ISSN: 0976-9102 (ONLINE)
DOI: 10.21917/ijivp.2019.0281

AN EFFICIENT AND EFFECTIVE TECHNIQUE FOR MARKER DETECTION AND POSE ESTIMATION USING A MONOCULAR CALIBRATED CAMERA

J. Jayakumari, S. Anand, M. Madan and R. Misha
Department of Electronics and Communication Engineering, Mar Baselios College of Engineering and Technology, India

Abstract
The basic idea of all image processing methods is to process images captures by the camera device. Using image processing techniques, it is possible to find and process markers, faces, natural objects, simple shapes and other objects. One of the techniques is based on the use of markers, both artificial and natural. Markers on image are used for determining objects around or about the current location of the user. This method is also known as marker-based tracking. Markers in this method can provide a reference coordinate system for producing graphical overlays over the real components on the images. Marker identification is an important part of marker-based tracking process. A good marker is considered to be a marker that can be easily and reliably detected under different circumstances. The process of marker detection consists of two stages: Image pre-processing and Identification of potential markers. Marker processing algorithm and steps related to image processing and detection of potential marker stages are: Acquisition of a source image, Image pre-processing and Detection of potential markers. This paper discusses a simple marker detection algorithm and its simulation in MATLAB for verification. A potential application of marker detection to estimate the pose of the marker in real time is also implemented and the results are discussed.

Keywords:
Marker, Processing, Detection, Tracking, Matlab

1. INTRODUCTION

Detecting and tracking markers is a useful process in many fields such as automated factories, vision based landing, augmented reality to name a few. Our project provides a simple way to detect and track marker or object in a real world scene and then to estimate its position and orientation, in 3 dimensional space from a video stream. Any point in a real world has freedoms of translation and rotation along their respected axis. This amounts to whole 6-dimensions of movable space. The image from a scene is picked using an image capturing device such as a camera and then processed through a number of well-defined steps. From there, it is trivial to overlay 3-dimensional content to the video stream in real-time, in a way that makes it appear consistent with the scene. Image processing methods is to process images captures by the camera or any image capturing device. Further on after preprocessing and processing, it is possible to find and Process markers, simple shapes and other objects. Markers on image are used for determining objects around or current location of the user. Identification or detection is an important part of marker-based tracking process. A good marker is considered to be one that can be easily and reliably detected under different circumstances. The additional errand of making the marker, unique and singularly detectable under nominal circumstances adds up to the credibility of the project. The process of marker detection consists of multitudes of steps through which the image from any video stream under any resolution is made to go through.

The outcome is to be able to have a set of well-defined points detected from the marker, in our project we detect the corners of a pre-defined fiducial marker that we have made. Fiducial markers already exist, the real sense of input is the algorithm to determine the corners, both inner and outer. Lastly with the help of the detected points, i.e. the corners, we compute the pose of the image w.r.t the camera element. The detected points along with the estimated output from the pose computation is displayed real time as the program is running.

2. PROBLEM STATEMENT

The major issue tackled in our project is to detect corners or strong points of a marker placed in a real world scene. This would necessarily include steps to remove noise and irregularity from the scenes, sharpen and make the edges and borders more definitive and finally to detect the marker per say. Now most of the existing technologies or algorithms that are found to do this, work on obtaining the corner lines and then to solve the multiple line equations and to formulate the points located at the edges. One most commonly used method is using Hough Transform and then to sequentially use those obtained values in the line equation and obtaining the corner points. Since we are working on MATLAB, there is an inbuilt function called ‘corners’, along with various supplements that find out corners. But the later method worked well only on perfect computer generated images or markers that were created using any graphic design software or CAD.

The former method involves stringent formulation and dedicated computational time to detect the set of corners. This would be acceptable in case of a still image, but when transcended to moving picture or motional domain, these computational times would add large overheads which is unacceptable. Say we have successfully been able to detect the corners of the marker. The next logical step is to estimate the camera pose. This would certainly add up to why we were detecting the corners of the marker. The main aim to detect corners were; that detection of the corners would essentially mean we have detected the whole marker per say and the corner co-ordinates could be used to compute the camera pose. In our project we are dealing with the pose estimation of a monocular camera that has been calibrated

3. PROPOSED ARCHITECTURE

The first step is to capture the image, still or from the moving picture. This is done by taking snapshot of moving video. Then this image is pre-processed to enhance the features in order to determine the corners/vertices. The outer most four points are obtained first and then for the purpose of pose estimation, the inner four points/corners are also detected.

The marker is checked with the original marker, which is already pre fed to ensure that only the desired marker is being taken. Using the array of corners detected, along with the global parameters of the marker, the pose of the marker is estimated. For the sake of easing complexity, monocular calibrated cameras are used.

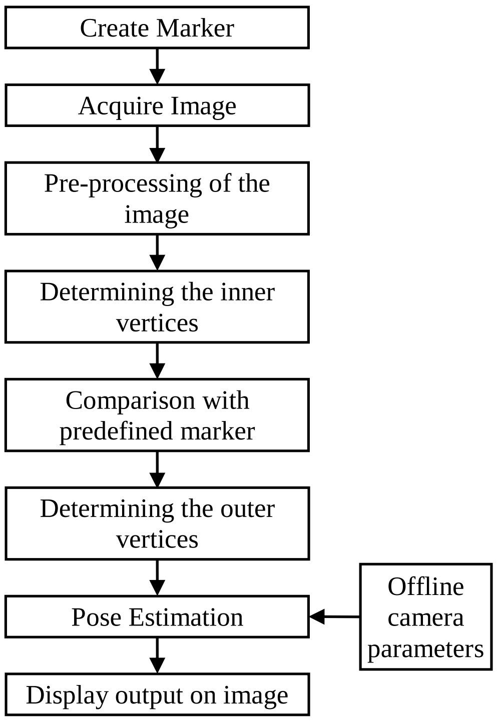
Create Marker
Acquire Image
Pre-processing of the image
Determining the inner vertices
Comparison with predefined marker
Determining the outer vertices
Pose Estimation
Offline camera parameters
Display output on image

Fig.1. Block diagram of the system

The camera used is calibrated beforehand with checkerboard patterns, and the camera intrinsic are obtained. Along with the pixel values (i.e. the corners), the global parameters and the camera intrinsic, the translation and rotation of the camera w.r.t the marker is found out.

4. FLOWCHART OF PROPOSED SYSTEM

The marker is placed horizontally on a flat surface and the camera is placed at a known distance with respect to the marker. The marker used is a fiducial marker, which is a binary marker. The first step is to obtain the image via the camera connected to PC, in which Matlab is functional. After acquiring the image, it is then converted into greyscale image by using the function. One may want to introduce an image processing tool using gray level images, as opposed to color, not because of the "format" of gray level images, but because the inherent complexity of gray level images is lower than that of color images.

The next step after converting the normal image to greyscale is to convert the greyscale image into binary image. Converting to binary is often used in order to find a Region of Interest -- a portion of the image that is of interest for further processing. The intention is binary, "Yes, this pixel is of interest" or "No, this pixel is not of interest".

Here the already converted binary image is inverted into its opposite image. It will make the marker which was previously white with black outers into one with black inners with white outers. This is to make the marker image in sync with the predefined fiducial marker. Those pixels which have a binary value equal to one, that is white. So basically whatever part of the binary image that was previously white is now inverted to black and vice-versa. 'imbinarize' in which we specify the threshold levels that is set to that of greyscale threshold of the greyscale image.

In this step the background details which includes the noise and lossy information present in the image is removed. 'imclearborder' function on Matlab which removes the white components present at the periphery of the border and turns them into black is made use of. The removal of white border is done by suppressing pixels lighter than that of its surroundings. This step is progressive towards getting the region of interest in the image.

The marker used is a binary marker. So it has two possible values for each pixel. When converted a grayscale image to a binary image, the marker would stand out compared to other objects in the image. In other words, its area would be larger compared to the area of other objects in the image. The function 'bwareafilter' is used to acquire largest areas in this image.

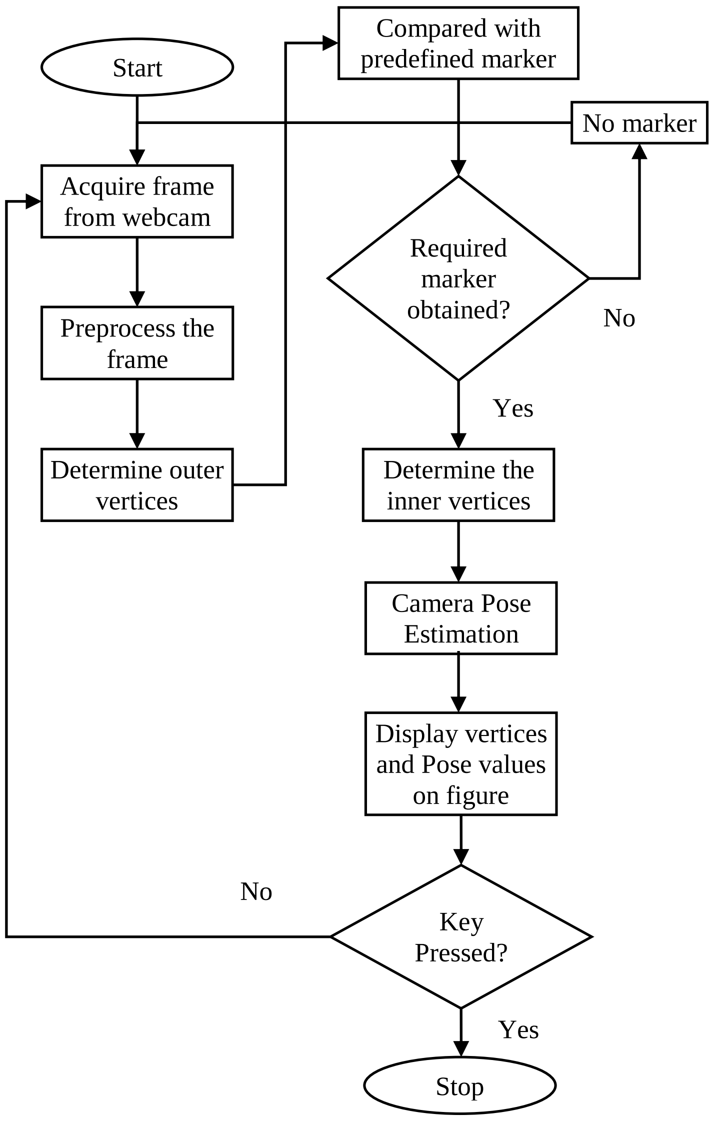
Start
Compared with predefined marker
No marker
Acquire frame from webcam
Required marker obtained?
No
Preprocess the frame
Yes
Determine outer vertices
Determine the inner vertices
Camera Pose Estimation
Display vertices and Pose values on figure
No
Key Pressed?
Yes
Stop

Fig.2. Flowchart representing the working

The function 'bwareafilter', it acquires the largest areas in this image. In our project, the specified function finds three largest areas. If the marker is in the field of view, then one of the largest areas would be at match with the input marker.

In order to detect the corners of the marker, we need to enhance the edges of the marker. Hence, morphological processes which removes these imperfections is used on the image. Opening is one of the morphological methods. The function to carry out opening is 'imopen'. Opening is erosion followed by dilation. Erosion is used for elimination unwanted detail, and dilation bridges the gap in the image. This process is carried out by using a structural element, which is a small template, and is positioned at every possible location in the image, and it compares with its neighboring pixels.

This step also does morphological image processing. In this step, closing is done, which is another method used in morphology. It is dilution followed by erosion, and this also uses a structuring element in order to carry out this process. Various structuring elements that can be used are: - Square, diamond, disc, line, cuboid, cube, etc. In the Fig.1 and Fig.2, we have used the structuring element cube, as we need a sharp corner for detection.

The marker that needs to be detected is created. This is done in order to compare the marker detected from the image to this predefined marker created. In doing so, it can be confirmed that this is the marker required, and then determine its configuration. The marker is created by using a set of binary values, and is stored in the destination file by using the function 'imwrite'.

After pre-processing the image, we can now detect the corners of the marker present in the image. The image obtained after pre-processing is analysed using the function 'regionprop', which gives us details of the objects present in the binary image. The details that can be determined are: area, euler number, PixelList, major axis length, minor axis length, etc. From these properties, first use Euler Number. Since the marker has a hole, that is a white region within a black region, its Euler number would be zero, as Euler number is equal to the difference between number of objects and number of holes, hence our object is selected.

The next step would be to determine its vertices. For this the property PixelList, which gives the location of each pixels in the image. After the object is selected, it is only needed for PixelList of that specific object. The x and y coordinates gives us the location in each pixels, hence in PixelList, we have x and y values for each pixels in that object. First the sum and difference of the pair is taken. From the set of sum and differences computed for coordinates of the pixels, the maximum and minimum values for each is taken. The row index of these corresponding values are used to get the pair of coordinates from the PixelList. These pair of coordinates are the vertices or corners of the marker.

From the pair of coordinates obtained, it allows us to crop the marker from the image. This is done in order to compare with the pre-defined marker. The function used here is 'fitgeotrans', where it fits the geometric transformation to the pair of points obtained.

The pre-defined marker is made to rotate to 90 degrees, 180 degrees, 270 degrees. All these orientations are stored in a variable for reference. If marker was placed at an angle, the fitgeotrans transforms the image to the closest orientation. It is then made to compare with the reference images. If the marker obtained from the image matches with any one of the orientations, the marker obtained is correct, also it can determine the top left position, used as reference when in situations of rotation.

As it is verified that the required image was obtained, need to determine the inner corner points of the image is necessary. The concept for finding the inner vertices is very similar to that of finding the outer vertices that has been estimated before. The only difference is that instead of using Euler number, the function 'bwboundaries' is used. From this function, the list of pixel coordinates that were used to create the inner boundary is obtained. Then, similar to the previous method, the sum and difference of the pixel coordinates, and from this set of sum and difference is found, thereby obtaining the maximum and minimum values. The row index of this maximum and minimum values are used to find the pixel coordinates of the vertices.

In this step we input the world location values of the marker. World location values are sets of values pertaining to the strong point corners that we have detected w.r.t. the periphery of the marker, i.e. w.r.t. the perimeter of the printed marker paper. Four unique sets of these world coordinates that is at par with the four side rotation that can happen in the marker with respect to the centre is taken. The camera internals have been computed beforehand by feeding in a set of checkerboard pattern, in all orientations and distances All these outputs obtained are then displayed on the image by using the functions 'insertShape' and 'insertText'.

5. RESULTS AND DISCUSSIONS

The program was formulated initially to address the marker detection on still images. The marker was first defined as a square black marker without any information contained in it. The initial steps were to detect the corners of this square marker. Through the progress of the project, it was realized that a basic square marker would not add credibility to the main aim. The square black marker was transcended to a binary marker or a fiducial marker that had a definitive spatial geometry to it. This would make the detection of the marker more definitive and non-redundant. The general condition for the detection of the marker as seen was it required normal or adept lighting. The amount of light is a decisive factor that would hinder the detection process if the scene is not lit up well. The results are given in Fig.3-Fig.8.

The original implementation was aimed at standard definition cameras that would provide well resolved image scenes without too much noise. The final source code can be implemented across a wide variety of cameras ranging from a normal laptop webcam to even high definition cameras. The catch in the selection of camera is that the lower the camera resolution, the real time detection and tracking would lag. This is mainly because of multiple reasons of which one being the resolution of the camera would be very low. The lower the resolution, more time the preprocessing stages would take and in turn the output would be available only a quirk slower.

The rate at which the frame is grabbed from the video scene is also a factor that affects tracking and detection. The better equipped camera would mean, better detection and tracking. Apart from the uniqueness of the marker being used, the color is independent. The color of the marker does not hinder tracking as long as the inner color remains white.

It determines the pose of a monocular calibrated camera. In order to calibrate the camera, we have taken sets of picture pertaining to checker board patterns of cell size 4.1cm each. The calibration information is necessary to be accurate and fulfilling as it would determine the internal parameters of the camera, in turn leading to pose estimation. Initially four of the corner points were detected, but since those points would cancel out while computing the rotation and translation, fetching and feeding more than four (eight in our project) became imperative.

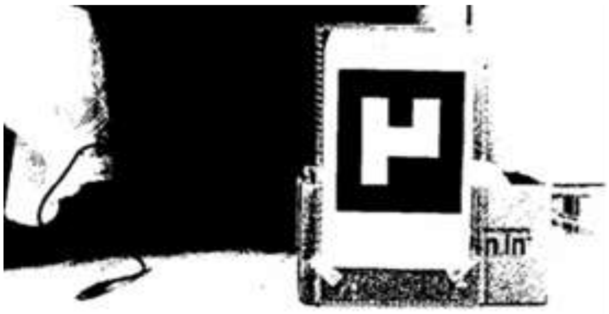
Fig.3. Marker used

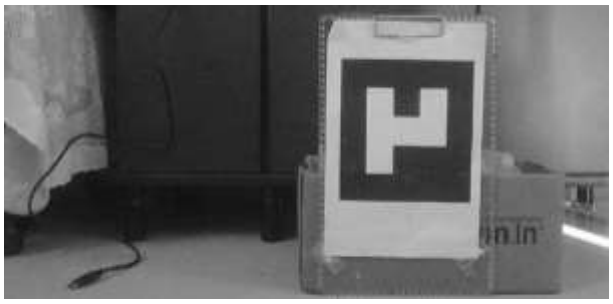
Fig.4. Greyscale Image of original scene

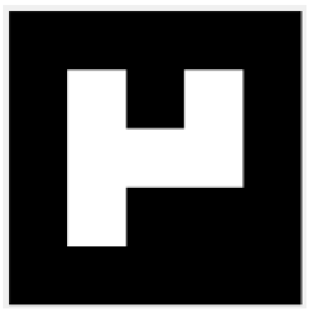
Fig.5. Binary Image of the original scene

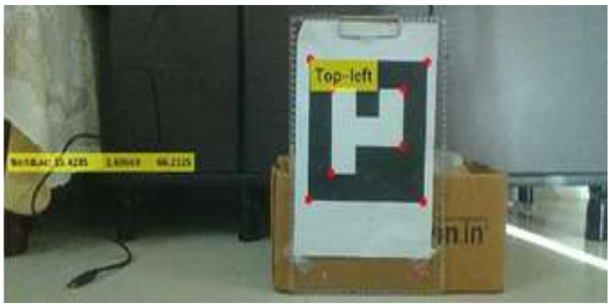

Fig.6. Inverted binary image

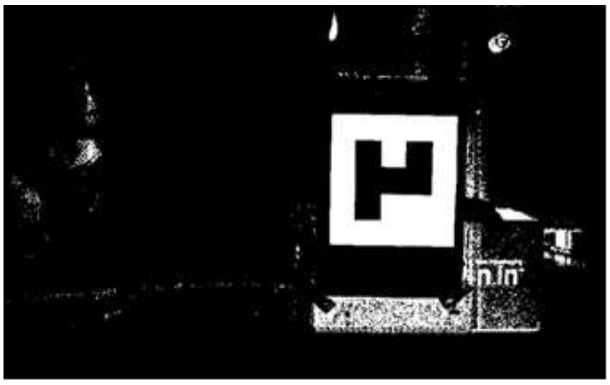
Fig.7. After removing all other elements

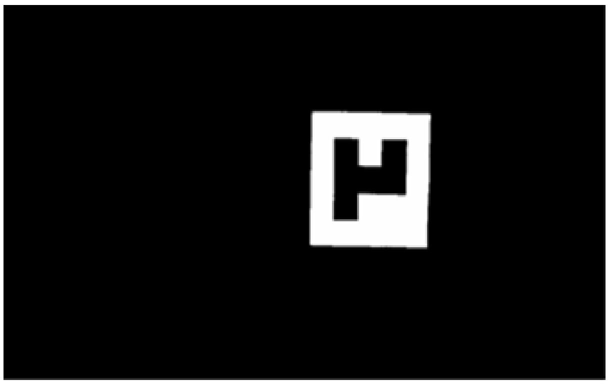
Fig.8. Output with detected points and pose estimated

6. CONCLUSIONS

The final software implementation of marker detection and camera pose estimation was carried out successfully by step by step prognosis and trial-error methods. The numerical constants used in the code are researched and found out parameters that would yield maximum results. The algorithm used was verified by placing the marker at known distances and tilted at all angles. The obtained values are at match with the original values, with minor distortions owing to the lighting and the frame rate of the camera used. The step by step output at various stages are shown below.

REFERENCES

[1] C.R. Jung and R. Schramm, "Rectangle Detection based on a Windowed Hough Transform", *Proceedings of 17th Brazilian Symposium on Computer Graphics and Image Processing*, pp. 20-24, 2004.

[2] Martin Hirzer, "Marker Detection for Augmented Reality Applications", Technical Report, Institute for Computer Graphics and Vision, Graz University of Technology, 2008.

[3] Vadim Beglov, "Object Information based on Marker Recognition", Master Thesis, Department of Computer Science, University of Eastern Finland, 2013.

[4] Sanni Siltanen, "Theory and Applications of Marker-based Augmented Reality", Available at: https://www.vtt.fi/inf/pdf/science/2012/S3.pdf.

[5] Oliver Toole and Dave Dolben, "Marker Detection and Tracking for Augmented Reality Applications", Available

at: https://stacks.stanford.edu/file/druid:hg315sw1225/Toole_Dolben.pdf

[6] Stefan Hinterstoisser, Selim Benhimane and Nassir Navab, “N3M: Natural 3D Markers for Real-Time Object Detection and Pose Estimation”, *Proceedings of IEEE 11th International Conference on Computer Vision*, pp. 14-21, 2007.

[7] E. Olson, “Apriltag: A Robust and Flexible Visual Fiducial System”, *Proceedings of IEEE 11th International Conference on* Robotics and Automation, pp. 3400-3407, 2011.

[8] K.M. Yi, E. Trulls, V. Lepetit and P. Fua, “Lift: Learned Invariant Feature Transform”, *Proceedings of IEEE International Conference on Computer Vision*, pp. 467-483, 2016.

[9] K. He, G. Gkioxari, P. Dollar and R. Girshick, “Mask R-CNN”, *Proceedings of IEEE International Conference on Computer Vision*, pp. 2980-2988, 2017.

[10] T. Simon, H. Joo, I.A. Matthews and Y. Sheikh, “Hand Key Point Detection in Single Images using Multiview Bootstrapping”, *Proceedings of IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 1145-1153, 2017.