Master Thesis

Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Vision based Understanding of the RoboCup Small Size League field

Felix Weinmann
Chair for Computer Science IV (Computer Vision)
Department of Computer Science & CAIDAS
University of Würzburg

Prof. Dr. Radu Timofte
Supervisor

Dr. Zongwei Wu
Coexaminer

Submission
07. November 2024

Abstract

In the robot soccer competition RoboCup Small Size League a camera above the playing field provides robot and ball position data for all participants. Long setup times and the inability to handle variable lighting conditions limit the currently used software SSL-Vision. A new SSL vision test dataset collection is proposed with corresponding measures to evaluate SSL related vision based algorithms. To enable an automated geometry setup line based camera calibration approaches used in sports player tracking are adapted to handle distortion and degenerate perspectives in this work. Multiple methods including a novel circular blob detector based on the color gradient dot product are investigated to detect color coded robots and balls in varying conditions. Relative position based blob assignment, k-Means color clustering and object tracking are used to determine object types and identities. The proposed approaches have been evaluated on multiple fields to determine accuracy, reliability and compliance to real-time limits under diverse conditions.

Zusammenfassung

Im Roboterfußballwettbewerb RoboCup Small Size League liefert eine Kamera über dem Spielfeld Roboter- und Ballpositionsdaten für alle Teilnehmer. Die aufwändige Einrichtung und fehlende Nutzbarkeit unter wechselnden Lichtbedingungen limitiert das derzeitig genutzte Programm SSL-Vision. Eine neuer SSL Bildverarbeitungs-Testdatensatz wird zusammen mit dazugehörigen Bewertungsmaßnahmen eingeführt, um die Performanz der Algorithmen beurteilen zu können. Um eine automatisierte Geometrieeinrichtung zu ermöglichen werden in dieser Arbeit linienbasierte Kamerakalibrierungsansätze aus dem Sportlertracking um die Behandlung von Verzerrung und degenerierten Perspektiven erweitert. Mehrere Methoden inklusive eines neuartigen Blobdetektors basierend auf dem Skalarprodukt des Farbgradienten werden untersucht, um die farbkodierten Roboter und Bälle unter unterschiedlichen Umständen zu erkennen. Zur Bestimmung der Objekttypen und -identitäten werden eine positionsbasierte Blobzuweisung, k-Means Farbgruppierung und Objektverfolgung verwendet. Die vorgeschlagenen Ansätze werden auf mehreren Feldern bewertet, um die Genauigkeit, Zuverlässigkeit und Einhaltung von Echtzeitanforderungen unter vielfältigen Umständen zu bestimmen.

Acknowledgements

This endeavor would not have been possible without Professor Radu Timofte, which I am very grateful for being open to this external topic and the many pointers provided throughout the thesis.

This thesis would not have started without Nicolai Ommer, who has initially suggested the topic as a potential master thesis topic, has provided me the TIGERs Lab and KIKS raw datasets and facilitated the distribution of my compound dataset. Many thanks for David Brand, who has supported this work through lending me an official SSL camera and computer and who has hang up additional camera setups at the Schubert Crailsheim, RoboCup German Open and RoboCup 2024 tournaments for my dataset recordings and experiments.

Lastly, I'd like to mention the encouraging words and positive feedback I've received from many of the RoboCup 2024 SSL participants.

Contents

1 Introduction

The annual RoboCup competition has been created to provide a competitive environment for robotics research, improving the capabilities of robots over time. The long term goals of the RoboCup initiative are winning a soccer game against the incumbent FIFA World Cup winner using humanoid robots by 2050 and promoting AI and robotics research by pursuing that goal [1], [2]. The RoboCup Small Size League (SSL) is one of the founding leagues focused on multi-agent coordination in a dynamic environment [3]. To enable this focus global ball and robot position data extracted from a shared camera above the playing field is sent out to all participants over a local network. Since 2011 the SSL-Vision software is used to process the camera data and generate the position information. This work investigates new ways to automatically calibrate the camera and detect these objects in visual data to replace the thresholding based detection approach of SSL-Vision which requires extensive calibration and static even illumination. This thesis therefore tries to support two long term goals [4] of the SSL: Reducing game interruptions and relaxing SSL-Vision constraints.

1.1 The RoboCup Small Size League field

The rules of the SSL [5] define the playing field, ball and robot requirements. The area of the playing field where robots and ball need to be tracked measures up to 12.6×9.6 m and is delimited by 10 cm high black walls. The surface of the playing field consists of green felt mat or carpet. White field lines mark the playing area, defense area and center circle. The goals consists of solid white walls and are open at the top to guarantee direct line of sight into the goal. As space and resources are oftentimes limited there are frequent deviations from the rules at local fields like smaller field sizes, wooden colored walls or gray carpets. These deviations should be supported by this work in order to keep compatibility.

Robots and balls are color coded to facilitate the visual object detection and tracking from above. Figure 1.1 showcases a SSL robot and ball with all color coded elements clearly visible. The 43 mm diameter golf ball is orange to ensure maximum color contrast to green field carpets. The 50 mm diameter yellow or blue blob at the center of the robot indicates the team color. The order of lime green and pink colored 40 mm diameter side blobs corresponds to the "jersey number" of the robot for unique identification. The butterfly pattern arrangement of the color blobs on top of the robot facilitates the determination of the robot orientation.

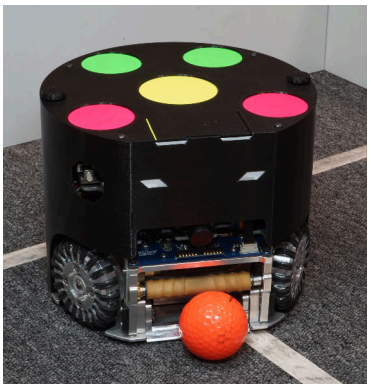


Figure 1.1: A SSL robot with blob pattern on top and orange golf ball in front showing the color-coded elements usable for detection.

To enable detection and tracking of robots and balls one or more cameras are positioned above the field. Typically they are positioned at the center of the area to observe to maximize the resolution over the whole field. As camera transition areas are a frequent source of misplays due to offsets in the detections the amount of cameras is usually the minimum amount necessary to capture the whole playing field at a sufficient resolution.

1.2 Novel contributions

This work introduces a new dataset collection with the SSL vision test data collection to enable the evaluation of vision approaches in the SSL. As the performance of SSL-Vision, the current visual object detection program used by the SSL, has not been methodically evaluated new measures are introduced to evaluate the proposed solution. The dRGB color space is proposed as novel illumination invariant color space. With the gradient dot product blob detection algorithm a novel circular color blob detection approach is proposed.

1.3 Outline

Chapter 2 describes motivation behind the proposed solution, discusses the issues with the current state and the resulting goals and requirements for this work.

In Chapter 3 the scope and ideas of previous solutions and research fields are discussed.

Chapter 4 will introduce the new dataset collection, hardware and novel proposed metrics used to evaluate the approaches in this work.

The approaches developed and tested are described in Chapter 5.

Chapter 6 contains the discussion about about the achieved results of the approaches and potential future developments.

The implementation details used to achieve the requirements are described Appendix A.

2 Motivation

The RoboCup Small Size League (SSL) currently uses SSL-Vision[1] as computer vision software providing robot and ball location information. As the participants of recent challenges without data from outside of the robot have not reached a state capable of regular play SSL-Vision remains the sole data source for global position and orientation information for all SSL teams, the automatic referees and the game controller. The color thresholding approach and manual calibration of the SSL-Vision software result in various robustness and usability issues reducing the availability and suitability of match occations and locations. This makes SSL-Vision a major restriction of the SSL, as further described in Section 2.1. The problem definition and requirements are then derived from these current constraints and capabilities in Section 2.2.

2.1 Restrictions of SSL-Vision

SSL-Vision uses the field lines as sole feature for a semi-automatic camera calibration. The typical perpendicular viewing angle can lead to multiple parameters becoming unbounded, as further discussed in Section 5.1.3. This leads to undesirable physical offsets and offsets between different cameras on elevated objects like robot patterns.

Figure 2.1: Images from fields (left Twente 2019, right Crailsheim 2023) with detection issues due to natural sunlight leading to uneven lighting and illumination changes over time.

[1] https://github.com/RoboCup-SSL/ssl-vision

The global color classification via lookup table (LUT) approach of SSL-Vision has great problems with temporal and spatial inconsistencies in the lighting. Uneven spatial lighting leads on to a reduced game quality due to blobs of different colors sharing a color in another area of the field. Changes in color temperature and image brightness due to natural sunlight or light flickering with line frequency causes color misclassification. Figure 2.1 shows two field setups where the natural sunlight influence has lead on to game interruptions due to the necessity of manual recalibration. Delays during games are undesirable due to the increase in game duration, loss of interest from spectators and continuous battery drainage. The need to reduce the influence of natural sunlight and guarantee even lighting over the field is a major venue restrictions, as the rarity of such places leads on to the inability to showcase the autonomity of the soccer robots at exhibitions. Other sources of frequent misclassification with phantom or failed detections come from chromatic aberration around bright blobs and the bilinear Bayer demosaicing step used in SSL-Vision at field line borders.

The large amount of colors in use (see Section 1.1) reduces the distance between these colors in all color spaces. This necessitates carefully chosen and standardized color shades to prevent overlaps that cannot be represented in the LUT. The manual calibration of cameras and colors required for SSL-Vision to operate takes multiple hours. The current monolithic architecture of the SSL-Vision software increases the setup time in the common case the cameras are connected to different computers due to the necessity of repeated configuration for every single instance. This causes reduced field availability at longer events such as weekend tournaments and the RoboCup and prevents the showcase of SSL matches at single day exhibitions completely.

On a typical competition field configuration with a Division B sized field and a single Blackfly S camera the best case camera resolution with perpendicular viewing angle and perfect crop is $3.92 \frac{\text{mm}}{\text{px}} = \frac{9.6 \text{ m}}{2448 \text{ px}}$. The high requirements to achieve acceptable position accuracy with SSL-Vision lead on to only using the optimal camera placement centrally over the observation area, resulting in increased field setup effort due to the requirement for trusses. The position differences in multi camera setups have led to increased desirability of single camera fields, resulting in an increased venue ceiling requirement.

Due to the Central Processing Unit (CPU) based processing approach the processing time of SSL-Vision is near the frame time limit despite hand-vectorized AVX2 code. Due to a reduced frame rate because of CPU overload when displaying or exporting the camera frames with current SSL-Vision no root cause analysis in case of reduced accuracy or detection rate is possible during gameplay, increasing the interruption time during matches. This also has lead on to a scarcity of test data for vision development.

2.2 Problem and performance definition

Detect the position and orientation of all robots and balls on the SSL field in camera data with minimized setup effort and improved robustness while keeping a similar or better accuracy, detection rate and repairability compared to SSL-Vision.

As SSL-Vision has been in continuous development since 2008 and is therefore relatively refined and stable inside its restrictions this work needs to fulfill the following requirements to be considered for adoption.

2.2.1 Accuracy

For fluid gameplay a sufficient position accuracy is necessary. Due to the small blob sizes and the necessity to annotate object positions in 3D space manual annotations are

less accurate than SSL-Vision and would require excessive effort. Therefore only object positions generated by SSL-Vision are available as ground truth for accuracy estimations. From the analysis of the SSL-Vision accuracy in static situations at recent RoboCup competitions (see Section 4.3.1) we can determine the best case accuracy achievable:

- Robot position: 3.47 mm
- Robot orientation: 7.20°
- Ball position: 3.50 mm

The butterfly pattern on top of the robot gives an estimate for the worst permissible accuracy, as larger deviations can lead to a misdetected orientation and subsequent misidentification of the robot. The distance between the side blobs is at the front 109.544 mm, at the side 91.924 mm and at the back 70 mm. The back and side distance become indistinguishable at $\frac{91.924\text{ mm}-70\text{ mm}}{2} = 10.96$ mm blob offset in the right direction. This can be used as an lower bound estimation where robots are starting to be misidentified. Therefore blob offsets should not exceed 10.96 mm.

An offset in the detections between cameras leads to issues with accepting ball passes as position changes and differences in the direction of ball movement result in misses. The accuracy of the proposed camera calibration in this work should therefore be better than SSL-Vision. As computed in Section 5.1.3, SSL-Vision has an average offset of 49.41 mm for robots and 21.38 mm for balls between camera perspectives.

2.2.2 Detection rate

The recall and precision of SSL robots and ball detections should not regress compared to SSL-Vision. From the analysis of the SSL-Vision accuracy in static situations at recent RoboCup competitions (see Section 4.3.1) we can determine the best case recall rate of SSL-Vision achievable in competition scenarios with 97.76% for robots and 95.51% for balls.

2.2.3 Robustness

The new approach is supposed to be capable of relaxing SSL-Vision constraints and therefore needs to be robust enough to support all fields currently using SSL-Vision. Apart from rule conformant fields like described in Section 1.1 this includes arbitrary field sizes with arbitrary additional field markings, gray carpets and wood colored field walls. To relax the biggest current constraint of SSL-Vision: uneven illuminated fields and temporal changes in lighting need to be supported as well.

2.2.4 Maintainability

As the camera is the only source of global position information stable robot and ball detections are a precondition for all gameplay. For the adoption of this work on fields outside of SSL-Vision constraints stable detections therefore have to be guaranteed. As the dataset contains only a limited field amount these guarantees cannot be definitely given. A user of the proposed solution therefore needs the capability to investigate, maintain and repair the problematic component. This requirement eliminates neural networks and other black box approaches for ball and robot detections as they cannot be easily investigated or repaired by design.

2.2.5 Usability

The long setup time required by SSL-Vision limits the usability during exhibitions and weekend competitions. The proposed solution therefore should reduce the required manual setup to a minimum. Configuration parameters should therefore be limited to the absolutely necessary and mostly determined automatically.

2.2.6 Latency

Most teams only have access to integrating sensors like odometric motion sensors or inertial measurement units which lead to accumulated errors over time. To minimize those errors minimal latency in the global position data provided to the teams is preferred. Therefore the proposed solution should not take more processing time than the current SSL-Vision, which has been measured around 7.40 ms on the NUC platform.

An upper bound of the available frame processing time is the camera frame rate. The cameras used at the RoboCup competition have a frame rate of 72 FPS, which results in a maximum total processing time of 13.89 ms.

3 Related work

As computer vision is a major research field many approaches and algorithms related to the visual understanding of the RoboCup Small Size League (SSL) field exist. Multiple approaches have previously been used to detect SSL robots and balls as the SSL uses external camera data since its founding. Section 3.1 discusses the camera calibration, robot and ball detection approaches previously used in the SSL. Multiple homography estimation and camera calibration approaches are discussed in Section 3.2. Related object and blob detection algorithms are discussed in Section 3.3. Blob assignment algorithms used in the proposed solution are presented in Section 3.4.

3.1 Previous approaches in the SSL

The SSL currently uses the SSL-Vision [6] software based on a modified CMVision blob detector [7]. A manually configured YUV lookup table (LUT) is used to classify pixels with blob colors. Based on a run-length encoding of the thresholded image contiguous color regions are generated. After multiple size, shape and position filters the colored blobs are used to construct robot and ball detections. While the CMVision algorithm only allows rectangular axis parallel class regions in the LUT SSL-Vision allows arbitrary class regions without the capability of assigning multiple classes for a single color. The usage of Graphics Processing Unit (GPU) resources has been investigated for the classification thresholding step during the development of SSL-Vision. Due to data transfer times exceeding Central Processing Unit (CPU) processing times a CPU based approach had been chosen. For the calibration of the intrinsic and extrinsic parameters of the camera, four outer line crossings of the visible camera extent have to be specified by the user, after which the ridge maxima are searched in the neighborhood of approximate line positions. The Levenberg-Marquardt algorithm [8] is then used to fit the camera model.

Prior to 2011 teams had to bring their own cameras and computer vision software. This approach was abandoned in 2011 due to increasing logistical issues with larger field sizes and greater field amounts. Color thresholding based approaches similar to SSL-Vision have already been documented in 2001 [9], usage of the CMVision algorithm already in 2004 [10]. Skuba used the CMVision algorithm in the Hue, Saturation, Value (HSV) color space in 2009 [11], Plasma-Z a similar HSV based LUT solution [12]. For the calibration Plasma-Z used dot patterns manually placed in the field edges. ZJUNlict used local region HSI color thresholding using a special pattern for threshold initialization [13] and manual field line edge point selection similar to SSL-Vision in combination with Tsais method

for camera calibration. B-Smart used a region of interest algorithm based on previous detections, frame difference and contrast regions and HSV based color segmentation based on the Fourier transformation [14].

Multiple alternative robot identification patterns have been used prior to SSL-Vision, like different blob shapes and colors by Plasma-Z, ZJUNlict and B-Smart in 2009. April tags [15] are an industry standard black white QR-code like marker and have been proposed for SSL-Vision in 2019 in an attempt to reduce the amount of colors needed to calibrate. This has not been adopted up to now due to remaining performance issues especially with the need to concurrently run the current color blob detector for ball detection. Due to the ball requiring a circular color blob detector no change to the robot pattern is being proposed in this work.

3.2 Camera calibration

As an accurate mapping of the image plane to the playing field is necessary for robot control the intrinsic and extrinsic parameters of the camera need to be calibrated. Traditional camera calibration algorithms like Zhengs approach [16] require point correspondences on different planes relative to the camera to eliminate singularities. As the only static feature available for self calibration on the SSL field are field lines and field line crossings. The frequent perpendicular viewing direction leads to singularities during camera calibration.

Human sport player tracking algorithms also need to determine the homography in sports videos for further processing and are therefore similar to the necessary self-calibration step. Similar to the camera calibration in this work lines are the dominant feature used. Different is the typically missing lens distortion determination and typical continuous camera attitude determination for each frame. Most approaches like Tari [17], Linnemann et al. [18] and Sun et al. [19] simplify the task by thresholding the image by dominant hue and subsequent morphological filter operations to estimate the field area. This method to estimate the field position is unsuitable for the SSL due to the existence of gray fields, which result in unstable hue values. Typical algorithms used by other RoboCup leagues like the Standard Platform League due to similar hue based field boundary estimations as documented e.g. by B-Human [20]. Farin et al. [21] propose a homography determination approach based on a field line detector without field boundary estimation by utilizing a ridge based white field line detector. As this approach does neither consider nor calibrate radial distortion or principal point offsets adaptations are necessary for usage in this work.

Thormählen et al. [22] propose the determination of distortion coefficients based on grouped line segments detected in a single frame. The proposed solution uses a derived approach thereof to calibrate the distortion and principal point.

Yu et al. [23] have compared multiple line segment detection and merging approaches. In this comparison the line segment merging algorithm proposed by Hamid et al. [24] has shown the best line segment merging capability for curved lines as necessary on fields with significant lens distortion.

As the predominant feature used for the camera calibration are the field lines detected line segments need to be assigned to field lines. Dedicated line segment matching algorithms like SMSLD [25] developed for matching two different images based on line segments use additional features like gradients in the image to construct a more expressive line descriptor. As these additional features are not present in the field line model these approaches are unsuited for the problem in this work.

3.3 Blob detection

Many state of the art approaches for object detection and localization are unsuitable due to the high resolution and framerate necessary combined with limited computation power. The comparatively small size of the ball (4.3cm) combined with the high maximal allowed velocity of 6.5 $\frac{\text{m}}{\text{s}}$ make a high resolution and frame rate necessary. For reference: the official SSL fields currently uses a Teledyne FLIR Blackfly S camera (72 FPS, 5 MP) in combination with an Intel NUC with i7-7567U processor to cover a playing field area of 9.6 m by 6.6 m. As the research focuses on more versatile and unique features than uniformly shaded blobs real-time capable color blob detection algorithms are a narrow field of research.

Other soccer leagues of the RoboCup also have the need for optical localization, object detection and tracking tasks but are limited to onboard sensors. The RoboCup Standard Platform League is especially interesting due to similar constraints regarding available computational power. Their current world champion B-Human [20] has thoroughly documented their dual object detection approach, consisting of a scan line based initial detector with neural network based ball classification and a low resolution single stage neural network detector. Due to the small size of the blobs in the SSL neither scan line nor low resolution images can be applied in this work. When using similar approaches to detect complete robots the necessary orientation and identification data would be still missing. As missing detections might only lead to rule violations of a single robot or the own team the required reliability is different to this work, where missing detections can endanger tournaments and lead to uncontrollability of robots. Neural network based approaches have also been investigated in the SSL for onboard vision detections [26] suffering the same performance and orientation data issues as in the Standard Platform League.

Mallat [27] proposed the usage of Haar-like features for pattern recognition, most commonly known from the Viola-Jones detector [28]. As the gradient dot product results in orthonormal negative and positive areas a Haar wavelet filter can be used for the blob detection. The area summation of this approach can be optimized with summed area tables [29]. The proposed gradient dot product generates Haar-like features

Circle detection algorithms have also been investigated due to the circular nature of the blobs. Rad et al. [30] use gradient pair vectors to detect circles by searching for pairs of gradient vectors opposing each other and cluster them according to their center point and distance. The gradient thresholding and accurate gradient directions required for this algorithm make it unsuitable for this work due to the low resolution of the color blobs. The generalized Hough transform [31] is a popular hypothesis based approach to detect circles. As this requires a clear threshold in the gradient magnitude which is difficult to determine due to the blurred response of curved balls a convolution based approach has been chosen instead.

Van de Weijer et al. [32] propose the color tensor as brightness invariant color gradient and use a circular Hough transform for color blob detection. The proposed color tensor has been evaluated in Section 5.2.7.1 but failed to achieve a sufficient blob response separation due to the smaller scale of the blobs.

The static camera perspective in the SSL enables the usage of background subtraction algorithms to generate regions of interest. Background subtraction methods use a background model to separate the image in background and foreground. Various real time capable background subtraction algorithms have been proposed [33]–[35]. As the proposed gradient dot product blob detector is fast and accurate enough when processing complete images no background subtraction algorithm is used in the proposed solution. Another issue of background subtraction algorithms is the embedding of long term still standing robots into the background model.

3.4 Blob assignment

To detect robots multiple neighboring blobs need to be combined into a robot. To facilitate a fast two dimensional neighbor search multiple tree data structures such as the Quadtree [36], range tree [37] and KD tree [38] have been proposed. Due to the simplicity of the implementation the KD tree has been chosen in the proposed solution.

For an adaptive blob color assignment blobs need to be classified depending on the color. As the amount of classes is known in advance k-Means clustering [39] is proposed to be used to assign the blob classes.

4 Proposed benchmarks

This chapter describes the datasets and measures proposed for the evaluation of different approaches and implementations. Section 4.1 gives an overview over the datasets used in the evaluation. Section 4.2 describes the hardware used during benchmarking. Section 4.3 introduces the measures used to evaluate the approaches.

4.1 Datasets

Prior to this work only the RoboCup 2019 and RoboCup 2022 datasets were publicly available. More datasets have been combined, recorded and preprocessed in the scope of this work to propose a new RoboCup Small Size League (SSL) vision test dataset collection. SSL-Vision only facilitates recording single images. As such only some particular static scenes as described in Section 4.1.2 were available prior to this work. The newly recorded video datasets presented in Section 4.1.3 feature moving scenes, unusual perspectives and variable illumination. All datasets used in this work are publicly available[1].

4.1.1 SSL game logs

The SSL game logs archive [40] contains all referee and vision messages sent over the network during past RoboCups. These vision messages contain timestamped robot and ball positions detected by SSL-Vision during the robot soccer games of past RoboCups. This dataset does not contain an image or video material. Used have been the game logs up to RoboCup 2023. As RoboCup 2021 was a virtual RoboCup within a simulation the game logs of RoboCup 2021 have been excluded.

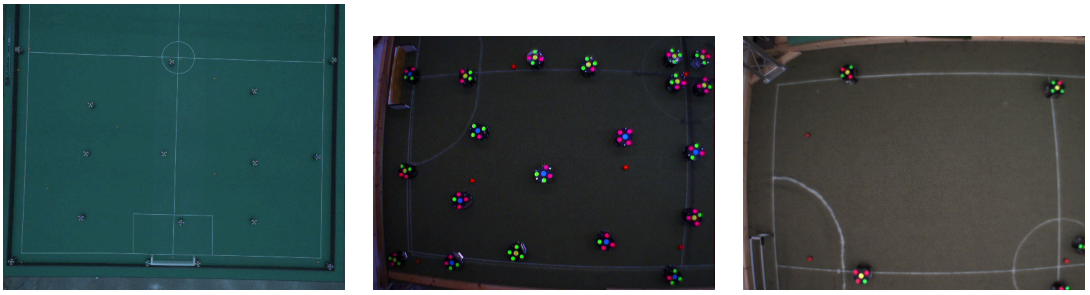
4.1.2 Preexisting image datasets

The preexisting image datasets depicted in Figure 4.1 have been recorded using SSL-Vision with lossless compression and feature static situations.

The **BUGA** dataset recorded at the Bundesgartenschau 2023 in Mannheim and depicted in Figure 4.1a features a dark environment with notably clear field lines and a gray carpet. The lens has a small pincushion distortion. The images used have been manually altered to remove humans standing on the field.

[1] `https://ssl.robocup.org/collected-data/`

(a) BUGA (b) KIKS Lab (c) RoboCup 2019



(d) RoboCup 2022 (e) TIGERs Lab (f) TIGERs Lab 2

Figure 4.1: Overview of scenes out of the preexisting image datasets showcasing the differences in extent, scale, carpet color, illumination, distortion and contrast.

The **KIKS Lab**oratory dataset depicted in Figure 4.1b consists of multiple situations in the overlap area between two cameras and features slightly overexposed blobs, very faint field lines, no blue robots and strong barrel distortion. The images used have been manually altered to remove tables standing on the other field side.

The **RoboCup 2019** dataset depicted in Figure 4.1c features a very dark environment with a low signal to noise ratio and small pincushion distortion. The images used have been manually altered to remove humans working on the field.

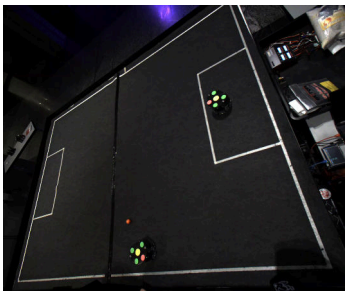
The **RoboCup 2022** dataset depicted in Figure 4.1d consists of multiple stationary situations viewed from two cameras and features reduced color contrast, small pincushion distortion and a blue-heavy white balance.

The **TIGERs Lab**oratory dataset depicted in Figure 4.1e consists of multiple stationary situations viewed from four cameras and features oversaturated colors, occational specular reflection glare of the robot top and barrel distortion. The **TIGERs Lab 2** dataset depicted in Figure 4.1f is similar to the TIGERs Lab dataset but has been separated due to a different camera setup.

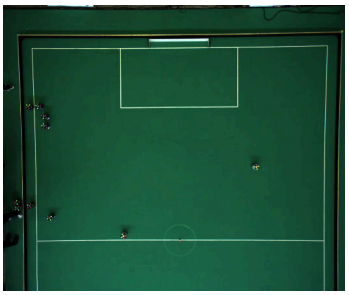
4.1.3 Video datasets

All video datasets have been recorded as part of this work. The videos have been recorded with the same cameras used at the RoboCup competition since 2018, Teledyne FLIR Blackfly S cameras. The lens used has a small pincushion distortion. The videos have been encoded at roughly 30 FPS with a h264 intel hardware encoder with quarter color resolution and 3500 $\frac{\text{kBit}}{\text{s}}$ data rate. No camera synchronization has been done on fields with multiple camera perspectives.

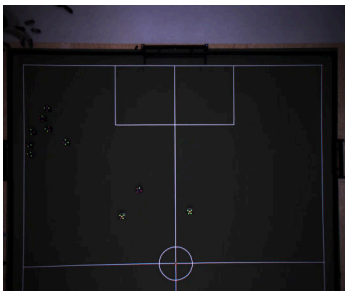
The **NOTG** dataset recorded at the “Night of the Graduates” gala in Mannheim and depicted in Figure 4.2a features a perspective from the edge of the field. The advantage of this perspective is the simpler setup compared to a camera over the center of the field. This simpler setup is crucial for enabling the usage of global vision data at exhibitions



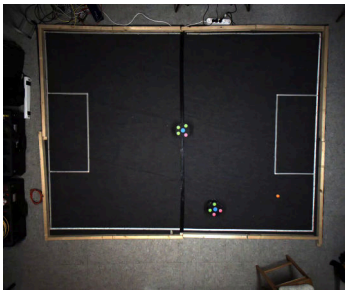
(a) NOTG



(b) RoboCup German Open



(c) Schubert Crailsheim



(d) TIGERs Weekend

Figure 4.2: Overview of scenes out of the contributed video datasets showcasing the differences in extent, scale, carpet color, illumination and perspective.

and fairs but creates additional challenges for the blob detection due to the uneven ground resolution. It's the only dataset where the ground plane is not parallel to the image plane which enables the automated field line based camera height calibration. The outer field line edge outside of the camera view poses an additional challenge for the geometry calibration.

The **RoboCup German Open** dataset depicted in Figure 4.2b consists of multiple complete ssl games from two cameras. It features some slight long term illumination changes through natural sunlight influences and camera gamma, gain and white balance changes between games.

The **Schubert Crailsheim** dataset recorded at a local tournament in Crailsheim and depicted in Figure 4.2c consists of multiple complete ssl games from two cameras. It features very uneven illumination over the area of the field and power frequency flickering lighting and camera parameter changes between games. One ssl game has been recorded with a doubled video resolution to capture the full camera resolution with the drawback of a lower FPS (around 22 FPS) count due to hardware encoder inadequateness.

The **TIGERs Weekend** dataset depicted in Figure 4.2d consists of multiple situations of robots driving and interacting with the ball. It features a comparatively high ground resolution of roughly 2 mm/px, flickering lighting and bright field walls directly neighboring the outer field lines.

4.2 Hardware

The real time nature of the task requires the adherence to frame time deadlines on diverse hardware. Additionally implementation details of Open Compute Language (OpenCL) runtimes can differ between different vendors, which necessitates testing the functionality of the OpenCL code on different platforms. Measures with processing time impact will therefore be evaluated on the following computer systems:

- Intel NUC with Intel i7-7560U processor and Intel Iris Plus Graphics 650 integrated Graphics Processing Unit (iGPU).

- Laptop with an AMD Ryzen 3700X processor and a NVIDIA GeForce RTX 2060 Mobile dedicated Graphics Processing Unit (dGPU).

These NUC computers are used at the RoboCup competition and various other local tournaments since 2018 and serve therefore as an important reference platform. Their integrated graphics card has a comparatively low processing power, but features unified memory with the Central Processing Unit (CPU). This enables zero-copy memory transfer between Graphics Processing Unit (GPU) and CPU.

The laptop is used to test the performance on a dGPU with comparatively higher processing power. Due to using separate memory data needs to be transferred to the dGPU. The CPU of the laptop has been locked at 2.2 GHz to eliminate clock boosting variances.

4.3 Performance measures

Prior to this work the performance of the geometry calibration and object detection has not been measured in the SSL. Performance measures therefore need to be introduced to evaluate the performance of the proposed solution compared to SSL-Vision. As the goal is are approaches that work reasonably well on all fields, measures will be averaged over each dataset with each dataset being weighted equally.

4.3.1 RoboCup SSL-Vision position accuracy

To determine the best-case detection and localization accuracy of the existing ssl-vision all game log files [40] from RoboCup 2016 to 2023 (excluding the virtual RoboCup 2021) have been analyzed. As measurement of the localization accuracy the standard deviation of the detected position and orientation during HALT phases (where no autonomous movement is permitted) has been chosen. Data samples inside the grace period for breaking [5] of 2 seconds have been removed. Every object deviating more than 10 cm from the average position or rotating more than 90° have been filtered for this camera per halt phase due to assumed human intervention. The average localization standard deviation of SSL-Vision is 3.47 mm for yellow robots, 3.48 mm for blue robots and 3.50 mm for the ball. The average orientation standard deviation is 7.1° for yellow robots and 7.3° for blue robots.

As measurement for the detection reliability the proportion of frames where at least once detected objects have been detected during HALT phases has been chosen. Data samples inside the grace period for breaking of 2 seconds have been removed. If any object deviated more than 10cm from the average position or rotated more than 90° during the HALT phase all data points of the corresponding HALT phase have been filtered due to potential occlusion from human intervention. Yellow bots have been detected with a reliability of 97.55%, blue bots with 97.97% and the ball with 95.51%.

As this data is from competitive matches during RoboCup after multiple setup days this data is considered a best-case scenario.

4.3.2 Field line point overlap

A metric to determine how good a camera calibration matches the physical field is necessary. Each pixel classified as field line by one of the algorithms described in Section 5.1.4 can be further classified for a given geometry calibration in points being part of a field line and points being outside of a field line. The ratio of field line points inside the field line model can then be used as a metric to compare the overlap of different calibrations. It can also be used to evaluate line pixel classification algorithms to determine the amount of false positives. A higher ratio corresponds to a better matching calibration model.

4.3.3 Overlap offset

The aim of the geometry calibration is the minimization of the offset between the detected and physical position of balls and robots. The generation of ground truth position data is difficult due to the limited precision of manual object placement and measurement. Therefore the precision of the calibrated geometry is determined on datasets with multiple cameras in the overlap area of the camera viewports. The average offset in the detected positions by SSL-Vision between cameras can be used as metric for calibration accuracy. A lower average offset indicates a more accurate geometry calibration.

This measure is limited to the following datasets with stationary robots and balls inside the overlap area between multiple cameras: KIKS Lab, TIGERs Lab, TIGERs Lab 2 and RoboCup 2022. The position detection accuracy of SSL-Vision is a lower bound for the attainable average offset.

4.3.4 Peak percentile ratio

As the main data reduction step the blob detector should minimize the amount of responses. To be able to define stable thresholds for blob classification a large difference between a response for a blob and the response for the rest of the image is necessary. The MOSSE object tracking filter [41] introduces the peak-to-sidelobe ratio as a metric of the detection quality. As the MOSSE filter is a single object tracker and no ground truth with perfect detection rate is available in this work the peak-to-sidelobe ratio needs adaptations.

For each robot and ball detected by SSL-Vision the corresponding blob locations are generated. The data point with the greatest response inside the blob area is selected as the peak of the specific blob. Responses which are not 4-way local peaks are excluded. The average peak response r_{blobavg} of all blobs inside the frame is then used as global peak.

As the majority of the image area is uniform colored carpet using the mean and standard deviation as values for the sidelobe is not informative. For a common visible extent of half a division A field 6.3m × 9.6m the possible area fraction of 22 robot blobs and one ball is 0.26%, for 22 robots and one ball is 0.93% and area fraction of all field lines (18 mm width) is 1.31%. As these values are approximations due to different camera extents, field sizes and field lines the 99th percentile $r_{99\%}$ of all responses has been chosen as a substitute for the sidelobe.

To normalize the results and add the capability of handling $r_{99\%} = 0$ the resulting Peak Percentile Ratio (PPR) m_{PPR} has been defined as follows:

$$m_{\text{PPR}} = \frac{r_{\text{blobavg}}}{|r_{99\%}| + |r_{\text{blobavg}}|}. \tag{4.1}$$

A PPR of 1.0 indicates a perfect separation, a value of 0.5 no separation between blobs and the field response and a value of less than 0.5 negative separation between blobs and the 99th percentile.

4.3.5 Blob position offset

A correct attribution of the butterfly blob pattern on top of the robots requires a position precision of better than 10.96 mm to prevent misidentification. To estimate the position offset the blob locations for each robot and ball detected by SSL-Vision are generated and compared to the peak responses of the blob detection algorithm inside the blob area. The distance of the peak to the SSL-Vision detection is then used as blob offset. The average

and standard deviation of these blob offsets is then reported as the blob position offset. Additionally for each robot the blob offsets are averaged and reported separately.

As only responses inside the predicted blob area are searched no offsets greater than 25 mm can be detected. The accuracy of SSL-Vision is a lower bound for the achievable position offset.

4.3.6 Recall and precision

The main task of this work is the correct detection of all robots and balls inside the frame. Recall and precision are two popular metrics to determine the accuracy of object detection algorithms.

Each image dataset has manual annotations about visible objects as ground truth for this metric. As the video datasets are too large to annotate each frame and robots and balls might leave the camera extent the visibility of objects is estimated. Each object is assigned a numerical value starting from 0. For each frame in which this object was detected by SSL-Vision or the proposed solution the value is incremented by 1 up to a maximal value of 15, corresponding to 0.5 seconds visibility. For each frame absent in SSL-Vision and the proposed solution the value is decreased by 1 up to a minimal value of -15. Each object with a score greater than 0 is assumed to be present.

According to the ground truth each detection is classified as true positive TP or false positive FP and missing detections as false negative FN. The precision $p = \frac{TP}{TP+FP}$ describes how many of the reported objects were actually in the frame. The recall $r = \frac{TP}{TP+FN}$ describes how many of the objects in the frame were detected.

The precision of the proposed solution should not fall significantly below the precision of SSL-Vision (around 95%). The approach with the greater recall is the better approach. As the ground truth of video datasets is estimated partially based on the proposed solution itself the SSL-Vision reference result changes.

5 Vision based approaches and performance

To simplify processing and report detections in phyiscal field positions an accurate mapping of the image plane to the playing field is necessary. Section 5.1 discusses the systematic issues with the semi-automatic calibration method used by SSL-Vision and explores other calibration approaches that minimize these systematic issues.

In section Section 5.2 various preprocessing steps and blob detection approaches are analyzed. A novel gradient dot product blob detector is proposed to detect and identify robots and balls as the properties of the ball and the identification of the robot require a blob detector.

Approaches for the classification and assignment of the detected blobs to balls and robots are discussed in Section 5.3. The proposed algorithm assigns the blobs based on relative position and classifies the blobs according to k-Means color clustering. As no prior region of interest filter is used due to the sufficient runtime and detection performance of the proposed approach this approach can be primarily classified as an object tracking by detection approach. To recover partial robot detections some information from prior frames is used.

5.1 Geometry calibration

The software needs to know intrinsic and extrinsic parameters of the camera relative to the playing field to be able to provide global position information. Knowledge about these parameters also helps with the detection of balls and robots as it opens the possibility of geometry normalization and size expectations.

Field lines are the always available feature on the playing field usable for camera calibration. Due to the coplanar nature of the field lines and the typically perpendicular viewing direction of the camera traditional point-to-perspective and camera calibration algorithms cannot determine the camera parameters.

5.1.1 Problem definition

Given an image of an small size league field, information about the field lines present and information on the expected extent (e.g. which half of the field) of the image determine the intrinsic and extrinsic calibration of the camera with minimal deviation from the field lines at goals, field boundaries and camera overlap areas. The height of the camera above the field is also given for cameras oriented perpendicular to the field.

5.1.2 SSL-Vision camera model

The SSL-Vision uses a camera model similar to the model proposed by Tsai [42]. The camera model of SSL-Vision has been reused in this work for compatibility reasons. Continued use of this camera model enables a better comparability with SSL-Vision during blob detection. Since robots frequently interrupt the line of sight between the camera and the ball multiple teams like ER-Force [43] use the published SSL-Vision camera model to determine areas of such occlusion effects. The camera model is also used by teams to determine the height of the ball above the field. Reusing the SSL-Vision camera model therefore eases the adaption by the teams and prevents changes to the network protocol.

The camera model allows for the translation of image points in the camera coordinate system C to the field coordinate system F and vice versa. C is a left handed coordinate system anchored at the principal point in the image plane with rightwards x-axis, downwards y-axis and backwards z-axis. F is a right handed coordinate system anchored at the field center point with the x-axis in goal direction, y-axis on the halfway line and upward z-axis.

Every image point i is normalized with the principal point p and the focal length f:

$$n = \frac{i - p}{f}. \tag{5.1}$$

The normalized point n is then undistorted with a single parameter k_2 radial Brown–Conrady [44] distortion model (with $\cdot$ denoting the dot product):

$$u = \begin{pmatrix} u_x \\ u_y \end{pmatrix} = n(1 + k_2(n \cdot n)). \tag{5.2}$$

Using the three dimensional camera ray $r_i = \begin{pmatrix} u_x \\ u_y \\ 1 \end{pmatrix}$ and the quaternion q describing the rotation from the camera coordinate system to the field coordinate system the camera ray r_f in field coordinates can be constructed (with $\star$ denoting the quaternion rotation):

$$r_f = \begin{pmatrix} r_{f_x} \\ r_{f_y} \\ r_{f_z} \end{pmatrix} = q \star r_i. \tag{5.3}$$

In the case r_{f_z} is 0 or negative the image point is at or above the camera height in the field coordinate system. Since the camera is always positioned above the field the special case of looking above the horizon will not be considered in the following plane intersection. Given the camera position in field coordinates c and the assumed height h over the field plane of the object corresponding to the image point the position in field coordinates o can be determined:

$$o = \frac{h - c_z}{r_{f_z}} r_f + c. \tag{5.4}$$

SSL-Vision uses the following closed form solution for the reverse mapping of the radial distortion model (Equation (5.2)) for $k \geq 0$:

$$
\begin{aligned}
r &= \sqrt{u \cdot u} \\
b &= \sqrt[3]{-9rk^2 + k\sqrt{12k + (9kr)^2}} \\
n &= u\frac{\frac{\sqrt[3]{\frac{2}{3}}}{b} - \frac{b}{k\sqrt[3]{18}}}{r}.
\end{aligned}
\tag{5.5}
$$

Since the cameras used at the RoboCup competition since 2017 feature pincushion distortion and as such $k < 0$ an iterative reverse radial distortion algorithm derived from the algorithm proposed by Peter Abeles [45] is used instead in this work:

$n \leftarrow u$;
for 8 iterations **do**
    $n \leftarrow \frac{u}{1+k(n \cdot n)}$;

A fixed iteration amount has been adopted instead of an convergence criteria to facilitate compiler and GPU optimizations through parallelization. The iterative solution has the additional advantage of also working with more complex distortion models, allowing for an easy replacement of the distortion model in the future.

5.1.3 Unbound parameters with field line point calibration and perpendicular viewing direction

To provide optimal resolution over the complete extent most cameras are placed directly above the center point of their field extent, resulting in a viewing direction perpendicular to the field. This results in unbound model parameters when calibrating all parameters simultaneously based on field line points.

To represent the restriction $C_z = F_z$ the quaternion q rotation is replaced by a sign swap (to accommodate the change between C as a left hand coordinate system and F as a right hand coordinate system) and a simple 2D rotation α in the following equations:

$$
r_f = \begin{pmatrix} r_{i_x} \cos\alpha - r_{i_y} \sin\alpha \\ -r_{i_x} \sin\alpha - r_{i_y} \cos\alpha \\ r_{i_z} \end{pmatrix}. \tag{5.6}
$$

The nonlinearities introduced by Equation (5.2) cannot be used as stabilizing factor in the focal length calibration as the distortion model is only an approximation of the actual lens distortion. The following proof therefore assumes $u = n$. All field line points are on the ground $h = 0$. Under these circumstances the focal length f is in direct relation to the camera height c_z whereby both parameters become unstable in a field line point based calibration:

$$
\begin{aligned}
o &= (h - c_z) \begin{pmatrix} \frac{i_x - p_x}{f} \cos\alpha - \frac{i_y - p_y}{f} \sin\alpha \\ -\frac{i_x - p_x}{f} \sin\alpha - \frac{i_y - p_y}{f} \cos\alpha \\ 1 \end{pmatrix} + c \\
&= \begin{pmatrix} -\frac{c_z}{f}((i_x - p_x)\cos\alpha - (i_y - p_y)\sin\alpha) + c_x \\ -\frac{c_z}{f}(-(i_x - p_x)\sin\alpha - (i_y - p_y)\cos\alpha) + c_y \\ 0 \end{pmatrix}.
\end{aligned}
\tag{5.7}
$$

As c_z and f can both be scaled by an arbitrary factor $x \neq 0$ without influencing the result of Equation (5.7) both parameters are unbound ($\frac{xc_z}{xf} = \frac{c_z}{f}$). SSL-Vision uses a manually configured camera height instead of a self calibrated camera height to prevent this instability. The proposed solution also allows a manually entered camera height in case of a perpendicular camera viewing direction for the same reason.

In the case of a perpendicular looking camera $C_z = F_z$ and a radial distortion of the camera close to zero $d = 0$ the principal point p_x, p_y and camera position c_x, c_y become unbound as well in a field line point $h = 0$ based calibration:

$$\begin{aligned} o &= (h - c_z) \begin{pmatrix} \frac{i_x - p_x}{f} \cos\alpha - \frac{i_y - p_y}{f} \sin\alpha \\ -\frac{i_x - p_x}{f} \sin\alpha - \frac{i_y - p_y}{f} \cos\alpha \\ 1 \end{pmatrix} + c \\ &= -\frac{c_z}{f} \begin{pmatrix} i_x \cos\alpha - i_y \sin\alpha \\ -i_x \sin\alpha - i_y \cos\alpha \\ \frac{1}{f} \end{pmatrix} + \frac{c_z}{f} \begin{pmatrix} p_x \cos\alpha - p_y \sin\alpha \\ -p_x \sin\alpha - p_y \cos\alpha \\ 0 \end{pmatrix} + c \\ &= -\frac{c_z}{f} \begin{pmatrix} i_x \cos\alpha - i_y \sin\alpha \\ -i_x \sin\alpha - i_y \cos\alpha \\ \frac{1}{f} \end{pmatrix} + \begin{pmatrix} \frac{c_z}{f}(p_x \cos\alpha - p_y \sin\alpha) + \frac{f}{c_z} c_x \\ \frac{c_z}{f}(-p_x \sin\alpha - p_y \cos\alpha) + \frac{f}{c_z} c_y \\ c_z \end{pmatrix}. \end{aligned} \tag{5.8}$$

As is visible in Equation (5.8) p_x, p_y and c_x, c_y can be changed independent of the image point i and field point o and are therefore unbound. SSL-Vision does not address this unboundedness in its calibration, which results in off-center principal points and camera positions. This results in problems on multi camera fields where there are large discrepancies in detected robot and ball positions in overlapping areas. This is shown in Table 5.1

Field	Robot offset/mm ↓	Ball offset/mm ↓	Overlap ratio ↑
KIKS Lab	40.07	37.49	0.1544
TIGERs Lab	95.58	21.18	0.4615
TIGERs Lab 2	12.55	11.35	0.4555
RoboCup 2022	49.45	15.49	0.3162
Average	**49.41**	**21.38**	**0.3469**

Table 5.1: Average offset of robot and ball detections among camera perspectives with SSL-Vision calibration showing the greater offset of the elevated robots. This indicates a camera position offset.

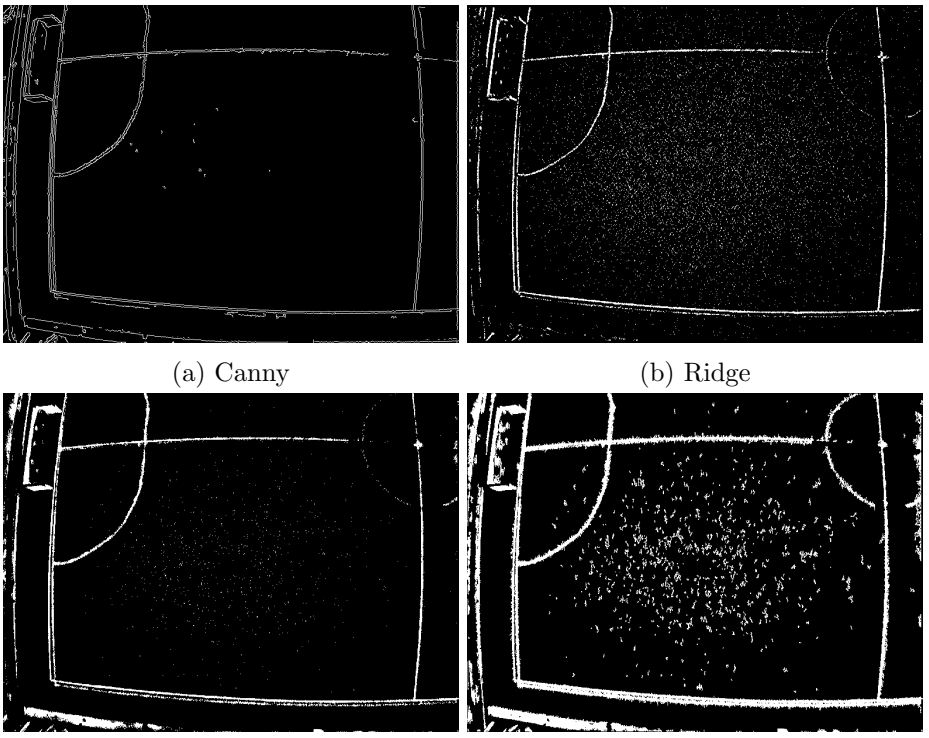
with the average offset of robots and balls between cameras on all datasets with static situations on multiple cameras. The average robot offset is significantly larger than the average ball offset on three of the four fields indicating calibration issues with increasing height through offset principal and camera position offsets.

5.1.4 Field line pixel classification

Most sports field homography estimation algorithms [17]–[19] generate a hue based field mask prior to the field line detection. This field extent estimation simplifies further processing through the elimination of clutter outside the field. Gray carpets prevent the usage of the dominant hue for the field extent estimation (see Figure 5.5b). Fields with uneven illumination or wooden field walls prevent the usage of brightness information. An

attempted thresholding in the color plane of the YUV color space did not manage to separate gray carpets from the surrounding black walls. Therefore no field extent estimation is available to limit the area of potential field line pixel candidates.

For all calibration approaches using field line information a field line pixel classification is necessary. The field lines and goals are the only white objects on the field and can be assumed as the only larger features on the carpet during the calibration. As no field extent estimation is available misdetections at other edges and ridges have to be minimized to aid further calibration steps.



(a) Canny (b) Ridge

(c) AdaMedian Otsu (d) AdaMedian hysteresis

Figure 5.1: Scene from the TIGERs Lab dataset thresholded with varying line pixel classification algorithms showcasing the differences in detection rate and noise.

Four different approaches have been evaluated for the field line pixel classification. The Canny edge detector [46] has been tested as a standard widespread edge detection algorithm. As visible in Figure 5.1a the Canny edge detector detects most field lines but also all other major edges like the field wall and goal post edges. The ridge detector proposed by Farin et al. [21] classifies the pixel at the position x, y as field line if the brightness $Y(x, y)$ is higher by more than a threshold t compared to pixels in a distance of the estimated half field line width h:

$$\begin{aligned}&((Y(x,y) - Y(x-h,y) > t) \wedge (Y(x,y) - Y(x+h,y) > t))\\ \vee&((Y(x,y) - Y(x,y-h) > t) \wedge (Y(x,y) - Y(x,y+h) > t)).\end{aligned} \tag{5.9}$$

The additional structure matrix filter has not been implemented due to the negligible impact of textured regions with typical RoboCup Small Size League (SSL) scenes. Due to the static threshold a tradeoff between field wall ridges, similar line like structures and faint field lines has to be done. As visible in Figure 5.1b the ridge detector detects most field lines and minimizes responses at larger bright areas and other edges. To improve the capability to detect faint field lines an adaptive median filter has been evaluated as a generalization of the axis parallel line preferring ridge detector. The adaptive median

is the subtraction of the grayscale image from the median blurred image with a square filter size double of the estimated field line width. This is supposed to have only field line sized bright spots as only positive values remaining. Negative values are clamped to 0 to facilitate a automatic threshold determination with the Otsu [47] algorithm. As visible in Figure 5.1c some faint lines are still undetected while the field wall produces a stronger response compared to the ridge detector. To detect missing faint lines a modified adaptive median variant using a hysteresis approach similar to the Canny edge detector has been evaluated with a pixel being classified as line pixel if it has a strong response or a weak response and is connected to a pixel with a strong response. As visible in Figure 5.1d this lead to all field lines being detected although the clutter and jagged field line edges make this approach unsuitable for further line segment detection.

Algorithm	Robot offset/mm ↓	Ball offset/mm ↓	Overlap ratio ↑
Canny	154.16	124.90	0.1572
Ridge	74.72	52.13	**0.2967**
AdaMedian Otsu	85.35	53.07	0.2398
AdaMedian hysteresis	**65.02**	**34.52**	0.1658

Table 5.2: Comparison of different line pixel classification algorithms with line based distortion calibration and four point extrinsic calibration showing the least offsets with the AdaMedian hysteresis and the best overlap with the Ridge detector.

Table 5.2 shows the results achieved with the different line pixel detection algorithms. The good result of the AdaMedian hysteresis approach has been discarded as a fluke due to the clutter and jagged edges causing issues with the line segment detection and grouping as visible in Figure 5.2. The ridge based detection remains as the best approach with the second best robot and ball offset and the highest overlap score.

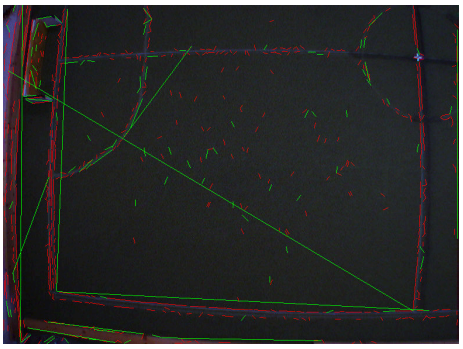


Figure 5.2: Scene from the TIGERs Lab dataset showcasing the detected line segments based on the AdaMedian hysteresis line pixel classifier marked in red and grouped lines marked in green. It shows the issues in the line segment detection and grouping caused by the jagged line border.

5.1.5 Direct calibration

A rough estimation of camera parameters and the position can be done based on the size of the field extent to observe and camera height if given. Minimizing the distance of the line model to the next detected field line pixel is a direct approach to calibrate the camera parameters. This is similar to the local search approach described by Okuma [48] for the initial model fitting of the homography estimation. For this the field line model has been sampled at a constant step size every 10 cm. The distance of the sampled points in the image plane to the next field line pixel is used as the error metric for the model fitting with the Levenberg-Marquardt algorithm [8].

This approach failed on most fields due to undetected field lines and detected lines outside the field leading to wrong local minima in the optimization. This did not improve when calibration the principal point and distortion separately with the method described in Section 5.1.7 or using a four point calibration beforehand.

5.1.6 Line segment detection

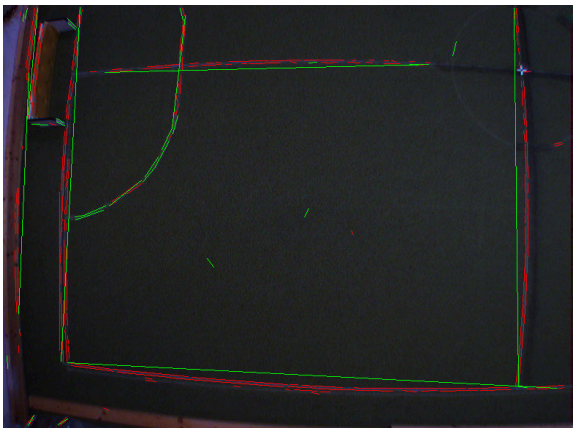


Figure 5.3: Scene from the TIGERs Lab dataset showcasing the detected line segments based on ridge line pixel classifier results marked in red and grouped lines marked in green. It shows the successful grouping of all straight field lines despite the noticeable barrel distortion.

For further processing the lines in the image need to be extracted. Using the line segment detector included in OpenCV [49] on a field line pixel thresholded image delivers a line segment for each side of the lines. Due to the distortion curvature the detected line segments need to be grouped to accurately represent a single field line.

Hamid et al. [24] group all line segments with an orientation angle and a horizontal and vertical distance between segment end points smaller than a threshold. To minimize processing time the line segments are processed in the order largest to shortest with grouped line segments being removed from further processing. Due to both sides of field lines being detected as line segments and the neighborhood condition based on the end points line segments are insufficiently grouped with this approach. The grouping approach of Thormählen et al. [22] has been adopted in this work as the approach introduces a perpendicular distance limit which allows for a large end point distance threshold. Thormählen proposes the usage of an orientation angle and a radial segment end point and perpendicular distance threshold as well. Segments shorter than 10 pixels are discarded due to their angular instability leading to erroneous results. Line segments are processed in arbitrary order with no removal of line segments from the processing queue. Line segments are grouped if they have an absolute orientation angle difference $\leq 3°$, a perpendicular distance < 10 px and a distance between the nearest endpoints < 200 px. This results in a line segment grouping as visible in Figure 5.3 where all straight field lines are successfully merged.

5.1.7 Intrinsic model calibration

Thormählen et al. [22] propose the calibration of the distortion by utilizing straight lines from a single image. The residual distance of each line pixel relative to the linear regression of the corresponding grouped line segment is minimized with the Levenberg-Marquardt algorithm [8]. As most field lines are straight lines the line segments grouped by Section 5.1.6 can be used as input for the algorithm. The approach can be modified to calibrate the principal point as well as the radial distortion is dependent on the principal point location.

Although the radial distortion is dependent on the focal length as well due to the prior normalization in Equation (5.1) the focal length cannot be calibrated with this approach as the distortion model is only a rough approximation of the true lens distortion. The focal length can be calibrated more precisely during the extrinsic parameter calibration. As the distortion is dependent on the focal length this intrinsic calibration has to be repeated each time the focal length is changed.

Filter	Robot offset/mm ↓	Ball offset/mm ↓	Overlap ratio ↑
no filter	148.94	128.16	0.1819
≥ 2 segments	140.77	125.15	0.2234
≥ 0.5 image height	**74.72**	**52.13**	0.2967
only distortion	82.65	65.31	**0.3159**

Table 5.3: Comparison of line filters for the line based distortion calibration with four point extrinsic calibration showing the least offsets with a length based line filter and the best field line overlap without principal point calibration.

As visible in Table 5.3 using all line segments leads to a worse result due to short clutter lines and curved lines like the center circle lead to an underestimation of the distortion coefficient. Filtering all line segment groups with less than 2 segments removes clutter improves the results slightly. Using only grouped line segments longer than half of the smaller image side limits the result to well detected long field lines and leads to the best estimation of the distortion. The worse offset without principal point calibration (with ≥ 0.5 height filter) show the necessity of the principal point calibration.

5.1.8 Homography calibration

For the mapping of the image coordinate system to the field coordinate system the homography consisting of focal length, camera position and camera orientation remains to be determined. Due to undetected faint field lines and too many detected lines outside the field all attempts at determining the homography based on the outer field lines failed. Therefore four outer edge point coordinates provided by the user are used to determine the homography by minimizing the position offset to the model points in the image plane with the Levenberg-Marquardt [8] optimizer. As the distortion coefficient is dependent on the focal length this calibration step is done alternating with the intrinsic calibration described in Section 5.1.7 until convergence occurs. Usage of user provided edge coordinates guarantees a stable calibration independent of detected lines and symmetry ambiguity.

5.1.9 Discussion

Field	Robot offset/mm ↓		Ball offset/mm ↓		Overlap ratio ↑	
	Proposed	SSL-Vision	Proposed	SSL-Vision	Proposed	SSL-Vision
KIKS Lab	153.33	**40.07**	145.16	**37.49**	0.0349	**0.1544**
TIGERs Lab	111.57	**95.58**	33.58	**21.18**	0.3349	**0.4615**
TIGERs Lab 2	19.75	**12.55**	19.49	**11.35**	**0.4750**	0.4555
RoboCup 2022	**14.23**	49.45	**10.27**	15.49	**0.3419**	0.3162
Average	74.72	**49.41**	52.13	**21.38**	0.2967	**0.3469**

Table 5.4: Comparison of the overlap ratio and average offset of robot and ball detections among camera perspectives showcasing the on average better results of SSL-Vision compared to the proposed solution.

As visible in Table 5.4 the proposed solution achieves an overall worse result than SSL-Vision due to the significantly worse result in the KIKS Lab and TIGERs Lab dataset. The RoboCup 2022 dataset shows how the calibration approach has the potential to produce a more accurate calibration. Further investigation is necessary if the calibration quality relative to SSL-Vision correlates with the amount of distortion as RoboCup 2022 is simultaneously the dataset with the least amount of distortion. The decreased accuracy with increasing object height is not as severe with the proposed solution compared to SSL-Vision, indicating a better principal point and position calibration.

Further improvements in the calibration accuracy can be expected with the additional usage of detected field line information during the homography estimation. This would also allow to prioritize the accuracy of the calibration in sensitive areas such as the camera overlap areas and the goals at the cost of accuracy in other areas. Another possibility is the usage of detected robots and balls on multi camera fields in the overlap area to further improve the calibration accuracy.

5.2 Color blob detection

On the playing field the ball and the robot butterfly pattern blobs are unique in both shape and color. As the ball is a circular blob and the robot orientation and identity is encoded into the butterfly blob pattern on top a circular color blob detector is necessary to successfully detect balls and robots. Potential rule changes replacing the robot pattern to improve robot detection rate and accuracy won't change this requirement as the ball will always be a circular blob.

For the selection of default parameters and the development of the proposed blob detection approach only the datasets KIKS, TIGERs Lab, RoboCup 2022, NOTG and TIGERs Weekend were used. Unless otherwise indicated all benchmarks use the following processing approaches and parameters:

- Resampling to the average pixel distance as described in Section 5.2.4
- dRGB color space as described in Section 5.2.5
- Color gradient dot product as described in Section 5.2.7.1
- SAT sum minimum blob score as described in Section 5.2.7.3
- Gradient scaling of $\frac{1}{3}\left\lceil\frac{25\text{ mm}}{\text{field scale}}\right\rceil$ as described in Section 5.2.7.4
- Square peak interpolation as described in Section 5.2.9

5.2.1 Problem definition

Given are an image of a RoboCup small size league field situation, a camera model and previous robot and ball detections. Find all orange balls, yellow and blue team identifying central robot blobs and green and pink numbering and orientation side blobs present in the image while minimizing the detected blob position offset, adhering to processing time limitations and maximizing the Peak Percentile Ratio (PPR).

5.2.2 SSL-Vision

SSL-Vision [6] uses a modified variant of the CMVision algorithm [7]. The pixels are classified using a color lookup table (LUT). Different from the CMVision algorithm the LUT is not limited to axis-aligned bounding boxes but uses a full color LUT. Afterwards the classifications are compressed utilizing run-length encoding. Regions are formed from

adjacent runs of the same classification. The average position of all pixels contributing to the region is then used as the center point of the region.

The immediate data reduction through the run-length encoding after the single LUT pass makes the algorithm used one of the fastest possible algorithms for color detection suitable for CPU processing. Due to the neighboring and often time overlapping color classes (although specifically chosen to be as distant as possible in most color spaces) SSL-Vision requires precise calibration and is unusable under variable lighting.

5.2.3 Bayer filter subsampling

The Bayer filter [50] is the typical method used for color detection in digital color cameras by applying red, green and blue color filters in an interleaved pattern prior to the sensor. As each pixel only contains one color channel a demosaicing step is required to generate a Red, Green, Blue (RGB) image. Due to the large area (9.6×6.6 m) that each Teledyne Blackfly S color camera used by the SSL needs to cover under typical circumstances the full resolution (2448×2048 px) is required ($3.92 \frac{\text{mm}}{\text{px}}$ in a best case scenario). Under these circumstances the best frame rate (72 FPS) can only be achieved using 8-bit Bayer filter data according to the vendor [51]. This work therefore needs to contain a demosaicing step.

Traditional Bayer filter demosaicing steps use various interpolation methods to preserve the image resolution. The processing time constraints make this data triplication undesirable. As the blob detection relies more on the color than the luminance (see Section 5.2.5) the full color resolution is more important. Therefore combining one RGGB block into a single pixel while averaging the green color channels has been investigated as demosaicing step. This leads to a loss of the second green channel and loss of resolution with a persistent subpixel red and blue color position offset. As many fast demosaicing algorithms struggle with color fringing [52] and chromatic aberration is a similar influence the color offset the subpixel offset has been ignored. The Bayer subsampling preserves the gradient strength and quarters the pixel count and data amount that need to be processed in further processing steps.

Bayer handling	Blob acc./mm ↓	PPR ↑	Frame time/ms ↓	
			Average	Schubert
Interpolated	$\mathbf{6.77 \pm 3.45}$	0.9669	131.99	14.38
Subsampled	9.88 ± 4.26	**0.9765**	**35.62**	**4.06**

Table 5.5: Blob position accuracy and PPR with Bayer interpolation and subsampling showing violated frame time constraints with interpolated images while producing a better position accuracy.

Table 5.5 shows the position accuracy and processing time on the NUC platform for all datasets without Bayer subsampling at recording time, which are all image datasets and a subset of the Schubert Crailsheim dataset. As lazy buffer allocation and OpenCL kernel compilation times significantly impact the processing time in single image situations the Schubert Crailsheim frame times have been denoted separately as a more representative time value. On subsampled images the blob positions are on average 1.5 times less accurate while the processing time is 3.5 times reduced. As the processing time without subsampling is greater than the camera frame rate permits (13.89 ms) Bayer subsampling is applied on cameras with raw Bayer filter data output.

5.2.4 Image rectification

The blob detection algorithm needs additional capabilities when the image is used directly. The lens distortion and camera perspective transformation lead to changing blob sizes

depending on the distance to the camera image plane and deformations of the circular blobs to ellipsoids. These effects are particularly strong in the NOTG dataset due to the camera location at the edge of the field. Blob sizes vary between 3 to around 35 pixels in diameter, which is particularly challenging for SSL-Vision due to pixel based size filters. The perspective distortion effects can be observed in the left part of Figure 5.4, where the blobs on top of the robot are strongly ellipsoid.

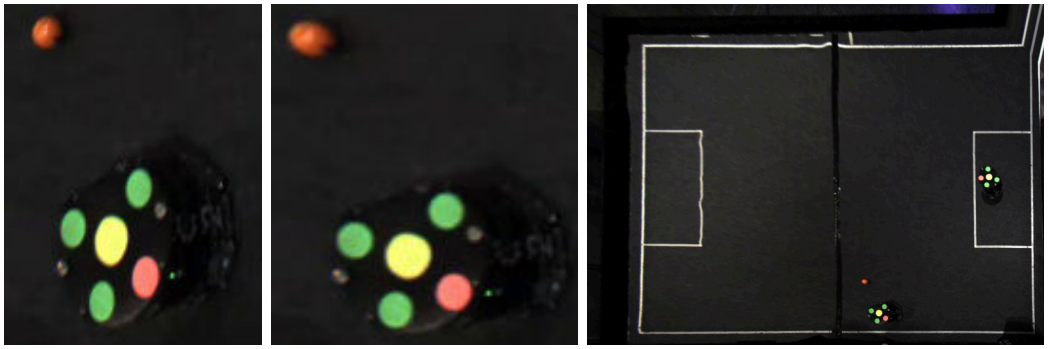


Figure 5.4: Scene from the NOTG dataset with a raw snippet on the left side, a rectified snippet in the center and the complete rectified image on the right showcasing the circularization of the robot pattern and elongation of the ball.

Since the camera position model, position and orientation is known at this point the image can be rectified to a fixed field height to remove these effects at a small processing time cost. Due to OpenCL offering functionality for sampling images with nearest neighbor or linear interpolation and functionality for various image border handling options, OpenCL drivers can utilize the specialized texture sampling hardware often present in GPUs. The effects of linear interpolation and "repeat" border handling can be seen in the right image of Figure 5.4. The top left corner is significantly more blurred due to less available pixels in the original image at that location. The top right corner showcases the border handling as the corner of the field itself is outside of the image. As the ball is a sphere and therefore circular from any perspective without lens distortion the ball becomes ellipsoidal when rectified. This is visible in the central image of Figure 5.4. As the ball is still relatively circular under a worst case scenario like in the NOTG dataset and as the size normalization effect of the resampling is still helpful for further processing this elongation effect will be ignored in future steps. The influence of blobs appearing smaller than expected due to a larger distance from the camera through lower height than maximum robot height are ignored due to the negligible magnitude of this effect with cameras positioned meters above the field and a maximum robot height of 15cm.

For the resampling robots are assumed to be the highest objects on the field. Potentially flying balls will therefore be ignored during the extent determination for the resampling. The maximum allowed robot height is therefore the optimal height to use for resampling and extent determination, as for cameras positioned above the field the highest object is the outermost object and for cameras positioned outside of the field the field wall mostly blocks line of sight below the maximum robot height.

On fields with multiple cameras each camera only sees a part of the field. To prevent uncaptured areas in the resampled image of a camera the extent necessary for the resampled image needs to be determined. As visible image borders are typically aligned to the field coordinate axes the visible extent is approximated with a simple 2D axis-parallel bounding box. For determining the visible extent the corresponding field position is determined for each pixel at the image border. The minimal and maximal x and y position values are then clipped to the field size and used as the visible extent.

The size of the resampled image impacts the performance of subsequent image processing steps. A lower image resolution minimizes the processing latency and preserves strong gradients in the image. A higher image resolution reduces the loss of information and enables a more precise localization. Through the computation of the minimum distance in the projected position between neighboring pixels an upper bound of the image resolution can be determined as more information isn't contained in the image. Further processing steps can be simplified by maintaining an lower bound of the smallest blob radius (20mm) as with this resolution the blob is guaranteed to fill one pixel. Another lower bound is the blob position accuracy required to prevent to prevent misidentification of the robot orientation, which is estimated to be 10.96mm (see Section 2.2.1). Choosing the maximum distance in the field position between neighboring pixels or lower resolution guarantees fairness through equal sample quality independent of the field position.

Resampling size	Accuracy/mm ↓ Blob	Ball	Robot	PPR ↑	Runtime ↓ Laptop
min distance	**6.62 ± 3.74**	**7.75 ± 4.06**	**5.21 ± 2.39**	0.8768	33m 36s
mean distance	6.69 ± 3.75	7.90 ± 3.75	5.41 ± 2.33	0.9081	31m 54s
NN interpol.	7.00 ± 3.89	8.36 ± 4.27	5.55 ± 2.46	**0.9401**	31m 50s
max distance	7.18 ± 3.80	8.96 ± 4.17	5.26 ± 2.09	0.9091	30m 50s
05 mm	7.24 ± 3.76	8.28 ± 4.02	5.87 ± 2.38	0.9091	40m 08s
10 mm	9.80 ± 4.14	10.32 ± 4.06	8.42 ± 2.65	0.8508	**23m 47s**
min, NOTG only	4.84 ± 3.37	5.06 ± 3.93	4.09 ± 2.39	0.8053	-
mean, NOTG only	5.73 ± 3.55	6.63 ± 4.22	4.72 ± 2.30	0.8646	-

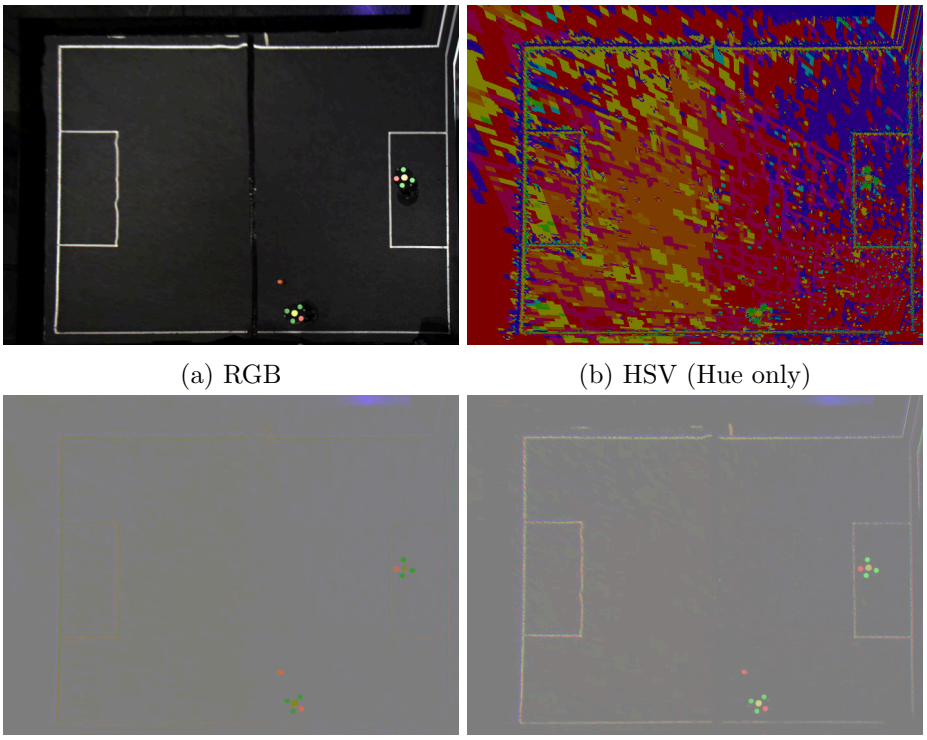
Table 5.6: Comparison of different resampling sizes and interpolation methods showing the correlation between resampling size, position accuracy and benchmark runtime.

Table 5.6 shows the impact of different resampling scales and interpolation methods. ± denotes the standard deviation. The runtime of the total benchmark on the laptop shows the impact of different resolutions on the performance. Using the nearest neighbor interpolation method delivers stronger gradients and therefore a better PPR but reduces the blob accuracy compared to using linear interpolation. Using the minimum pixel distance gives overall the best position accuracy at a slight runtime cost and reduced PPR due to more interpolated gradients. As the NOTG dataset is the dataset with the biggest size difference the min and mean distance results are shown separately. The max distance benchmark of NOTG yielded no result due to the gradient scaling resulting in a gradient distance of 0. As the most sensible tradeoff between PPR, runtime and position accuracy the mean pixel distance has been chosen as the default resampling resolution.

5.2.5 Color spaces

While robot blobs have generally a uniform color and diffuse reflection, the spherical shape and rubber material of the ball result in specular reflection and shading differences. The field lines, goal posts and walls oftentimes feature a strong brightness contrast to the surrounding field carpet. Therefore the usage of a brightness invariant color space can aid the blob detection task by converging the appearance of ball and robot blobs and reducing the contrast between carpet and field lines. Definitions for the RGB, Hue, Saturation, Value (HSV) and YUV color spaces can be found in [53].

The RGB color space uses three channels for one color each and corresponds as such to the Bayer matrix with color filters most color cameras use. Due to the direct correspondence with the light response brightness changes are encoded in the color information, as is visible in Figure 5.5a.



(a) RGB (b) HSV (Hue only)

(c) YUV (UV only) (d) dRGB

Figure 5.5: Comparison of the color components of a rectified NOTG scene showing the influence of brightness and saturation in different color spaces.

The HSV color space uses a single cyclical value channel to encode the color, separate from the brightness information. As visible in Figure 5.5b the color information is unstable in white, gray and black areas. The reported color in low saturation areas is also heavily dependent on the white balance. As the hue channel does not reduce gradient responses with gray carpets and at the field lines the HSV color space was not further investigated. The quadrilateral patches visible in Figure 5.5b are compression artifacts from the video compression and not present in direct camera data.

The YUV color space (also known as YCbCr) separates the brightness/luminance channel (Y) and color plane (UV) with a linear transformation of the RGB color space. As visible in Figure 5.5c the UV components manage to reduce field line and carpet responses through the achieved brightness invariance.

As the YUV color space is optimized for human visual perception a novel three channel color space delta RGB (dRGB) is being proposed. The dRGB color space uses $R - 0.5G - 0.5B$ as red channel, $G - 0.5R - 0.5B$ as green channel and $B - 0.5R - 0.5G$ as blue channel. As visible in Figure 5.5d the dRGB color space manages to reduce brightness contrast similar to the YUV color space.

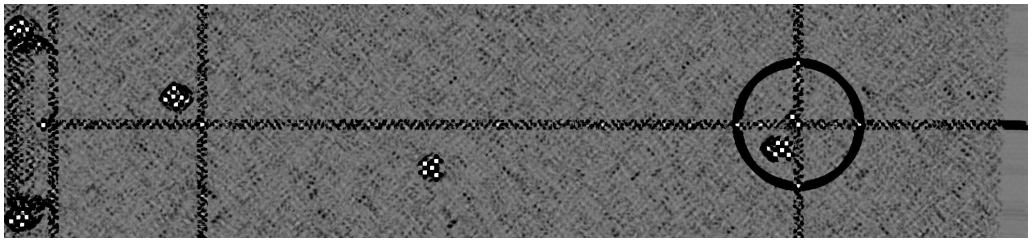
Figure 5.6 shows the blob scores of a cutout from the RoboCup 2022 dataset with brighter areas corresponding to a greater response. A response of 0 corresponds to a gray level of 0.5. In Figure 5.6a the issues with the RGB color space become apparent. Strong responses at field line crossings and at the side of the robot due to white and black livery make the RGB color space less useful. Another issue is the missing response at the left side of the cutout on the robot blobs neighboring the goal posts. Figure 5.6b and Figure 5.6c in YUV and dRGB color space respectively do not exhibit these flaws.

As Table 5.7 shows, all three color spaces are yielding acceptable results with the biggest difference being in the PPR. The RGB ball accuracy is worse than the other approaches

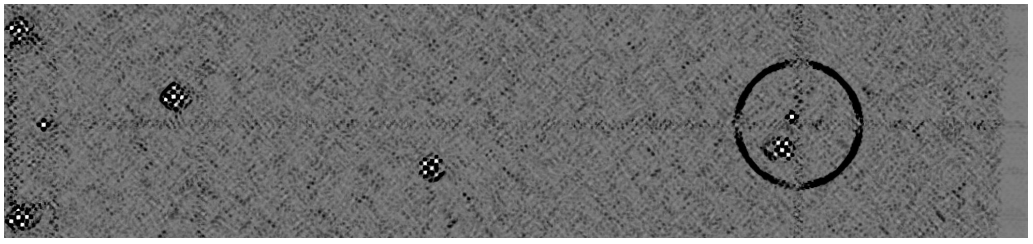
Color space	Accuracy/mm ↓ Blob	Ball	Robot	PPR ↑
RGB	**6.40 ± 3.86**	8.71 ± 4.33	**5.35 ± 2.59**	0.9279
YUV	6.69 ± 3.82	7.98 ± 3.75	5.37 ± 2.32	**0.9541**
dRGB	6.69 ± 3.75	**7.90 ± 3.75**	5.41 ± 2.33	0.9081

Table 5.7: Comparison of position accuracy and PPR in different color spaces showing similar accuracy in all color spaces.

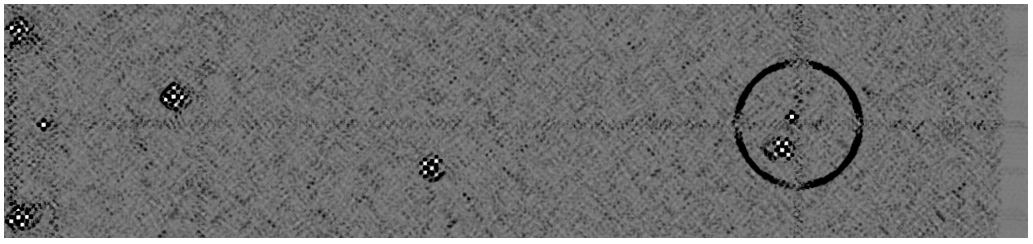
due to the influence of the ball curvature and shadow. The PPR in RGB is high despite the additional field line crossing responses due to the utilization of the brightness change of the robot blobs on the black background. The issues with the RGB color space become only apparent with a visual inspection of the blob scores. The conceptually similar YUV and dRGB color spaces yield overall very similar position accuracy results with the only significant difference in the PPR mainly due to a significant difference in the RoboCup German Open dataset.



(a) RGB



(b) YUV (UV only)



(c) dRGB

Figure 5.6: Comparison of blob scores in the RoboCup 2022 dataset across color spaces showing the false responses at field line crossings in the RGB color space.

5.2.6 Template matching

Template matching is an obvious approach as ideal color, shape and size are known in advance. The deliberation of determining additional parameters previously undetermined through the modeling of additional effects also speaks for template matching. Through the modeling of motion blur object velocity can be estimated from a single frame theoretically. Testing different ball sizes allows for an estimation of the height of the ball above the ground.

Naive template matching compares the image I to the template T at each position and uses the Squared Sum of Differences (SSD) relative to the template as scoring metric S [54]:

$$S(x, y) = \sum_{i,j} (I(x + i, y + j) - T(i, j))^2. \tag{5.10}$$

Other correlation based scoring metrics for template matching like the normalized cross correlation compensate for differences in magnitude between image and template. This linear scaling leads to an inability to differentiate neighboring colors such as orange and yellow in the (d)RGB and YUV color spaces. Convolutions are unsuitable due to the same issue.

The evaluated templates were derived from the ideal form and color of the blobs. As the blobs are circular with known radius, all templates investigated are equally in circular shape.For the following tests template matching is done for each blob type and color separate with the maximum response for a given pixel used as score. The baseline approach assumes all blobs as a uniformly shaded circle with known radius and ideal colors and uses the RGB color space. "Opt. color" uses manually chosen colors from the RoboCup 2022 dataset to estimate the performance with optimized colors. dRGB uses the dRGB color space. "Ball gradient" changes the template for ball blobs and tries to model the diffuse shading of the spherical ball by assuming a linear gradient starting from the center point. "Center blob" changes the template for center blobs to additionally model the guaranteed black space around center blobs. "Center & ball" combines the ball gradient and center blob changes.

Approach	Accuracy/mm ↓	PPR ↑			Runtime ↓
	Blob	Average	Image	RoboCup 2022	Laptop
Baseline	**4.73 ± 3.04**	**0.3070**	**0.5111**	**0.5101**	43m 24s
Opt. colors	5.12 ± 3.62	0.2979	0.4960	0.5045	43m 09s
dRGB	5.64 ± 3.57	0.3048	0.5070	0.4980	**41m 44s**
Ball gradient	5.84 ± 4.73	0.3010	0.5012	0.5021	44m 26s
Center blob	8.88 ± 5.90	0.2987	0.4973	0.4935	107m 10s
Center & ball	8.29 ± 5.77	0.2991	0.4980	0.4940	108m 44s

Table 5.8: Comparison of the blob accuracy and PPR with different template models and colors showing the uniform shaded baseline blob model as best template matching based approach.

Table 5.8 shows the results of the approaches described above. The PPR for all approaches is near zero for all video datasets. This has been attributed to compression artifacts. Therefore the average over all image datasets is used as more representative value. Runtime describes the total processing time for all datasets on the laptop platform.

As the PPR is always around 0.5 none of the tested approaches achieves a satisfactory separation between blobs and field. All improvement attempts over the baseline approach achieved worse performance. Using optimized colors for the RoboCup 2022 dataset did regress the performance due to a similarly smaller distance in the average field color. The center blob approach was significantly slower due to the larger template and had a worse position accuracy due to better responses in dark areas. The template matching approach has therefore been abandoned as it provided no adequate performance.

5.2.7 Color gradient

All blob borders have a strong change in the color gradient in common, as the orange ball has the opposite color of the green carpet and the robot pattern is always on a black surface. Since the carpet has no sudden changes aside from the field lines a gradient based blob detection is an obvious approach.

Each gradient based approach needs two nearly independent steps to be solved:

1. Determining the gradient metric.
2. Determining the center point of the blob location.

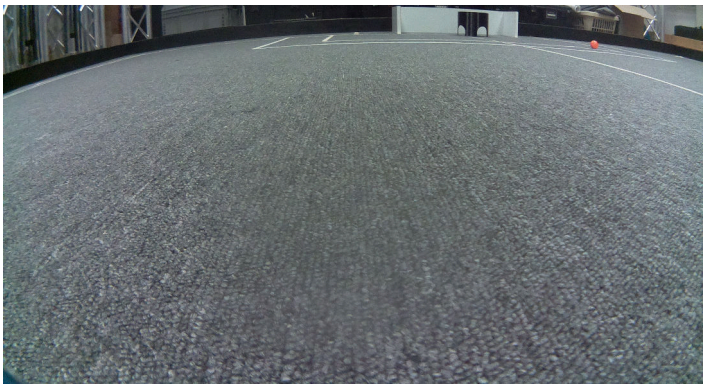


Figure 5.7: The carpet at close proximity shows the inconsistent shading caused by uneven dye and the carpet pattern. This leads to random blobs of similar shading with varied sizes up to robot pattern and ball sized blobs.

One challenge is the texture of the the carpet resulting in a nonuniform color and brightness at scales that can reach sizes comparable to blobs. An example of the carpet texture at close proximity is displayed in Figure 5.7.

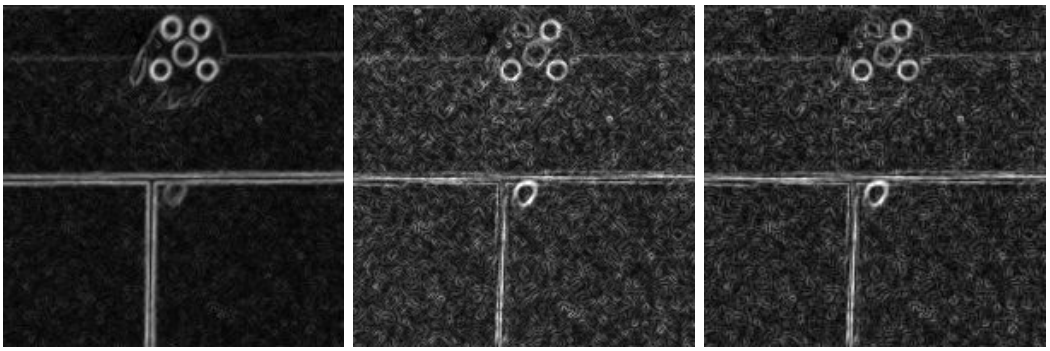
5.2.7.1 Color gradient type

The gradients $g_x(x, y)$ and $g_y(x, y)$ describe the change in the image i at the position x, y:

$$\begin{aligned} g_x(x, y) &= i(x + 1, y) - i(x - 1, y) \\ g_y(x, y) &= i(x, y + 1) - i(x, y - 1). \end{aligned} \tag{5.11}$$

To maintain positional symmetry the values used for the gradient determination are two pixels apart. This simple gradient has the benefit of requiring less memory reads compared to more popular and accurate gradient variants like the Sobel filter [55]. As shown in Figure 5.8 the Sobel filter brings no significant benefit in this usecase while increasing the processing time on memory bandwidth limited integrated Graphics Processing Unit (iGPU).

Typically the magnitude $m = \sqrt{\sum_C g_x^2 + g_y^2}$ (summed over all color channels C) and gradient direction $d = \text{atan2}(g_y, g_x)$ are used for further processing. Figure 5.8 shows why the usage of either the magnitude or direction alone is insufficient. The strong response of the field lines in the RGB color space as seen in Figure 5.8a and blurred out response at the ball near the field line increase the difficulty of successful blob detection. When reducing the impact of the brightness information with the dRGB color space as seen in Figure 5.8b the texture of the carpet is emphasized. The response of the green top left blob



(a) RGB (b) dRGB (c) dRGB Sobel

Figure 5.8: Comparison of the gradient magnitude in the RoboCup 2022 dataset showing the impact of brightness invariance and gradient type on noise levels and field line responses.

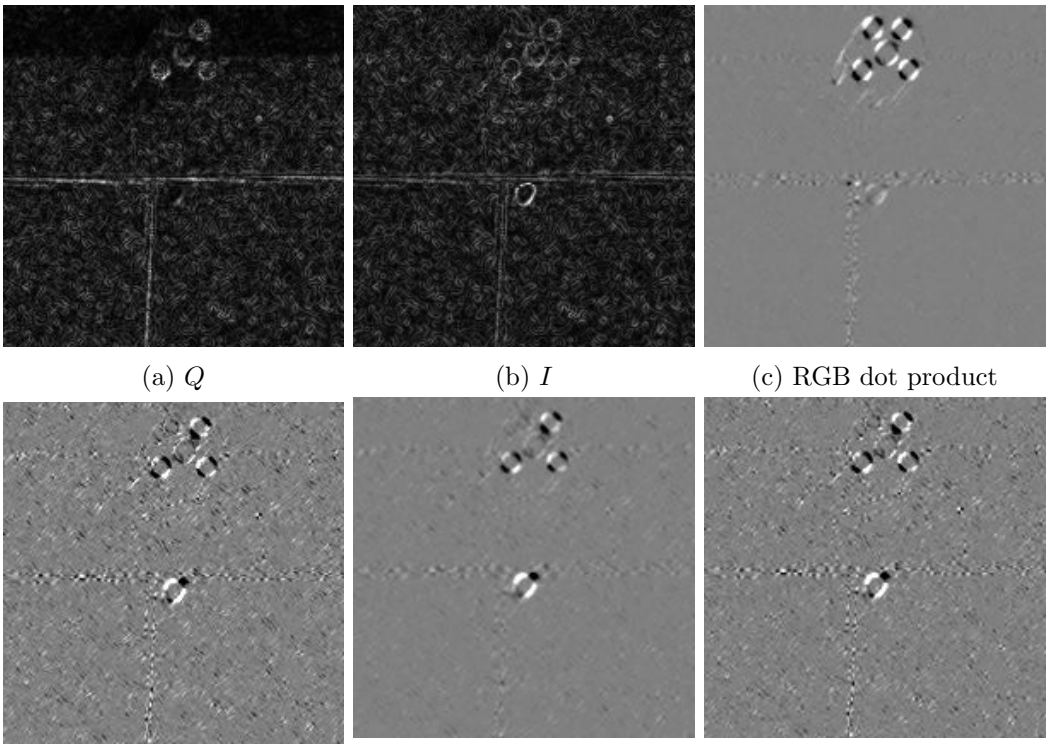
is significantly reduced due to the cyan heavy white balance in the RoboCup 2022 dataset. Using the Sobel filter doesn't improve these issues significantly as seen in Figure 5.8c. This carpet texture noise as well as the small scale of the blobs reduces the usability of the gradient direction. The small size of the blobs makes a detection approach based on the gradient direction difficult. An approach purely based on the direction of the gradient showcases issues with balls at or above field lines as visible from Figure 5.8a. Additional blur doesn't help as the overall gradient response is reduced similar to the carpet texture due to the similar scale of the carpet texture compared to the blob size.

As an alternative to the simple gradient magnitude Joost van de Weijer et al. have developed multiple approaches to quantify the color change through the color tensor [32]. They suggest among other methods using the magnitude of the shadow-shading-specular quasi-invariant Q and invariant I transformations for colored object detection. These are defined for the axis $z \in \{x, y\}$, the color channels R, G and B and the dot product denoted as $\cdot$ as follows:

$$\begin{aligned} h_z &= i_R(g_{z_B} - g_{z_G}) + i_G(g_{z_R} - g_{z_B}) + i_B(g_{z_G} - g_{z_R}) \\ Q &= \sqrt{h_x^2 + h_y^2 + \sqrt{(h_x^2 - h_y^2)^2 + (2h_x h_y)^2}} \\ I &= \frac{Q}{\sqrt{2(i \cdot i - i_R i_G - i_R i_B - i_G i_B)}}. \end{aligned} \tag{5.12}$$

Figure 5.9 showcases the magnitude of the shadow-shading-specular quasi-invariant (Figure 5.9a) and invariant (Figure 5.9b). While the invariant shows a stronger response than the quasi-invariant around the ball, both approaches show similar noise levels compared to the dRGB color space gradient magnitude (Figure 5.8b) while minimizing the field line response more.

On a rectified image the blobs are the only features with neighboring diagonal gradients as goal posts and field lines are mostly parallel to the resampled coordinates. The center circle and robot hulls are both of significantly larger scale. Therefore the dot product has been tested as a metric combining magnitude and direction. As visible in Figure 5.9c the dot product of the gradients in x and y direction results in strong positive responses in the "even" quadrants and strong negative responses in the "uneven" quadrants, resulting in a checkerboard like pattern. Carefully chosen color spaces like visible in Figure 5.9d can furthermore reduce the response along field lines and improve the response at balls. Gaussian blur (Figure 5.9e) or the Sobel filter (Figure 5.9f) can help in minimizing the response noise at the cost of additional processing time.



(a) Q (b) I (c) RGB dot product

(d) dRGB dot product (e) dRGB dot product with Gaussian blur (f) dRGB dot product with Sobel filter

Figure 5.9: Comparison in the RoboCup 2022 dataset showing the differences in the response between color gradient type and preprocessing steps.

5.2.7.2 Magnitude blob center determination

After determining the gradient magnitudes the blob center and score has to be determined. For magnitude based gradient approaches the blob center is surrounded by a circle with high magnitude.

One popular approach to determine circle center positions is the Hough transform [31]. The Hough transform has not been investigated in this work as the usage of the Hough transform necessitates an additional threshold which gradient positions vote. This threshold is difficult to determine due to sometimes blurred gradient responses on balls (e.g. Figure 5.8a). Furthermore the need for atomic synchronization and more writes than other approaches increase the processing time cost of the Hough transform, especially in memory bandwith limited circumstances.

The inverted approach of using a convolution for the cross-correlation with the expected response pattern does not have the same issues as the Hough transform. As the magnitude response is expected to be similar around the complete blob circle, the convolution pattern can be further simplified to only use 1 and 0 as factors and therefore reduce the computational requirement to a sum. As all blob sizes $b_{\text{center}}, b_{\text{side}}, b_{\text{ball}}$ and the uncertainty through resampling size s are known, the radius r to the center in which responses are expected can be computed: $b_{\text{side}} - \frac{s}{2} \leq r \leq b_{\text{center}} + \frac{s}{2}$. The sum of all responses M inside this ring normalized by the amount of samples n can be used as a blob center score S.

$$S(x,y) = \frac{\sum_{b_{\text{side}}-\frac{s}{2} \leq r \leq b_{\text{center}}+\frac{s}{2}} M(x+i, x+j)}{n}. \tag{5.13}$$

As visible in Table 5.9 all magnitude based gradient types lead to similar PPR results. The shadow-shading-specular quasi-invariant and invariant resulted in a significantly worse

Gradient type	Accuracy/mm ↓ Blob	Ball	Robot	PPR ↑ Image
Magnitude dRGB	7.82 ± 5.39	$\mathbf{7.56 \pm 3.81}$	4.35 ± 2.13	0.6759
Magnitude RGB	7.95 ± 5.78	8.94 ± 4.79	$\mathbf{4.00 \pm 1.81}$	0.6383
Magnitude dRGB Sobel	$\mathbf{7.66 \pm 5.31}$	7.59 ± 4.25	4.45 ± 2.24	0.6805
SSSQI	13.40 ± 5.88	13.39 ± 5.13	5.81 ± 2.61	0.6397
SSSI	9.07 ± 5.28	9.20 ± 4.71	4.06 ± 2.05	**0.6967**

Table 5.9: Comparison of the position accuracy and PPR results with Magnitude based ring summation showing good accuracy and unsatisfactory response separation with magnitude based approaches.

blob position accuracy. Usage of the Sobel operator for the gradient determination only lead to marginally better results. The achieved PPR is better than the results achieved by template matching, but still an insufficient separation between blobs and field. Performance improvements through the approximation of the ring with the midpoint algorithm at only one specific radius did lead to further regressions of the PPR.

5.2.7.3 Gradient dot product blob center determination

The gradient dot product delivers responses with different sign depending on the quadrant. Multiple approaches to derive a blob center score from the gradient dot product have been evaluated.

As the gradient dot product G can be blurred due to ball curvature or gradient scaling (see Section 5.2.7.4) a sum over each quadrant square with the maximum blob radius r can be used as as blob score for the position x, y similar to the Haar-like features used in the Viola-Jones object detector [28]:

$$S(x,y) = \frac{\sum_{i=1,j=1}^{r,r} G(x-i,y-j) + G(x+i,y+j) - G(x+i,y-j) - G(x-i,y+j)}{4r^2}. \tag{5.14}$$

This method leads to phantom blobs in the area between blobs due to strong quadrant responses if one or more quadrants are at the side of another blob. To compensate for this, a score using the minimum of the quadrant sums has been devised ($i, j \geq 1$):

$$\begin{aligned} S(x,y) = \min\Big(&\sum_{i,j}^{r,r} \frac{G(x-i,y-j)}{r^2}; \sum_{i,j}^{r,r} \frac{G(x+i,y+j)}{r^2}; \\ &-\sum_{i,j}^{r,r} \frac{G(x+i,y-j)}{r^2}; -\sum_{i,j}^{r,r} \frac{G(x-i,y-j)}{r^2}\Big). \end{aligned} \tag{5.15}$$

The quadrant square sum can be accelerated using a Summed-Area Table (SAT) as further discussed in Appendix A.3. As the blobs are circular and not rectangular a quadrant sum using circle shaped sections has also been evaluated. Due to the nature of the cross product the largest responses are on the diagonals. Therefore a cross shaped pattern was investigated as a tradeoff between frame time and accuracy:

$$S(x,y) = \min\Big(\sum_{i=1}^{r}\frac{G(x-i,y-i)}{r};\sum_{i=1}^{r}\frac{G(x+i,y+i)}{r};\\ -\sum_{i=1}^{r}\frac{G(x+i,y-i)}{r};-\sum_{i=1}^{r}\frac{G(x-i,y+i)}{r}\Big). \tag{5.16}$$

For further frame time improvements starting i at the minimum blob radius r_{min} was also evaluated.

Type	Accuracy/mm ↓ Blob	Ball	Robot	PPR ↑	f.t./ms ↓ NOTG
Quadrant Min	6.69 ± 3.75	7.90 ± 3.75	5.41 ± 2.33	0.9081	3.90
Quadrant Sum	6.91 ± 3.95	8.06 ± 3.42	5.32 ± 2.37	0.5678	4.11
Circle Min	$\mathbf{5.60 \pm 3.58}$	$\mathbf{6.72 \pm 3.83}$	$\mathbf{4.45 \pm 2.20}$	0.9058	10.49
Cross Min $i \geq 1$	6.73 ± 3.97	8.07 ± 4.02	4.67 ± 2.39	0.8424	3.45
Cross Min $i \geq r_{\min}$	7.80 ± 5.20	8.19 ± 4.22	5.03 ± 2.33	**0.9282**	**2.63**

Table 5.10: Comparison of gradient dot product summation types showing circle quadrants with the best position accuracy and all minimum based summation types with good field response separation. "f.t." indicates the average frame time on the NOTG dataset.

As visible in Table 5.10 using the minimum quadrant sum instead of the signed quadrant sum as score leads to a significantly better PPR. Using an exact circular shape results in a better accuracy but comes at a significant frame time cost as the SAT approach cannot be used directly. Using the cross approximation leads to either a PPR reduction or a accuracy penalty. As the frame time cost of the exact circular shape is prohibitive the minimum quadrant sum has been chosen as the blob center score.

5.2.7.4 Gradient scale

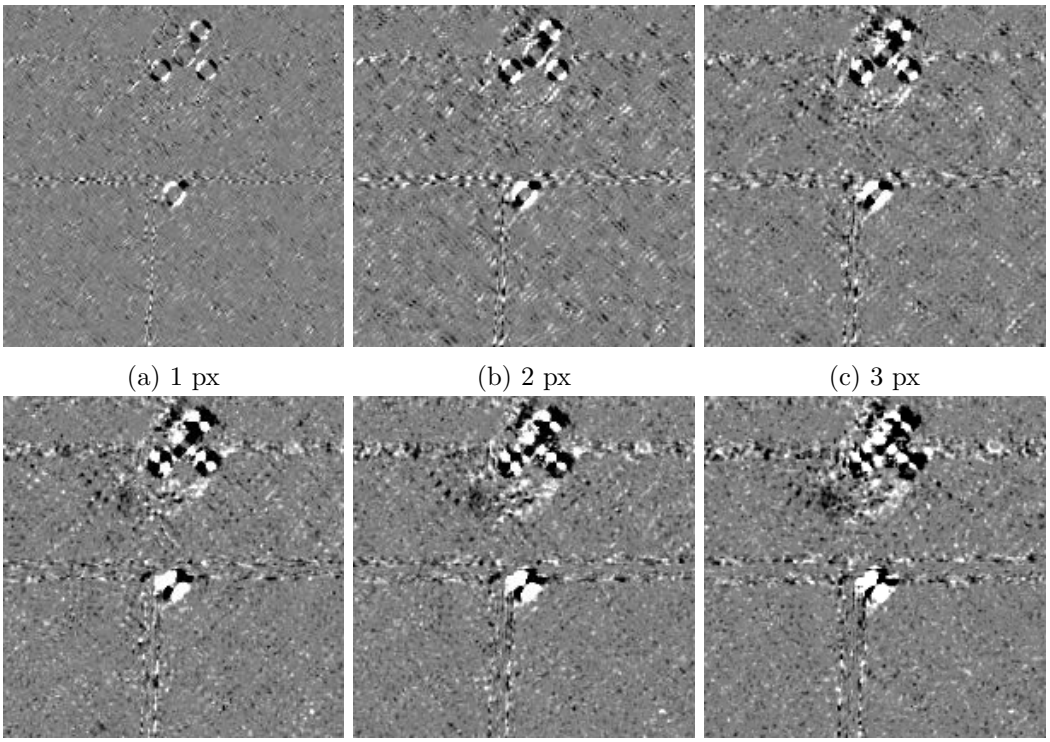
As an alternative to blurring the distance of the values used to determine the gradient can be changed to improve the response at blurry edges caused by demosaicing, chromatic aberration or resampling interpolation. As the gradient response is distributed over a larger area this also improves the response with area summation based blob scores like described in Section 5.2.7.3.

Scale/px	Scale/mm	Accuracy/mm ↓ Blob	PPR ↑
1	4.03	7.96 ± 4.24	0.9623
2	8.06	7.17 ± 3.66	0.9784
3	12.09	$\mathbf{7.16 \pm 3.80}$	0.9866
4	16.12	7.36 ± 4.01	**0.9878**
5	20.15	7.70 ± 4.12	0.9866
6	24.18	9.51 ± 5.03	0.9784

Table 5.11: Position accuracy and PPR with varying gradient scale on the RoboCup 2022 dataset showing improvements with a scale around half of the expected blob radius.

Figure 5.10 shows the gradient dot product with different distance of the datapoints used for the gradient determination. The area of the checkerboard response increases until

neighboring blobs interfere with each other. Table 5.11 shows the corresponding blob accuracy and PPR results for the RoboCup 2022 dataset, with a gradient scale of 3 delivering the best accuracy and 4 the best PPR. As the RoboCup dataset has a field resolution of 4.03 $\frac{\text{mm}}{\text{px}}$ two adaptive approaches based on the field resolution f and the minimum or maximum blob radius have been evaluated: $\left\lceil \frac{20\text{ mm}}{2f} \right\rceil$ and $\frac{1}{3}\left\lceil \frac{25\text{ mm}}{f} \right\rceil$.



(a) 1 px (b) 2 px (c) 3 px

(d) 4 px (e) 5 px (f) 6 px

Figure 5.10: Visual comparison of the gradient dot product with different gradient scales in the RoboCup 2022 dataset showing increased response areas and artifacts with increasing gradient scale.

Algorithm	Accuracy/mm ↓			PPR ↑
	Blob	Ball	Robot	
$\frac{1}{3}\left\lceil \frac{25\text{ mm}}{f} \right\rceil$	$\mathbf{6.69 \pm 3.75}$	$\mathbf{7.90 \pm 3.75}$	$\mathbf{5.41 \pm 2.33}$	0.9081
$\left\lceil \frac{20\text{ mm}}{2f} \right\rceil$	6.71 ± 3.77	8.13 ± 3.67	5.50 ± 2.38	0.8634
1 px	7.14 ± 3.97	8.36 ± 4.06	5.48 ± 2.35	**0.9222**
2 px	6.72 ± 3.79	8.24 ± 3.77	$\mathbf{5.41 \pm 2.33}$	0.8810

Table 5.12: Position accuracy and PPR with different gradient scale algorithms on the RoboCup 2022 dataset showing the best position accuracy with a gradient scale of $\frac{1}{3}\left\lceil \frac{25\text{ mm}}{f} \right\rceil$.

As visible in Table 5.12 $\frac{1}{3}\left\lceil \frac{25\text{ mm}}{f} \right\rceil$ delivers the best accuracy. Although a scale of 1 px provides a higher PPR, is has not been selected due to the significantly worse blob accuracy.

5.2.8 Blob selection

For further processing blobs have to be extracted from all responses. Due to the good response separation of the gradient dot product quadrant sum as discussed in Section 5.2.7.3

a manually selected threshold is sufficient as the results of Section 5.3 show. A response is classified as a blob if the response $S(x, y)$ is a 4-way local peak and greater than the manually selected threshold $t = 25$.

5.2.9 Quadratic peak interpolation

As the blob score is mostly continuous the peak can be interpolated to get a more accurate blob center position. The quadratic peak interpolation [56] is a common method to approximate the peak position $x_{\text{peak}}, y_{\text{peak}}$ by fitting a parabola with the data points neighboring a local peak $s(x, y)$:

$$\begin{aligned} x_{\text{peak}} &= x + \frac{s(x-1,y) - s(x+1,y)}{2(s(x-1,y) - 2s(x,y) + s(x+1,y))} \\ y_{\text{peak}} &= y + \frac{s(x,y-1) - s(x,y+1)}{2(s(x,y-1) - 2s(x,y) + s(x,y+1))}. \end{aligned} \tag{5.17}$$

Peak interpol.	Accuracy/mm ↓ Blob	Ball	Robot
Yes	**6.69 ± 3.75**	**7.90 ± 3.75**	**5.41 ± 2.33**
No	6.98 ± 3.84	8.40 ± 4.42	5.47 ± 2.39

Table 5.13: Comparison of the position accuracy with and without quadratic peak interpolation showing position accuracy improvements with interpolation.

As shown in Table 5.13 blob position accuracy did increase with quadratic peak interpolation.

5.2.10 Discussion

With the gradient dot product a novel color blob detection approach that fulfills all requirements is being proposed with acceptable position accuracy, runtime and PPR.

Bayer filter subsampling preserves almost full color resolution while reducing processing time requirements to an acceptable level. The resulting blob precision reduction is necessary to keep processing time requirements. Further investigation could look into the usage of the Alpha channel as second green channel to preserve all data and ways to better preserve the channel localization during image resampling.

Image rectification is the necessary step to simplify subsequent processing steps at a low runtime cost. The average pixel distance has been chosen as the default resampling size as tradeoff of processing time, PPR and blob accuracy. The impact of different resampling sizes on fairness regarding the evenness of detection accuracy would be an interesting further research area.

The dRGB and YUV nvestigated succeed in suppressing responses outside of blobs through the reduction of the brightness impact with the YUV chosen as optimal color space. Automatic gain adjustment in the camera or during processing might further improve the brightness invariance.

Naive template matching did not achieve satisfactory separation between blob and field responses. As this prevents the blob classification this has been abandoned. Further investigation into the usage as a postprocessing step to improve blob position accuracy and test additional parameter extraction like motion blur based velocity or 3D ball position estimation.

Usage of the gradient magnitude or derived metrics like the shadow-shading-specular invariant did similarly not yield sufficient separation between blob and field responses. The gradient dot product quadrant square sum as a combination of amplitude and direction information did yield good response contrast and sufficient blob accuracy. Potential runtime improvements of the exact circle with the midpoint algorithm to eliminate the need for square roots and partial SAT usage have to be investigated further.

Quadratic peak interpolation has successfully increased the blob accuracy. A potential area of future research is the investigation of a more accurate peak model to further increase the blob accuracy.

5.3 Blob assignment

The in the previous step detected blobs need to be classified and combined to detect robots and balls. Evaluations use the same default method for the blob detection as described in Section 5.2. In Section 5.3.1 the precise problem definition is given. Section 5.3.2 discusses the determination of the robot position and orientation for a given tuple of blobs. Section 5.3.3 describes how the relative position of blobs can be used to detect robots. In Section 5.3.4 the hue of the blob is used for classification and filtering, while in Section 5.3.5 a method to use the position of the blob for an adaptive color clustering classification is described. Section 5.3.6 describes partial robot detection through previous detected positions. In Section 5.3.7 the achieved results are discussed.

5.3.1 Problem definition

Given a set of blobs B detect all robots and balls present in the current frame and determine their position and orientation. Each blob consists of the position, blob score and mean color. Maximise the recall of all objects across all datasets while keeping the precision above 0.9.

5.3.2 Robot position and orientation

The robot position and orientation can be determined with at least two blobs. The angles $a_{i,j}$ between two pattern positions are known in advance. The robot orientation $\alpha_{i,j}$ for two given blobs b_i, b_j with $i, j \in \{C, 1, 2, 3, 4\}$ is:

$$\alpha_{i,j} = \text{atan2}(b_{i_y} - b_{j_y}, b_{i_x} - b_{i_x}) - a_{i,j}. \tag{5.18}$$

For more than two given blobs the robot orientation α is the average of all blob pair orientations:

$$\alpha = \text{atan2}(\sum_{i,j} \sin \alpha_{i,j}, \sum_{i,j} \cos \alpha_{i,j}). \tag{5.19}$$

As the pattern position offset p_i of all blobs is known, the robot position p is a simple average for a blob amount n:

$$p = \frac{\sum_i b_i - \alpha \star p_i}{n}. \tag{5.20}$$

$\star$ denotes a 2D rotation.

5.3.3 Position based blob assignment

For identification and orientation determination all robots have a unique blob pattern on the top. The unique fixed position of the robot blobs relative to each other allows for robot detection without the usage of any color information.

For each tuple of five distinct blobs $b_C, b_1, b_2, b_3, b_4 \in B^5$ the position p and orientation α can be computed according to Section 5.3.2. To compare and select different robot hypothesis a scoring function is necessary. The scoring function $s_P(b_C, b_1, b_2, b_3, b_4)$ should be continuous, keep the score in the range between 0 and 1 to double as a confidence score, be computationally inexpensive, punish offsets greater than 10.96 mm as that likely indicates a misdetection according to Section 4.3.5 and minimize the punishment for small expected deviations.

$$s_P(b_C, b_1, b_2, b_3, b_4) = \min_i \frac{1}{1 + \left\| \frac{b_i - \alpha \star p_i}{10\ \text{mm}} \right\|^2}. \tag{5.21}$$

The selected scoring function Equation (5.21) fulfills all of the aforementioned requirements. The 10 mm act as a scaling parameter and mark the offset which results in a 0.5 score.

Using a fivefold Cartesian power B^5 to generate all possible bot hypothesis is computationally too expensive. Therefore the blobs tested as side blobs are limited to a 90 mm radius, the maximum robot radius, around the selected center blob. As the expected blob positions are sorted with increasing angle only blob combinations with increasing angle are tested to further reduce the amount of combinations to test.

To extract actual detections from all hypotheses filtering is necessary. As a blob can only be part of one robot only the best scored hypothesis for a given center blob b_C is retained. Robot hypotheses with a score s_P worse than 0.1 are filtered as likely misdetections. The SSL rules [5] mandate that every robot top covers at least an area of a circle with 85 mm radius with a linear cut with 55 mm distance to the center in the front of the robot. Only the best scored hypothesis of hypotheses intersecting each other according to the minimum size above is retained. As this filtering step scales quadratically with the amount of hypotheses it is done as last filtering step.

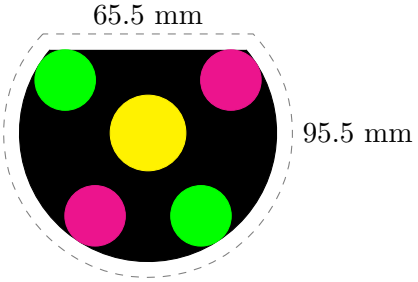


Figure 5.11: Illustration showing the minimal distance between robot and ball center point based on rule constraints.

The orange color of the ball is oftentimes similar to the pink color used in robot patterns. To prevent detecting multiple balls on top of a robot due to multiple blob centers detected for a single blob, the special case of the ball being on top of the robot will be excluded. The rules stipulate that “at least 80% of the [ball area] when viewed from above have to be outside the convex hull around the robot” at all times. This results in the minimum distance between the ball center and the robot center illustrated in Figure 5.11.

5.3.4 Hue based color classification

As each blob is coded in a different color depending on the blob function an obvious approach is blob assignment based on the nearest color class hue (according to the HSV color space). For each yellow and blue blob all green and pink blobs inside a 90 mm radius are used to construct robot hypothesis using scoring and filtering like described in Section 5.3.3. All orange classified blobs outside of robots are used as balls.

Field	Recall ↑		Precision ↑	
	Ideal hues	RoboCup 2022	Ideal hues	RoboCup 2022
TIGERs Lab	0.9578	0.0928	0.9827	0.8148
TIGERs Lab 2	1.0000	0.2222	1.0000	1.0000
KIKS Lab	0.3333	0.3333	1.0000	1.0000
RoboCup 2019	0.1905	0.0952	1.0000	1.0000
RoboCup 2022	0.0600	**0.7333**	1.0000	1.0000
BUGA	0.3333	0.3333	1.0000	1.0000
NOTG	0.3529	0.3523	0.9806	0.9995
TIGERs Weekend	0.4096	0.2609	0.9879	0.9982
RoboCup German Open	0.1488	0.1386	0.9060	0.9997
Schubert Crailsheim	0.4394	0.1834	0.9601	0.9996
Average	**0.4226**	0.2745	**0.9817**	0.9812

Table 5.14: Recall and precision results with hue based color classification showing an unsatisfactory recall rate even with field specific hue values.

The evaluation has been done with the following ideal hues: orange 30°, yellow 60°, blue 210°, green 120°, pink 300°. "RoboCup 2022" annotated values have been computed with hue values handpicked from the RoboCup 2022 dataset: orange 38°, yellow 146°, blue 208°, green 182°, pink 252°. As visible in Table 5.14 the results vary significantly depending on the white balance, exposure and saturation of the specific dataset. The oversaturation in the KIKS dataset leads to issues with similar orange and pink colors, the underexposure in the RoboCup 2019 and BUGA dataset and the blue shifted white balance of the RoboCup 2022 dataset lead to hue stability issues. Even with adjustments to handpicked hue values the recall of the RoboCup 2022 field does not exceed 0.7333.

5.3.5 Position based color classification

Through prior assignment of each blob as robot center blob, robot side blob or potential ball the amount of eligible color classes per blob is reduced to two. Center blobs can either be yellow or blue and side blobs only green and pink. Through the introduction of a field color a similar binary decision can be constructed for ball candidates. The blob color c_b class C_b can therefore be assigned to the neighboring class color c_C according to the L2 norm in arbitrary color spaces.

The ideal reference colors will be a bad match for most fields. The reference colors therefore need to be updated according to detected colors with a clustering algorithm. The maximum amount of classes for the clustering is 2 as the type of the blob has already been determined based on the position. As there is only up to one ball visible in typical situations the algorithm needs to work with small sample sizes. Due to the influence of saturation no a priori distance in and between different color classes is known. k-Means clustering [39] is a common clustering algorithm that has as only input the amount of classes k (in this case $k = 2$) and works for small cluster sizes. k-Means repeatedly assigns each blob the nearest class and recomputes the class color as the average color of all assigned blobs until convergence occurs:

$$C_i = \min_{C_k} \|c_i - c_k\| \tag{5.22}$$

$$c_k = \frac{\sum_i c_i}{n_k}. \tag{5.23}$$

In unmodified k-Means clustering a random blob color is chosen as initial class color to guarantee at least one sample per class. As a rough color estimate already exists from the previous frame (or the reference in the first frame) the blob color nearest to the previous reference color is used as initial class color.

k-Means clustering itself always clusters into the fixed amount of classes independent of the amount of clusters present in the data. One popular solution to determine the correct amount of clusters is the silhouette score [57]. The silhouette score compares the distances inside the cluster to the distances outside the cluster. As two samples are required per cluster to determine an inner distance the silhouette score is not applicable. Due to the existence of multiple types of colors we can use a definitively different color as substitute for the intra cluster distance. Using the smallest distance between two samples d_s, the smallest distance between a sample and the external reference color d_e and the distance between the new class colors d_c the existence of two separate clusters can be estimated:

$$d_c \geq 0.5(d_e - d_s). \tag{5.24}$$

As each robot pattern is guaranteed to have at least 2 side blobs with the same color and a differently colored center blob this methodology is also used per robot to get more accurate local colors for the robot identification determination.

As the type of a blob can be misdetermined, e.g. due to missing pattern blobs, a full redetermination of the class color c_c in each frame as determined by the k-Means clustering is inadvisable. As no significant changes are expected between video frames a retention force $f_p \in [0, 1]$ to the previous frame color c_p has been added. To prevent color inversions a reference force $f_r \in [0, 1]$ to the reference color c_r has been added. As such the class color is updated c_u as follows:

$$c_u = f_p c_p + f_r c_r + (1 - f_p - f_r) c_c. \tag{5.25}$$

f_p	f_r	Recall ↑	Precision ↑
0.0	1.0	0.7352	0.8574
0.0	0.0	0.8062	**0.9206**
0.7	0.1	**0.8173**	0.9142

Table 5.15: Recall and precision results with position based color classification showing the best recall with active forces to the previous and reference color.

As visible in Table 5.15 the adaptive position based color classification does yield overall good recall results. An explanation of the field differences can be found in Section 5.3.7.

5.3.6 Tracking based blob assignment

Objects detected in previous frames disappear seldomly. As such tracking can be used to recover the detection of robots with missing blobs. The robot position in the last two frames can be used to predict the current position of the robot with linear extrapolation. Robot

acceleration estimates[1] of the current world champion TIGERs Mannheim can be used to estimate the uncertainty of the prediction. TIGERs Mannheim estimate a maximum breaking acceleration of 6.0 $\frac{m}{s^2}$, to improve resistance to outliers 6.5 $\frac{m}{s^2}$ has been chosen as maximum robot acceleration. To prevent tracking failures due to a position uncertainty smaller than the blob accuracy a minimum uncertainty of 20 mm is used. To prevent a runaway frame time due to a excessive position uncertainty the time difference t used in the position uncertainty calculation has been capped at 0.05 s.

From the extrapolated robot position and orientation the position of the blobs is determined. At each predicted blob position all blobs in the uncertainty radius are used as potential candidates for this pattern position. The effect of potential angular acceleration of the robot is hereby ignored. A robot hypothesis is then generated for each blob combination with at least two blobs.

Tracking	Recall ↑ Video	Precision ↑ Video
No	0.8789	0.9781
Yes	**0.9277**	**0.9850**

Table 5.16: Recall and precision comparison with additional tracking based blob assignment in Video datasets showing significant improvements with tracking.

As visible in Table 5.16 using tracking improves the recall in video datasets significantly.

5.3.7 Discussion

Field	Recall ↑ Proposed	Recall ↑ SSL-Vision	Precision ↑ Proposed	Precision ↑ SSL-Vision
TIGERs Lab	**0.9873**	0.8692	0.9710	**0.9952**
TIGERs Lab 2	1.0000	1.0000	1.0000	1.0000
KIKS Lab	1.0000	1.0000	1.0000	1.0000
RoboCup 2019	0.4762	**0.8095**	0.4762	**1.0000**
RoboCup 2022	0.5200	**0.9467**	0.7800	**1.0000**
BUGA	0.6667	**1.0000**	1.0000	1.0000
Image average	0.7750	**0.9376**	0.8712	**0.9992**
NOTG	**0.9316**	0.8731	0.9872	**0.9926**
TIGERs Weekend	**0.9828**	0.9545	0.9866	**0.9978**
RoboCup German Open	**0.9491**	0.6431	**0.9861**	0.9775
Schubert Crailsheim	**0.8471**	0.5750	**0.9799**	0.9270
Video average	**0.9277**	0.7614	**0.9850**	0.9737
Total average	0.8361	**0.8671**	0.9167	**0.9890**

Table 5.17: Comparison of the proposed solution and SSL-Vision in recall and precision over all datasets showing the significant improvements in video datasets.

As visible in Table 5.17 the performance of the proposed solution relative to SSL-Vision cannot be described with the overall recall and precision. In the image datasets SSL-Vision is nearly unbeatable as SSL-Vision has been calibrated using the first image (or video) and as such a significant part of the image datasets. The proposed solution struggles especially in very dark image datasets like RoboCup 2019 and BUGA and in the image dataset RoboCup 2022 due to its bad white balance. The image datasets do not allow

[1] `https://github.com/TIGERs-Mannheim/Sumatra/blob/master/config/botParamsDatabase.json`

for partial robot detection with tracking as each scene only consists of a single frame. RoboCup 2022 does contain for example one situation with only yellow robots, which are misdetected as blue robots due to the shifted white balance. In the video datasets the proposed solution has a significantly higher recall, especially in the datasets with inconsistent lighting (Schubert Crailsheim) or changing illumination (RoboCup German Open).

6 Conclusion and future work

The proposed solution with the novel gradient dot product minimum quadrant sum blob detector and position and tracking based blob classification with minimal a priori information achieves an overall satisfactory performance. An validation experiment with ball passing robots using data from the proposed solution at the RoboCup 2024 has shown successfully robot control. During this experiment the rate of successful passes was higher with the proposed solution compared to SSL-Vision at the same field.

Datasets	Accuracy/mm relative to SSL-Vision ↓		Recall ↑		Precision ↑	
	Ball	Robot	Proposed	SSL-Vision	Proposed	SSL-Vision
Image	6.44 ± 2.79	5.30 ± 2.13	0.7750	**0.9376**	0.8712	**0.9992**
Video	10.09 ± 5.20	5.57 ± 2.63	**0.9277**	0.7614	**0.9850**	0.9737
Average	7.90 ± 3.75	5.41 ± 2.33	0.8361	**0.8671**	0.9167	**0.9890**

Table 6.1: Overall accuracy, recall and precision results in comparison to SSL-Vision showing the advantage of the shape based proposed solution to the color LUT approach of SSL-Vision in imperfectly calibrated scenarios like the video datasets.

The proposed solution achieves a blob accuracy of 7.90 mm relative to SSL-Vision, which is worse then the prior estimated accuracy of SSL-Vision of around 3.50 mm but still within the estimated accuracy limit of 10.96 mm. The better accuracy achieved with template matching (4.73 mm) or circular minimum quadrant summation (5.60 mm) show potential ways for further improvement. As template matching did not achieve a good separation between blob and field this could only be done as a peak refinement step, while a more performant implementation is necessary for the usage of the circular minimum quadrant summation. Combining the Bayer subsampling and image rectification steps might also lead to further accuracy improvements.

The detection rate of robots and balls of the proposed solution is even to or better than SSL-Vision on the majority of fields. It is worse in some image datasets (BUGA, RoboCup 2019 and RoboCup 2022) due to SSL-Vision being configured on a significant part of the dataset and no opportunity to learn the colors over multiple frames. Due to the usage of brightness invariant color spaces and adaptive color clustering the proposed solution performs significantly better than SSL-Vision on fields with spatially inconsistent lighting like Schubert Crailsheim, temporally changing illumination like RoboCup German Open and from a side looking perspective like NOTG. To further improve robustness adaptive

exposure, gamma and white balance adjustments to maximize the color contrast at all brightness levels might be possible, as typical automatic exposure modes of cameras tend to overexpose the blobs due to their small size.

As the proposed solution has been designed with minimal necessary configuration (field size, camera amount, camera id and camera driver) the necessary setup time is significantly shorter than a SSL-Vision setup. This improves the usability. The reduced configuration parameter count leads to a slight reduction in repairability compared to SSL-Vision. The camera parameters gain and white balance, which influenced the performance of the proposed solution the most, have been designed to be manually adjustable to mitigate the impact. , although the absence of pretrained data

The proposed solution is faster than SSL-Vision (7.40 ms) with an average frame time of 5.96 ms for the NOTG dataset on the NUC platform and therefore fulfills all frame time requirements.

Further research could look into improved ball detection, as the current solution has the tendency of detecting phantom balls at field line crossings after the ball has been invisible for a prolonged time. Furthermore 3D ball position and single frame blob velocity estimation are potential future enhancements.

List of Figures

List of Tables

Acronyms

API Application Programming Interface

CPU Central Processing Unit

dGPU dedicated Graphics Processing Unit

dRGB delta RGB

GPGPU General Purpose computing on Graphics Processing Unit

GPU Graphics Processing Unit

HSV Hue, Saturation, Value

iGPU integrated Graphics Processing Unit

LUT lookup table

OpenCL Open Compute Language

PPR Peak Percentile Ratio

RGB Red, Green, Blue

SAT Summed-Area Table

SSD Squared Sum of Differences

SSL RoboCup Small Size League

Bibliography

[1] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara, „Robocup: A challenge problem for ai“, *AI magazine*, vol. 18, no. 1, pp. 73–85, 1997.

[2] R. Federation. „Objective“. (2024), [Online]. Available: `https://www.robocup.org/objective` (visited on 06/10/2024).

[3] R. Federation. „Robocup small size league“. (2024), [Online]. Available: `https://ssl.robocup.org/` (visited on 09/11/2024).

[4] R. S. E. Committee. „Robocup small size league (ssl) principles and goals“. (2023), [Online]. Available: `https://robocup-ssl.github.io/ssl-goals/sslgoals.pdf` (visited on 06/10/2024).

[5] R. S. T. Committee. „Rules of the robocup small size league“. (2023), [Online]. Available: `https://robocup-ssl.github.io/ssl-rules/sslrules.pdf` (visited on 05/29/2024).

[6] S. Zickler, T. Laue, O. Birbach, M. Wongphati, and M. Veloso, „Ssl-vision: The shared vision system for the robocup small size league“, in *RoboCup 2009: Robot Soccer World Cup XIII 13*, Springer, 2010, pp. 425–436.

[7] J. Bruce, T. Balch, and M. Veloso, „Fast and inexpensive color image segmentation for interactive robots“, in *2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, vol. 3, 2000, pp. 2061–2066.

[8] D. W. Marquardt, „An algorithm for least-squares estimation of nonlinear parameters“, *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[9] C. Wang, H. Wang, H. Wang, and W. Y. Soh, „The vision system for robocup small robot league“, in *Second Asian Symposium on Industrial Automation and Robotics*, BITEC, 2001.

[10] L. A. Martinez-Gomez and A. Weitzenfeld, „Real time vision system for a small size league team“, in *Proc. 1st IEEE-RAS Latin American Robotics Symposium, ITAM, Mexico City, October*, 2004, pp. 28–29.

[11] J. Srisabye, P. Wasuntapichaikul, C. Onman, *et al.*, „Skuba 2009 extended team description“, 2009. [Online]. Available: `https://tdpsearch.com/#/tdp/soccer_smallsize__2009__Skuba__0` (visited on 10/10/2024).

[12] C. Chuengsatiansup, T. Charoensripongsa, K. Wongsuphasawat, *et al.*, „Plasma-z extended team description paper“, 2009. [Online]. Available: `https://tdpsearch.com/#/tdp/soccer_smallsize__2009__Plasma-Z__0` (visited on 10/10/2024).

[13] Y. Wu, X. Qiu, G. Yu, *et al.*, „Extended tdp of zjunlict 2009“, 2009. [Online]. Available: `https://tdpsearch.com/#/tdp/soccer_smallsize__2009__ZJUNlict__0` (visited on 10/10/2024).

[14] T. Laue, A. Burchardt, S. Fritsch, *et al.*, „B-smart extended team description for robocup 2009“, 2009. [Online]. Available: `https://tdpsearch.com/#/tdp/soccer_smallsize__2009__B-Smart__0` (visited on 10/10/2024).

[15] E. Olson, „Apriltag: A robust and flexible visual fiducial system“, in *2011 IEEE international conference on robotics and automation*, IEEE, 2011, pp. 3400–3407.

[16] Z. Zhang, „A flexible new technique for camera calibration“, *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[17] S. Mohammadi Tari, „Automatic initialization for broadcast sports videos rectification“, University of British Columbia, 2011.

[18] A. Linnemann, S. Gerke, S. Kriener, and P. Ndjiki-Nya, „Temporally consistent soccer field registration“, in *2013 IEEE International Conference on Image Processing*, 2013, pp. 1316–1320.

[19] L. Sun and G. Liu, „Field lines and players detection and recognition in soccer video“, in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, 2009, pp. 1237–1240.

[20] T. Röfer, T. Laue, A. Hasselbring, L. M. Monnerjahn, P. Reichenberg, and ... „B-Human code release documentation 2024“. (2024), [Online]. Available: `https://docs.b-human.de/master/` (visited on 10/13/2024).

[21] D. Farin, S. Krabbe, P. H. N. de With, and W. Effelsberg, „Robust camera calibration for sport videos using court models“, in *Storage and Retrieval Methods and Applications for Multimedia 2004*, International Society for Optics and Photonics, vol. 5307, SPIE, 2003, pp. 80–91.

[22] T. Thormählen, H. Broszio, and I. Wassermann, „Robust line-based calibration of lens distortion from a single view“, *Mirage 2003*, pp. 105–112, 2003.

[23] Q. Yu, G. Xu, Y. Cheng, and Z. H. Zhu, „Plsd: A perceptually accurate line segment detection approach“, *IEEE Access*, vol. 8, pp. 42 595–42 607, 2020.

[24] N. Hamid and N. Khan, „Lsm: Perceptually accurate line segment merging“, *Journal of Electronic Imaging*, vol. 25, no. 6, p. 061 620, 2016.

[25] B. Verhagen, R. Timofte, and L. Van Gool, „Scale-invariant line descriptors for wide baseline matching“, in *IEEE Winter Conference on Applications of Computer Vision*, IEEE, 2014, pp. 493–500.

[26] K. Nguyen, L. T. V. Ngo, K. T. V. Huynh, and N. T. Nam, „Empirical study one-stage object detection methods for robocup small size league“, in *2022 9th NAFOSTED Conference on Information and Computer Science (NICS)*, 2022, pp. 264–268.

[27] S. G. Mallat, „A theory for multiresolution signal decomposition: The wavelet representation“, *IEEE transactions on pattern analysis and machine intelligence*, vol. 11, no. 7, pp. 674–693, 1989.

[28] P. Viola and M. Jones, „Rapid object detection using a boosted cascade of simple features“, in *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, IEEE, vol. 1, 2001.

[29] F. C. Crow, „Summed-area tables for texture mapping“, in *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’84, Association for Computing Machinery, 1984, pp. 207–212.

[30] A. Rad, K. Faez, N. Qaragozlou, *et al.*, „Fast circle detection using gradient pair vectors“, *Proc. VIIth Digital Image Computing: Techniques and Applications*, pp. 879–888, Jan. 2003.

[31] D. H. Ballard, „Generalizing the hough transform to detect arbitrary shapes“, *Pattern recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[32] J. Van De Weijer, T. Gevers, and A. W. Smeulders, „Robust photometric invariant features from the color tensor“, *IEEE Transactions on Image Processing*, vol. 15, no. 1, pp. 118–127, 2005.

[33] G. Cocorullo, P. Corsonello, F. Frustaci, L.-l.-A. Guachi-Guachi, and S. Perri, „Multimodal background subtraction for high-performance embedded systems“, *Journal of Real-Time Image Processing*, vol. 16, pp. 1407–1423, 2019.

[34] D. Bloisi, L. Iocchi, *et al.*, „Independent multimodal background subtraction.“, in *CompIMAGE*, 2012, pp. 39–44.

[35] M. Zhao, J. Bu, and C. Chen, „Robust background subtraction in hsv color space“, in *Multimedia systems and Applications V*, SPIE, vol. 4861, 2002, pp. 325–332.

[36] R. A. Finkel and J. L. Bentley, „Quad trees a data structure for retrieval on composite keys“, *Acta informatica*, vol. 4, pp. 1–9, 1974.

[37] G. S. Lueker, „A data structure for orthogonal range queries“, in *19th Annual Symposium on Foundations of Computer Science (sfcs 1978)*, 1978, pp. 28–34.

[38] J. L. Bentley, „Multidimensional binary search trees used for associative searching“, *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[39] S. Lloyd, „Least squares quantization in pcm“, *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[40] R. SSL. „Game logs“. (2024), [Online]. Available: `https://ssl.robocup.org/game-logs/` (visited on 05/29/2024).

[41] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, „Visual object tracking using adaptive correlation filters“, in *2010 IEEE computer society conference on computer vision and pattern recognition*, IEEE, 2010, pp. 2544–2550.

[42] R. Tsai, „A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses“, *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.

[43] P. Bergmann, T. Engelhardt, E. Gareis, *et al.*, „Er-force 2023 extended team description paper“, p. 14, 2023. [Online]. Available: `https://www.robotics-erlangen.de/wp-content/uploads/etdp2023.pdf` (visited on 08/14/2024).

[44] D. Brown, „Decentering distortion of lenses“, *Photogrammetric engineering*, vol. 32, no. 3, pp. 444–462, 1996.

[45] P. Abeles. „Inverse radial distortion formula“. (2016), [Online]. Available: `http://peterabeles.com/blog/?p=73` (visited on 08/16/2024).

[46] J. Canny, „A computational approach to edge detection“, *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[47] N. Otsu *et al.*, „A threshold selection method from gray-level histograms“, *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[48] K. Okuma, J. J. Little, and D. G. Lowe, „Automatic rectification of long image sequences“, in *Asian conference on computer vision*, vol. 9, 2004.

[49] R. G. Von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall, „Lsd: A line segment detector“, *Image Processing On Line*, vol. 2, pp. 35–55, 2012.

[50] B. E. Bayer, *Color imaging array*, US Patent 3,971,065, Jul. 1976.

[51] F. I. I. S. Inc. „Specifications and frame rates“. (2022), [Online]. Available: http://softwareservices.flir.com/bfs-u3-50s4/latest/Model/spec.html (visited on 10/05/2024).

[52] M. R. Gupta and T. Chen, „Vector color filter array demosaicing“, in *Sensors and Camera Systems for Scientific, Industrial, and Digital Photography Applications II*, SPIE, vol. 4306, 2001, pp. 374–382.

[53] H. D. Cheng, X. Jiang, Y. Sun, and J. Wang, „Color image segmentation: Advances and prospects“, *Pattern recognition*, vol. 34, 12 Dec. 2001.

[54] N. S. Hashemi, R. B. Aghdam, A. S. B. Ghiasi, and P. Fatemi, „Template matching advances and applications in image analysis“, *American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)*, vol. 26, no. 3, pp. 91–108, 2016.

[55] I. Sobel and G. Feldman, „A 3x3 isotropic gradient operator for image processing“, *A talk at the Stanford Artificial Intelligence Project (SAIL)*, 1968.

[56] J. O. Smith. „Quadratic interpolation of spectral peaks“. in: Spectral Audio Signal Processing. (2011), [Online]. Available: https://ccrma.stanford.edu/~jos/sasp/Quadratic_Interpolation_Spectral_Peaks.html (visited on 08/19/2024).

[57] P. J. Rousseeuw, „Silhouettes: A graphical aid to the interpretation and validation of cluster analysis“, *Journal of computational and applied mathematics*, vol. 20, pp. 53–65, 1987.

[58] A. Lake. „Getting the most from opencl 1.2: How to increase performance by minimizing buffer copies on intel processor graphics“. (2014), [Online]. Available: https://www.intel.com/content/dam/develop/external/us/en/documents/opencl-zero-copy-in-opencl-1-2.pdf (visited on 08/26/2024).

[59] NVIDIA. „Opencl best practices guide“. (2011), [Online]. Available: https://developer.download.nvidia.com/compute/DevZone/docs/html/OpenCL/doc/OpenCL_Best_Practices_Guide.pdf (visited on 08/26/2024).

[60] A. M. Devices. „Amd app sdk opencl optimization guide“. (2015), [Online]. Available: https://www.amd.com/content/dam/amd/en/documents/radeon-tech-docs/programmer-references/AMD_OpenCL_Programming_Optimization_Guide2.pdf (visited on 08/26/2024).

[61] F. I. I. S. Inc. „Acquisitionuserbuffer.cpp“. (2024), [Online]. Available: https://softwareservices.flir.com/Spinnaker/latest/_acquisition_user_buffer_8cpp-example.html (visited on 10/18/2024).

[62] J. Sellner. „Buffer vs. image performance for applying filters to an image pyramid in opencl“. (2017), [Online]. Available: https://www.milania.de/blog/Buffer_vs._image_performance_for_applying_filters_to_an_image_pyramid_in_OpenCL (visited on 10/18/2024).

[63] I. Corporation. „Opencl developer guide for intel processor graphics“. (2019), [Online]. Available: https://cdrdv2-public.intel.com/773090/opencl-sdk_developer-guide-processor-graphics_2019.4-773088-773090.pdf (visited on 10/18/2024).

[64] Y. Emoto, S. Funasaka, H. Tokura, T. Honda, K. Nakano, and Y. Ito, „An optimal parallel algorithm for computing the summed area table on the gpu“, in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, IEEE, 2018, pp. 763–772.

[65] W. Kahan, „Pracniques: Further remarks on reducing truncation errors“, *Communications of the ACM*, vol. 8, no. 1, p. 40, 1965.

[66] jonas-apple. „Clcreateprogramwithil-alternatives“. (2022), [Online]. Available: `https://forums.developer.nvidia.com/t/clcreateprogramwithil-alternatives/216557` (visited on 10/17/2024).

[67] Youka, sfstewman, I. Grudev, and N. Stiurca. „Embed resources (eg, shader code; images) into executable/library with cmake“. (2019), [Online]. Available: `https://stackoverflow.com/q/11813271` (visited on 10/17/2024).

[68] killogre, Engineer, and M. Galindo. „Difference between quadtree and kd-tree“. (2012), [Online]. Available: `https://stackoverflow.com/q/13487953` (visited on 10/17/2024).

A Implementation details

As usability and latency are parts of the motivation for this work implementation considerations are important. Appendix A.1 describes the architectural considerations to achieve the goals. In Appendix A.2 the tradeoffs regarding memory are discussed. Appendix A.3 discusses the processing acceleration of axis aligned sum queries through Summed-Area Table (SAT)s. The integration of the OpenCL kernels into the program is discussed in Appendix A.4. Efficient range queries to maintain processing times are described in Appendix A.5.

A.1 Architecture

SSL-Vision has been designed as monolithic software. The graphical user interface necessary for configuration and manual calibration, image processing and publishing the geometry data are all bundled as one program. This architecture does not match the current usage at major tournaments, where a computer is located directly next to each camera. This leads to the necessity of remote desktop control software for configuration. On fields with more than one camera field lines and sizes have to be calibrated on each computer equally due to each instance publishing the geometry with the own camera calibration.

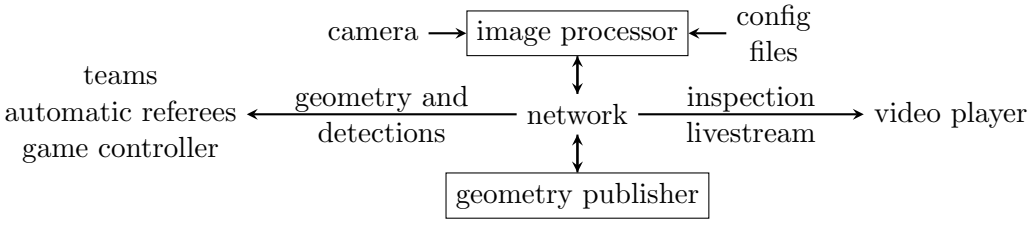


Figure A.1: Decentralized software architecture of the proposed solution

The proposed solution has been designed with these challenges in mind with a decentralized compartmentalized architecture depicted in Figure A.1. The design target of minimized configuration with no manual calibration enables the removal of a graphical user interface, replacing it with configuration files and an inspection video stream. The separate, central geometry publisher acts as a single source of truth for field information. As the geometry publisher component itself is not subject to the real-time requirement a easier maintenance, but less performant programming language has been chosen for this component with Python. Different from SSL-Vision the image processor component has only been designed to process input from a single camera to simplify the necessary processing architecture, as multiple instances of the image processor can also be started on a single computer.

The emergence of General Purpose computing on Graphics Processing Unit (GPGPU) in the last two decades has massively increased computational resources available for image processing tasks. The utilization of GPGPU resources has been tested during SSL-Vision

development [6]. This approach has been aborted due to the high memory transfer times with the dedicated Graphics Processing Unit (dGPU) and the bad adaptability of the following data reduction steps of the CMVision algorithm used [7]. The data transfer speed increases due to newer PCI Express generations, integrated Graphics Processing Unit (iGPU) zero-copy transfer capabilities and new programming interfaces allowing for asynchronous data transfer like Open Compute Language (OpenCL) change the usability for Graphics Processing Unit (GPU) usage in the context of this work. OpenCL has been chosen as GPU processing framework due to the widespread support over most graphics card vendors and operating systems.

While Rust has been evaluated as modern main processing programming language missing low-level library interfaces have lead to C++ as chosen programming language. The main processing is done single threaded without pipeline to minimize the latency and prevent resource competition at the cost of a potentially lower frame rate.

A.2 Memory management

dGPUs typically have a separate memory to the Central Processing Unit (CPU), in contrast to iGPUs, which typically have a unified memory architecture. Data locality and transfer are therefore important considerations.

Data transfer can take a significant amount of time. During development of the SSL-Vision software a 2.4 times increase of processing time when using GPU based processing in comparison to CPU based processing was observed due to data transfer times [6]. Newer Application Programming Interface (API)s with asynchronous data transfer techniques, hardware improvements and the zero-copy potential on systems with iGPU have increased the potential of GPU based processing acceleration. The Intel NUC reference platform utilizes an integrated GPU using the same memory as the CPU, which enables the zero-copy data transfers [58]. On Intel systems the memory requires a page and cache line boundary alignment, so a 4096 byte buffer alignment and 64 byte length alignment. Therefore the usage of OpenCL driver allocated memory with CL_MEM_ALLOC_HOST_PTR is preferred due to automatic correct alignments and advantages on other platforms like the usage of direct memory access and pinned memory on the NVIDIA [59] and AMD [60] platforms. clEnqueueMapImage and clEnqueueUnmapMemObject are then utilized to ensure correct synchronization. Some camera APIs like the Spinnaker SDK allow the usage of user provided buffers [61] which is used in this work to eliminate the need for an additional copy into correctly aligned and assigned memory.

OpenCL has two data memory types that can be used for image processing: buffers and images. Buffers offer more flexibility and data transferability due to their nature as linear memory arrays. Images are more limited in data types and data transfer due to their origin in graphics textures but on the other hand offer more interpolation, border handling options and potential hardware acceleration through texture caches and dedicated circuits. OpenCL images have been observed to achieve a comparable speed to local memory [62]. As all memory accesses are cached in the L3 cache local memory optimizations don't improve performance on the intel iGPU platform, while image textures are faster due to the utilization of the L1 and L2 texture sampling cache [63]. OpenCL images have therefore been selected as main memory type.

As the iGPU shares the memory with the CPU DDR optimized for latency is used instead of GDDR optimized for the throughput necessary for typical GPU operations. Most processing tasks proposed in this thesis have a low ratio of computation operations per memory operation and are therefore comparatively memory intensive. As the camera calibration does not change between frames the rectification could be done with a lookup table (LUT) to minimize comparatively computationally expensive undistortion operations.

A.3 Summed-area table

As the quadrant sum (see Section 5.2.7.3) is axis parallel this approach can be optimized with a SAT, also known as integral image. The values for each pixel are summed that each pixel contains the sum of itself and every pixel top, left and top-left of itself [29]. The sum of the values in the bottom right and top left corner subtracted by the values in the top right and bottom left corner equals the sum of all values in the rectangle in the original image. This reduces the computational complexity as the areas are already summed.

The generation of the SAT has been implemented as a simple row and column wise progressive prefix sum. Further parallelizing GPU optimizations like [64] have not been evaluated due to the naive approach being fast enough and concerns over performance regressions on iGPUs due to less optimal memory access patterns and increased memory bandwidth demands. Numerical differences compared to a direct summation without SAT have been observe on the laptop platform due to floating point imprecisions. Attempts at using the Kahan summation technique [65] have resulted in no differences due to compiler optimizations despite appropriate `volatile` annotations.

A.4 OpenCL kernel loading

In contrast to standardized CPU instruction sets GPU instruction sets and architectures differ between vendors and products. Programs intended to run on GPUs like OpenCL kernels therefore need to be compiled at runtime and distributed in source code or intermediate representation.

As the capability of loading the intermediate representation SPIR-V has apparently not been implemented by Nvidia [66] and as an additional external compilation command is necessary to generate the intermediate representation source code distribution has been chosen. Embedding the kernels as strings in the C++ code has the drawbacks of long recompilation times, no syntax highlighting and difficult multi-line kernel coding. Reading external OpenCL source files at runtime does limit the binary program to a fixed working or installation directory. CMake can be used convert OpenCL source files into C byte array or C multi line strings, although these methods either require the creation of an additional helper program or run at CMake configuration time instead of build time [67]. Using the linker to directly embed the binary file into the executable is the fastest embedded method but not portable across linker implementations. As compilation speed and portability have been prioritized during development direct usage of the linker has been chosen as the OpenCL kernel loading method.

A.5 Range queries

For the evaluation of robot hypothesis a range query is necessary for the blob assignment. Multiple spatial tree types like quad trees [36], KD trees [38] or range trees [37] are optimized for range queries. As the improved query complexity of range trees is not required the other tree types have been preferred due to simpler construction and implementation. As blob clustering is expected at robot positions KD trees are preferred due to fewer visited nodes according to “killogre” and “Engineer” [68]. A KD tree has therefore been chosen as data structure for blob range queries.