Article

Enhanced Checkerboard Detection Using Gaussian Processes

Michaël Hillen *, Ivan De Boi, Thomas De Kerf, Seppe Sels, Edgar Cardenas De La Hoz, Jona Gladines, Gunther Steenackers, Rudi Penne and Steve Vanlanduit

InViLab, University of Antwerp, Groenenborgerlaan 171, B-2020 Antwerp, Belgium; ivan.deboi@uantwerpen.be (I.D.B.); thomas.dekerf@uantwerpen.be (T.D.K.); seppe.sels@uantwerpen.be (S.S.); edgar.cardenas@uantwerpen.be (E.C.D.L.H.); jona.gladines@uantwerpen.be (J.G.); gunther.steenackers@uantwerpen.be (G.S.); rudi.penne@uantwerpen.be (R.P.); steve.vanlanduit@uantwerpen.be (S.V.)

* Correspondence: michael.hillen@uantwerpen.be

Abstract: Accurate checkerboard detection is of vital importance for computer vision applications, and a variety of checkerboard detectors have been developed in the past decades. While some detectors are able to handle partially occluded checkerboards, they fail when a large occlusion completely divides the checkerboard. We propose a new checkerboard detection pipeline for occluded checkerboards that has a robust performance under varying levels of noise, blurring, and distortion, and for a variety of imaging modalities. This pipeline consists of a checkerboard detector and checkerboard enhancement with Gaussian processes (GP). By learning a mapping from local board coordinates to image pixel coordinates via a Gaussian process, we can fill in occluded corners, expand the board beyond the image borders, allocate detected corners that do not fit an initial grid, and remove noise on the detected corner locations. We show that our method can improve the performance of other publicly available state-of-the-art checkerboard detectors, both in terms of accuracy and the number of corners detected. Our code and datasets are made publicly available. The checkerboard detector pipeline is contained within our Python checkerboard detection library, called PyCBD. The pipeline itself is modular and easy to adapt to different use cases.

Keywords: checkerboard detection; Gaussian processes; occlusions

MSC: 60G15; 68T10

check for updates

Citation: Hillen, M.; De Boi, I.; De Kerf, T.; Sels, S.; Cardenas De La Hoz, E.; Gladines, J.; Steenackers, G.; Penne, R.; Vanlanduit, S. Enhanced Checkerboard Detection Using Gaussian Processes. *Mathematics* **2023**, *11*, 4568. https://doi.org/10.3390/math11224568

Academic Editor: Snezhana Gocheva-Ilieva

Received: 6 October 2023
Revised: 24 October 2023
Accepted: 31 October 2023
Published: 7 November 2023

1. Introduction

Checkerboard detection is a fundamental tool in computer vision applications such as camera calibration [1–6], projector-camera systems [7,8], simultaneous localisation and mapping (SLAM) [9], and robotics in general [10–12]. This topic is of such high importance that it has received a large amount of attention from the community over the past decades and a large variety of detection methods have been developed.

Generally, checkerboard detection involves corner detection, corner refinement, and checkerboard structure recovery [2,3,13]. The locations of the inner corners of a checkerboard pattern (X-type corners) in an image are found with a corner detector. In many cases, the additional refinement of the found corner locations is necessary to achieve sub-pixel accurate locations. The structure of the checkerboard is recovered by utilising the locations and the topology of the detected corners, and the detected corners are mapped to local checkerboard coordinates.

A wide variety of automatic checkerboard detectors already exist. OpenCV's [14] standard checkerboard detector, findChessboardCorners, is based on the work of [15]. An improved version of this detector is proposed in [16], which is better suited for blurred and distorted images. More recently, a new detector called findChessboardCornersSB was added to OpenCV [17], which is more robust to all sorts of noise and runs faster on large images compared to findChessboardCorners. The major drawbacks of the previously mentioned detectors are that they require the size of the checkerboard, and a thick white edge around the checkerboard is necessary for optimal performance. A checkerboard detector that does not require the size of the checkerboard was introduced in [13]. Their structure recovery algorithm cannot cope with occlusions, which is particularly problematic when an occlusion occurs near the centre of the checkerboard. OCPAD [18], which is a successor to ROCHADE [19], and the newer version of OCamCalib [20] are both able to detect occluded checkerboards. Convolutional neural networks (CNNs) are becoming more ubiquitous for a large variety of detection tasks, including checkerboard corner detection [2–4,21–23]. Both [21] and [3] have developed CNN-based checkerboard detectors. Chen et al. improved upon the structure recovery algorithm by Geiger et al. so it can be used to detect occluded checkerboards. At the time of writing, both OpenCV implementations, OCamCalib, and the detector by Geiger et al. are publicly available. Zhang et al. share their code and data, but the trained weights for the CNN are omitted.

Many real world applications that rely on checkerboard detection suffer from low quality images, drastically impacting the corner detection performance. This could be due to low resolution, artefacts in the lenses of the camera used such as scratches or faults in the glass, contamination on the lens itself, heavily warped images from fisheye lenses, etc. An example where these factors accumulate is provided in Section 4.3. The image was taken by a camera used in endoscopy. Besides the fisheye warping and the contamination in the lens, we can also see some corners being missed due to the specular reflection. The result is a checkerboard that is only partly detected.

In this work, we address the above mentioned problems by proposing a new checkerboard detection pipeline with additional checkerboard enhancement using Gaussian processes (GP) [24], which can be performed after standard checkerboard detection. This machine learning technique is used to learn a mapping from local board coordinates to pixel coordinates, which can be used to predict the image coordinates for undetected or occluded corners and to smooth out the checkerboard corner image locations.

All of our code and the used datasets are publicly available on GitHub. The checkerboard detector and enhancement algorithms are contained within our Python checkerboard detection library, called PyCBD (https://github.com/InViLabUAntwerp/PyCBD, accessed on 30 October 2023) . This library is modular and easy to adapt to different use cases.

The rest of the paper is structured as follows: Section 2 describes the methodology implemented in the enhancement using Gaussian processes. Section 3 explains our experimental setup. We provide the results in Section 4 and discuss them in Section 5. Finally, we formulate our conclusions in Section 6.

2. Methodology

The complete pipeline is summarised in Figure 1. In this section, we will explain the Gaussian process enhancement.

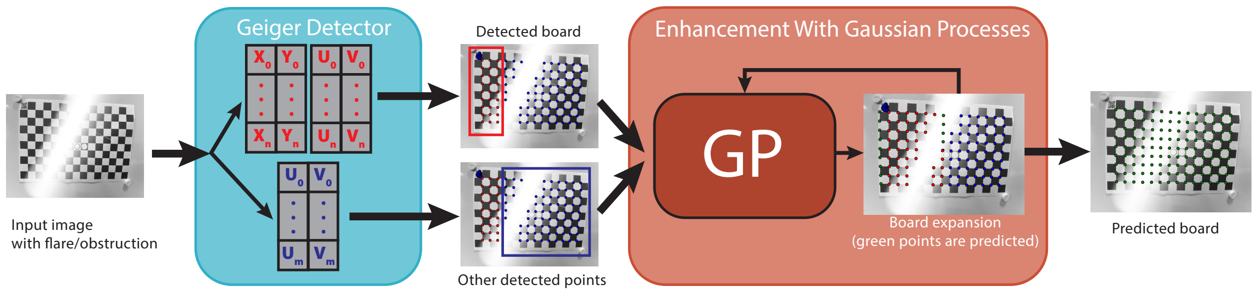


Figure 1. The proposed processing pipeline for enhancing a detected checkerboard using GP.

2.1. Gaussian Processes

As a probabilistic machine learning technique, Gaussian processes (GP) can be used to perform non-linear regression [24]. In our work, GPs are used to learn a function or mapping from local board coordinates to pixel coordinates on an image.

Let $\mathcal{D} = \{X, \mathbf{y}\}$ be a dataset of n observations, where $X = \left[\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\right]^T$ is an $n \times d$ matrix of n input vectors of dimension d and $\mathbf{y} = \left[y_1, y_2, ..., y_n\right]^T$ is a vector of continuous-valued scalar outputs. These data points are also called training points. Regression aims to find a mapping $f : \mathbb{R}^d \to \mathbb{R}$,

$$y = f(\mathbf{x}) + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2), \tag{1}$$

with ϵ being identically distributed observation noise. We implement a Gaussian process to realise this mapping.

By definition, a Gaussian process can be defined as a continuous collection of random variables, any finite subset of which is normally distributed as a continuous multivariate distribution. In our work, the data point inputs are the local xy-coordinates of corners in the checkerboard and the outputs are the uv pixel coordinates of those corners. Those inputs are jointly Gaussian, which means they are not independent but co-vary. The uv-coordinates of one corner provide some information about a corner nearby. This amount of information decreases with distance.

A Gaussian process is fully defined by its mean $m(\mathbf{x})$ and covariance function $k(\mathbf{x}, \mathbf{x}')$, in which $\mathbf{x}$ and $\mathbf{x}'$ are inputs. It is denoted as $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$. We can always normalise the data to zero mean [24]. The covariance function on the other hand shows us how much the data points influence each other as a function of their relative distance. Thus, a GP yields a distribution over functions that have a joint normal distribution,

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{m}_X \\ \boldsymbol{m}_{X_*} \end{bmatrix}, \begin{bmatrix} K_{X,X} & K_{X,X_*} \\ K_{X_*,X} & K_{X_*,X_*}, \end{bmatrix}\right) \tag{2}$$

where X and X_* are the input vectors of the n observed training points and n_* the unobserved test points, respectively. The mean values for X and X_* are given by $\boldsymbol{m}_X$ and $\boldsymbol{m}_{X_*}$. The covariance matrices $K_{X,X}$, K_{X_*,X_*}, $K_{X_*,X}$, and K_{X,X_*} are constructed by evaluating k at their respective pairs of data points. In practice, we do not have access to the latent function values directly, but are dependent on the noisy observations $\mathbf{y}$.

The conditional predictive posterior distribution of the GP can be written as:

$$\mathbf{f}_* | X, X_*, \mathbf{y}, \boldsymbol{\theta}, \sigma_\epsilon^2 \sim \mathcal{N}(\mathbb{E}(\mathbf{f}_*), \mathbb{V}(\mathbf{f}_*)) \tag{3}$$

$$\mathbb{E}(\mathbf{f}_*) = \boldsymbol{m}_{X_*} + K_{X_*,X}\left[K_{X,X} + \sigma_\epsilon^2 I\right]^{-1}\mathbf{f} \tag{4}$$

$$\mathbb{V}(\mathbf{f}_*) = K_{X_*,X_*} - K_{X_*,X}\left[K_{X,X} + \sigma_\epsilon^2 I\right]^{-1}K_{X,X_*}. \tag{5}$$

There exists a large variety of covariance functions, also called kernels [25]. A common choice is the squared exponential kernel, which we also implement in our work. It is infinitely differentiable and thus yields smooth functions. This is a reasonable assumption to make, as our checkerboards do not show discontinuities. It has the form:

$$k_{SE}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{|\mathbf{x} - \mathbf{x}'|^2}{2l^2}\right), \tag{6}$$

where σ_f^2 is a height-scale factor and l the length-scale that determines the radius of influence of the training points. For the squared exponential kernel, the hyperparameters are σ_f^2 and l. The values for the hyperparameters, which we combine in the symbol

$\boldsymbol{\theta}$, are learned by using a gradient based optimisation algorithm to maximise the log marginal likelihood,

$$\log p(\mathbf{y}|\theta, X) \propto -\frac{1}{2}\left[\mathbf{y}^T\left[K_{X,X} + \sigma_\epsilon^2 I\right]\mathbf{y} + \log|K_{X,X} + \sigma_\epsilon^2 I)|\right], \tag{7}$$

which is a combination of a data fit term and a complexity penalty and thus automatically incorporates Occam's Razor [24]. This guards the GP against overfitting. In our experiments, we use L-BFGS, a quasi-Newton method.

2.2. *Allocation of Detected Corners*

Many checkerboard detectors perform checkerboard corner detection followed by a structure recovery algorithm that allocates these detected corners to positions in a grid, based on their location and topology. It is possible that not all detected corners are allocated to this grid because they do not fit according to the structure recovery algorithm. These detected corners could either be false positives, i.e., detected corners that are not checkerboard corners, or checkerboard corners that are somehow missed by the structure recovery step. The latter, for instance, could be the result of a large flare across the image, splitting the checkerboard into two distinct islands of ordered corners. An example of this is given in Figure 2.

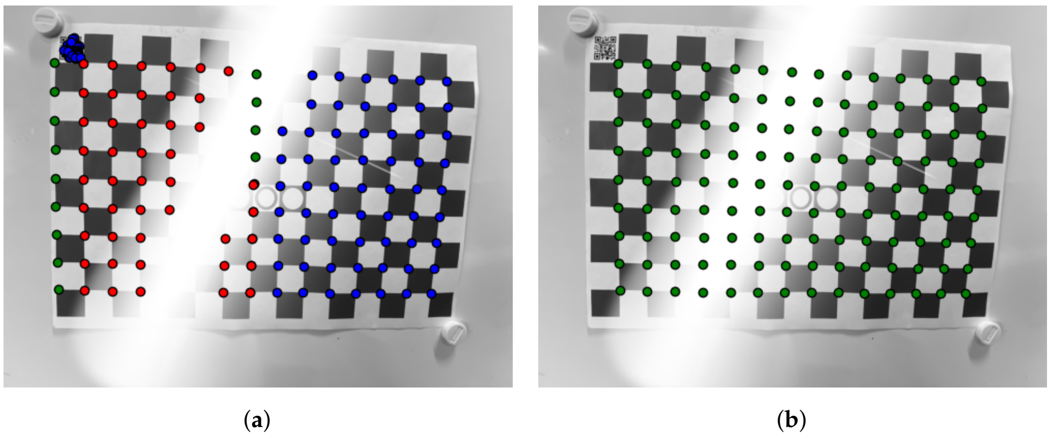


Figure 2. (**a**) Visualisation of iteration 4: red are corners for which a local coordinate is found, blue are the remaining corners without a local coordinate and green are the predicted corner locations. Notice how the original corner detection also detected corners in the QR code in the upper left corner of the image. As these corner coordinates do not correspond to any predicted values of the GP, they are deemed as false positives and filtered out. (**b**) The entire re-predicted checkerboard is in green.

In this section, we describe an algorithm to expand a partially detected checkerboard with detected corners that are part of the checkerboard pattern, but were not assigned to the grid by the structure recovery algorithm. By learning the map from local board coordinates to pixel coordinates, we can make predictions for missing grid points, which in turn can be used to add unassigned corners to the grid. This section relates to the last (red) box of our method in Figure 1. Our Algorithm is given in pseudo-code in Algorithm 1.

Our algorithm takes as input a partially detected checkerboard, which has two sets of coordinates for each corner: a set of image coordinates (boardUV) and a set of local grid coordinates (boardXY), and the image coordinates of the unassigned detected corners (cornersUV), which do not have local grid coordinates. The aim of the algorithm is to find the corresponding local grid coordinates (xy) for any remaining corners, filter out corners that are not part of the checkerboard, and find occluded corners. Each of these three sets could be empty. This is not known at the start.

The central idea behind our algorithm is the mapping from local grid coordinates (xy) of points on a checkerboard to pixel coordinates (uv) of points in a given image. This mapping is learned via Gaussian processes. We implemented two distinct GPs in parallel to perform regression from xy-coordinates to both the u-coordinates and the v-coordinates of detected corners. They both take a 2D point as input and each produce a scalar value. In Algorithm 1, the image coordinates of the corners are called cornerUV. The image and local coordinates of the corners that are already on a (possibly impartial) checkerboard are called boardUV and boardXY, respectively.

Algorithm 1 Algorithm to allocate all detected corners.

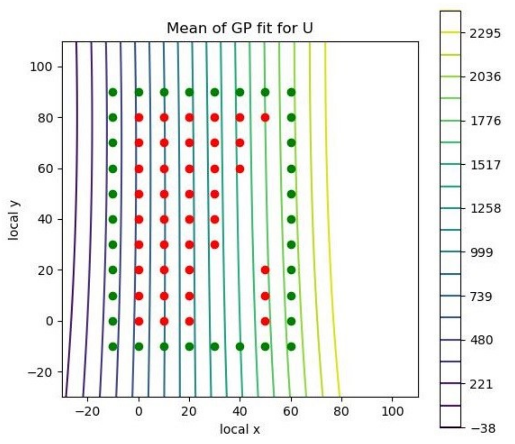
```
Require: $boardXY, boardUV, cornerUV$
  maxNrOfIterations = 10
  currIteration = 1
  while $true$ do
      $GPs \leftarrow boardXY, boardUV$
      Expand local grid of corners with $newXY$
      Predict $newUV$ for given $newXY$
      Match $newUV$ with existing $cornerUV$
      Augment $boardXY, boardUV$
      currIteration++
      if $cornerUV = boardUV$ or
       currIteration > maxNrOfIterations or
       no $newUV$ in image
       then
          break                                         ▷ leave loop
      end if
  end while
  No more corners to be allocated
  return $boardUV$                                      ▷ possibly augmented
```

In the second step, for each of the predicted corners, we search for a viable detected corner. If a corner is found, then it is added to the list of checkerboard corners. Its xy-coordinates are the input of the GP and its uv-coordinates are the detected values (not the predicted ones). If no corner can be found within a reasonable distance, which is a fraction of the distance between the checkerboard corners, then this prediction is dropped. This fraction is a hyperparameter that can be tuned for the specific application. If the value is too large, this might result in more false positives being accepted or corners being attributed to the wrong position in the grid. Naturally, this is less of a problem when there are few or no false corner detections. If the value is too small, it might result in too few corners being added to the set of accepted corners. The latter manifests itself more when we are dealing with heavily warped images.

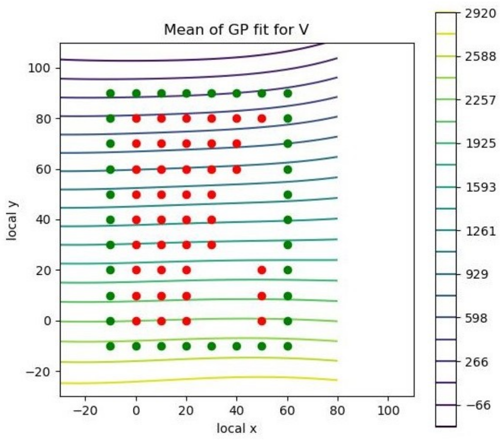
We repeat these steps, each time with a list of corners that is extended with the newly found ones. If no new corners can be found, then we expand the local xy-coordinates with two rows or columns instead of one. This allows us to bridge large gaps.

We keep iterating these steps until a maximum number of steps have been reached, no new points are within the image, or no more corners are without corresponding xy-coordinates (all corners are accounted for). Corners not close to a predicted corner are deemed false positives or outliers. These are falsely detected corners and are ignored on the checkerboard.

At the end of the algorithm, all detected corners are either part of the checkerboard or considered a false positive. A visualisation of a situation after a few iterations is given in Figure 3.



(**a**)



(**b**)

Figure 3. An example of a situation given by the algorithm after four iterations. The red dots are the locations of detected checkerboard corners on a perfect checkerboard. The green dots are locations where we predict a value for the u and v coordinates of those corners, in (**a**,**b**), respectively. The values for the coordinates are given by the contour lines. Notice how these contour lines capture the curvature of the checkerboard in the real world. A flat checkerboard would result in straight lines for the contour lines. The deviation from this is the result of the bending of the paper checkerboard as seen in Figure 2.

2.3. Gaussian Process Refinement

After running the algorithm described in Section 2.2, we retrain the GPs one more time on all found corners. This allows us to exploit the GP in two ways.

First, we make predictions for the local xy-coordinates of corners that are without corresponding uv-coordinates. These are locations in the image where no corner is detected, even though the checkerboard tells us there should be one. This enables us to fill in occluded corners and even corners that are outside the borders of the image. An example can be seen in Figure 2, where a large flare hides some corners.

Second, we re-predict the uv-coordinates for all detected corners. In this work, we implemented a squared exponential kernel, which yields smooth functions that are infinitely many times differentiable [24]. This results in an extra refinement step, which removes noise from those predicted coordinates. This is due to the fact that the GP utilises the uv-coordinates of all corners to predict a single one and does not base its prediction only on the surrounding pixels.

3. Experimental Setup

3.1. Dataset Generation

In this work, two datasets are generated. The first is a synthetic dataset, where checkerboards are generated artificially and drawn on random background images. The background images are generated through DALL-E 2 and are made to resemble a typical laboratory environment. When creating the checkerboards on the background images, the ground truth points are known. When augmenting the images, these reference points are transformed in the same way, thus obtaining the actual ground truth for the checkerboard locations. The second dataset, referred to as the "real dataset", is generated using two types of cameras, including a thermal infrared (IR) camera (Xenics Ceres) and a snapshot multispectral imaging (MSI) camera (Photonfocus MV0-D2048x1088-C01-HS02-G2). The multispectral and thermal infrared cameras are specifically chosen because they pose a challenge to current checkerboard detection due to heavy blurring or limited contrast between checkers.

When using a synthetic dataset, the exact sub-pixel location of the corners can be easily determined with high accuracy. This level of precision is difficult to obtain when working with real datasets. Several different types of distortions are used to create realistic environmental conditions [26]. These techniques include Gaussian blur, shot noise, optical distortions—such as pincushion, moustache, and barrel distortion—perspective transformations, and rotational transformations, as is common in the literature [21]. For the Gaussian blur, multiplicative noise, and projective distortions, different levels of augmentations are used, from slightly distorted to heavily distorted. For each category and level of augmentation, 100 images are generated and evaluated.

3.2. Evaluation Methods

For the synthetic dataset, two metrics are used to evaluate the different methods: the amount of fully detected checkerboards and the pixel error of the corner locations. A fully detected checkerboard refers to cases where all the checkerboard corners are correctly detected by the algorithm. A checkerboard corner is considered to be correctly detected if its predicted location is within a Euclidean distance of two pixels from the true location. The pixel error is computed as the average Euclidean distance between the predicted and true locations of all the checkerboard corners in the image. The lower the value of this metric, the better the accuracy of the algorithm in predicting the location of the checkerboard corners. The subsequent processing of the image will yield better results.

4. Results

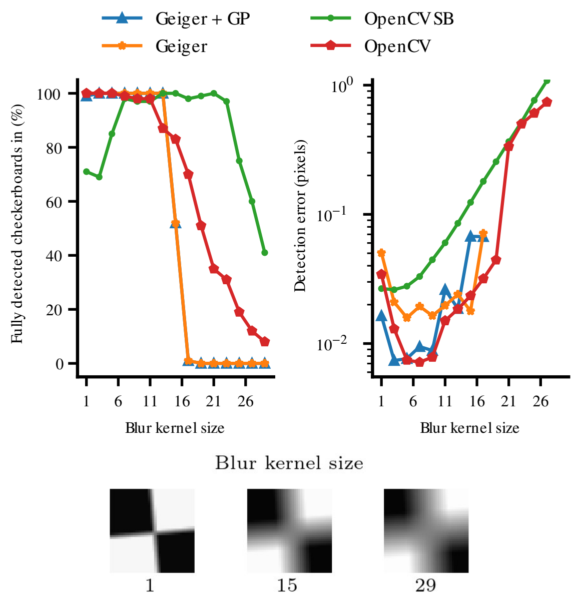
The results of both datasets, simulated and real, are discussed separately below. Four different methods were compared to each other: the method as proposed by Geiger with and without Gaussian enhancement (Geiger, Geiger + GP) and the industry standards OpenCV (OpenCV) and OpenCV Sector Based approach (OpenCV SB) [17]. The Geiger detector we used is a lightly modified version of the implementation in the libcbdetect library [27]. It should be noted that the simulated images are tailored for use with OpenCV: there are clear white and black edges, a thick white border around the checkerboard and the checkerboard are always completely in the image; not doing so would result in a lower detection rate.

4.1. Simulated Data Results

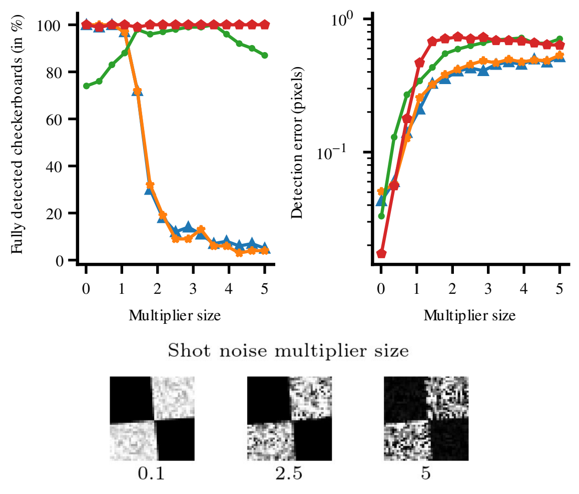
The results of the synthetic images, with varying degrees of blur, shot noise, and perspective transformations, are displayed in Figure 4a–c. Each of these figures is composed of two graphs. The left graph displays the number of fully detected checkerboards as a percentage of the total. The right graph shows the average error for each method. To obtain a fair comparison between the methods, the average of a single checkerboard is only taken into account when four or more methods find all the points of the respective checkerboard.

The Geiger detector, and by extension the proposed Geiger + GP method, fails to detect corners under a lower degree of blurring and shot noise compared to the OpenCV detectors, as can be seen in the left graphs in Figure 4a,b. The right graphs in Figure 4a,b show that the GP corner refinement is able to reduce the corner position error when blur and shot noise is introduced into the images. The Geiger method is able to detect more corners under higher degrees of perspective transformation compared to the OpenCV methods, and the GP enhancement is able to improve this further for higher degrees of perspective scaling. The OpenCV SB method performs notably worse for all degrees of perspective transformation. This is shown in the left graph of Figure 4c.

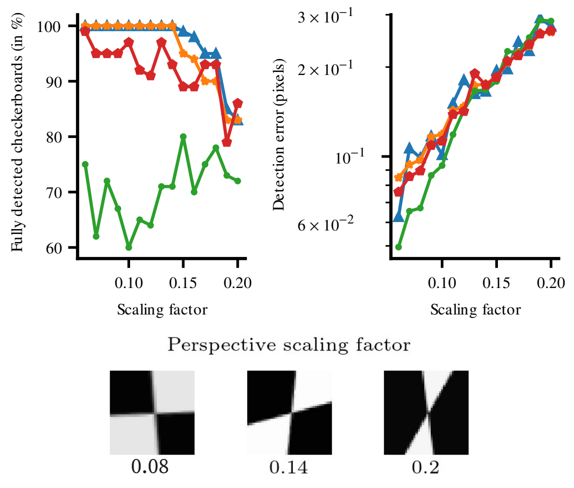
When different types of optical distortion are applied, all checkerboard detection methods are able to detect most if not all checkerboards, except for OpenCV SB, as shown in Table 1. The mean pixel error is similar for all methods but worst for the Geiger detector. The GP corner refinement is able to slightly improve this result, but not enough for it to be better than the OpenCV methods, as shown in Table 1.



(**a**)



(**b**)



(**c**)

Figure 4. Results of testing different methods for their performance on (**a**) checkerboard blur, (**b**) shot noise, and (**c**) perspective distortion.

Table 1. Results for different types of distortion.

Fully Detected Checkerboards (%)				
Type of Distortion	**Geiger**	**Geiger + GP**	**OpenCV**	**OpenCV SB**
Barrel	100	100	100	65
Mustache	100	100	100	49
Pincushion	85	100	100	85
Average Corner Error (Pixels)				
Type of Distortion	**Geiger**	**Geiger + GP**	**OpenCV**	**OpenCV SB**
Barrel	0.078	0.069	0.051	0.032
Sunglints	0.085	0.066	0.063	0.048
Pincushion	0.179	0.178	0.170	0.140

When there are occlusions added to the images, we see that, using the GP enhancement step, it is still possible to obtain fully detected checkerboards. The average pixel error for the occluded pixels is still in the sub-pixel range, as can be seen in Table 2.

Table 2. Results for increasing amounts of occluded corners introduced into the dataset. Only the results for the Geiger + GP method are shown here. The OpenCV methods are not included as they are unable to cope with occlusions. The reported pixel error is only for specific occluded pixels.

Number of Missing Corners	Average Corner Error (Pixels)
1	0.0226
3	0.0589
5	0.1023

4.2. Real Data Results

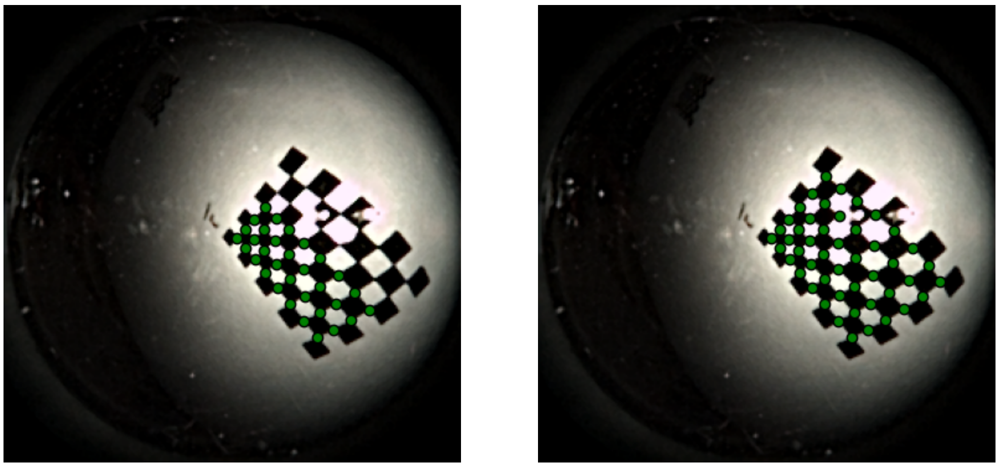
When reviewing the results in Table 3, it becomes evident that significant variations exist among the different datasets. However, the Geiger method with the proposed Gaussian process enhancement consistently demonstrates the best overall performance. The checkerboards in the IR dataset originally have white checkers and dark borders; in other words, the checkerboards in the images look like the inverse of regular checkerboard targets. Therefore, we performed two tests on the IR dataset—with the original images and with images whose grayscale was inverted so the checkerboards look like regular checkerboards. We noticed a substantial disparity between the IR and IR-inverted datasets, indicating that the performance of both OpenCV methods is adversely affected when the images are not preprocessed beforehand. Upon inverting the infrared images, we observed improvements across all methods, with the most significant enhancement occurring in the case of the OpenCV method, resulting in an eightfold increase in the amount of detected corners. Furthermore, a noteworthy observation can be made when analysing the results of the MSI on the large dataset. Surprisingly, the Geiger method outperforms the others, even though the simulated dataset results indicated that it struggled with increasing augmentations compared to the alternative methods. The application of Gaussian process enhancement further bolsters the detection of checkerboards, particularly in the case of large checkerboards (14×9). This substantial increase can be attributed to the higher chances of missing a corner in larger checkerboards. While a similar trend is observed with small checkerboards, the OpenCV method performs relatively better but still falls short of the performance achieved by the Geiger method with the proposed Gaussian process enhancement.

Table 3. Results for the different checkerboard detection methods for both the infrared (IR) images and the multispectral images (MSI). For the IR images, one set of results is without inverting the images and one with inverting the images. For the MSI images, a large and a small checkerboard are evaluated. The values in bold represent the highest scores.

	Detected Checkers (%)			
Method	**IR**	**IR Inverted**	**MSI Large CB**	**MSI Small CB**
OpenCV	12.65	**96.15**	14.23	58.4
OpenCV SB	6.44	46.15	24.21	25.6
Geiger	67.25	84.61	22.54	13.43
Geiger + GP	**95.77**	**96.15**	**77.64**	**88.8**

4.3. Use Case: Endoscopic Camera Calibration

As a final demonstration of our method, we describe the use case of working with images of calibration patterns taken to calibrate a Pillcam COLON2 double endoscope camera capsule. Several difficulties arise when calibrating this device. The resolution is low (320 × 320), the warping of the image is significant (172° field of view), and even the most minute artefacts in the glass casing are visible in the image. Detecting a checkerboard with the Geiger method fails, as not every corner is detected. This results in a significant portion of the corners not being used in the checkerboard. The current workaround is to add them manually, which is labour-intensive. Our method can infer the missing corners. We demonstrate this on images from the work of [28], taken from their accompanying online GitHub repository. An example can be seen in Figure 5. This image of a checkerboard is of very low quality, which resulted in several occlusions. The upper right part can no longer be fitted in a checkerboard by the Geiger method. Our Gaussian process method expands the initially-found checkerboard to include other corners, and predicts corner uv-coordinates for occluded corners as well. Moreover, the entire checkerboard is refined.



(**a**) (**b**)

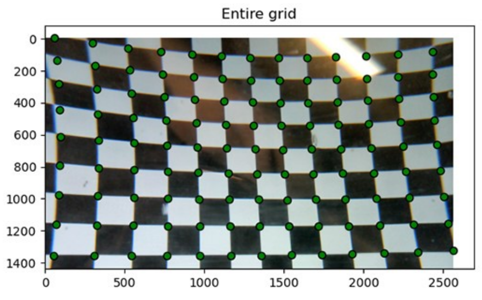
Figure 5. An example of a low quality image of a checkerboard detected by (**a**) Geiger and (**b**) Geiger plus the Gaussian process enhancement.

5. Discussion

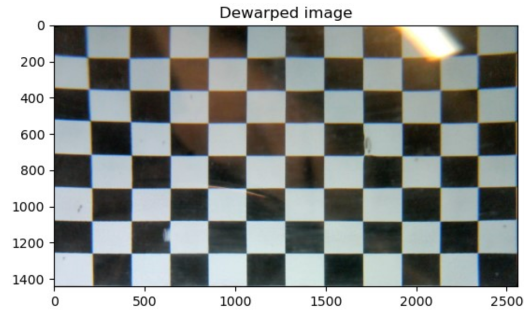
In this research, we are working with the mean of the posterior distribution of the Gaussian processes (the predicted uv-coordinates in an image for a given local xy-coordinate on a checkerboard); this tends to smooth out the results [24]. A squared exponential kernel is implemented, resulting in infinitely differentiable, and thus smooth, functions. The overall result is a checkerboard that has been smoothed out. This can be seen as removing noise or jitter on the found corners. We exploit this feature as an extra final step of corner refinement. However, caution is advised when working with heavily warped checkerboards. When working with GPs, one could overly smooth out the corners in cases of heavily warped checkerboards or images from fisheye lenses. This could be addressed with more complex kernels or even deep Gaussian processes that will be investigated in future work.

Another way to look at this is through the insight that a GP will adjust the coordinates for a single corner based on the coordinates of all other corners. In a GP, all data points—in our case corners—are assumed to be jointly Gaussian [24]. This means they are not independent of each other—they co-vary. Other refinement methods are only based the values of neighbouring pixels, not the whole (possibly warped) grid. The justification for this is the fact that the underlying truth is a regularly spaced rectangular pattern.

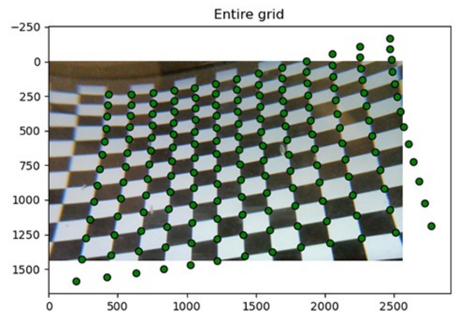
The GP enhancement method can be used to interpolate and extrapolate point locations in between the corners and even outside the checkerboard, assuming the curvature pattern can be described by the points in the board. This allows us to unwarp and frontalise the image. We simply predict the corresponding pixel uv-coordinate for a dense grid of xy-coordinates. A demonstration can be seen in Figure 6.



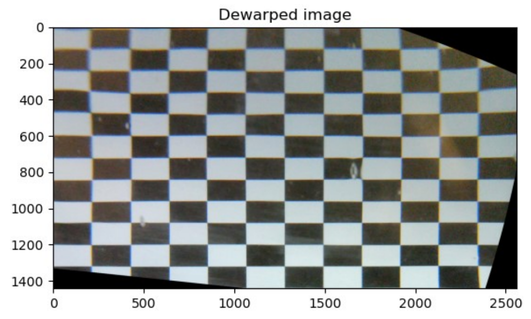
(**a**)



(**b**)



(**c**)



(**d**)

Figure 6. (**a**) A Gaussian process predicted the (green) corners of a checkerboard. Notice how the occlusions from the lighting are also filled in. (**b**) This model can be used to dewarp and frontalise an image. (**c**) A checkerboard with points beyond the image borders. (**d**) The dewarped and frontalised result, leaving pixels outside the original image black.

In future research, we will explore the possibility of working with fiducial markers [29] to assert the orientation of the board. As our current method is able to predict corners and, in fact, rows and columns that are not even on the image, the addition of fiducials becomes necessary to decide to expand the board in the image to the left, right, up, or downward directions.

Even though Gaussian processes can perform well in low data regimes [24], some numerical instabilities and oddities might occur. For instance, when working with a slightly warped 3×3 checkerboard, there is not much information upon which to build a solid statistical model. The danger is that, from a Bayesian point of view, the corners could be explained by considering their coordinates as pure noise [30]. This results in patterns for the predicted checkerboard corners that simply do not make sense. In this case, our software (version 1.2.2) allows for end-users to incorporate prior knowledge in the model by adjusting some of the options and hyperparameters in the software. In the code itself, we provide a lot of comments to guide the end-user. We suggest two things to try: First, put a lower limit on the length scales. This will automatically force the model to consider configurations in which the points are on a smooth grid, which in fact they are. Second, consider an upper level on the noise. This will force the model to stay close to the values of the detected corners. All of this is included in comments in the code.

6. Conclusions

In this paper, we designed a new checkerboard detection pipeline, consisting of a checkerboard detector and a checkerboard enhancement with Gaussian processes. This research has shown that our Gaussian process enhancement is able to improve checkerboard detection results. By learning a mapping from local board coordinates to image pixel coordinates via a Gaussian process, we can fill in occluded corners, expand the board beyond the image borders, allocate detected corners that do not fit an initial grid, and remove noise on the detected corner's locations. We explained the role Gaussian processes play in enhancing the results. Lastly, we provide all our code as open source, in the form of a modular and easy-to-use Python checkerboard detection library called PyCBD.

The findings will be of interest to people working on camera calibration, camera localisation, SLAM, stereoscopic vision, depth sensing, and many others.

Author Contributions: Conceptualization, I.D.B.; software, I.D.B., M.H., S.S. and E.C.D.L.H.; validation, M.H. and J.G.; formal analysis, T.D.K.; investigation, I.D.B., M.H. and T.D.K.; resources, G.S., S.V. and R.P.; data curation, T.D.K.; writing—original draft preparation, I.D.B., M.H. and T.D.K.; writing—review and editing, S.S., S.V. and R.P.; visualization, J.G.; Project administration, M.H.; supervision, G.S., S.V. and R.P.; funding acquisition, G.S., S.V. and R.P. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by ETF Fund—PhairywinD project by Belgium SPF Economy and the University of Antwerp BOF [FFB200259, 42339, 39928].

Data Availability Statement: The code and datasets used in this article will be made available on a public GitHub repository (https://github.com/InViLabUAntwerp/PyCBD, accessed on 30 October 2023) and distributed through a PyPi package under the name PyCBD.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SLAM	Simultaneous Localisation and Mapping
CNN	Convolutional Neural Network
GP	Gaussian Process
MSI	Multispectral Imaging
IR	Infrared

References

1. Lochman, Y.; Liepieshov, K.; Chen, J.; Perdoch, M.; Zach, C.; Pritts, J. BabelCalib: A Universal Approach to Calibrating Central Cameras. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), Montreal, BC, Canada, 11–17 October 2021; pp. 15253–15262.
2. Chen, B.; Liu, Y.; Xiong, C. Automatic checkerboard detection for robust camera calibration. In Proceedings of the 2021 IEEE International Conference on Multimedia and Expo (ICME), Shenzhen, China, 5–9 July 2021; pp. 1–6.
3. Zhang, Y.; Zhao, X.; Qian, D. Learning-Based Distortion Correction and Feature Detection for High Precision and Robust Camera Calibration. *IEEE Robot. Autom. Lett.* **2022**, *7*, 10470–10477. [CrossRef]
4. Kang, J.; Yoon, H.; Lee, S.; Lee, S. Sparse Checkerboard Corner Detection from Global Perspective. In Proceedings of the 2021 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), Kuala, Terengganu, 13–15 September 2021; pp. 12–17. [CrossRef]
5. Sels, S.; Ribbens, B.; Vanlanduit, S.; Penne, R. Camera Calibration Using Gray Code. *Sensors* **2019**, *19*, 246. [CrossRef]
6. Albarelli, A.; Rodolà, E.; Torsello, A. Robust Camera Calibration using Inaccurate Targets. In Proceedings of the British Machine Vision Conference, Aberystwyth, UK, 30 August–2 September 2010; BMVA Press: Durham, UK, 2010; pp. 16.1–16.10. . [CrossRef]

7. Sun, W.; Yang, X.; Xiao, S.; Hu, W. Robust Checkerboard Recognition for Efficient Nonplanar Geometry Registration in Projector-Camera Systems. In Proceedings of the 5th ACM/IEEE International Workshop on Projector Camera Systems, Marina del Rey, CA, USA, 10 August 2008; Association for Computing Machinery: New York, NY, USA, 2008; PROCAMS '08, pp. 1–7. [CrossRef]
8. Juarez-Salazar, R.; Diaz-Ramirez, V.H. Flexible camera-projector calibration using superposed color checkerboards. *Opt. Lasers Eng.* **2019**, *120*, 59–65. [CrossRef]
9. Ito, A.; Li, J.; Maeda, Y. SLAM-Integrated Kinematic Calibration Using Checkerboard Patterns. In Proceedings of the 2020 IEEE/SICE International Symposium on System Integration (SII), Honolulu, HI, USA, 12–15 January 2020; pp. 551–556.
10. Enebuse, I.; Foo, M.; Ibrahim, B.S.K.K.; Ahmed, H.; Supmak, F.; Eyobu, O.S. A Comparative Review of Hand-Eye Calibration Techniques for Vision Guided Robots. *IEEE Access* **2021**, *9*, 113143–113155. [CrossRef]
11. Gui, Y.; Wu, Y.; Wang, Y.; Yao, C. Visual Image Processing of Humanoid Go Game Robot Based on OPENCV. In Proceedings of the 2020 Chinese Control And Decision Conference (CCDC), Hefei, China, 22–24 August 2020; pp. 3713–3716. [CrossRef]
12. Vaníček, O.; Chaluš, M.; Liška, J. 3D Vision Based Calibration Approach for Robotic Laser Surfacing Applications. In Proceedings of the 2022 20th International Conference on Mechatronics-Mechatronika (ME), Pilsen, Czech Republic, 7–9 December 2022; pp. 1–6.
13. Geiger, A.; Moosmann, F.; Car, O.; Schuster, B. Automatic camera and range sensor calibration using a single shot. In Proceedings of the 2012 IEEE International Conference on Robotics and Automation, St. Paul, MN, USA, 14–18 May 2012; pp. 3936–3943. [CrossRef]
14. Kaehler, A. *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*; O'Reilly Media: Sebastopol, CA, USA, 2016.
15. Vezhnevets, V. OpenCV Calibration Object Detection. 2005. Available online: https://www.graphicon.ru/oldgr/en/research/calibration/opencv.html (accessed on 3 March 2023).
16. Rufli, M.; Scaramuzza, D.; Siegwart, R. Automatic detection of checkerboards on blurred and distorted images. In Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems IROS, Nice, France, 22–26 September 2008; pp. 3121–3126. [CrossRef]
17. Duda, A.; Frese, U. Accurate Detection and Localization of Checkerboard Corners for Calibration. In Proceedings of the British Machine Vision Conference, Newcastle upon Tyne, UK, 3–6 September 2018; pp. 1–10.
18. Fuersattel, P.; Dotenco, S.; Placht, S.; Balda, M.; Maier, A.; Riess, C. OCPAD—Occluded checkerboard pattern detector. In Proceedings of the 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), Lake Placid, NY, USA, 7–10 March 2016; pp. 1–9. [CrossRef]
19. Placht, S.; Fürsattel, P.; Mengue, E.A.; Hofmann, H.; Schaller, C.; Balda, M.; Angelopoulou, E. ROCHADE: Robust Checkerboard Advanced Detection for Camera Calibration. In Proceedings of the Computer Vision—ECCV 2014, Zurich, Switzerland, 6–12 September 2014; Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T., Eds.; Spring: Cham, Switzerland, 2014; pp. 766–779.
20. Scaramuzza, D.; Martinelli, A.; Siegwart, R. A Toolbox for Easily Calibrating Omnidirectional Cameras. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–13 October 2006; pp. 5695–5701. [CrossRef]
21. Chen, B.; Xiong, C.; Zhang, Q. CCDN: Checkerboard Corner Detection Network for Robust Camera Calibration. In Proceedings of the Intelligent Robotics and Applications, Newcastle, NSW, Australia, 9–11 August 2018; Proceedings, Part I; Chen, Z., Mendes, A., Yan, Y., Chen, S., Eds.; Spring: Cham, Switzerland, 2018; pp. 324–334.
22. Wang, G.; Zheng, H.; Zhang, X. A Robust Checkerboard Corner Detection Method for Camera Calibration Based on Improved YOLOX. *Front. Phys.* **2022**, *9*, 819019. [CrossRef]
23. Wu, H.; Wan, Y. A highly accurate and robust deep checkerboard corner detector. *Electron. Lett.* **2021**, *57*, 317–320. [CrossRef]
24. Rasmussen, C.E.; Williams, C.K.I. *Gaussian Processes for Machine Learning*; The MIT Press: Cambridge, MA, USA, 2006.
25. Duvenaud, D.K.; College, P. Automatic Model Construction with Gaussian Processes. Ph.D. Thesis, University of Cambridge, Cambridge, UK, 2014.
26. Buslaev, A.; Iglovikov, V.I.; Khvedchenya, E.; Parinov, A.; Druzhinin, M.; Kalinin, A.A. Albumentations: Fast and Flexible Image Augmentations. *Information* **2020**, *11*, 125. [CrossRef]
27. ftdlyc. Libcbdetect: A Library for Chessboard Detection. Available online: https://github.com/ftdlyc/libcbdetect (accessed on 2 March 2023).
28. Ozyoruk, K.B.; Gokceler, G.I.; Bobrow, T.L.; Coskun, G.; Incetan, K.; Almalioglu, Y.; Mahmood, F.; Curto, E.; Perdigoto, L.; Oliveira, M.; et al. EndoSLAM dataset and an unsupervised monocular visual odometry and depth estimation approach for endoscopic videos. *Med. Image Anal.* **2021**, *71*, 102058. [CrossRef] [PubMed]
29. Kalaitzakis, M.; Cain, B.; Carroll, S.; Ambrosi, A.; Whitehead, C.; Vitzilaios, N. Fiducial markers for pose estimation: Overview, applications and experimental comparison of the artag, apriltag, aruco and stag markers. *J. Intell. Robot. Syst.* **2021**, *101*, 71. [CrossRef]
30. Lázaro-Gredilla, M. Bayesian warped Gaussian processes. In Proceedings of the 25th International Conference on Neural Information Processing Systems—Volume 1, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1619–1627.