporating additional information to solve for the scale using via vehicle odometry or LIDAR would be a good future addition to this project.

This method is relatively computationally intensive so we also developed an interesting replacement for SIFT in our calibration pipeline.

1.2.3 PatchNet Keypoint Extraction and Descriptors

Initially, SIFT was used as the algorithm for keypoint extraction and description. While we were very happy with its success, we decided to build incrementally upon it by introducing learned descriptors to replace the handcrafted descriptors computed by SIFT. To accomplish this, we trained a compact CNN architecture on a curated dataset of image patches.

Dataset

The dataset used for this portion of the pipeline was the HPatches dataset [12]. This dataset contains a large number of curated image patches that draw from a 116 diverse scenes. The curators of this dataset introduced a variety of photometric and geometric distortions to these patches. These distortions provide an excellent training set for the potential visual distortions that might be experienced by stereo cameras in the driverless application. Secondly, as these patches are grouped with the original image and subsequent patches in increasing order of distortion, this provides us an opportunity to explore various triplet mining approaches to improve the model's performance. These attempts will be touched on in the triplet mining section.

Architecture

The architecture for this network is presented in Table 1 and is constituted of 5 layers. The first four layers are composed of several blocks of convolutional filters, followed by max-pooling and sigmoid non-linearities. This section of the network computes a feature map over the input image patch. Finally, these feature maps are then fed into a dense layer that computes the final descriptor vector of length 128.

Name	Output Dim.	Kernel Size	Stride
conv1-1	32	7 x 7	1
pool1	32	2 x 2	2
conv2-1	64	6 x 6	1
pool2	64	2 x 2	2
dense	128	N.A	N.A

Table 1: PatchNet Architecture

Optimization

The final model was trained using Stochastic Gradient Descent on batches of 256 triplets using a learning rate of 0.1, which decays at a rate of 10^{-6}. The learning rate is also annealed every 10 epochs to 0.8 of its previous value. Experiments were also performed with the Adam optimizer, but these tended to result in poor local optima during training. Training results over a range of epochs are shown in Figure 5.

Loss

The following standard triplet formulation of margin ranking loss was used to enforce the embedding of patches into the descriptor space.

$$Loss = \sum_{i=1}^{N}(\|f_i^a - f_i^p\|_2^2 - \|f_i^a - f_i^n\|_2^2 + \alpha)$$

Several experiments were conducted with alternatives, such as Lossless Triplet Loss. This formulation presents promise to improve triplet learning as it forces triplets even within the margin to still contribute to the loss. However, these resulted in far less stable training than the original formulation. Additional experimentation was pursued with the selection of the margin. Work done by Hermans *et al.* [1] indicated that larger margins might result in improved accuracy on validation datasets; however, the selection of a relatively small margin of 1 proved to be the most effective. As in Balntas *et al.* [13], positive and anchor swapping were used in the loss computation to provided some hard-negative mining within triplets at no additional computational expense.

Triplet Mining

One of the major challenges of triplet-based metric learning is the proliferation of "easy triplets", defined as $d(f_i^a, f_i^p) + \alpha < d(f_i^a, f_i^n)$ where d is the distance function used in triplet loss. Consider the above formulation of triplet loss, clearly, these pairs will contribute 0 to the loss computation once the condition is satisfied. After several epochs, the vast majority of triplets used will fall into this category. Thus, this causes only a few "hard triplets", defined as $d(f_i^a, f_i^n) < d(f_i^a, f_i^p)$ and "semi-hard triplets", $d(f_i^a, f_i^p) < d(f_i^a, f_i^n) < d(f_i^a, f_i^p) + \alpha$, to contribute to training loss after several epochs.

To remedy this, several triplet mining approaches have been posed. These techniques rely on mining "hard triplets" and limit training to these, allowing the network to train on only triplets that violating the margin. As mentioned in the previous subsection, we utilize in-triplet hard negative mining. To further improve training, we experimented with hard positive mining by leveraging the structure of the the training dataset. Since positive pairs must be drawn from a specific image patch and some distorted version of it, we selected pairs consisting of a reference patch and a "tough" version of it as the positive pair. This produces a positive pair that are visually disparate, forcing the network to improve its embedding of such positive pairs in the metric space.

This produced a 1-2% improvement in validation accuracy of the final model. Both of these techniques effectively allow the mining more challenging triplets. In future work, triplet mining could be expanded to additional online methods; however, within the time constraints of this work such attempts proved to impossible to implement.

1.3. Experimental Results

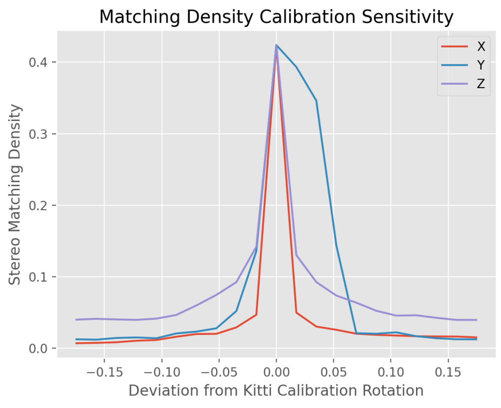


Figure 2

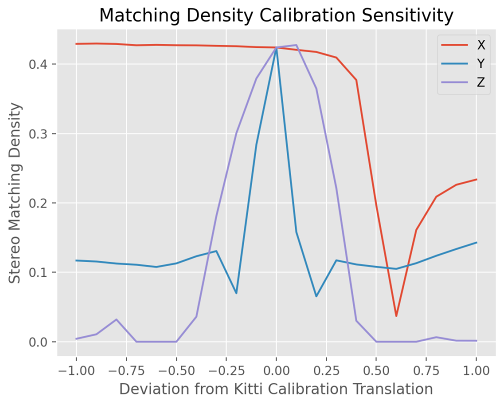


Figure 3

We found stereo density to be quite sensitive to extrinsic rotation calibration and moderately sensitive to translation calibration as can be seen in figures 2 and 3. These figures show the average stereo matching density from a sample of 1000 KITTI images against the proportional deviation

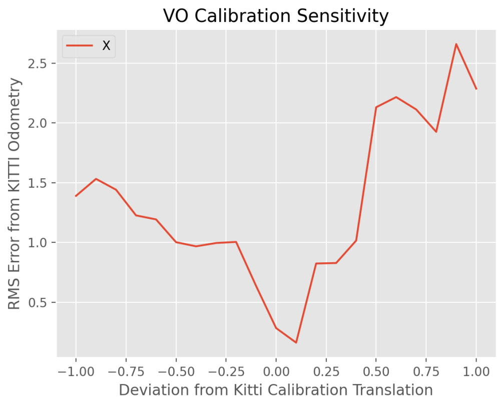


Figure 4

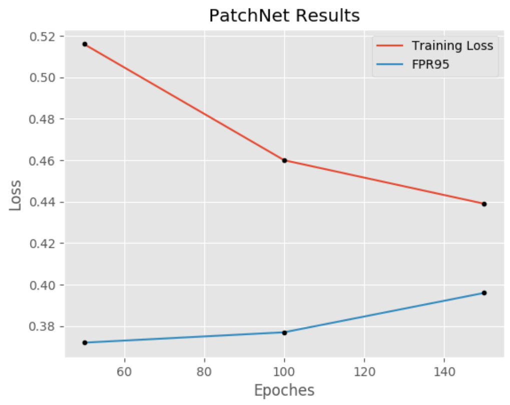


Figure 5: Training Results

in calibration parameters. We observed that stereo density was quite sensitive to the calibrated rotation and moderately sensitive to the calibrated translation. Translation along the baseline (X in KITTI coordinates) is an exception which is to be expected given that this transformation preserves epipolar lines.

Visual odometry was also a fairly effective metric as can be seen in figure 4. The error however, was rather high even for calibrated camera. A sample of 100 sequences of 10 KITTI images had an average root-mean-squared proportional deviation 0.27 with a standard deviation of 0.1. We think it is possible this is due to inaccuracies in the GPS odometry or with our method for performing visual odometry.

We can also use these two utilities to measure the accuracy of our online calibration method.

	KITTI	SIFT	PatchNet
Stereo Density	0.47	0.35	0.35
VO RMS Error	0.27	0.51	0.52

In Figure 5, the results of training the final version of PatchNet over a range of epochs are shown. The graph includes training loss, as well as false positive rate at 95% recall(FPR95), which was used as our evaluation metric. Although training loss decreases steadily, FPR95 seems to plateau after the initial epochs suggesting overfitting of the model.

Despite promising results in stereo density and visual odometry, we found that the PatchNet model demonstrated subpar performance when benchmarked against SIFT on Hpatches data. The chart 6 below includes the results for a standardized matching task described in [12]. The results are shown in terms of mean average precision, mAP.

We suspect that this under-performance is a result of the aforementioned overfitting caused by the large size of training samples. Further, this was likely exacerbated by the

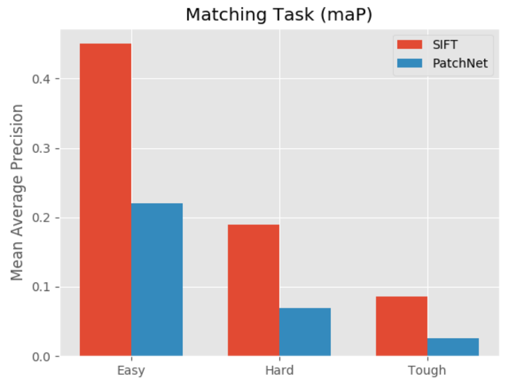


Figure 6: Matching Task Benchmark

difficulty of patch matching on the Hpatches dataset. In future work, additional hard triplet mining and adjustments to the architecture should bring PatchNet's performance in line with the results demonstrated in Balntas *et al.* [13] on the Phototour dataset [11].

The speed of the PatchNet model was also benchmarked against SIFT on standard keypoint extraction as shown in Table 2. Although SIFT appears to outperform the current version of Patchet, the benchmarking was performed on a NVIDIA K80 GPU provided by AWS. If this were to be run on MIT/DUT 2019's NVIDIA T4 GPU, we expect that the Patchnet implementation would be several times faster.

SIFT	PatchNet
0.274	0.778

Table 2: Keypoint Extraction Benchmark(in seconds)

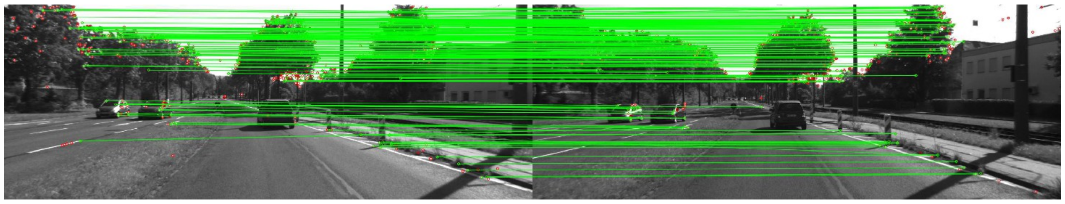


Figure 7: SIFT matching points

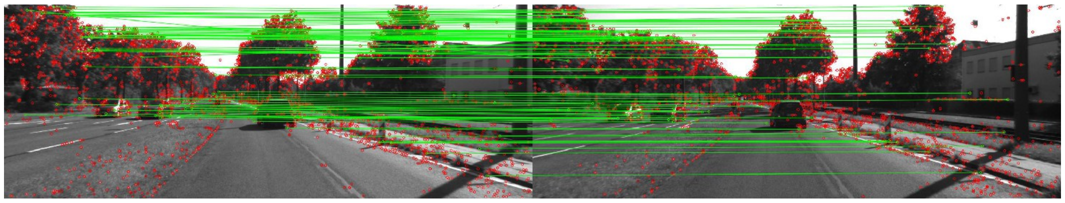
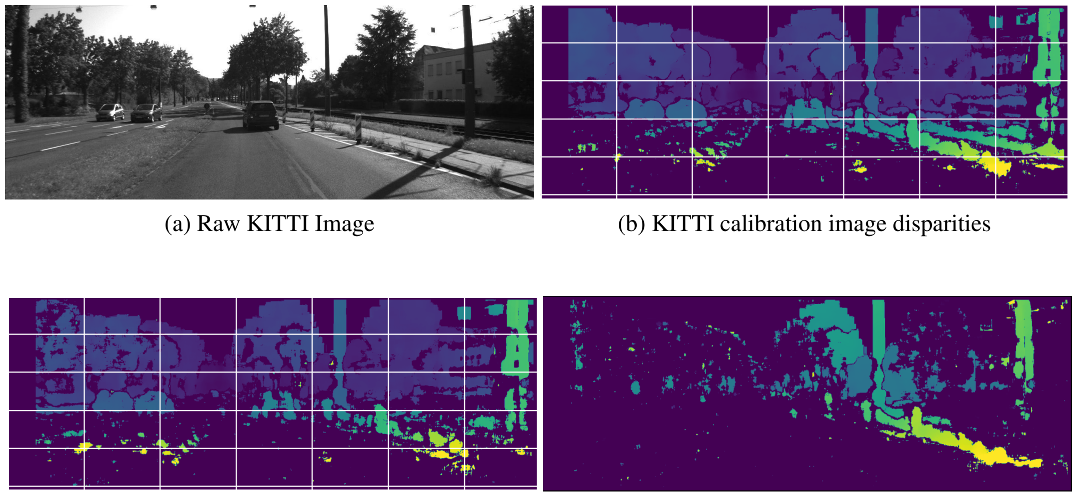


Figure 8: PatchNet matching points



(a) Raw KITTI Image

(b) KITTI calibration image disparities

(c) Our calibration disparities

(d) Uncalibrated disparities

1.4. Contributions

1.4.1 Charles Vorbach

I am responsible for all work using KITTI data. I built the stereo density and visual odometry utilities for monitoring calibration. I also designed and implemented the procedure for determining calibrations online.

1.4.2 Max Tell

I am responsible for keypoint extraction and matching. I wrote the keypoint extraction pipeline. Most of my work on this project was focused developing and training PatchNet as a model to compute learned descriptors.

1.5. Conclusion

Our project produced a robust and effective system for monitoring and maintaining stereo extrinsic calibrations using epipolar constraints, visual odometry, and as simple online calibration scheme.

We also developed a neural network model for computing keypoint descriptors to be used for stereo correspondences. This produced relatively robust and efficient correspondence matching with speed gains from GPU accelerated hardware.

The overall calibration monitoring and online calibration pipeline promises to be an effective tool for the MIT/DUT 2019 vehicle as well as for future stereo vision applications.

References

[1] B. L. Alexander Hermans, Lucas Beyer. In defense of the triplet loss for person re-identification. 2017.

[2] T. Dang, C. Hoffmann, and C. Stiller. Continuous stereo self-calibration by camera parameter tracking. *IEEE TRANSACTIONS ON IMAGE PROCESSING*, 18(7):1536–1550, 2009.

[3] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[4] R. I. Hartley. Theory and practice of projective rectification, 1998.

[5] K. Konolige. Small vision systems: Hardware and implementation. In Y. Shirai and S. Hirose, editors, *Robotics Research*, pages 203–212, London, 1998. Springer London.

[6] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004.

[7] K. T. S. M. Paolo Di Febbo, Carlo Dal Mutto. Kcnn: Extremely-efficient hardware keypoint detection with a compact convolutional neural network. CVPR, 2018.

[8] X. H. Qiangliang Guo, Jin Xiao. New keypoint matching method using local convolutional features for power transmission line icing monitoring. *Sensors*, 18(698), 2018.

[9] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[10] E. Rehder, C. Kinzig, P. Bender, and M. Lauer. Online stereo camera calibration from scratch. *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1694–1699, 2017.

[11] M. B. Simon A. J. Winder. Learning local image descriptors. CVPR, 2007.

[12] A. V. K. M. Vassileios Balntas, Karel Lenc. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. CVPR, 2017.

[13] D. P. K. M. Vassileios Balntas, Edgar Riba. Learning local feature descriptors with triplets and shallow convolutional neural networks. BMVC, 2016.

[14] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov 2000.