International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 02 | Feb 2019 www.irjet.net p-ISSN: 2395-0072

Automated Pan-Tilt Unit for Target Tracking using Computer Vision

Sushant Patil[1], Prashansa Gharat[2], Priyanka Shetty[3], Ajinkya Nigade[4], Prof. Shudhodhan Bokefode[5]

[1,2,3,4]B.E. Computer Engineering, Dept. of Computer Engineering, Terna Engineering College, Maharashtra, India
[5]Professor, Dept. of Computer Engineering, Terna Engineering College, Maharashtra, India

---

Abstract - *This paper describes about object detection and tracking using pan-tilt mechanism. The BLOB detection is applied for detecting regions in digital image which vary in properties such as brightest pixel and color. Computer vision play important role to track moving objects. Registration of object is done using bmp file format which is capable of storing two-dimensional digital images. In this paper we are introducing tracking of object using grassfire algorithm.*

Key Words: **Pan-tilt mechanism, bmp file format, grassfire algorithm, Computer vision**

1. INTRODUCTION

Object detection is integral part of any vision-based Computer application. Object detection algorithm decides whether object of interest is present in scene or not and if it is present it locates the position of object in scene [4]. An efficient object detection algorithm should be able to decide whether object of interest is present in a scene or not irrespective of scaling and rotation of an object, depending on change in camera views point and illumination variations. BLOB extraction is intended to isolate the BLOBs (objects) in a binary image. A BLOB consists of a linked pixel group. The connectivity defines whether two pixels are connected or not, i.e. which pixels are neighbours and which are not [2]. A number of different algorithms exist for finding the BLOBs and such algorithms are usually referred to as connected component analysis or connected component labelling. In the following paper we use one of these algorithms known as grassfire algorithm. Grassfire can be performed both recursively and sequentially. Then finally calculating centroid of image object detection is done. And using this value the offset of centroid from centre of frame is calculated. This offset is used to calculate the pan-tilt angle of servos. Then the command for servo is provided to track the BLOB in continues manner.

1.1 Related Work

[1] John G. Allen, Richard Y. D. Xu, Jesse S. Jin The Continuously Adaptive Mean Shift Algorithm (Cam Shift) is an adaptation of the Mean Shift algorithm for object tracking that is intended as a step towards head and face tracking for a perceptual user interface. In this paper, we review the Cam Shift Algorithm and extend a default implementation to allow tracking in an arbitrary number and type of feature spaces.

[2] Israr Ahmed Khan Saeed This paper describes the application of a real - time vision system to track moving objects dynamically using image movement. The study encompasses experimental development of an intelligent system that works in ordinary light and is capable of tracking moving objects with unpredictable motion. In this research the task of mobile target tracking using a pan-and-tilt camera is considered with motion-based dynamic target tracking to deal with complex targets (shape and motion pattern unknown).

[3] Yuncheng Li, Yukun Zhu, Rui Zhang, Jun Zhou This paper describes a novel approach for generating object proposals for ball detection. This method can capture possible contours of balls and then transfers them into proposal bounding boxes. These proposal bounding boxes can be further used in class-specific object detection task.

2. Methodology

The aim of this paper is to design a system which performs auto-tracking. It is realized by performing single object detection and tracking.

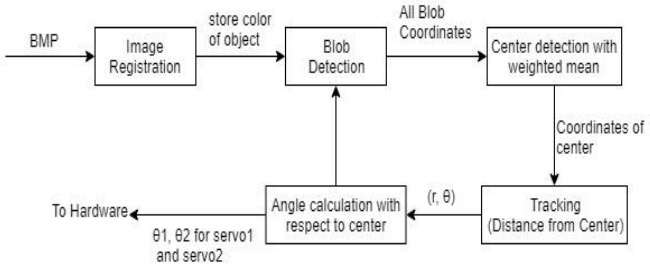

Fig No.1 Block diagram of Pan-tilt mechanism

The above figure illustrates how the workflow of the mechanism is carried out.

A digital image is nothing but the data numbers which indicates variations of blue, green, and red at particular location on the grid of pixels. On the computer screen we view these pixels as miniature rectangles that are sandwiched together [3].

In general, we view the screen as collection of pixels and these pixels on the screen having an X and Y position in a

two-dimensional window. However, the array has only one dimension, so it stores values in linear form.

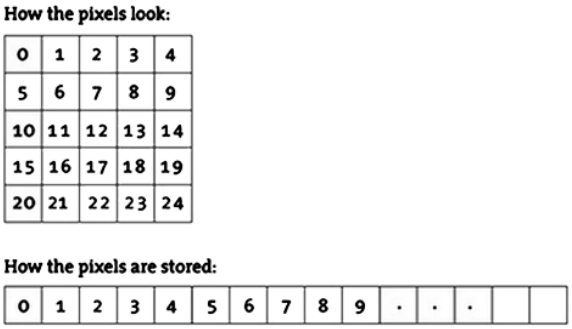

Fig No.2 Pixel arrangement

The below is the pseudo-code.

1. The image file is loaded in the Image Object
2. Retrieve the pixel's color for each pixel in the Image object and then set the display pixel to that color.

It is convenient to write the new pixels to a destination image. We'll understand this technique while implementing another simple pixel operation: threshold.

A threshold filter displays each pixel of an image in only, black or white. The state is set according to a particular value called threshold value. The pixel is colored white if its value is greater than the threshold and colored black if its value is less than threshold. Here the arbitrary threshold value is taken as 100.

Fig No.3

Fig No.4

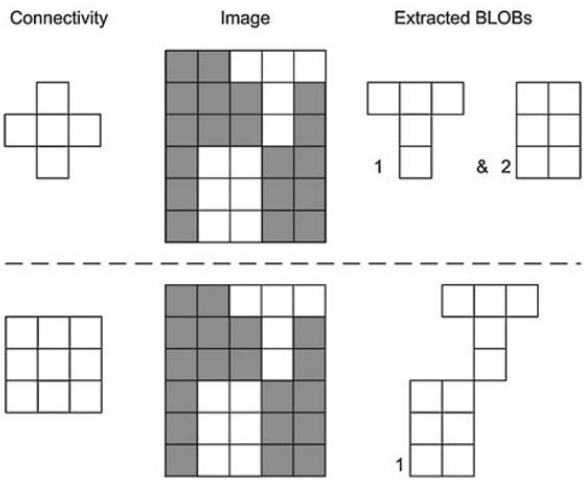

Fig No.5

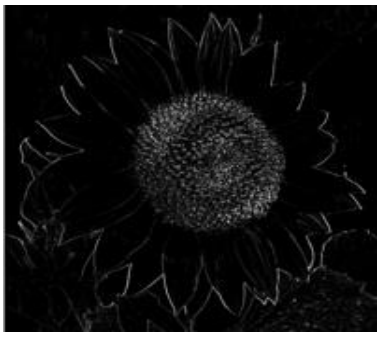

Fig No.6

The BLOB extraction is done to isolate the BLOBs in a binary image [5]. A BLOB consists of a group of connected pixels. The connectivity is defined by checking which pixels are neighbors and which are not.

There are various existing algorithms available for finding the BLOBs and these algorithms are usually referred to as connected component labelling or connected component analysis. An algorithm named Grass-Fire algorithm is defined below. Here 4-connectivity is used for simplicity.

Now the next step is of classifying the different BLOBs. For example, if the circles are taken then we need to classify each BLOB as either a circle or not a circle and another example is human vs. non-human BLOBs.

2.1 Grass Fire algorithm

The grassfire can be implemented recursively as well as sequentially.

2.1.1 Recursive Grassfire

The recursive grassfire algorithm starts its process from the upper left corner of the binary image. It then scans the image from top to bottom and from the left to right.

At some point the scan detects the white pixel (object pixel) and the grass fire algorithm comes into existence. The algorithm searches for all four direction around the detected pixel and tries to find another white pixel (object pixel). The process continues for all such object pixels until all the pixel are visited and detected. Firstly, in the output image it gives this pixel an object label (basically a number) and secondly it "burns" the pixel in the input image by setting it to zero (black). Setting it to zero indicates that it has been burned and will therefore not be part of yet another fire.

[1] The algorithm can be implemented very efficiently by calling functions. Such an algorithm is said to be recursive. Along with that the care should be taken to ensure that the program is terminated properly as recursive algorithms have no built-in termination strategy. Another danger is that a computer has a limited amount of memory allocated to function calls. And since the grass-fire algorithm can have many thousands function calls queued up, the computer can run out of allocated memory.

2.1.2 Sequential Grassfire

In sequential grass-fire algorithm the scanning of the image is done similar to recursive grass-fire algorithm i.e., from top left to the bottom right. On finding the object pixel it does two things. Initially, this pixel in the output image is given an object label 1, and then after this step it "burns" the pixel in the input image by setting it to the object label zero i.e., black. The next step is checking the neighbors and to look if any of them is an object pixel. Till these steps it works exactly the same as the recursive grass-fire algorithm. But now comes the part which differs from recursive grass-fire algorithm, if any of the neighbor is an object pixel, they are labelled 1 in the output image and burned i.e., set to zero in the input image. And then they are placed in a list. Next step is performed by taking the first pixel from the list and checking its neighbors. If any of them are found to be the object pixels they are labelled in the output, set to zero in input image and then placed in the list. This process is continued until all pixels in the list have been investigated. The algorithm keeps working continuously by following the scan path until it meets the next object pixel, which is then labelled 2, and begins a new grass-fire.

2.2 Extracting the Blob

This includes classifying the different blobs. This process called classification process consists of two steps. In the first step each BLOB is represented by a number of denoted features, characteristics, and secondly some method is applied in order to compare the features of each BLOB with the features of the type of object we are searching for. For example, to find circles we need to calculate the circularity of each BLOB and then compare the results to circularity of a perfect circle.

Feature extraction is about keeping the important and relevant information and ignoring the rest. A number of features of the blob can be calculated following are the features with their descriptions:

2.2.1 Centre of mass

It is point on the object where you can place the finger in order to balance the object. It is given by-

$$x_c = \frac{1}{N}\sum_{i=1}^{N} x_i, \qquad y_c = \frac{1}{N}\sum_{i=1}^{N} y_i$$

Where, N is the number of pixels in the BLOB
(xc, yc) is center of mass

2.2.2 Area of a BLOB

It is the number of pixels the BLOB consists of. This feature is used to remove the Blob that is too tiny or to large.
Bounding box of a BLOB: - It is the minimum rectangle that contains the BLOB. It is defined by tracing through all pixels for the BLOB and then finding the four pixels with the minimum x and y value and maximum x and y value. The height of the bounding box is given by ymax – ymin and width is given by xmax – xmin.

Bounding box ratio of a BLOB: - It is ratio of the height of the bounding box to the width. It determines the elongation of the BLOB, i.e., is the BLOB high, long or neither.

2.2.3 Compactness of a BLOB

It is defined as the ratio of the BLOB's area to the area of the bounding box. For example, this can be used to distinguish between hand with outstretched fingers vs. fist.

$$\text{Compactness} = \frac{\text{Area of BLOB}}{\text{width} \cdot \text{height}}$$

3. Hardware

Fig No.7 Pan-Tilt Mechanism with Servos connected

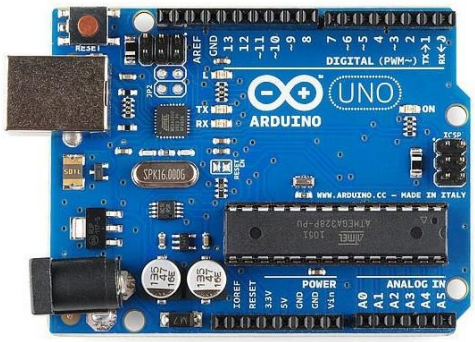

Fig No.8 Arduino UNO

1. Logitech camera (pre-attached USB type A)
2. Arduino Uno(microcontroller) for controlling servo.
3. Pan-tilt mechanism (Figure No. 7)
4. 2x micro-servos (SG-90)
5. Power Bank for servo power supply
6. Laptop for image processing
7. Mini USB Cable USB 2.0 Type A to Mini B Cable Male (for connecting Arduino Nano to computer)
8. 6 jumper wire (male to female) (connecting servo to Arduino)

4. Conclusion

In this paper, we have proposed technique for object detection. We have shown detection of BLOB using grassfire algorithm. Using this we try to capture all pixels in frame. The results showed that the proposed method performance is comparable to other methods are better which is reported in the literature. We'll be in the future Develop a more precise detection framework Performance in case of non-homogeneous colored object Distribution. On the basis of different experiments tracking model has better real-time performance and accuracy.

ACKNOWLEDGEMENT

We thank Professor Shudhodhan Bokefode for contributing their valuable guidance and time.

REFERENCES

[1] John G. Allen, Richard Y. D. Xu, Jesse S. Jin. Object Tracking Using CamShift Algorithm and Multiple Quantized Feature Spaces. ACM-ICPS 2004

[2] Israr A. Saeed, Nitin V Afzulpurkar. Real Time, Dynamic Target Tracking using Image Motion International Conference on Mechatronics, 2005 IEEE

[3] Yuncheng Li, Yukun Zhu, Rui Zhang, Jun Zhou. Shape detector for generic ball detection. DOI: 10.1109/BMSB.2015.7177234

[4] Balasubramanian, A., Kamate, S., & Yilmazer, N., 2014. Utilization of robust video processing techniques to aid efficient object detection and tracking. Procedia Comput.Sci.36,579–586 doi: 10.1016/j.procs.2014.09.057

[5] Ben Ayed, A., Ben Halima, M., & Alimi, A.M., 2015. MapReduce-based text detection in big data natural scene videos. Procedia Comput. Sci. 53, 216– 223. doi: 10.1016/j.procs.2015.07.297

[6] Chakravarthy, S., Aved, A., Shirvani, S., Annappa, M., & Blasch, E., 2015. Adapting Stream Processing Framework for Video Analysis. Procedia Comput. Sci. 51, 2648–2657. doi: 10.1016/j.procs.2015.05.372

[7] Chandrajit, M., Girisha, R., & Vasudev, T., 2016. Multiple Objects Tracking in Surveillance Video Using Color and Hu Moments. Signal Image Process. An Int. J. 7, 15–27. doi:10.5121/sipij.2016.7302

[8] Coşkun, M. & Ünal, S., 2016. Implementation of Tracking of a Moving Object Based on Camshift Approach with a UAV. Procedia Technol. 22, 556–561. doi: 10.1016/j.protcy.2016.01.116

[9] Elhariri, E., El-Bendary, N., Hassanien, A.E. & Snasel, V., 2015. An Assistive Object Recognition System for Enhancing Seniors Quality of Life. Procedia Comput. Sci. 65, 691–700. doi: 10.1016/j.procs.2015.09.013

[10] Ha, H. & Ko, K., 2015. A method for image-based shadow interaction with virtual objects. J. Comput.Des. Eng. 2, 26–37. doi: 10.1016/j.jcde.2014.11.003