A self-targeting missile system using computer vision

Kit Axelrod (305153692), Ben Itzstein (305128329), and Michael West (305159240)
MTRX 4700 Experimental Robotics Major Project
University of Sydney

Abstract

The use of visual sensing to acquire and track targets has the potential to significantly reduce cost in comparison to active sensors. We present preliminary results for a system that combines a webcam with a toy foam missile launcher on an altitude-azimuth mount. The controller uses foreground segmentation to identity new elements in a scene. It then applies SURF feature detection and a pinhole camera model to determine their position in the real world. Based on our analysis of the ballistic flight characteristics for the foam missiles, the system can thus generate the altitude and azimuth launch angles that will allow a missile to intercept the target. We interface with the missile launcher via USB, and implement a second vision system based on gradient detection to provide pose feedback for the launcher barrel. The system is consistently able to intercept targets, within the restricted parameters of our testing.

1 Introduction

Target acquisition and tracking are frequently the domain of active sensing methods such as radar, ultrasound, or laser scanning. The ability to track targets at manipulation range without resorting to these active methods can significantly reduce the cost and complexity of manipulator control. Modern webcams, in particular, provide an ideal platform for experimental development in computer vision. They are cheap, readily available, and increasingly possessed of high-resolution sensors and quality optics. Applications range from robotic security systems to environments such as production lines where identification and manipulation of objects are routine tasks with potential for wide-scale automation.

In this paper, a toy missile launcher is modified to automatically locate and aim at a stationary target, such as a balloon, marked with a pre-determined symbol and placed at a pre-defined range. Further developments could relax these restrictions by allowing range detection from the video image and implementing tracking and prediction of a moving target, but these features proved impossible to include within our timeframe. Target acquisition occurs via processing of an image stream from a single webcam, making use of foreground segmentation and SURF feature detection, together with a calibrated pinhole model to convert from pixel distances into real-world Cartesian coordinates. Because the missile launcher has no sensors to provide feedback on its pose, described in terms of the altitude and azimuth angles of its barrel, we also present results for a visual servoing system. This uses a camera mounted on the barrel to read a calibrated fan pattern printed behind the launcher base, providing pose feedback by detecting and recording movement from a defined origin. We also show results for ballistic flight tests conducted on the foam missiles, which allow the calculation of the desired launcher pose given a target location.

The paper thus covers three principal strands: target acquisition, visual servoing, and ballistic flight characterisation for the missiles. In section 2, we assess current techniques available in the literature and determine how they apply to our system. In section 3, we describe the design of our system, including approaches that proved to be unsuccessful. In section rd, we describe the results achieved in each of the three major aspects of the project, and discuss their significance. Finally, we summarise the project's main advances and indicate possible directions for future work.

2 Background

2.1 Target acquisition

The video targeting system described in this section is a step towards using monocular video as an effective targeting method at manipulation range. Video imaging can be used to determine the location of a target in the image plane of the camera. Unfortunately there is no reliable way to translate this information into three-dimensional Cartesian world-frame coordinates without some understanding of the range to target. This is a fundamental limitation of the pinhole camera model and moreover, all perspective geometry. Once range is known, however, it is a simple calculation to produce the world-frame position. A typical solution to this problem is the application of stereo or multi-baseline stereo vision and much literature exists on this topic (see, for example, [10]), however it is not always practical to employ two cameras and frequently stereo matching algorithms are computationally expensive. This poses a problem frequently encountered in mechatronics applications, that is, how to determine target range with only one camera. A cue in this direction is the target's scale. If the scale of the target is observable and can be compared to a model then range may be estimated. Working with this estimate it is then trivial to extrapolate from image pixel space to real-world coordinates.

Before object scale can be estimated by comparison to a predefined object model it is necessary to identify a target object. This is a fundamental problem of computer vision and has been addressed in two main modes, dense and sparse methods. Dense methods refer to processing the entire image space or a region of interest whereas sparse methods extract a feature set from input images and process the key-points and descriptors. Generally when dealing with static images, for applications such as image-stitching, it is often advantageous to work with sparse methods as the dataset may be reduced dramatically. In the case of video however, the reduced cost associated with a rarefied dataset may be offset by the necessity to define features for each frame. To perform object recognition, an algorithm must be able to define an object template in terms of transformation-invariant constituent features. As the object may appear translated, rotated, occluded, and illuminated differently, the features extracted must be robust to these changes.

In a recent paper, Hu [6] shows a solution to visual tracking using Scale-Invariant Feature Transform (SIFT) points. It is demonstrated that a target object may be found under conditions of translation and rotation, affine transformation and even partial occlusion. This is achieved by finding the SIFT features of the object and each video frame and then attempting to match the two feature sets to locate the target object. The SIFT algorithm [9] is a method to extract distinctive features from an image. The features are descriptors of local image patches and are defined in such a fashion that they are invariant to scaling translation and rotation, and robust with respect to affine distortion, noise and partial occlusion.

SURF is a similar algorithm, but is reportedly significantly faster to compute than SIFT and capable of higher reliability in repeatability, robustness and comparison speed [1]. SURF was developed specifically as a fast competitor to SIFT. SURF key-points are selected using the determinant of Hessian of the integral image. This value is estimated using a Difference of Gaussian convolution, which is a simplification of the Laplacian of Gaussian convolution used in SIFT [4]. For scale-invariance, Hessian values are calculated in state-space, which is formed in SURF by varying the size of the Difference of Gaussian convolution templates. Conversely, SIFT forms scale space by down-sampling the image, creating an image pyramid. Key-points are selected as having the highest Hessian value amongst all neighbouring pixels in X, Y and scale. Once key-points are calculated, each key-point has an orientation and descriptor defined. The orientation is required to immunise the key-point to rotation. This is calculated by applying the Haar Wavelet template to all pixels within a set radius around the key-point. The Haar Wavelet is a fast gradient discriminator. The responses, in X and Y, are graphed and a 15 degree radial window is rotated about the origin of the graph. The orientation of the window which holds the most points is defined as the key-point's orientation. The descriptor is then determined as a vector containing the X and Y returns from Haar Wavelet convolution relative to the principle orientation calculated in the previous step. The result is a histogram of gradients similar to SIFT; however, it is calculated much faster due to preference of the Haar Wavelet.

SURF thus provides a robust and rapid method of identifying features, which can be suitably applied for our purpose of visual tracking.

2.2 Visual servoing

Visual servoing is the process of using visual information to provide closed-loop feedback on the pose of a robot, relative to a target feature [7]. It is generally applied to more complex systems than our missile launcher, which only has 2 degrees of freedom. Most implementations are used to control an end-effector mounted on a multi-joint manipulating arm, and track image features rather than using the simpler gradient detection we discuss in section 3.3. Nevertheless, some general points are applicable here. Current visual servoing systems are divided into two major categories, based on whether the error in pose is defined in terms of image or real-world coordinates [11]. The goal is to minimise this error. If features from the image are extracted and used to generate a geometric model of the current pose, the system is labelled as position-based. The system can then use this model to estimate control values that will minimise the error in real-world coordinates. In contrast, if the control values are generated directly to minimise errors defined in pixel distances, then the system is image-based. Such systems may be less prone to errors in camera calibration, and can reduce computational delays. However, directly generating the control values is a non-trivial task, particularly when dealing with complex kinematics. Our system is position-based, and the target can be thought of as a particular point on the table behind the missile launcher, which corresponds to the desired launch angle.

In general, visual servoing systems work by capturing an image of the target. Closed-loop systems also capture an image of the end-effector that they are aiming to move to coincide with the target, whereas open-loop systems rely on a model that links the end-effector's motion to given control values. The functioning of the system will also depend on whether the camera is mounted on the robot, so that commanded motion leads to changes in the camera position, or whether it is external and stationary. The former case is known as the "eye-in-hand" problem [3], and describes the situation in this project. A mounted camera was chosen due to hardware limitations, notably the poor resolution of the simple webcam used for visual servoing (in comparison to the more advanced one applied to target acquisition), which made it difficult to accurately sense the motion of the launcher from an external image.

Ultimately our system does not draw heavily from previous work in visual servoing, due to fundamental differences in scope and implementation. Whereas most established systems require a camera with a wide field of view to identify the relative positions of the target and manipulator, our camera has only a short view down to the calibration grid. Because of the very limited motion of the launcher, it is not necessary to use the generic approach of identifying relative target and pose locations, and finding multiple control values that will cause them to coincide. Instead, a simpler methodology can be used where the visual servoing process serves only to provide a constantly-updated estimate of the current pose, which can be compared to the target values. This form of closed-loop feedback is entirely sufficient for our purposes.

2.3 Launcher interface

The Dream Cheeky USB Missile Launcher comes bundled with a simple control application which communicates with the hardware over a USB cable. The only control options presented are graphical or keyboard based, which are both undesirable for integration in an autonomous system. For our integration of computer vision with the USB Missile Launcher, it was desirable to control the device using C++, as the OpenCV library is written in C/C++. This required the development of novel USB controller interface software, using the Developer Manual provided by Dream Cheeky for the technical details.

The manual documents the necessary format of the low level communication messages used to control the device [8]. Communication is in the form of simple bit-field serial messages. The message contents completely specify the state of the missile launcher, and hence multiple instructions can be bundled into a single message. The state of the missile launcher specifies movement in either direction on either of its two rotational axes, as well as a fire command. The missile launcher will continue to execute the most recently received message until another message is received.

The most important limitation of the interface is a lack of feedback as to the angle about both rotational axes of the missile launcher. This lack of an encoder in the missile launcher to measure position means there is no built-in way to accurately command the missile launcher to assume a desired pose in space.

3 Design

3.1 Target acquisition

Once the location of the target was reliably identified in the pixel space of the camera image, a transform was needed to calculate where it appeared in the real world, relative to the launcher tip. We firstly simplified the problem by constraining the Z-depth of the target from the camera to a constant 2 metres. This enabled us to apply a simple pinhole camera model to create a linear scaling factor to change between pixel distance and actual distance. Under a pinhole model, such a scaling factor would change depending on the Z-depth.

Camera calibration

Intrinsic camera parameters calibration (see, for example [13]) was not considered for this application, however it was necessary to calibrate the position and orientation of the camera relative to the position of the missile launcher. Automation of this process was briefly considered however was decided to be beyond the scope of the project. It proved sufficient to measure the location of the image centre and align the vertical axis of the image plane with the line at zero elevation and azimuth of the missile launcher. The setup is diagrammed in Fig. 1.

Applying the standard camera model:

$$\mathbf{p} = \mathbf{R}(\mathbf{P} + \mathbf{t}) \tag{1}$$

where $\mathbf{p}$ is the point on the image plane, $\mathbf{R}$ is the rotation matrix, $\mathbf{P}$ is the point in the world frame and $\mathbf{t}$ is the translation vector. The rotation matrix is the identity matrix as there is no rotation. The translation vector is defined as

$$\mathbf{t} = \begin{bmatrix} x_{CAM} & y_{CAM} & 0 \end{bmatrix} \tag{2}$$

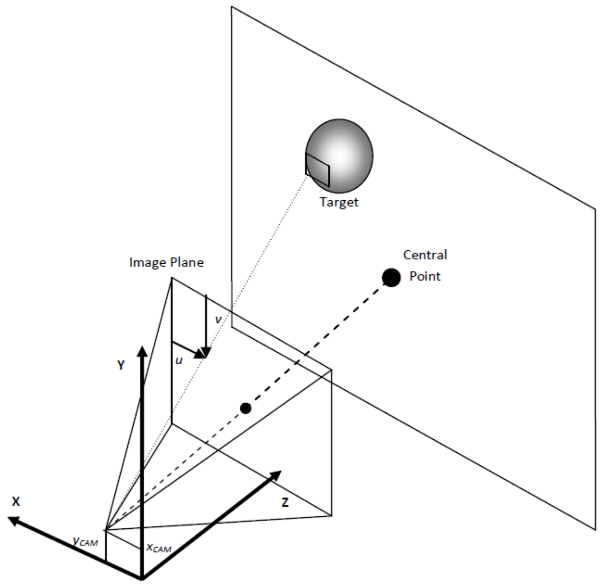

Figure 1: Camera calibration setup, showing the defined axes, the target, and the central point of the captured image.

where these are distances from the world frame origin, defined as the principle point of the foam missile launcher.

Object matching

The method described in Hu *et al.* [6] was applied using SURF features instead of SIFT. Target object features were extracted using the OpenCV SURF modules [2], then frames from a video were acquired and SURF key-points were extracted from these. The target used was selected as key-points obtained were high in scale and Hessian value. These values are particularly important, as key-points with large scale require a lower resolution to extract while high Hessian values represent key-point reliability. As the value of the Hessian shows the strength of the scale-space extremum which is selected as a key-point, this is vital to determining the repeatability and robustness of a key-point. Descriptor matching was achieved by adapting *find_obj*, a SURF implementation demonstration available with OpenCV. The function determines a key-point correspondence by summing difference values between the two vectors. A match is found only if lowest total difference is significantly smaller than the next closest. It was possible to determine the scale and orientation of the target object in the image based on the correspondences, however, problems arose at greater range as key-points in the target could not be matched accurately. This was first considered to be a resolution problem as results with larger targets were generally better, however increasing the video stream resolution resulted in a severe speed reduction with little performance improvement.

As discussed later in the results, it was noted that some matches were usually present. However, when trying to match a fully defined target, the matches were too rare to determine scale and orientation. Accordingly, this approach was abandoned, in favour of matching a single salient key-point with a unique descriptor, to two instances of a known target containing such a key-point, separated by a known spacing.

Initially, and for much of testing, the metric for selecting a key-point was size. This is a pixel representation of the scale of the convolution template used to obtain the key-point. The assumption used was that a larger key-point is easier to extract in a blurred or low resolution image, however this was flawed as a key-point with a strong Hessian value will be dominant over one with large size. This is especially true if the two key-points have similar size to begin with. SURF descriptors are a histogram of 128 gradients in regions around the key-point oriented with respect to the principle direction of the feature [4]. These can be compared directly to determine similarity. The strategy used was to find a strong key-point to use as a template and compare its descriptor to key-points extracted from the image frame. Two such points are selected from a target prototype and found in the image. Assuming that the target is always parallel to the image plane, it is then possible to obtain the pixel separation between the two points which is used as a scaling estimate to approximate z-distance. To account for clutter and false-positive matches for these descriptors, each point is modelled as a Gaussian and the results are α-filtered to reduce noise. Further clutter removal is discussed in the section on foreground segmentation below.

Foreground segmentation

Lens limitations on the Logitech Quickcam Orbit webcam used for target acquisition make it difficult to resolve images at a range greater than approximately three metres. By default, the camera operates at 320×240 pixels, however this can be improved to 640×480 or 800×600 pixels for better resolution whilst maintaining a capture rate of 30 frames per second. Increasing the resolution beyond this results in a decrease in frame-rate. It was found that at the lowest resolution, a frame could be processed for SURF key-points in approximately 20 milliseconds, giving a potential frame-rate of 50 frames per second. As expected, frame-rate fell with increase in resolution, as shown in the table in the results section. Extraction time depends on the complexity of the image and particularly the presence of regular repeating patterns. The solution to this was to create an image mask by subtracting the current image from a learned background model and round the pixel values. The background model was acquired by accumulating frames of the background with no target present, which were α-filtered to account for changes in luminance from fluorescent lighting, a method similar to one used by Matsuoka *et al.* [10].

3.2 Launcher interface

We developed a C++ interface for communicating with the Dream Cheeky USB Missile Launcher, so that it could execute in the same application as the OpenCV code. The Missile Launcher is recognised by the host operating system as a generic USB HID device, which allows it to be controlled by the open source USB-HID library available at Code Project for Microsoft Common Language Infrastructure languages such as VC++, C# and VB.NET [12].

The C++ interface allows the missile launcher to be commanded to move in any of the four directions, as well as simultaneously fire a missile. A "primed" fire can be achieved by pressurising the the compression chamber to the brink of firing pressure, and then stopping the fire process. Our results indicated the complete firing process took about 4.3 s. Thus to prime a missile ready for quick release we pressurised for 4.0 s and then stopped.

Furthermore we extended the set of available actions by implementing a method for performing slow movements in one axis at a time. This was achieved by simultaneously driving the motor on the other axis in both directions, whilst driving the motor on the axis of interest in the desired direction. The alternation between motors resulted in a much slower movement in the axis of interest, whilst oscillating very slightly in the other axis. The system operates either under timed movements, or fully managed movements where feedback is used to control the start and stop signals directly.

3.3 Visual servoing for launcher pose

The position of the missile launcher relative to the target tracking camera is known, based on the experimental setup. However in the absence of a built-in position encoder in the missile launcher, we cannot directly determine the angle of the missile launcher barrel with respect to the target tracking camera. Our initial investigations found that timed movements did not prove to be accurate enough to position the launcher using dead reckoning. This was because the motor speed varied considerably under different circumstances depending on: the direction the motor was previously moving; the size of the commanded motion; and whether the missile launcher was simultaneously moving and firing. Even under the same conditions there was still some unpredictability. This was unacceptable, given the need to provide as much accuracy as possible.

We thus implemented a self-localisation approach using an external camera to determine the current pose of the missile launcher in space. We use a camera attached to the missile launcher, capturing a video feed of a calibrated pattern. This calibrated pattern is a radar plot, with increments of 2.5° in azimuth and 5° in altitude, that can be read to allow the missile launcher to localise itself with respect to the environment, by counting grid squares as it moves away from an origin. The origin pose in the calibrated pattern is well defined, as described in section 3.3.

The experimental configuration consists of a camera attached to a metal bracket, which is attached to the missile launcher, such that the camera faces down towards the table surface behind the missile launcher. The attached camera and calibrated plot can be seen in Fig. 2. The angle of attachment was tuned such that the missile launcher base was never in the view of the camera. The view attained resulted in the camera having a bird's eye image of the calibrated pattern at high angles, meaning the pattern was very close to linear. At lower angles, the camera view of the calibrated pattern has a perspective effect. This is mostly accounted for during the generation of the calibrated pattern, however the high angle positioning is definitely more accurate, as it is a better approximation of a linear conversion between camera pixels and angle.

To aid in the self-localisation process, which requires a sharp, constant video stream when closing in on the target pose, we used the slow movement method outlined in section 3.2 when nearing the target position. During the longer distance movements we were able to move the missile launcher at full speed, due to a well designed calibrated pattern, as described in section 3.3.

Calibrated pattern

The calibrated pattern was generated by attaching a laser pointer to the barrel of the missile launcher and manually moving the missile launcher so that the laser lined up with the previously calculated positions of altitude (varying between -5° and 30°, in increments of 5°) on a wall at 2 metres. The centre of the camera view was marked on a piece of paper that was underneath the missile launcher. These altitude markings were plotted as semi-circles around the base of the missile launcher, and separated into 2.5° increments of azimuth.

The reason for the difference in increment size between the azimuth and altitude is that the azimuth motor tended to be more accurate than the altitude motor. The azimuth motor moved more slowly and with considerably less backlash when changing directions. In addition, the firing consistency of the missile launcher was lower in the altitude than in the azimuth, and so this larger uncertainty reinforced the need for larger in-

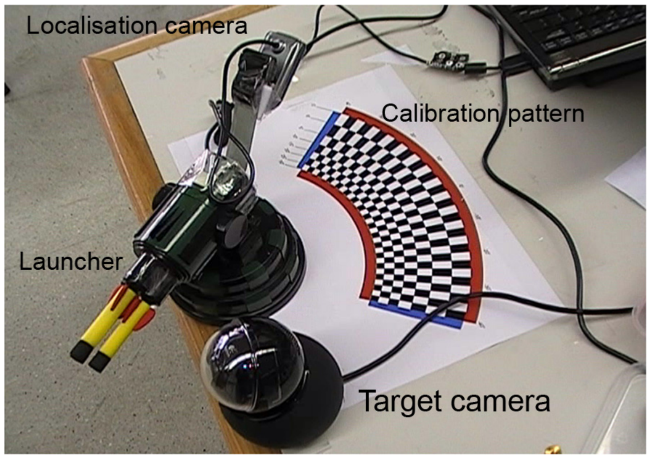

Figure 2: The complete system, with the two cameras, launcher, and calibration pattern visible. The controlling laptop is visible in the background.

crements.

The boxes in the calibrated pattern were alternately filled black and white to permit accurate tracking at high speeds, as described in section 3.3. The boundary areas were filled red in the altitude direction and blue in the azimuth direction. The design of the calibrated pattern is quite simple, but it is effective. We experimented with various types of calibrated patterns, involving greyscale gradients and use of colours, however these analog approaches produced results which were inconsistent under different lighting. The digital approach of alternating black and white boxes did not suffer these inconsistencies, under a wide range of lighting conditions. No particular attention was paid to the calibration differences between firing each barrel, as the missile grouping was acceptable and tracking which barrel was due to fire next without hardware feedback proved problematic.

Locating the origin of the calibrated pattern

When the system starts up, the missile launcher is in an unknown position, somewhere within the legal range of positions. This means the camera is viewing the calibration pattern, but it doesn't have any well defined point of reference. To solve this issue, we perform an initial reset operation, to locate the origin position. The origin position is one of the corners of the calibration pattern, specifically the highest altitude and lowest azimuth. This position is found by moving the altitude up until the camera hits the red boundary area. The red boundary area is readily distinguished from the standard black and white squares of the pattern, by its high relative red intensity, even in low light. It then moves left until it reaches the end of the red boundary, which means it is directly above the origin box. The missile launcher moves back down until the camera is centred in the origin box. The system is now ready to count boxes as described below.

Tracking movement using the pattern

The calibration pattern essentially consists of a series of lines, which appear roughly horizontal or vertical from the centre of the field of view of the camera. When a single line is tracked by the camera, such that the line remains in the centre of the camera view, the missile launcher azimuth or altitude is known and remains constant (azimuth is constant along vertical lines and altitude is constant along horizontal lines). When the camera view is centred between the lines in either dimension, the current angle can be estimated by interpolation, based on the distance of the centre pixel of the image to the nearest line on either side. For simplicity we use linear interpolation, which is a good estimate in the azimuth position, but will underestimate the altitude angle. This is because of a perspective effect, resulting from the angle of the camera becoming non-perpendicular to the calibration patten plane, as the missile launcher azimuth angle reduces.

We attempted to use Sobel and Canny edge detectors to detect the lines between boxes, but these had some difficulties with the corners between four boxes, as well as inconsistent performance under different lighting conditions. We were able to produce better results by analysing the gradients directly, using a 6×6 pixel window. The window began in the centre of the camera image and was slid in all four directions, until the first line gradient was located. This method worked well because we had the prior information of the gradient direction and colour.

Practically, the video stream from the camera is used to describe the pose of the missile launcher in integral and fractional components of a box, in the two dimensions. The integral distance component of the camera view, relative to the well defined origin pose, is measured in full boxes. This figure is reached by counting boxes from the origin position, as the missile launcher, and hence the camera, moves. The origin pose is detailed in section 3.3. The fractional components of the pose are then obtained once the missile launcher has stopped, or slowed greatly, and are calculated based on where within the target row of boxes the camera is centred.

Because the camera moves relatively quickly across the calibration pattern, when counting boxes, we chose to alternately fill the boxes in the calibration pattern black and white. This way the camera can easily track the alternating transitions from black to white of the centre 3×3 pixels, without becoming confused between two adjacent boxes and skipping or counting a box twice at high speed. The black and white boxes provide excellent contrast even in poor lighting conditions. Since we command the camera to move, we know which direction it is moving in, and so there can be no confusion as to which direction a transition (or edge crossing) has occurred in.

Once the camera is within one box of its goal position it can slow down and use the linear interpolation process described above to attain the correct position for the given dimension. In our implementation this is done by moving until the camera has just seen it is in the perfect position, then it signals the missile launcher to stop moving. This naturally results in a slight overshoot, but the slow speed means the overshoot was minor. A better implementation could use a PID controller to smooth the approach to the goal position, and correct any overshoot.

The missile launcher bounds were hard-coded in to the program, as azimuth between -38.75° and +38.75° and altitude between -2.5° and +27.5°. The original plan was to use the red and blue boundary areas on the calibrated pattern for delineating the limits, at -40° and +40° azimuth and -5° and +30° altitude. Unfortunately due to time constraints we were unable to use the boundary regions for anything other than the initial reset procedure.

3.4 Ballistic flight model

An important aspect of achieving target interception was developing an accurate model for the flight path of the missiles. Since the launcher is sold as a toy, these are made of lightweight foam. The unmodified missiles were tested to determine if they were suitable for use in their native state. Observations with both the naked eye and video recording showed that they were highly inaccurate, often not shooting in a straight line, and achieving very poor repeat performance at hitting a target. The hypothesised cause was that the missiles' low mass made them very susceptible to minor misalignments in their tail-fins and small air currents, allowing them to easily rotate off course. To increase the weight, brass thumbtacks were attached to the end with an adhesive. Two orientations were used: with the tack's blunt cap exposed, and with the tack's sharp pin exposed. We assessed the difference in air resistance between the two forms as negligible. The blunt missiles were used for most testing, for safety reasons, while the sharp missiles were used to pop target balloons as a demonstration. See Fig. 3 for an illustration of the missile types.

The modified missiles were tested by firing them with the launcher set at known altitude and azimuth angles, and measuring their point of impact with a large target. The impact location was assessed with the naked eye. The placement of the target was arbitrary; we used a depth of 2 metres from the launcher to match our desired final target. This meant the missiles were hitting it at roughly the top (apogee) of their arc trajectory, as confirmed by recording video of the missiles in flight. An example frame is shown in Fig. 4.

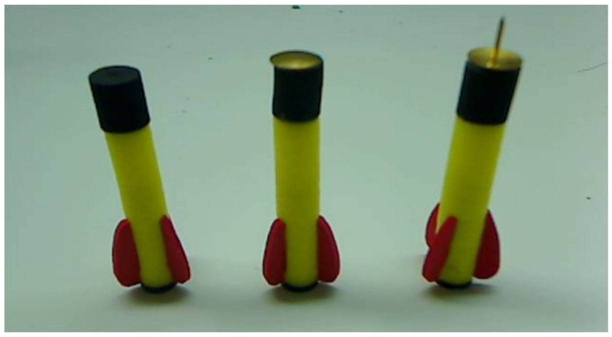

Figure 3: The three missile designs tested. From left to right: the unmodified form; the blunt tack form; and the sharp tack form.

Figure 4: Video capture for flight path viewing. The rapidly moving missile is visible as two short blurred lines within the marked white box.

The launcher was tested at three different altitudes, for each of two azimuth angles. Three missiles were used in each test and we assessed the centre-point of the impacts. For every launch, the height and horizontal distance of the impact point relative to the launcher were recorded. These results were plotted in Matlab, together with the ideal projectile motion calculation assuming no air resistance. This plot is shown in Fig. 8. The experimentally measured results matched the ideal case quite closely, meaning that using the equations describing ideal flight was a reasonable approximation. The fit is better at higher altitude angle. The discrepancy at 30 degrees is only 1 cm, but at 20 degrees, the measured impact height was up to 10cm below the ideal projection. To reduce this, an offset was added to the calculated altitude angle, which was inversely proportional to that altitude, so that low values were boosted but high values were left unchanged. This model would have to be re-assessed if a different type of missile was used, since the offset or even the validity of the projectile motion approximation may not be applicable.

Once the use of ideal projectile motion was proven to be acceptable, algorithms were implemented to calculate the required altitude and azimuth angles, given a target position in 3D Cartesian coordinates relative to the launcher. The azimuth calculation is simple, given a depth Z and a horizontal displacement X.

$$\phi = \text{atan}(X/Z) \tag{3}$$

The altitude angle calculation, given X and Z as well as the vertical height Y, is more complex. It cannot be solved analytically so a numerical approach was implemented. Angles in 1-degree increments across the launcher's range of motion were checked, and the best one chosen. The metric was the vertical displacement between the desired Y height and the calculated missile height after travelling the relevant distance under ideal projectile motion. This required distance to the target was determined using a simple hypotenuse calculation given X and Z. This was implemented in Matlab and shown to be a rapid calculation (0.01 s), especially in comparison to the time for servoing and target identification.

4 Results and Discussion

4.1 Target acquisition

Target object matching using the Hu *et al.* method was consistently a success at ranges around 1 metre and below for any rectangular target with a minor dimension no greater than 3.5cm. This corresponds to a minimum visual angle of approximately 2°. Increasing the object size to 6cm increases the effective range to approximately 1.8 metres. This is expected as the visual angle is maintained at 2°. Typical results are shown in Fig. 5.

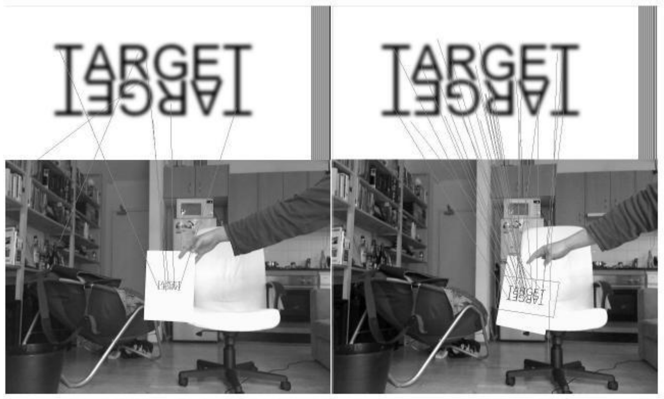

Figure 5: Object matching performance at 2 m range, using a small target (left), and a large target (right).

Performance of single descriptor matching was similar to that of the full object match. Fig. 6 below shows the target that was used and the key-point chosen by either size or Hessian value.

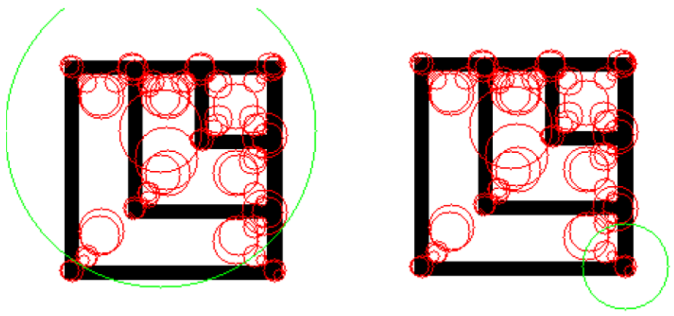

Figure 6: Descriptor matching targets, selected by size (left) or by Hessian (right).

Performance is not significantly improved by either choice and the correct key-point is typically recoverable reliably at a range of 2.8 m using a 12×12 cm target. This corresponds to a visual angle of 2.64°, which is significantly larger than the value required for object matching. This performance is shown in Fig. 7, discussed below. To match with lower precision, to within the dimensions of the larger balloon, a target of 4×4 cm can be used at a range of 2 m, requiring a visual angle of 1.14°.

The process of extracting SURF key-points is very costly and scales with the number of pixels being considered. The mean time taken to extract features on a 1.6 GHz Celeron for varying image sizes is presented in Table 4.1.

Image size (pixels)	Time (ms)
320 x 240	20
640 x 480	160
800 x 600	600
960 x 720	1100

The result of applying an image mask was that only newly introduced objects in the image were processed for SURF points, significantly reducing extraction and comparison times. In addition, clutter features in the background were effectively discarded. Extraction times vary depending on the fraction of new data in an image therefore it is not meaningful to include these as results. It is sufficient to note that this technique allowed effective target segmentation and a great improvement on SURF extraction time. Fig. 7 shows the application of the mask and identification of the target. Correct placement on a 12 cm target at long range is demonstrated. This represents the current state of the system.

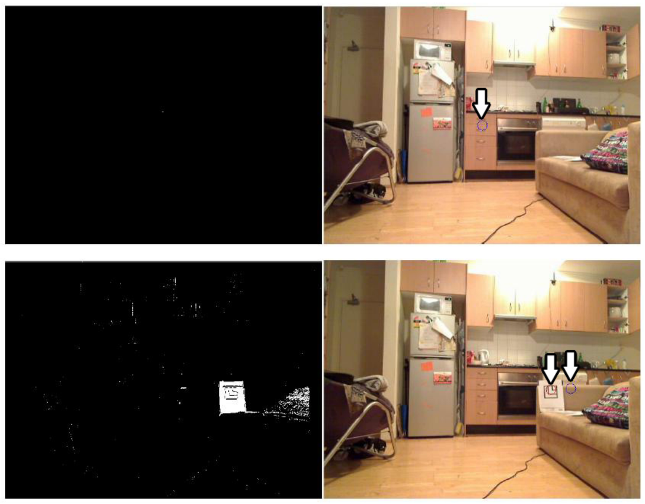
Figure 7: Descriptor matching precision and foreground segmentation. Images on the right are video input, with masked images shown on the left. The top two images show the learned background, while the bottom two images demonstrate the introduction of a target and resultant descriptor matching. Arrows have been superimposed to highlight the computer's placement of the SURF key-point. Note that the point is placed centrally when there is no target (top right), but introduction of a target creates a foreground (bottom left) and allows accurate placement of the key-point (bottom right).

Foreground segmentation from a known background model is thus very useful in reducing the time spent extracting image features, however the background model is static and would not deal adequately with long-term changes in the environment, such as moving a usually stationary object, or time-variant areas such as computer screens, or areas of high traffic. Adaptive background-modelling techniques exist however these tend to update the background model with every new image frame [10]. These techniques cannot be applied when the target may be static as it will become part of the background model.

At present the system uses image differencing to determine pixels that have changed value exactly. It may be beneficial to account for changes in luminance by performing segmentation in HSV space rather than RGB while also blurring the background and image frame during comparison to negate small changes in orientation or position; for example, see [5]. The two methods described for obtaining target object scale from features both have limited success. Object recognition and tracking using the Hu *et al.* [6] method is highly successful at close range, around 1 metre, with targets sized to system specification. Range can be increased but only by increasing target size. Scale and orientation are fully recoverable given a viewing angle of 2° or greater.

Descriptor matching is a probabilistic approach to target identification with similar performance to object recognition in terms of visual angle required for detection. Performance degrades gracefully with reduced visual angle, due to the method's statistical nature, and precision to target size can be achieved at a visual angle of 1.1°.

The question of why the feature match fails at low visual angles must be addressed. A possible explanation is that while key-points are successfully extracted, Hessian values for these points are stronger for a larger scale. Thus the area covered by the image key-point descriptor is greater than the template descriptor and there is a mismatch of Haar wavelet gradients during comparison. It may be possible to account for this by encoding key-points at all scales, however computation time and memory requirements would be severely increased.

4.2 Visual servoing

The visual servoing subsystem worked reliably and effectively, moving the launcher barrel to specified positions entered by the user. It was also successfully integrated with the rest of the system, allowing missiles to accurately and repeatedly hit targets in an automated manner. The reset procedure was quick and effective, accurately finding the bottom corner of the target pattern. The final routines were significant improvements over early versions which would often traverse several boxes before stopping and run off the end of the calibration grid. This illustrates the vital importance of dealing carefully with special cases such as moving along a boundary, where detecting gradient changes is difficult. The merit of the binary white and black colour scheme was also demonstrated. Multiple patterns were tested, involving gradients and different colours, but these proved to be inconsistent under different lighting conditions.

The excellent targeting accuracy is a result of the careful construction and calibration of this pattern. It is very specific to our particular system setup, however, and took a considerable amount of time to manually generate. It would also difficult to automate its creation, given the need to refer to real-world targeting data for a particular launcher.

Overall, hardware difficulties posed the most significant problems. Occasional delays in the USB camera feed meant that the system would skip a box before the vision data was processed. The accuracy penalty associated with skipping one box is relatively small, but can be problematic, especially in the altitude direction where small variations can lead to large changes in the missile position at target range. More significantly, the azimuth drive motor broke and lost all function, even under control from the manufacturer's included software.

We could not ascertain whether this was a problem with the motor itself or with any control electronics. It is possible that the repeated movement at low speed caused it to burn out, as this functionality is not provided by the manufacturer. As a temporary measure, the system was modified to provide instructions to the user to slowly rotate the launcher turret clockwise or anti-clockwise, thus allowing them to act as the motor. The visual servoing system was still used at this time to provide position feedback, either telling the user to continue rotating the turret, or determining that the position was correct and firing the missiles. It is important to note that slow movement of the launcher is necessary in order to provide sufficient resolution in the pose, so if this was indeed the cause of the motor malfunction, the system must be adapted to allow slow motion without stressing the motor. A stepwise approach could be applied, allowing the motors to pause between each traversal of a box.

4.3 Ballistic flight model

Fig. 8 shows the results of the ballistic flight path testing, comparing the simulation of ideal projectile motion with the measured missile impact points. It demonstrates the close correspondence that validated our use of ideal projectile motion as the basis for determining missile launch angles. It also illustrates the deviation of the ideal model from experimental data as altitude angle is decreased, motivating the addition of a sliding offset that scales with the altitude.

The angle calculation algorithms derived from this flight model proved to be very effective. They reliably achieved missile impact with the target balloon on multiple occasions. Target grouping was typically within approximately 5 cm. Use of the sharp tack missiles allowed the balloons to be popped, except in some demonstrations where the targets were incorrectly set at a height above the launcher's maximum vertical reach.

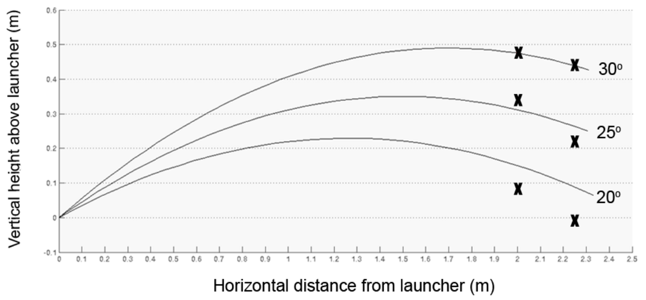

Figure 8: Missile flight paths for varying altitude angles. The three lines correspond to the ideal projectile motion calculations, while the crosses illustrate measured data using the blunt missiles.

5 Conclusions and Future Work

The system presented is a successful implementation of a basic self-targeting missile launcher, achieved within a restricted timeframe. It successfully demonstrated the use of foreground separation and SURF feature detection to identify and locate a target within an image. The visual servoing system provided accurate pose feedback and rapid reset behaviour. The ballistic path analysis enabled the system to generate accurate estimates of the altitude and azimuth angles required for the foam missiles to hit targets with a high degree of reliability. Given further time and resources, more expansive goals could be attempted, such as tracking a moving object. Some possible directions for future work in this field are presented below.

In the area of target acquisition, SURF features are very useful for object recognition in static images, but cannot be successfully used to locate a moving target because of several shortcomings. First, while SURF features are robust to blurring due to the scale-space processing, there is no provision for motion smear. This problem can be mitigated using a faster rate of capture, although this is not practical for most webcam-style cameras. Second, whereas a target object may be examined for SURF key-points in isolation to produce a reliable set of descriptors, there is no way to guarantee that features extracted from the video image containing the target object will be identified at a corresponding scale. It is not yet a simple task to implement video tracking using object recognition. Another key limitation of the current targeting system is that the range (Z-distance) must be input externally, although X and Y coordinates are determined from the image processing. Future effort should be directed towards finding a way to extract the range using the scale of a distant object image, by comparing the apparent size to a pre-known actual size. Likely, this means investing effort in a more robust feature and descriptor.

The visual servoing subsystem is largely complete, but possible improvements exist. A form of analogue colour gradient could be used for position feedback. For example, the red channel could represent altitude and the blue channel azimuth, with the camera reading both components to determine its pose. However, there are two caveats. Lighting conditions would need to be more stable, perhaps via the use of external illumination. Also, detailed testing would be required to determine whether particular cameras have sufficient accuracy to reliably distinguish small changes in the colour signals, given the need for finely resolved position information.

Finally, when developing systems such as these, greater attention should be paid to safety. Our launcher is not powerful, using only a small air compressor to fire the missiles, but the sharp tacks still pose a threat of eye injury. A safeguard to indicate when the system is about to fire, probably requiring user input to verify that it is safe and the target area is clear, would be an important inclusion. A more rigid missile, retaining a low mass but made of a material stronger than foam with tighter manufacturing tolerances, would further improve reliability at hitting targets.

Acknowledgments

This project was carried out as part of the senior course in Experimental Robotics at the University of Sydney, which provided facilities and support. The authors would like to thank Stefan Williams, Tom Allen and Stephen Barkby for productive discussions and feedback on the project goals and design.

References

[1] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346 – 359, 2008.

[2] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc., 1st edition, October 2008.

[3] F. Chaumette and S. Hutchinson. Visual servo control, Part I: Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4):82–90, December 2006.

[4] Christopher Evans. Notes on the OpenSURF Library. Technical Report CSTR-09-001, University of Bristol, January 2009.

[5] M. Harville, G. Gordon, and J. Woodfill. Foreground segmentation using adaptive mixture models in color and depth. In *Detection and Recognition of Events in Video, 2001. Proceedings. IEEE Workshop on*, pages 3–11, 2001.

[6] Xuelong Hu, Yingcheng Tang, and Zhenghua Zhang. Video object matching based on SIFT algorithm. In *Neural Networks and Signal Processing, 2008 International Conference on*, pages 412–415, June 2008.

[7] Seth Hutchinson, Greg Hager, and Peter Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12:651–670, 1996.

[8] Dream Link. USB Missile Launcher - Developer Manual v1.0.pdf. `http://www.dreamlink.info/dream/forum/download/file.php?id=10&sid=508ada005b29c7b0d67e7aa723ce91e4`, November 2008.

[9] David G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 1150 – 1157, Kerkyra, Greece, 1999.

[10] Masayoshi Matsuoka, Alan Chen, Surya P. N. Singh, Adam Coates, Andrew Y. Ng, and Sebastian Thrun. Autonomous helicopter tracking and localization using a self-surveying camera array. *International Journal of Robotics Research*, 26(2):205 – 215, 2007.

[11] A.C. Sanderson and L.E. Weiss. Image-based visual servo control using relational graph error signals. In *Proc. IEEE*, pages 1074–1077, 1980.

[12] CodeProject user 'wimar'. A USB HID Component for C#. `http://www.codeproject.com/KB/cs/USB_HID.aspx`, March 2007.

[13] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, Nov 2000.