Computer Vision - Proposal

Argentina Ortega Sáinz
Nicolás Alfonso Laverde

November 20, 2014

Virtual Whiteboard.

The idea is to develop a tracking system for a predefined marker that can be used to paint as if it was a virtual whiteboard. The system would have the capabilities to recognize different "virtual markers" (but one at a time), each one with a different color, and to detect the strokes or traces made with them. Detected strokes are then draw. The aim behind it is to be able to project the drawings (traces of the marker) into a screen using of a video beam, and even to the point of saving them. This will allow the digitalization of notes made by the user of the system (marker carrier), such as a teacher, for later distribution and inspection.

Objective

Design a vision system able to recognize the strokes made with a predefined marker in a flat surface and to transform the strokes into drawings for later projection.

- Design, calibrate and synchronize an orthogonal array of two cameras in order to capture videos simultaneously of a scene from different perspectives.
- Process two videos from different perspectives in order to detect strokes performed by a pre-defined marker.
- Simulate a whiteboard by drawing the detected strokes and later projecting the draws into a screen for visualization.

Motivation

Obtaining the position of a certain object in real time has other applications, most interesting for us the computation of position of a robot. By using a simple example such as the virtual whiteboard to familiarize ourselves with the Computer Vision concepts, it could be possible in the future to extend this method to a much more complex application.

Description

We will use an orthogonal arrange of cameras. This will allow us to reconstruct the 3D coordinates of the marker using 2D synchronized images, following a similar approach as mentioned in [1]. The board plane contains the "x" and "y" coordinates, which are

obtained by the re-projection of points from the two view to the board plane. The "z" component (depth) will make two modes distinguishable: the "writing mode" and a "moving mode" . A similar approach is mentioned in [2], but using hand gestures instead of markers.

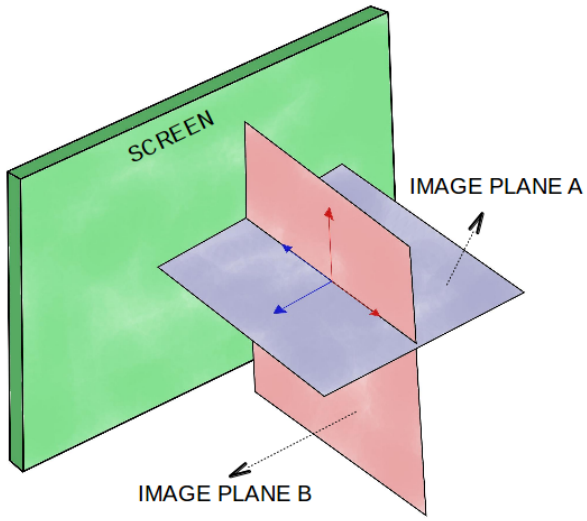

Figure 1: Planes Scheme.
The figure shows the three plains relevant to the system; the screen plane (in green) and the two images planes (red and blue).

The "writing mode" implies that the marker is close enough to the surface (screen) to enable the detection of the strokes, and therefore, drawing. The "moving mode" means that the marker can be moved, but the system will not detect the strokes of such movements, and therefore, no drawing is performed.

In other words, an "active drawing" volume is defined, which allow us to make the distinction between the modes. This active plane is parallel to the screen plane.

For the the system's output (drawing), we chose to back-project into the screen using a video-projector. Doing so, we can test the system at the same time; i.e. making the strokes and seeing the drawing correspondent to the movement of the marker. Projecting for the back allows us to avoid blocking the projected beam.

Detection of different colors in different markers will allow the user to paint on the screen. A possible approach is mentioned in [3], [4] or [5], to name a few.

Set-Up

The setup consists of two cameras arranged orthogonally; one camera is placed on the ceiling, over the board, and the other one is place in the side. Both cameras are to be mounted as close to the board as possible.

The selected cameras are two LiveCam from Microsoft, connected via USB. They have a frame rate of 30fps and a high resolution (up to 1920x1080). Additional, it offers an auto-focus feature from 0.1m to 10m.

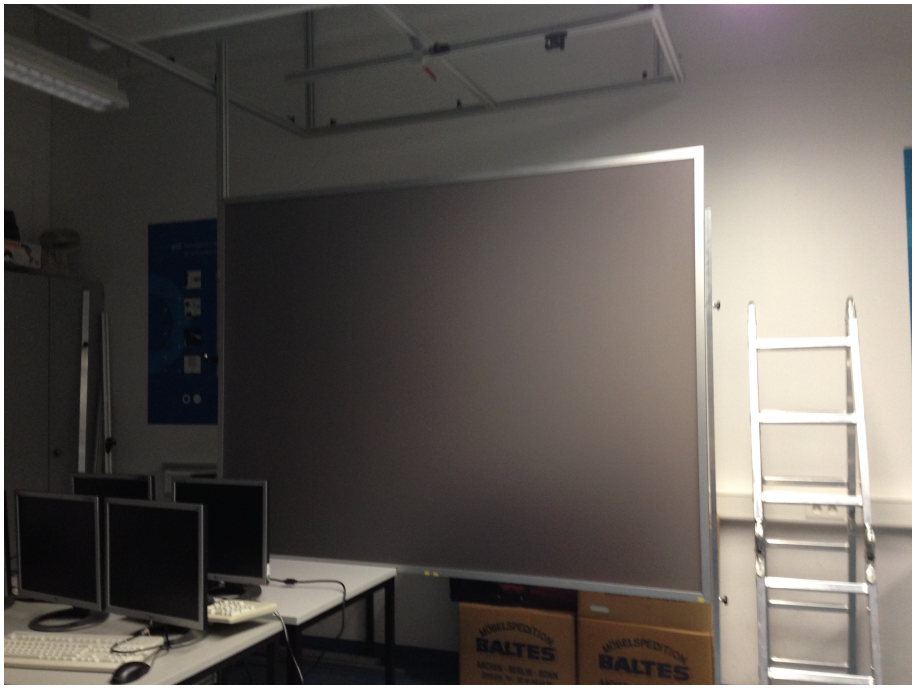

Figure 2: Initial setup.
In the picture, the glass screen which allows us to back-project. The setup also illustrates the position of the cameras as well as the additional light source.

A halogen lamp was added as a result of non-uniform lighting in the room (as seen in the picture).

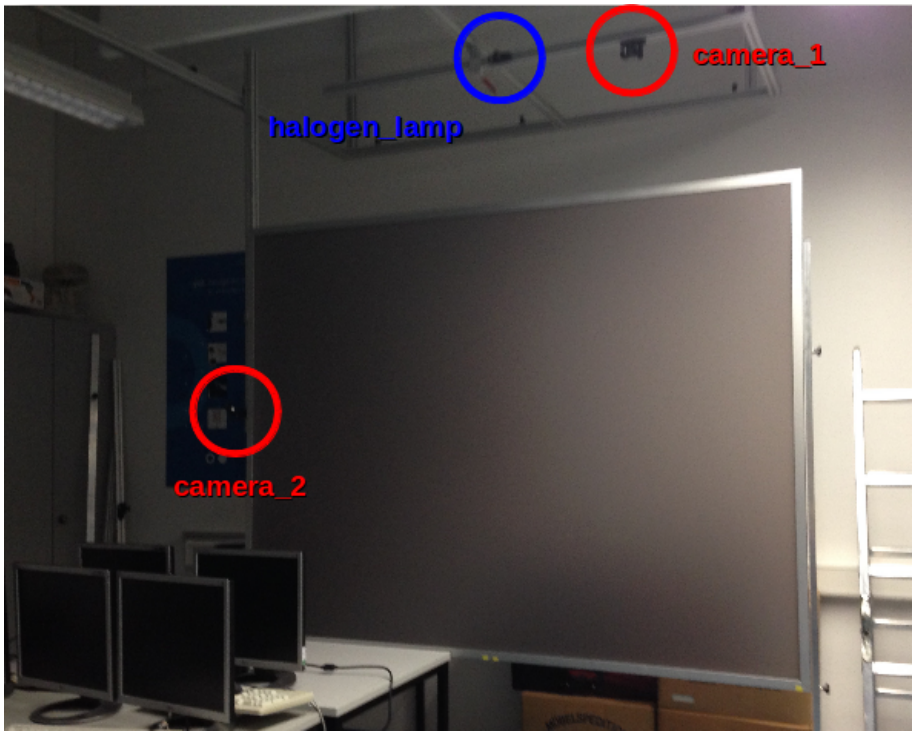

Figure 3: Camera mounts and Light source positions.
The red circles indicate where the mounts for the cameras are. The blue circle indicates the position of the halogen lamp.

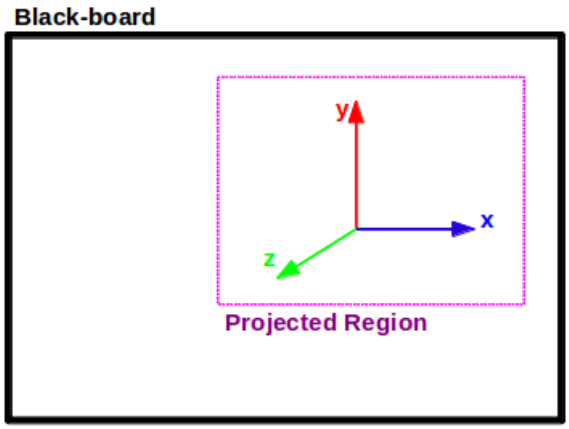

Figure 4: Board coordinate frame.
The purple dashed line corresponds to the effective area, where the video-beam will project the drawn strokes. This due to limitation in the distance between the video-beam and the physical screen

Data

As initialization a snapshot, one for each camera, is taken of the static background in order to be able to use them for segmentation later in the project. In addition, four

snapshots are taken with the marker on each corner

Our data consists of snapshots of each perspective taken by each camera respectively. The resolution of the images is 1920 x 1080 and a predefined number of frames was recorded while the strokes were being drawn.

These images are then processed by a PC generating the output, which consists of an image where the strokes are reprojected.

Approach

The project was divided in the following sub-tasks:

- Video capture. This stage consist in the recording of the video from the two cameras. This is in general terms our input data. For this stage we have taken into account:
  - Control of the environment. Lighting conditions, external sources of noise and occlusions were addressed during Set-up.
  - Calibration of the cameras each time we want to record in a different work session. The calibration is needed given that the board (glass screen) can be moved from session to session. Autocalibration techniques consist of obtaining the intrinsic parameters of the camera based on principles of multiple view geometry, obtaining the fundamental matrix or using a Euclidean reconstruction. Two methods are available: Direct [7] or stratification [8]. For more information see the references.
  - Synchronization of the videos. This was done by syncing threads through the use of thread events. Each time the first thread is ready to take a snapshot it notifies the second one, and then both proceed to take a snapshot at approximately the same time.

- Marker detection. This was accomplished by using a selected Region of Interest, which had been converted to HSV color space. Previously selected lower and upper boundaries help us isolate the marker from the background. A median blur is applied. We use Otsu's binarization threshold to segment the marker from the scene, which calculates the threshold based on the histogram of the region of interest. This process is repeated for each frame, where only the region of interest is used during calculation in order to reduce the amount of data processed.

- Color recognition. Once the marker was detected, the area with the marker was analyzed iby means of histograms. The color was detected in the RGB color space by using the RGB values found at the center of the circle that encloses the marker at the corner configuration sequence.

- Position reconstruction. Once we have the marker detected in each frame, the next step is to get the world coordinates back from the image coordinates. Assuming we have calibrated the cameras to obtain the intrinsic and extrinsic parameters, and we are able to reconstruct the matrix P of the cameras, we could then transform the image coordinates of the marker to the world coordinates. Exploiting the fact that our cameras share an axis, from which we will determined if the marker is close enough to the board to activate the drawing mode, we can extract the $x_{camera-i}$ and $y_{camera-j}$ coordinates of the marker, transform them to world coordinates and

re-project them to the frame of the board. The output of this step is a vector with the world coordinates which describes a stroke in the world frame.

- Stroke draw. The stroke vector obtained from the position reconstruction is used to draw the stroke back into the screen, using a video-beam to project. For this step we need to draw the stoke with the color previously recognized.

Results

All intermediate test will be done using the video recorded in the initial setup.

- Video captured simultaneously from both cameras.

Figure 5: Two snapshots taken simultaneously.

- Identification of at least 3 colors to be used on three different markers.

Figure 6: Marker detection
Marker detected in both views. This is then used to detect the color of the stroke.

- Set of (x, y) coordinates of the movement "painted"
- Algorithm that draws a line connecting a given set of points.

References

[1] Robert Laganière. *OpenCV 2 Computer Vision Application Programming Cookbook: Over 50 recipes to master this library of programming functions for real-time computer vision.* Packt Publishing Ltd, 2011.

[2] Xenophon Zabulis, Haris Baltzakis, and Antonis Argyros. Vision-based hand gesture recognition for human-computer interaction. *The Universal Access Handbook. LEA*, 2009.

[3] P. Fieguth and D. Terzopoulos. Color-based tracking of heads and other mobile objects at video frame rates. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 21–27, Jun 1997.

[4] M. Mason and Z. Duric. Using histograms to detect and track objects in color video. In *Applied Imagery Pattern Recognition Workshop, AIPR 2001 30th*, pages 154–159, Oct 2001.

[5] Stephen J. McKenna, Yogesh Raja, and Shaogang Gong. Tracking colour objects using adaptive mixture models. *Image and Vision Computing*, 17(3–4):225 – 231, 1999.

[6] N. Conci, P. Ceresato, and F.G.B. De Natale. Natural human-machine interface using an interactive virtual blackboard. In *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, volume 5, pages V – 181–V – 184, Sept 2007.

[7] Paulo RS Mendonça and Roberto Cipolla. A simple technique for self-calibration. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1. IEEE, 1999.

[8] Olivier Faugeras. Stratification of three-dimensional vision: projective, affine, and metric representations. *JOSA A*, 12(3):465–484, 1995.

[9] Francisco Martín and Manuela Veloso. Effective real-time visual object detection. *Progress in Artificial Intelligence*, 1(4):259–265, 2012.