EINDHOVEN UNIVERSITY OF TECHNOLOGY

PDEng Mechatronic System Design

Mathematics and Computer Science Department

Autonomous Referee System

Feasibility Report of Stationary Camera

Project Team 2021

March 4, 2022

Contents

1 Introduction

In the robotic laboratory of TU/e, there are five stationary cameras on the roof, which enable full monitoring of the entire site. Therefore, the game situation can be analyzed and judged using the video stream of the cameras. The 2021 project team decided to try out the cameras as a way to build the referee system during their second week of group meetings. In the fourth week, due to the breakthrough of the referee system based on the football robot itself, the team decided to completely abandon the use of stationary cameras. In this report, in order to facilitate the review of the subsequent project team, information about the configuration and use of the camera is introduced, as well as the possibility of using the camera as a supplement to other systems.

2 Objective

Take video streams from five stationary cameras and apply image processing algorithm to obtain information about the game (such as the position and speed of the soccer ball or robots, etc.). Use this information as a supplemental validation of other data sources or use this data to build an automated referee system.

3 Advantages and Disadvantages Analysis

Compared with other solutions, the solution using the camera has the following advantages:

- The data obtained is more comprehensive: Compared with the data from the robot, the camera can monitor the whole field, so it can obtain some data that may not be collected by other solutions(such as robots themselves).
- The data obtained is unbiased: The data obtained from the camera is more credible than the solution obtained from the robot. The camera is equivalent to the audience outside the field. As a third party, the data obtained are often rarely interfered by other factors in the field. Referees can use this data without taking into account the possibility of false data provided by players as in other schemes.
- The original video stream can be directly obtained: In the scheme of using robot data, due to the different specifications and camera angles of the cameras used by each competition team on the robot, in order to make a decision more efficiently, we can only obtain the processed data from each competition team. and cannot make judgments directly based on the original video. It is difficult to make judgments when there are large differences in information from different teams. Using stationary cameras can solve this problem.

However, using stationary cameras also has the following disadvantages:

- Image processing requires a lot of work: As mentioned before, the camera gets raw video streams, and converting it into usable game data requires a lot of video processing.
- The generality of this scheme is relatively low compared to other schemes: first, there is no guarantee that all playing fields will have similar cameras available (a possible solution to this problem is presented in Chapter 5). Secondly, if the camera specifications and camera angles used in different venues are different, the parameters need to be adjusted each time the solution is used.

4 User Manual

4.1 Get the Video Flow[1]

The camera model used in robotics laboratory of TU/e is Bosch Flexidome IP Starlight 8000i. The cameras have 4K resolution and 3 different view axis: azimuth, elevation and roll. We can obtain the video in the software provided by Bosch, or obtain the video stream directly through camera IP. The two methods will be introduced separately:

1. Connect to the following WiFi[2]:

   Name: CoreMinicarWiFi
   Pass: xxxxxxxxx

   and the IP addresses of the cameras in the LAN are as follows:

#	IP Address
1	192.168.3.101
2	192.168.3.102
3	192.168.3.103
4	192.168.3.104
5	192.168.3.105

Table 1: IP address of the cameras

2. If you want to get the video frames from the software, then install:

   Project Assistant App version 2.1.1.48 for Windows:

   `https://www.boschsecurity.com/xn/en/solutions/video-systems/project-assistant/`

   Select Single camera assistant. And then select Enter IP address and enter the IP address to connect to the camera (shown in Figure 1 and 2 respectively).

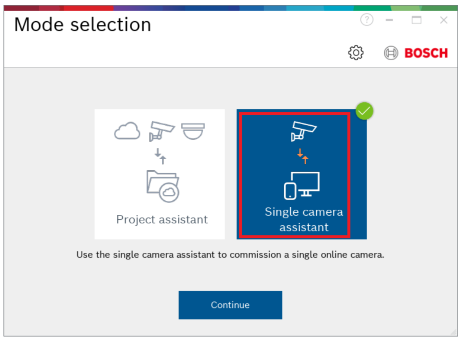
Mode selection
BOSCH
Project assistant
Single camera assistant
Use the single camera assistant to commission a single online camera.
Continue

Figure 1: Step 1

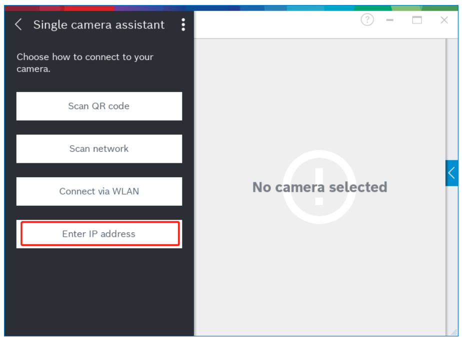

Figure 2: Step 2

   The login information of all the cameras is as follows[2]:

[1]The following information is updated in March 2022. When subsequent teams use this information, please confirm with the laboratory manager that the information has not changed.
[2]Passwords are hidden for privacy reasons, please ask lab management for the latest password.

Name: service
Pass: xxxxxxxxx

3. If you want to get the video flow directly with the python code, then the following command can be applied with opencv in python:

```
camera = cv2.VideoCapture('rtsp://service:core_PTZcam-123!@192.168.3.105/1')
```

where the "192.168.3.105" can be changed to the IP address of the camera that you want to connect to.

4.2 Image Processing

Before the project team decided to abandon the solution, we had tried some preprocessing of the video and made good progress. In this subsection, we will summarize the work that has been done.

The code we use comes from [1] (which is also included in the appendix). The main idea of the code is as follows:

- In the first frame of the video, we need to ensure that there are no moving objects or other distractions in the video, that is, a completely still background. After that, the images contained in the first frame of each frame of the video will be removed to achieve the principle of eliminating the background.
- We then convert each frame of the processed video to a black and white image, showing areas of the image where pixel intensity values vary significantly. The small white area is regarded as noise and discarded. The remaining white areas represent moving objects.

Figure 3 and 4 show the effect of video processing.

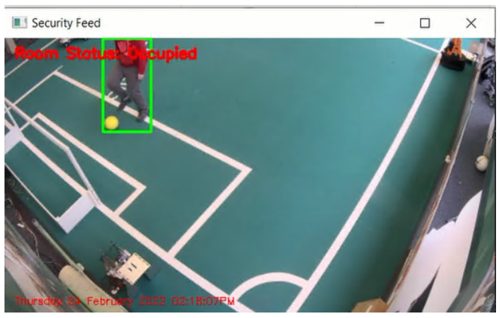

Figure 3: Effect of Image Processing 1

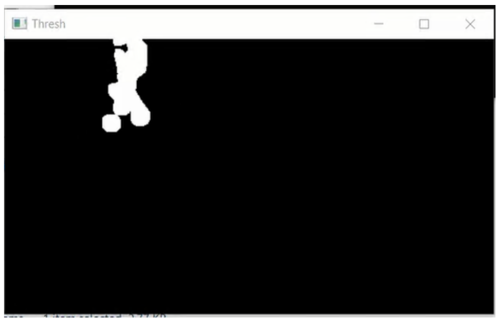

Figure 4: Effect of Image Processing 1

In Chapter 5, the steps that need to be taken to continue using the camera are outlined and introduced.

4.3 Remote Connection of the Cameras (Optional)

As mentioned above, if we want to connect the camera, we need to connect to the local network in the laboratory to access it. Due to privacy reasons, we cannot use the external network, such as the campus network, to directly access the camera (the method of port mapping for stationary cameras is not allowed for privacy reasons). However, using port mapping to connect other devices under the LAN is not blocked. In order to make debugging and video processing more convenient, the 2021 project team decided to use port mapping to remotely access the camera through the Raspberry Pi. The basic steps are as follows:

1. Install the Raspberry Pi OS and configure SSH and VNC. For the specific operation process, please refer to `https://www.youtube.com/watch?v=XZUx9ngvJTE`.
2. Install OpenCV and other necessary toolboxs for the Raspberry Pi.
3. Connect the Raspberry Pi to the LAN and set up DDNS and port mapping for it from the router management page.

Next, you can access the Raspberry Pi through the set domain and corresponding port for remote operation.

5 Further Improvement

5.1 Locating Moving Objects

Since the camera is fixed and the captured area doesn't change, we can calculate the actual object's position relative to the pitch from the position of the pixels represented by the moving objects in the video relative to the photo. There is a certain proportional relationship between the two.

5.2 Using Image Recognition to Distinguish Football and Players

Since the contours of soccer robots and soccer balls are different, we can use a certain contour recognition method to distinguish the two. In this way, we can automatically obtain the position and motion information of the target.

5.3 Replacing Stationary Cameras with Drones

We also mentioned above that one of the limitations of using stationary cameras is that some football fields do not have such cameras, but this method can be achieved by using drones. At present, most drones have a hovering function. We just need to adjust the height and position of the drone, and adjust the angle of the camera to cover the entire court, and we can achieve a similar effect to the fixed camera. But it should be noted that every time we use this code, we need to adjust some parameters (such as the ratio of the scene in the video to the actual scene).

6 Conclusion

Although the project team has abandoned the use of stationary cameras, the author still believes that these cameras are an important and effective way to verify other data sources or obtain fair data. The author still recommends the follow-up project team, if there is a chance, please try to use the camera or drone as a source or supplement to achieve a better referee effect.

7 Appendix

```python

import argparse
import datetime
import imutils
import time
import cv2


ap = argparse.ArgumentParser()
ap.add_argument("-v", "--video", help="path to the video file")
ap.add_argument("-a", "--min-area", type=int, default=500, help="minimum area size
    ")
args = vars(ap.parse_args())


if args.get("video", None) is None:
    camera = cv2.VideoCapture('rtsp://service:core_PTZcam-123!@192.168.3.105/1')
    time.sleep(0.25)


else:
    camera = cv2.VideoCapture(args["video"])


firstFrame = None


while True:

    (grabbed, frame) = camera.read()
    text = "Unoccupied"


    if not grabbed:
        break


    frame = imutils.resize(frame, width=500)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray, (21, 21), 0)


    if firstFrame is None:
        firstFrame = gray
        continue


    frameDelta = cv2.absdiff(firstFrame, gray)
    thresh = cv2.threshold(frameDelta, 25, 255, cv2.THRESH_BINARY)[1]


    thresh = cv2.dilate(thresh, None, iterations=2)
    (cnts, _) = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
        cv2.CHAIN_APPROX_SIMPLE)


    for c in cnts:

        if cv2.contourArea(c) < args["min_area"]:
            continue


        (x, y, w, h) = cv2.boundingRect(c)
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        text = "Occupied"


    cv2.putText(frame, "Room Status: {}".format(text), (10, 20),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 255), 2)
    cv2.putText(frame, datetime.datetime.now().strftime("%A %d %B %Y %I:%M:%S%p"),
        (10, frame.shape[0] - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.35, (0, 0, 255), 1)


    cv2.imshow("Security Feed", frame)
    cv2.imshow("Thresh", thresh)
    cv2.imshow("Frame Delta", frameDelta)
    key = cv2.waitKey(1) & 0xFF


    if key == ord("q"):
        break


camera.release()
cv2.destroyAllWindows()
```

References

[1] Noah Zhang, "Detect and track moving objects with python and opencv," [EB/OL], 2018, https://noahzhy.github.io/2018/02/02/%E7%94%A8-Python-%E5%92%8C-OpenCV-%E6%A3%80%E6%B5%8B%E5%92%8C%E8%B7%9F%E8%B8%AA%E8%BF%90%E5%8A%A8%E5%AF%B9%E8%B1%A1/.