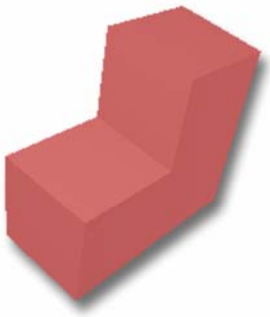


DEVELOPMENT OF AN EMBEDDED 3D ROBOT VISUAL SYSTEM

By Malcolm Stagg

5 Page Summary

April 29, 2004

1. ABSTRACT

Current methods creating virtual models of 3D objects from two cameras match the raw pixel data, generating raster information useless to robots.

Microcontroller programs I have created and implemented generate vectors of object edges and reconstruct with point matching. Another system may interpret the output for recognition and scene understanding.

2. OBJECTIVE

A. To create a system of circuit boards which can be used with several computer programs to create a vector model of a simple object in three dimensional space.

B. To find which positions and orientations of a single-faced object will show up most accurately in 3D space, and also preserve the area of the face.

3. HYPOTHESIS

If testing of different transformations in 3D space for a simple single-faced object is completed, then the most accurate results in terms of transformation measures and area of the face will occur when the object is pointing towards the cameras, and is as close as possible to the cameras; this is because the less the visibility of the object, and the greater the distance to the object, small inaccuracies in measurement can lead to very poor results.

4. MATERIALS

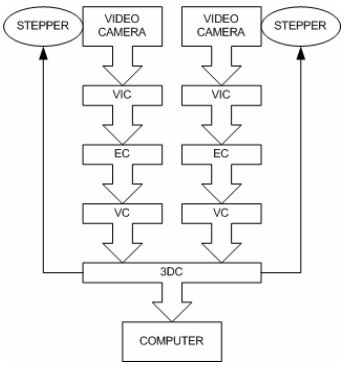
- 2 small grayscale video cameras
- 7 PIC18F452s, and 1 PIC16F628
- 2 LM1881 video sync separators
- 2 TDA8708 video ADCs
- 2 MSM518221A video RAMs
- 4 74LS164 shift registers
- 2 DS1230-120 256k bit RAMs
- PAKII math co-processor
- 2 L293D stepper motor drivers
- 2 0.9 deg/step unipolar stepper motors
- Connectors for power and debugger
- 2 video camera power and RCA plugs
- Several LM7805 & LM7812 regulators
- 3 4049 Invertors
- 2 47ACT11008 AND gates
- Discrete components

5. BACKGROUND

A. HARDWARE

In this system, I have designed seven circuit boards to implement each component of my robot vision algorithm. Each board contains at

least one PIC microcontroller, as well as several other components. Parallel ports link together the boards, and the output of any board may be disconnected from the system, and connected to a computer for data transfer.



The input of the system consists of two cameras mounted on unipolar stepper motors, at a constant defined distance from each other. The video outputs of the cameras are each fed into a Video Interface Controller (VIC) board – more or less a frame grabber. The outputs of each of these VICs go to an Edges Controller (EC) board. The EC board finds edges of objects, and thins the edges. The outputs of each EC board are sent to a Vector Controller (VC) board. This VC will draw vector lines representing the edges. The output of each of the two VCs will go to a single 3D Controller (3DC) board. This board implements a point matching algorithm to match together the left and right images, and generate a 3D result. It also can control the stepper motors. The output of the 3DC is sent to a computer.

B. SOFTWARE

For preliminary research into viable algorithms to improve, implement, or redesign, I created a large demo program in Visual Basic 3.0. This program was begun last year, in my project comparing edge detection methods **[10]**. This program demonstrates many algorithms which I may or may not have implemented.

After the effectiveness of each algorithm to implement had been verified, microcontroller assembly code for the PIC was started.

The software for a frame grabber (the VIC) seems quite insignificant; however this was certainly a challenge to write. For one thing, everyone has a different idea about standard NTSC timing, and none seemed to be completely correct. Also, it was necessary to remove noise in the video output with a mean filter in this microcontroller program. For greatest effectiveness, this was not a true mean filter, but it weighted the sum of pixels surrounding the current pixel as the same as the current pixel.

a	b	c
d	e	f
g	h	i

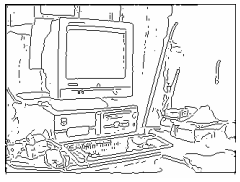
1	1	1
1	8	1
1	1	1

Kernels work by multiplying a neighborhood of pixels in an image by set values. The result of a kernel is the sum of each of the set values times each of the pixel values. This process is repeated for each pixel in an image. The mean filter described above uses the kernel shown above, ÷8 to get the final result.

-1	0	1
-2	0	2
-1	0	1

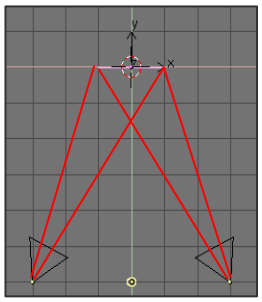
1	2	1
0	0	0
-1	-2	-1

The edge detection program (EC) was also an interesting challenge to write. This uses the Sobel edge detection



kernel, a 3x3 kernel finding horizontal and vertical edges separately using a kernel comparing pixels to the left and right, or top and bottom of the current pixel. The horizontal and vertical edges can be combined together by $G=\sqrt{G_X^2 + G_Y^2}$, or approximately $G=(G_X + G_Y) / 2$. The kernels for G_X and G_Y respectively are shown above. Thinning is completed with a simple implementation of the Non-Maximal Suppression algorithm, as described in **[2]**.

Vectorization (VC) uses a rather complex method originally described by Dr. Jim Parker in **[7]**. Extreme improvements have been made to this method, to allow for the process to be completed in two stages. First, line following is used to find areas in the image with low linearity. These areas are marked with a corner, representing the end of a line segment. The next stage is basically connect-the-dots. The microcontroller will attempt to trace between all the corners, determining where line segments are needed.



The 3D transform uses a simple point matching algorithm of sorting the left and right vertices by their y position, then x position. Assuming vertices have the same connectivity information in the left and right images, a direct match between the vertices is attempted. 2D to 3D transformation is computed using the distances between cameras, camera angles, focal length, and camera height, to determine the positions of the vertices in 3D space.

6. PROCEDURE

A. CONSTRUCTION

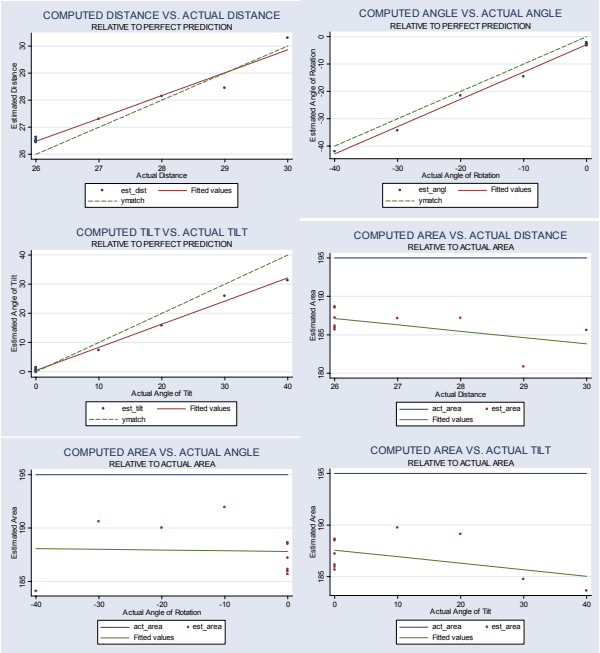
1. Research and design necessary algorithms
2. Write demo software to test methods
3. Create schematics for circuits
4. Build demo circuits on breadboards
5. Write programs in MPASM to implement.
6. Debug and test programs
7. Test circuits with functional programs
8. Design and manufacture circuit boards

B. TESTING

1. Create background for testing against
2. Cut test shapes from thin wood
3. Paint shapes white and mount, allowing rotation and tilt
4. Place triangle in 5 positions, 5 rotations, and 5 tilts, while capturing data on laptop
5. Find ideal vertices from edges image by hand for each image; also try vectorization for the images
6. Create 3D models using stereoscopic calculations. Calculate average distance and orientation.

7. RESULTS

The following are the graphs for the preliminary experimentation. Tests were completed to measure the accuracy of distance, angle of rotation, and tilt. Area was measured for each of these as well.



8. ANALYSIS

In graphs of computed vs. actual distance, there is a line showing the ideal values with a slope of 1. Linear least squares regression was done on these data to generate a straight line. With these data, the 95% confidence interval of the slope is (.71 to .98). These bounds do not include 1, so the calculated line is slightly significantly different from the line of perfect prediction.

The graph of area values at the different distances was done similarly. Linear least squares regression is done on these data to obtain a straight line. The 95% confidence interval of the slope is (-1.8 to .17), which includes 0, showing that area does not change significantly with distance.

Similar plots were repeated for angle of rotation and tilt. Regression of the computed rotation data on the actual showed that the slope was not significantly different from the ideal. The regression of the area values shows that area does not change significantly with the angle of rotation.

For the tilt data, the regression line indicated that the slope is significantly different from the ideal value. The graph shows that as the angle of tilt increases, the accuracy decreases. The regression of area showed that the area does not change significantly with tilt.

9. CONCLUSION

In conclusion to the experiment described in Objective – part B, my hypothesis was only partially correct. From these results, it does not seem that accuracy is improved when the object is closer to the camera. It also shows that the distance is calculated quite well. Distance vs. area data shows that the area calculations are rather poor.

The results for angle of rotation showed similar conclusions to the distance data. Accuracy of the calculated angle of rotation was very close to the ideal values, as it was for distance. The area calculated after each of the rotations seems to once again be quite poor.

When angle of tilt is examined, it can be seen that accuracy starts out very good, and gradually becomes worse. The area data for angle of tilt are poor.

Overall, I am quite pleased with the accuracy of the distance, rotation, and tilt calculations. However, the area calculations were overall poor.

10. SOURCES OF ERROR

One main source of error is the alignment of the cameras. One of the cameras had the CCD shifted down vertically by the equivalent of 28 pixels. This was corrected using math, but showed that the accuracy of the cameras may be quite poor. A new camera will be obtained for future experimentation. Also, the orientation of the cameras may not be completely perfect, as well as the actual position and orientation of the object. There may have been some small inaccuracies caused by poor measurement.

11. APPLICATIONS

There are a huge number of applications where a system like this could be used, when it is fully optimized and constructed. The main application I was looking at during the construction of the electronics, though, was a basic robotic device. Such a device could be extended to be capable of finding an object and where it is in a scene, adding to a virtual model of an object, and recognizing the identity of the object.

12. SPECIAL THANKS

1. **Dr. Peter Stagg**
   My uncle for his kind help and support especially with the 3D calculations and ideas for the experimentation.
2. **Vicki Stagg**
   My mother for all of her help, especially for her help with Stata for the analysis.
3. **Andrew Stagg**
   My brother, for all his computer related help and support.
4. **David Wells**
   Auton Engineering, Ltd.
   For his kind donation of a PicStart Plus PIC microcontroller programmer, and his circuit board troubleshooting ideas
5. **Damon Chu**
   Microchip Technology, Inc.
   For generously sending me a MPLAB ICD 2 PIC microcontroller debugger at no charge
6. **Cory Mills**
   National Semiconductor, Inc.
   For generously sending me a COP-8 microcontroller emulator at no charge
7. **Laura Neilsen**
   Alberta Printed Circuits
   For kindly manufacturing my circuit board layouts at no charge
8. **John Carney**
   Cadence
   For the demo of Orcad Layout PC Board design software

13. REFERENCES

[1] Brown, Christopher. Advances in Computer Vision Volume 1. Academic Press, 1989.

[2] Desai, Ujjaval et al. Edge and Mean Based Image Compression. http://www.ai.mit.edu/people/bkph/AIM/AIM-1584.pdf. 1996.

[3] Freeman, Herbert. Machine Vision for Three-Dimensional Scenes. Academic Press, 1989.

[4] Guzman, Adolpho. Computer Recognition of Three-Dimensional Objects in a Visual Scene. http://ncstrl.mit.edu/Dienst/UI/2.0/Composite/ncstrl.mit_lcs/MIT/LCS/TR-59/1?nsections=13. 1968.

[5] Iosoft Ltd. Chipvid Project. http://www.iosoft.co.uk/chipvid.php.

[6] Marr, David. Vision. W. H. Freeman and Company, 1982.

[7] Parker, James. Practical Computer Vision Using C.

[8] Roberts, L. G. Machine Perception of Three-Dimensional Scenes. MIT Press, 1980.

[9] Shirai, Yoshiaki. Three Dimensional Computer Vision. Springer-Verlog Berlin Heidelberg, 1987.

[10] Stagg, Malcolm. Evaluation of 3D Object Recognition Methods. 2003.

[11] University of Edinburgh. Image Processing Learning Resources. http://www.dai.ed.ac.uk/HIPR2/hipr_top.htm. 2002.

[12] Wahl, Freidrich. Digital Image Signal Processing. Artech House, Inc., 1987.