16-385 Computer Vision Term Project Final Report

Scott Martin
semartin@andrew.cmu.edu

1. Introduction

In this paper, the plan and implementation of a term project for 16-385 Computer Vision is discussed. This project attempts to refine the control of a homemade NERF gun turret by designing an algorithm that tracks and predicts the motion of human targets using a rotating camera.

2. Hardware & Control

The turret consists of a NERF Stampede gun mounted to a plastic turntable, which is capable of 360° of rotation. By using a continuous-rotation servo motor [1] and an onboard power supply, the turret has completely unrestricted, continuous horizontal rotation.

The PWM signal for the servo motor and the trigger relay that fires the gun are controlled using an Arduino Uno microcontroller board, which receives command from a mounted laptop through a USB serial port. The laptop controls odometry and tracking, pulling a video stream from the webcam sighted down the gun barrel and using this information to control the servo and the gun. See Fig. 1 for an overview of how control works.

Fig. 1: Control loop for NERF turret

3. Proposal & Objectives

Previous attempts at automatic target tracking and elimination used a fixed camera, which greatly simplifies tracking but is deficient for two reasons. Firstly, this limits the field of view of the turret to the viewing angle of the camera, which may allow enemies to approach from an untracked direction and disable the turret. Secondly, the method of tracking with a fixed camera also introduces a cable attached to the ground at some point, which constrains the rotation of the turret and may prevent it from tracking to an enemy. The solution explored in this research paper removes the fixed positioning of the camera. Instead, the camera is mounted to the base of the plastic turntable and sighted along the barrel of the gun. Without any external fixed points, the turret is allowed to rotate indefinitely in either direction in its search for enemy targets.

The computer vision problem becomes more complicated. Instead of a fixed background with simple motion tracking, the algorithm must filter out the moving background in the image caused by rotation of the turntable in order to track to an enemy. However, the movement of the background also allows the potential to estimate odometry of the servo motor. By providing this real-time angular speed estimate, we can predict the future angle of the gun and use this information to further refine our tracking algorithm.

4. Planned Procedure

The algorithm was developed sequentially in three steps.

The first step was implementation of background homography and rotation speed estimation. First the servo must rotate the camera a full 360° to capture the background scenery, which will be removed in order to detect targets in part 2. To accomplish this, homography is established between the image frames through the use of cross-correlation of several horizontal pixel ranges. Once the background is

acquired, the algorithm then estimates the rotation speed of the servo motor by tracking the speed at which the background moves. This is useful in later tracking.

The second step in implementation factored in the added complication of a moving target. Background tracking and odometry was revisited with an added moving foreground object to test robustness and accuracy.

The third, final, and most satisfying step is testing the capabilities of the NERF gun in eliminating targets. Foreground objects must be identified as either passive - such as an incorrectly subtracted background object or camera noise - or as an enemy that requires pacification. Once identified, the target's motion is anticipated and it is fired on at a high enough rate and accuracy as to discourage further ingress.

5. Preliminary Research

For background calibration, rotation speed of the servo can be set such that consecutive frames deviate by a very small number of pixels. Combined with a fixed movement axis and a known direction, matching can be exceedingly simple. Images will likely be able to be paired by calculating squared error of a small number of distributed image patches – this will run very fast. In order to average out camera rotation, a few frames could be gathered from fast servo rotation. Either matching via homography or Lucas-Kanade flow estimation could be used to detect motion deviation from horizontal.

Another consideration is the width of the panoramic image buffer used to store the background image. This will vary from camera to camera, but a standard webcam with roughly 60° FOV will require a buffer roughly six times as wide as the image.

Background subtraction requires odometry to determine our camera's viewpoint in the background image buffer, but odometry also requires background subtraction to remove foreground objects which should not contribute to background motion. Assuming the foreground object occupies less than half of the image, a voting scheme is one possible solution to this – rotation speed can be measured in various locations of the image and the majority of the responses is calculated. Our knowledge of the direction and speed of servo rotation can also be an influencing factor.

For detection of enemies, the subtracted background can be thresholded to a binary (black and white) image, all holes filled with *imfill(IM, 'holes')*, and independent components detected with the *bwconncomp* function [3]. The centroid of the object, which presents a good starting point for targeting, can be calculated from the mean of the object's pixels.

Once targets have been established, particle filters are a good first choice in predicting the velocity of each target [2]. Target priority could be calculated by some weighted measurement of the number of pixels of the target (i.e. target size), the speed estimate from the particle filter, closeness of the target to the gun's crosshairs, and human-like shape. Rudimentary face detection may possibly be used to exclude non-human targets.

6. Project Schedule

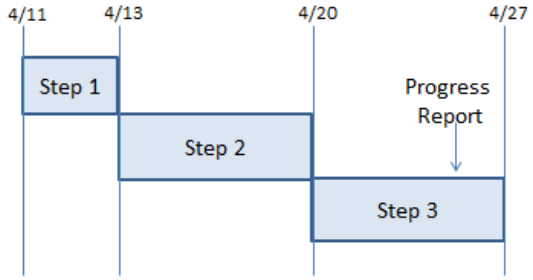

Fig. 2: Gantt chart of project schedule

April 11th: Project proposal (this paper) is due.

April 11th-13th: Step 1 implementation. Begin testing robustness of odometry against foreground objects.

April 13th-20th: Step 2 implementation. Begin target identification.

April 20th-27th: Step 3 implementation. Revisit project requirements and revise schedule.

April 25th: Single page progress report & additional slide is due.

May 8th: Term project final report is due

May 11th: Term project class presentation

7. Hardware Constrution

The NERF gun turret hardware was improved substantially since the project proposal. In place of the old 180-degree servo motor that allowed restricted movement of the gun, a new system is installed that grants full continuous rotation as designed.

In addition to the changes to the platform, the gun itself was made much more compact – nonessential parts (such as the handle, forward barrel, and battery compartment) are cut away, leaving only the firing mechanism and clip holder. In addition to giving the turret a smaller profile that fits more easily onto the turntable, the reduced weight reduces the load on the servo and makes the turret react faster to changes in movement.

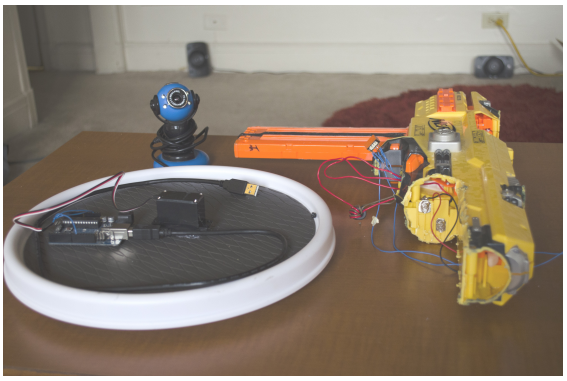

Fig. 1: Turntable (left) and modified NERF gun (right)

It was discovered that a cardboard box made for a great enclosure for the turret. PVC pipe was fitted in place to angle the gun upwards for greater range. Two holes were cut to allow the camera to see and bullets to escape, and a side hatch provides easy access to the magazine when the gun runs out of ammunition. An extra cut was made to allow the servo and servo connector to poke through the enclosure.

With the combined weight of all parts, no attachment to the turntable was necessary – the box rests on the turntable and rotates without slipping.

3. Software Implementation – Calibration

The MATLAB script can read camera frames from a USB port, stitch consecutive frames together to form an image panorama, and localize with high accuracy despite corrupted images and no knowledge of prior position.

The webcam capture script shows a very smooth frame rate (roughly 24fps) at the native 320x240 resolution of the webcam, with some affective motion blur. The images are clear and undistorted enough to stitch together quite well.

Background acquisition is tested using panoramic images from Google Image search. A 320x240 window is panned across the image, with jitter and Gaussian image noise to simulate conditions from the webcam. When the software detects wraparound, the algorithm stops itself, matches the start and endpoint of the panorama, and removes any excess.

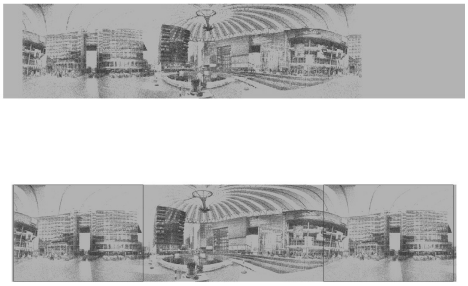

Fig. 2: Panorama in progress (top) and complete (bottom)

Localization testing is done by selecting a (uniformly) random 320x240 window from the panorama, applying motion blur, image noise, and several obscuring "foreground" rectangles, and running it through the algorithm in an attempt to determine its position. Deviation by more than one pixel counts as a failed test. Even with such noisy data, the algorithm still achieves 81% accuracy. Removing the foreground objects and motion blur (and leaving image noise) yields upwards of 96% accuracy in recovering the window's position.

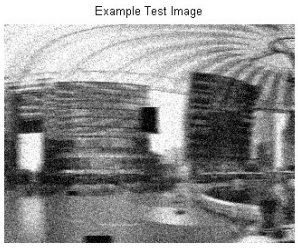

Fig. 3: Blurred, noisy, obstructed test image.

3. Software Implementation – Odometry

Initially, it was expected the cross-correlation method used in the calibration step would also be enough to determine the orientation of the turret. However, early experimentation with a moving foreground target indicated that this was not the case. Over time, small errors in correlation due to the foreground objects added up and caused the panoramic background to lag behind the actual image. This produced many false positives, and the algorithm fell apart.
Feature tracking solved this particular problem. By finding SURF features in each frame and matching them to features in the previous frame (via

RANSAC), the pixel distance between frames is computed with high accuracy and the background panorama easily keeps up with the video feed.

Unfortunately, the increased complexity of the SURF feature matching makes this computation step very intensive. The frame rate was reduced from roughly 20 frames per second using cross-correlation to 3 or 4 frames per second. Planned inclusion of prior knowledge of speed and direction will greatly reduce this computation time, as a sum of squared distances (SSD) algorithm could be used around the predicted location of each feature.

See test_turret_calibration.m or test_panorama.m for an example of the current calibration implementation.

4. Software Implementation – Tracking

At the heart of the tracking algorithm is threshold detection. After the odometry step, the background is subtracted out, leaving background areas with near-zero pixel value and foreground areas with (mostly) nonzero values. A gaussian blur is applied to mitigate noise, and the pixels with large enough difference are treated as potentially belonging to a foreground object. After these potential pixels are detected, a few calls to the *bwareaopen* function remove false positives.

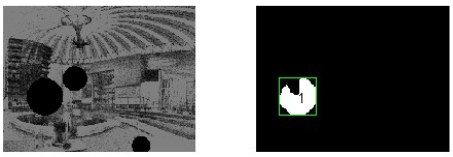

Fig. 4: Blob detection test. The smaller blobs are thresholded out.

After this detection step, several blobs are now known to be potential targets, and it must be decided which target we should track. The current test scripts (such as test_turret_tracking.m) run the tracker on a black ball, which is superimposed on a previously calculated panoramic image background. The tracker decides which blob is the ball by searching for the object with the most black pixels. In a human-tracking environment, this discrimination step will likely include a face detector to reject background objects.

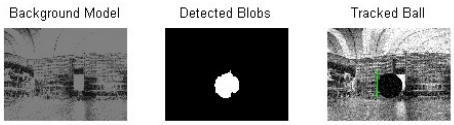

Fig. 5: Background subtraction and tracking

Once the target blob is fully identified, the turret tracks towards the target blob's centroid using a PID loop. While this resulted in very smooth and reliable motion in the test script, experiments with the actual turntable showed that the turret's motion is too fast with any more than a 1-bit deviation from the servo's motionless state. Thus, PID for the real hardware was determined to be overkill and will be removed in future versions.

5. Lessons Learned & Future Plans

This project emphasizes the difficulties involved with the use of computer vision with real-world data. In particular, simulation is very useful in anticipating problems, but the assumptions used in simulation may still prevent the algorithm from being much use in the field, as evidenced by the expected need for PID when the servo would be better off using bang-bang control. This also served as a lesson in real-time control, and how sophisticated computer vision algorithms such as SURF feature matching potentially create the greatest bottleneck in the control loop.

Finally, this project demonstrated the fine line between assumption and generalization. Algorithms with heavy assumptions produce cleaner, simpler and faster code, while generalized computer vision algorithms apply to more situations and are more useful in the long term. A proper balance between these two aspects is crucial for an accurate and robust algorithm.

This project is far from over – more research into motor control, blob detection, and background tracking is required before the turret can perform as expected. In particular, the performance of the feature matching step of background estimation must be greatly improved, as this represents the majority of computation time of the algorithm. The PID loop may be replaced with bang-bang control, or the microcontroller may potentially modulate the duty cycle of the PWM servo control signal to allow for more varied control. Finally, the blob discrimination algorithm must be improved to track human features and exclude false positives from the background.

If the desired performance is still not achieved after these improvements, potential options include the use of a Microsoft Kinect or similar depth-based camera, or inclusion of an encoder to the base of the turret to more accurately predict motion.

7. References

1. "Continuous Rotation Servo (#900-00008)" . Parallax Inc., 29 4 2004. Web. 11 Apr 2013. <http://www.parallax.com/dl/docs/prod/motors/crservo.pdf>.

2. Kaijen, Hsiao. "Particle Filters and their Applications." . MIT, 11 Apr 2005. Web. 11 Apr 2013. <http://web.mit.edu/16.412j/www/html/Advanced lectures/Slides/Hsaio_plinval_miller_ParticleFiltersPrint.pdf>.

3. "bwconncomp - find connected components of a binary image." MathWorks. N.p.. Web. 11 Apr 2013. <http://www.mathworks.com/help/images/ref/bwconncomp.html>
.