Art Gallery Positioning System

Kaveh Boghraty

May 17, 2007

1 Introduction

The Art Gallery Positioning System is part of a larger project called Sonic Gallery[1]. Although this tracking system was built around the requirements of Sonic Gallery, the methods that are used can be adapted to a wide variety of scenarios. Before a detailed description of the system, a little information about its development context is needed.

1.1 Sonic Gallery

The vision of the Sonic Gallery project consists of an art gallery that uses location-dependent music to enhance a person's understanding and enjoyment of the visual works of art displayed in the gallery. Visitors of the gallery are provided with a small Pocket PC with headphones, and upon approaching a painting, a corresponding piece of music gradually immerses them in a state of mind that is more characteristic of the theme of the artwork.

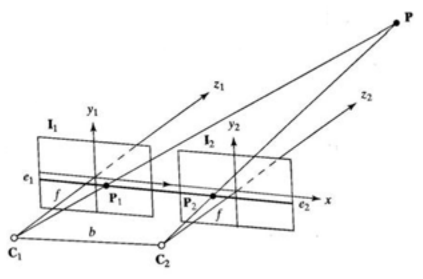
Among other factors, the implementation of this vision depends on the capability of simultaneously estimating the position of several people within a room. Despite the successful implementation of other indoor tracking systems, such as Cricket[2], a less costly route involving network cameras was considered by the Rhode Island School of Design (RISD). However, despite the successful development of several methods for tracking human movement using cameras, there is a shortage of methods that can track several people simultaneously and distinctively through one set of cameras.

1.2 Stereo Camera Triangulation

With the appropriate calibration, two cameras placed next to each other with a significant amount of gap and facing the same way can provide enough information to calculate the missing coordinate of an object (depth) relative to the 2D images provided. For this project the major challenge is to get the cameras to each locate the 2D coordinates of the same target point. By maximizing the gap between the cameras without losing too much field-of-view we can minimize the amount of error in the depth estimate caused by the inaccuracies of 2D tracking.

1. Master Project by Ha Tran - http://www.cs.brown.edu/publications/theses/masters/2007/tran.pdf

2. MIT's indoor location system http://cricket.csail.mit.edu/



1.3 Goal

To demonstrate that the above method can potentially be used as a solution to the demands of the Sonic Gallery project, this project must result in the successful tracking of subjects though a single pair of cameras. Provided one pair of cameras can track the location of a subject within its field of view, the expansion of the tracking environment through the addition of extra sets of cameras is a trivial process.

1.4 Approach

After experimenting with a few different tracking methods, the most successful and by far the fastest of these consisted in background subtraction followed by color detection. Both stages involve the use of a threshold value that is very crucial in isolating the target without ruling it out as background. Among the unsuccessful routes explored are shape detection using image derivatives and eigenvectors.

2 Equipment

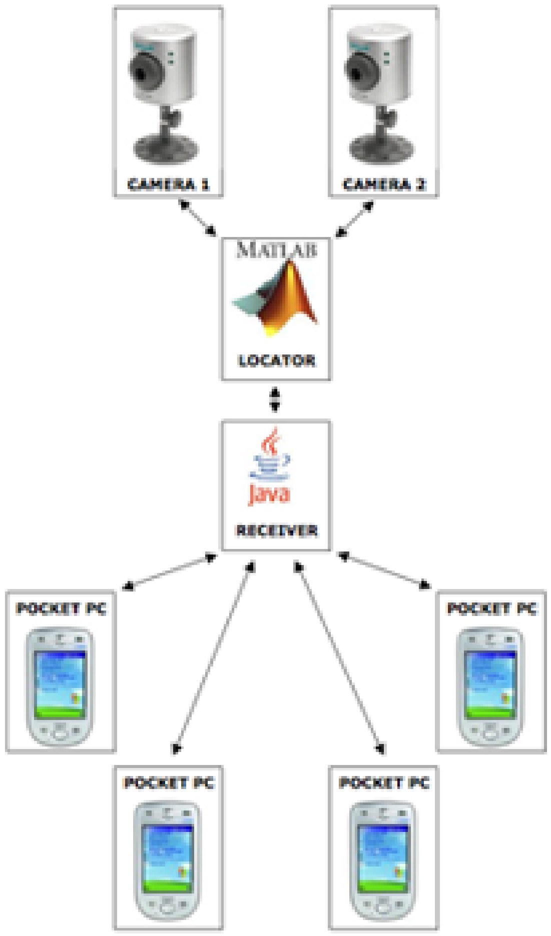
2.1 Calibration Toolbox

This package by Caltech students, called Camera Calibration Toolbox for Matlab[3], provides not only a relatively straightforward program for calibrating stereo cameras after placement, but also a wide range of matlab functions that facilitate interaction with the cameras. The calibration process makes use of a checkerboard positioned a many different angles in front of both cameras.

2.2 Setup

The Art Gallery Positioning System consists of four major categories connected by the wireless network: the cameras, the locator program, the receiver program, and the Pocket PCs (not available for final demo). Here is a simple diagram illustrating the network links:

3. Courtesy of Caltech Vision Department, www.vision.caltech.edu/bouguetj/



Here is an overview of the equipment used in the final demonstration.

2.3 Cameras

The specific model of the two cameras used in the tracking is D-Links DCS-900W. This model can be accessed by Ethernet as well as 802.11b (Wireless). For this project, the wireless interface was used. The settings were adjusted to provide the maximum resolution of 640x480 pixels.

2.4 Wireless Router

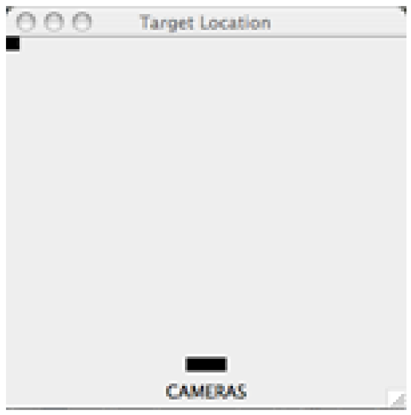
The 802.11b router that connected all components was Apples AirPort Express Base Station. Since the network did not need to be connected to the Internet, no cables were used in the setup of this project. The AirPort router was plugged directly into the nearest outlet and setup wirelessly.

2.5 Laptops

The locator program (Matlab) ran on a Apple PowerBook G4, while the receiver program (Java) ran on a Apple Intel MacBook Pro.

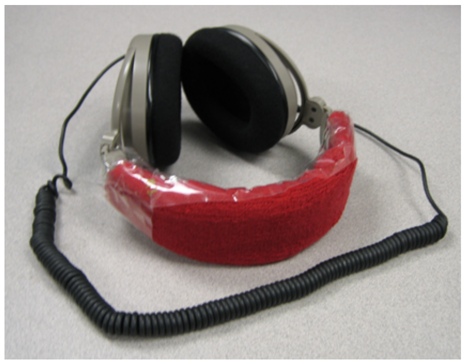
2.6 PocketPC

Unfortunately there were no working PocketPCs available for the last demo. Instead, I modified the receiver program so that it displayed a simple diagram showing the position of the people being tracked in relation to the cameras. This diagram is updated upon the receipt of each set of coordinates from the locator program.



2.7 Labeled Headphones

The color label used to distinguish the various people in the gallery is mounted on the band of the headphones used to hear the PocketPC audio.



3 The Algorithm

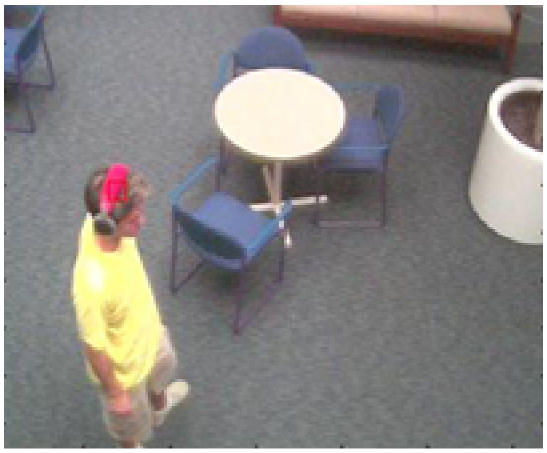
The central engine of this project is the one used to find the 3D coordinates of a subject. This process involves several stages with independent algorithms. To get an idea of the entire process we can track a single pair of frames from the time they are captured from the camera to the 3D coordinates that result from the processing of the frames.

3.1 Calibration Process

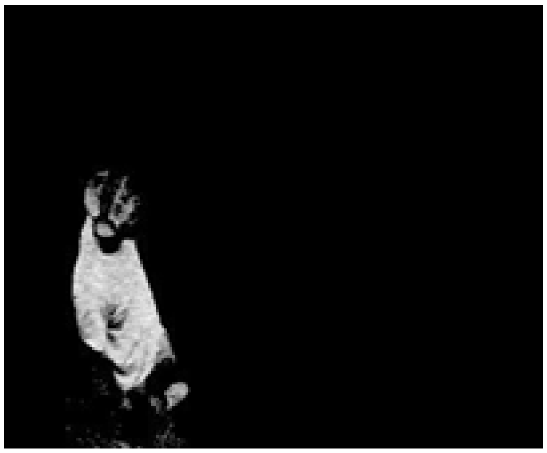
This calibration is not to be confused with the stereo calibration discussed above. Before the main loop is started the program needs a pair of images from the positioned cameras without any people in the field of view. These will be used to rule out background information from every frame once the tracking is begun.

3.2 Background Subtraction

Here is a typical frame as downloaded:



Background subtraction is a very efficient way of reducing the chance of contamination from unwanted objects with the same color of the target label. In this part, the calibration frames stored before the tracking are subtracted from the new frames. If the new frames contain any new objects, these will stand out in the resulting difference of images. Since introducing a new object in a scene also slightly changes the lighting in the rest of the room, a threshold is used to allow some tolerance in the difference values. If the difference value of a pixel is higher than the threshold, that pixel is considered for the color detection process. The following image is the result of subtracting the background from the above frame (converted to grayscale):



There is no safe rule of thumb for the threshold value, besides determination by trial. The value of this argument is very important. If too small of a value is chosen, the chances of contamination from the background are substantial. If the value is too big, the pixels of the target label can be unintentionally ruled out.

3.3 Color Detection

The pixels that survived the background subtraction are organized into a list, and their x and y coordinates are saved. Each pixel has 3 values, one for each of the RGB colors. These can be treated as a set of 3D coordinates, so that the 3D distance between this pixel's color and the target color can be calculated.

A different threshold is used to determine which pixels are close enough to the target to be considered for the Center of Mass. This time, pixel distance values greater than the threshold are disposed of, leaving pixels with the colors closest to what we are trying to detect. Here is the same frame after the color detection stage:

The border was added for display in this paper. Although this example shows a successful localization of the red headphone label, you can also see that a substantial portion of the label was removed in the background subtraction process. This result hints that the difference threshold value is a bit too large (leading to the removal of more than just background information).

3.4 Center of Mass

If the threshold values for the background subtraction and color distance are not too far off track, at this point only the pixels corresponding to the label surface in the frames should be left. The center of mass is calculated by averaging all the x and y coordinates into one set, which is then passed to the next stage.

3.5 Triangulation

At this point we have one set of 2D coordinates from each camera, corresponding to the position of the target label. We can now use the Camera Calibration Toolbox for Matlab to obtain a set of 3D coordinates. The coordinates returned by the triangulation function are not always intuitive. For each stereo calibration, the 3D origin that the coordinates are based on is different. It is, however, fairly straightforward to figure out the origin by running a few controlled trials. The coordinates can now be transmitted to the receiver program.

4 Limitations

Although this tracking method is fast and requires fairly inexpensive equipment, it also has some major drawbacks when compared to systems using actual sensors. Below are some of the limitations that became frustratingly clear throughout the project:

4.1 Subject Number

When using colors to distinguish multiple people within the same environment the maximum target number is very limited. Since a surface's color can vary in tone and brightness depending on its position with respect to light, a certain degree of flexibility must be allowed in the color detection process of the tracking algorithm. This means that in order to successfully distinguish various colors, they have to be distant enough on the color spectrum so their detection ranges do not overlap. Moreover, depending on the setup location certain colors should be avoided because they are too common in the environment.

4.2 Subject Labeling

In order for the cameras to have a reasonable field of view, they need to be fairly distant from the subjects. Consequently, the size of the color labels used to track the subjects needs to be large enough so that it can be detected from that distance. A large color label usually requires uncomfortable or unwanted extra clothing (hats) or accessories (large headphones). This may discourage people from making use of the tracking system.

4.3 Accuracy

There are numerous potential sources of inaccuracy in this method. Here are some of the most important ones:

- Items other than labels that match label color range can lead to an incorrect center of mass calculation.
- Line of vision from camera to label can be blocked by people or objects.
- Direct light source can cause the same label surface to produce a wide range of color values depending on position and angle with respect to light.
- A large label surface area can cause a pair of cameras to focus on different points on the same label and result in reduced triangulation accuracy.

Being familiar with the sources of inaccuracy can help adjust the environment to improve the odds. For example, a room's light source can be sometimes adjusted to create more ambient light and less spotlight. Image difference threshold values, as well as label sizes, can be fine tuned for specific conditions.

5 Conclusion

Although there is no way to significantly increase the limit of people that can be simultaneously tracked within an art gallery, the other major limiting factors can be minimized by tweaking the setup parameters (camera positions, light source, ...) to the system's advantage. In other words, this tracking system could be successfully implemented as part of the Sonic Gallery project with a small number of people. This would be an inexpensive way to get an idea about the appeal of Sonic Gallery's main concept to various types of people, which would help determine weather it is worth it to further expand the scale of the project.

References

[1] J.H. ter Bekke, *The Cricket Location-Support system*, Boston, MA, August 2000.

[2] Nissanka B. Priyantha, Anit Chakraborty, Hari Balakrishnan, *Thinking Forth, a language and philosophy for solving problems*, Prentice Hall, ISBN 0-13-917568-7, 1984.

[3] Klaus Strobl, Wolfgang Sepp, Stefan Fuchs, Cristian Paredes, Klaus Arbter,*Camera Calibration Toolbox for Matlab*, Pasadena, CA

[4] Sturm and Maybank,*On Plane-Based Camera Calibration: A General Algorithm, Singularities, Applications*, Reading, RG6 6AY, United Kingdom, 1999

Appendices

A Appendix A: Matlab code for locator

A.1 locator.m

A.2 locate_color.m

```
% ==============================================================
% locator.m
% Kaveh Boghraty
% ==============================================================

clear all;

% ==============================================================
% Please set the program parameters here
% ==============================================================

% IP Address for camera 1 (left)
cam1_address = '10.0.1.31';

% IP Address for camera 2 (right)
cam2_address = '10.0.1.32';

% Directory where locator files are run from
main_dir = '~/Desktop/locator/';

% Difference treshold: if pixel value difference between
% calibration frame and current frame is smaller than this,
% the pixel is ingnored for color detection
imdiff_cutoff = 0;

% Color distance treshold: if 3D distance between target
% color and pixel color is grater than this value, pixel is
% not considered for center of mass
colordist_cutoff = 40;

% Debug on/off: set this to 1 if you want debug info,
% including diff and color distance images to be displayed
imdebug = 0;

% Set this to 1 if you want to send coordinates to a server
% of choice
server_connect = 0;

% Server IP and port
hostname = '10.0.1.2';
port = 2000;

% ==============================================================

cd(main_dir);

% Setup and initialize camera
javaaddpath './java/';
addpath './TOOLBOX_calib/';
import CamView;

cv1 = CamView(cam1_address, 80);
cv2 = CamView(cam2_address, 80);
cv1.start();
cv2.start();

% Wait for cameras to come online
% Pause duration may need to be adjusted based on network
pause(5);

% Setup and open connection to server
```

```
if( server_connect )
    socket = tcpip(hostname, port);
    fopen(socket);
end

% Load calibration files
caldir = './calibration/';
cal = load( [caldir 'Calib_Results_stereo.mat'] );

% load images in source folder
sourcepath = './colors/';
colors = loadImagesRGB(sourcepath);
h_size = size(colors);
him1 = reshape( hats(2,:,:,:), h_size(2), h_size(3), h_size(4) );
red = reshape( mean(mean(him1)), 1, 1, 3 );

% Load background calibration frames. Make sure there no temporary objects
% or people in view
calim1 = frame(cv1);
calim2 = frame(cv2);

out = '0;0;0;';

while(true)
    pause(.25);
    % Receive frames from cameras
    fim1 = frame(cv1);
    fim2 = frame(cv2);
    % Subtract background
    diffim1 = imabsdiff( rgb2gray(fim1), rgb2gray(calim1) );
    diffim1( find( diffim1 < imdiff_cutoff ) ) = 0;
    diffim2 = imabsdiff( rgb2gray(fim2), rgb2gray(calim2) );
    diffim2( find( diffim2 < imdiff_cutoff ) ) = 0;
    % Find target in new frames
    [y1,x1, distmat1] = locate_color( fim1, calim1, red, tile, imdiff_cutoff, colordist_cutoff );
    [y2,x2, distmat2] = locate_color( fim2, calim2, red, tile, imdiff_cutoff, colordist_cutoff );

    % If target could not be located 0 is returned
    if( y1 == 0 || y2 == 0 )

        NAval = sprintf( '%d;%d;%d;\r\n', [0;0;0] );
        % Send coordinates to server
        if( server_connect )
            fprintf( socket, '%s', NAval );
        end
        % Print coordinates
        if( imdebug )
            fprintf( '%s', NAval );
        end

        if(imdebug)
            subplot(2,3,1)
            imagesc( fim1 );
            subplot(2,3,4)
            imagesc( fim2 );
            subplot(2,3,2)
            colormap gray
            imagesc( diffim1 );
            subplot(2,3,5)
            imagesc( diffim2 );
            subplot(2,3,3)
            imagesc( distmat1 );
            subplot(2,3,6)
            imagesc( distmat2 );
            drawnow;
        end

    else
        % Triangulation happens here
        [XL,XR] = stereo_triangulation( [x2;y2], [x1;y1], cal.om, cal.T, cal.fc_left, cal.cc_left, cal.kc_left, cal.↙
alpha_c_left, cal.fc_right, cal.cc_right, cal.kc_right, cal.alpha_c_right);

        if(imdebug)
            subplot(2,3,1)
            imagesc( fim1 );
            hold on;
            rectangle('Position', [x1-10,y1-10,20,20], ...
                'Curvature', [0,0], 'EdgeColor', 'red');
            subplot(2,3,4)
            imagesc( fim2 );
            hold on;
            rectangle('Position', [x2-10,y2-10,20,20], ...
                'Curvature', [0,0], 'EdgeColor', 'red');
            colormap gray
            subplot(2,3,2)
            imagesc( diffim1 );
            subplot(2,3,5)
            imagesc( diffim2 );
            subplot(2,3,3)
            imagesc( distmat1 );
            subplot(2,3,6)
            imagesc( distmat2 );
            drawnow;
        end

        % Construct output
        out = sprintf( '%d;%d;%d;\r\n', (int16(XL)) );
        % Send coordinates to server
        if( server_connect )
        fprintf( socket, '%s', out );
        end
        % Print coordinates
        if( imdebug )
        fprintf( '%s', out );
        end
    end
end
```

```
function [y,x, distmat] = locate_color( fim, cim, color, tile, diff_tresh, dist_tresh )


% store image dimensions
imsize = size(fim);

% expand destination color to size of image
colormat = repmat( color, [ imsize(1), imsize(2) ] );
% compute difference matrix of color and image
diffmat = double(fim) − double(colormat);
% compute distance matrix between image and color
distmat = ( (diffmat(:,:,1).^2) + (diffmat(:,:,2).^2) + (diffmat(:,:,3).^2) ).^(1/2);


% use background image to narrow matches
diffim = imabsdiff( rgb2gray(fim), rgb2gray(cim) );
indeces = find( diffim < diff_tresh );

% discard values outside diff match
distmat(indeces) = max(max(distmat));

% get all pixels with values close to destination color
[y_vals, x_vals] = find( distmat < dist_tresh );

% for return image, remove all out−of−range pixels
distmat( find( distmat > dist_tresh ) ) = max(max(distmat));


% make sure we have a match
if(length(y_vals) == 0)
    x = 0;
    y = 0;
    disp 'No match';
else
    y = mean( y_vals );
    x = mean( x_vals );
end
```