공학석사학위논문

Object Tracking with Low Complexity SIFT Matching

2012년 8월

서울대학교 대학원

전기 컴퓨터 공학부

Brian David Fransioli

Abstract

Local invariant features as a method for object recognition in images has become a widely researched area in the field of computer vision. The more powerful features which exhibit scale, rotation and viewpoint invariance require expensive scale-space searches which involve Gaussian convolutions or their approximations for detection. In addition, robust, distinct feature description and matching adds to the computational complexity. SIFT is one such robust, local invariant feature popular in image matching for stereo vision and object recognition. It comes with disadvantages such as high computational complexity and suffers from the curse of dimensionality. Features which exhibit scale and rotation invariance, such as SIFT, are notorious for expensive computation time, and often overlooked for real-time tracking scenarios. This paper proposes methods to rectify speed issues, even while improving matching accuracy.

In order to harness the power of local invariant features such as SIFT in the realm of tracking, this work introduces a reduced complexity tracking system that exploits temporal locality and uses a radial metric to consistently match features between consecutive frames. In addition, descriptor modifications are performed to rectify high dimensionality errors, and poorly tracked keypoints are pruned and replaced as need, maintaining an ideal count of features in the tracking database for accuracy and low complexity. This system outperforms the standard, naive SIFT tracking system in processing time and maintains better error rates.

This radial matching idea is further extended, and a descriptor-less matching algorithm is proposed which is based on finding the geometrically closest candidate to each tracked reference feature in the database using motion vectors between consecutive frames. Descriptor-less matching forgoes expensive SIFT descriptor extraction, which is the most expensive component of the SIFT tracking process, without loss of matching accuracy; D-Less matching exhibits dramatic speed-up compared to traditional, naive SIFT based trackers.

The proposed Radial Matching system runs at 14-15 fps on a low-end Intel dual core machine without optimizations. As well, D-Less SIFT tracking runs in real-time on an Intel dual core machine at an average of 25 frames per second.

Keywords : SIFT, SURF, Object Tracking, Computer Vision

Student# : 2010-24082

List of Figures

List of Tables

Table Of Contents

1 Introduction

1.1 Foreword

Automated object detection and tracking in computer vision systems has become a well researched area with many developments over the last decade. The task of recognizing a reference object in a test image is more daunting a task than one might imagine. It has often been the case that one system built for a specific type of object can not easily be generalized to other objects. The complexity of the object to be tracked also introduces problems for successful redetection. Deformable geometric objects can take on a range of poses which usually require a detection system to be able to distinguish between one of several poses, increasing memory and processing complexity. The rotation of objects further complicates the process by forcing the system to be resilient to both in-plane and out-of-plane rotations. Various continued complications arise such as [severe] lighting changes, full or partial object occlusion, and multiple instances of the object of interest.

For all of the challenges inherent in object detection and tracking, there have been a variety of tools and techniques developed to combat each challenge. In this paper, we examine the progression of object detection and tracking to the current state-of-the-art forms, and discuss and show test results for a reduced complexity system for the general case of object tracking in real-time.

1.2 Object Detection Problem Specification

Given a reference class which is a description of an object of interest, we want to identify the location of the object of interest in future images. The object may take on different poses and orientations, and may be present in a range of different environments. The task of object detection is to maximize the probability of detection of the object of interest in an arbitrary test frame.

1.3 Extension to Object Tracking

Object tracking in a continuous video sequence can be viewed as an extension of the object detection problem; the task is to detect in every frame the object of interest, recording its trajectory through the test sequence. While object tracking is a restatement of the problem of object detection, video sequences provide an extra dimension of information - temporal locality between frames - which we will use in solving the object tracking problem.

1.4 Book Materials

All source code, test sequences, ground truth, and results for both the tracking system and the files for creating this document are available at my website.[1] You are welcomed to use these in your own endeavors or to improve upon mine. Please direct all inquiries related to this document to **terranpro@capp.snu.ac.kr**. Enjoy!

[1] http://terranpro.org

2 Previous Works

The progression of object detection and tracking in computer vision has developed considerably over the past decade. In order to understand the state-of-the-art, a brief history of and introduction to this progression is given. Finally, a working knowledge of the field of previous work in this research (SIFT, RANSAC, Optical Flow) is given to allow the reader to follow the discussion without needing to reference other papers.

2.1 Template Matching

Early forms of object detection and tracking were based off template matching schemes. A reference image or set of reference images is scanned across a test image using a difference metric for similarity. A resultant density image is created which its pixel values correspond to the result of the difference metric. A common scheme is to find the maximum regions in the resultant image which have the strongest vote for the presence of the reference object. The conceptual workings of a simple, template matching system are visualized in Figure 1.

It should be obvious there are severe limitations present in this type of detection system. Even minor angles of rotation will reduce the likelihood of a detection. As well, variations to scale are impossible to detect without special considerations, such as multi-pass scanning at differing scales. During a single pass, the window shift offset is another consideration. Shifting the window by one pixel for each subsequent window calculation results in more calculations, but better

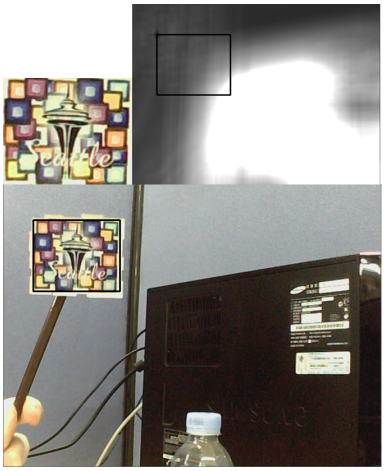

Figure 1: Template matching using normalized squared difference.

accuracy; similarly, a larger window offset results in less computations but lowered accuracy for detection where boundaries for the object in the template and the query image do not line up. Even on modern desktop machines, the time for computation when changing from a coarse to a fine offset increases dramatically - quickly eliminating any potential for real-time execution.

2.2 Image Features

Much of the state-of-the-art relies on the idea of decomposing image information into a number of interest points[2] which convey high amounts of information. The benefits are multi-fold. By reducing the number of points to process from the set of pixels in an entire image

[2]Also called keypoints, features, feature points, or points of interest. The various forms are seen in the literature, and differences between these are slim - if perhaps only with respect to connotation. Keypoints may distinguish an interest point and its location only, without its description. Whereas, features may be a more encompassing term, usually referring to both the keypoint and its description.

to a relatively smaller group of highly distinct keypoints, we stand to drastically reduce the processing time for detection. As well, by detecting distinct points in an image, they are easily reproducible and descriptive of the objects that contain them.

2.3 Relation to the Human Vision System

Much of the corner point and interest point detector theory has roots in the way psychology understands human vision processing and perception. Biederman[2] hypothesizes that human vision processing is based on identifying components and the organization of components into objects. This process begins with a similar feature and edge detection process, which then forms the building blocks for larger, components. From these components, their organization creates a matching representation which we use to distinguish one particular object from another.

Biederman also shows through testing that elimination of corner contours in an image results in lower identification rates, whereas elimination of areas with smooth curvature (i.e. straight lines connecting corners) can still be readily identified. This is illustrated in figure 2. The study suggests that component recognition is necessary for object recognition.

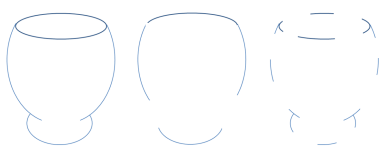

Figure 2: Illustrating non-recoverable vs. recoverable contour deletion.

2.4 Corner Detector

Another popular feature used in image processing and computer vision is the standard, Harris corner detector[5]. In general, detection works by examining a local region around a point - a patch of some size - and examining the auto-correlation of adjacent patches shifted by a small amount in various directions. In the simplest case, an edge is detected when a shift results that is perpendicular to the edge; a corner is detected for any shift (perpendicular or parallel) when the minimum difference between all shifted patches is large.

Harris extended the most basic case of patch shifts at fixed, 45 degree angled directions to examine *small shifts* as well, by incorporating directional gradients of the image. This extension results in detection which features rotation invariance. Denoting the change in intensities produced by a patch shift as $E(x, y)$, all small shifts about a patch can be written as:

$$E(x, y) = Ax^2 + By^2 + Cxy \tag{1}$$

where

$$A = \frac{\partial^2 I}{\partial x^2} \otimes w$$
$$B = \frac{\partial^2 I}{\partial y^2} \otimes w$$
$$C = \frac{\partial^2 I}{\partial x \partial y} \otimes w$$

Where w is a window used for convolution. It can be weighted or unity. Examining the eigenvalues (directly proportional to the curvatures) of a matrix concise form of $E(x, y)$, three observations can be made:

- If both values are small, the local auto-correlation is relatively small; the region is roughly of constant intensity, and **not interesting**
- If one value is high and the other low, this indicates a ridge shape of the auto-correlation function, and the presence of an **edge**.
- When both values are high, then shifts in any direction will increase the $E(x, y)$ function; this indicates the presence of a **corner**.

The popularity of the Harris corner detector became a starting point for future research in local feature detection. The original Harris detector, while exhibiting rotation invariance, was not scale invariant, and future research moved towards this direction.

2.5 SIFT

The Scale Invariant Feature Transform (SIFT) is a method of transforming image data into scale invariant feature points; these keypoints are robust to scale changes, rotation, illumination variations, and viewpoint changes. As well, keypoint description of the surrounding region is highly distinctive, which allows for accurate matching of features between images or inside a large feature database. An

overview of the critical aspects of SIFT will be covered here; see [12],[13] for more about the original SIFT algorithm.

2.5.1 Overview

SIFT keypoints are well localized in both spatial and frequency domains. SIFT consists of four major stages.

- Scale-Space Extrema Detection
- Keypoint Localization
- Orientation Assignment
- Keypoint Description

2.5.2 Scale-Space Analysis

Detection of keypoints begins by examining the image at different scales with variable Gaussian blurring. The key idea is that strong features are repeatable, and will occur at multiple scales (or sizes) of the image.

In order to detect features that are invariant to scale, localization of stable features across multiple scales is used in a space known as *scale-space.*[21] Scale-space measures the ambiguity of scale in an organized, progressive manner. It is built by convolutions of a signal (image) with Gaussians of a continuum of sizes; the scale-space is then collapsed together similar to a tree form, and stability criterion are used to identify features present across large scale changes.

SIFT uses an efficient difference-of-gaussian (DoG) function as an approximation to the convolution with a gaussian to build a scale-space pyramid of images. The first step of constructing the scale-space pyramid is to consecutively blur the images using a Gaussian filter. The original image is repeatedly blurred using a sigma value, for example, $\sigma = \sqrt{2}$. At the end of *level* the original image is scaled down, and the process is repeated. This is seen pictorially in Figure 3.

Figure 3: Gaussian Blurred Pyramid

The next step is scale-space construction is to take the difference between consecutively blurred images. This difference is what is referred to as the *Difference of Gaussians*. In a similar manner as above, the DoG scale-space pyramid is constructed and can be seen visually in Figure 4.

Finally, extrema (minimum and maximum) points are detected in the image, by collapsing together images from adjacent scales. These extrema points that are found to be repeatable at multiple scales are what constitute the scale-invariant property of SIFT.

A pixel is compared to its neighboring pixels at the current level

Figure 4: DoG Scale-Space Pyramid

and levels above and below in scale-space. If this pixel is either a minimum or maximum, the process is repeated at the corresponding location on the next scale-space level. In this manner, pixels that are not extrema are quickly rejected, and only serious candidates for features are examined at multiple scale space levels.

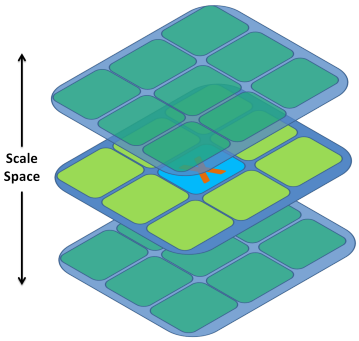

Figure 5: Scale Space Collapsed for Maxima/Minima Detection

2.5.3 Keypoint Localization

Points that pass the extrema detection phase are subject to further criteria to insure they are strong, repeatable features. The Taylor

expansion of the scale-space function is used to determine sub-pixel accurate localization (and possibly reject) of a feature. As well, the curvature, similar with the discussion earlier in 2.4, is used to distinguish an edge from a corner. See [12] for details.

Test	Reason
Location Interpolation	Accurate Sub-pixel Localization
Magnitude Thresholding	Robustness to Illumination
Principal Curvature	Remove Edges

2.5.4 Orientation Assignment

At this stage, each point that has passed the previous tests is deemed a point-of-interest, or feature. Each feature has a location at a certain scale, σ, and using this scale space location, a feature is first assignment a gradient magnitude and orientation by using the Gaussian smoothed image from earlier.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{2}$$

$$\theta(x,y) = tan^{-1}\left(\frac{L(x,y+1) - L(x,y-1)}{}L(x+1,y) - L(x-1,y)\right) \tag{3}$$

A histogram of orientations is formed with 36 bins, representing the range of 360 degrees around a keypoint. Samples located around the feature are added to the histogram using a Gaussian weighted circular

window with $\sigma = 1.5$ times the scale of the keypoint. This weighted, circular window helps avoid sudden changes in the descriptor, and emphasizes gradients that are closer to the center of the keypoint. From this, the dominant orientation of the feature is found[3].

Figure 6 shows an example of the SIFT detector's final feature candidates along with their dominant orientations. The *size* of the feature is also directly related to scale it was located at.

Figure 6: Rich Display of Final, Detected SIFT Features

2.5.5 Keypoint Description

For description, first the dominant orientation of the feature is used to rotate the coordinate system - achieving rotation invariant description. Gradient magnitudes and orientations are sampled around

[3]There are also cases where multiple orientations for a single keypoint are used. This is a detail not closely relevant to our need for conceptual understanding. See Lowe's journal article for more.

the area of the keypoint, again using the scale to determine which Gaussian blurred image is used.

The descriptor itself is vector which is a flattened combination of orientation histograms. For SIFT, `8` bins in the orientation histogram are used, and the patch around the feature is split into separate, $4x4$ regions each with its own orientation histogram. After the collection process is done, these histograms are flattened into a single, 128 dimensional vector (`4*4*8 = 128`).

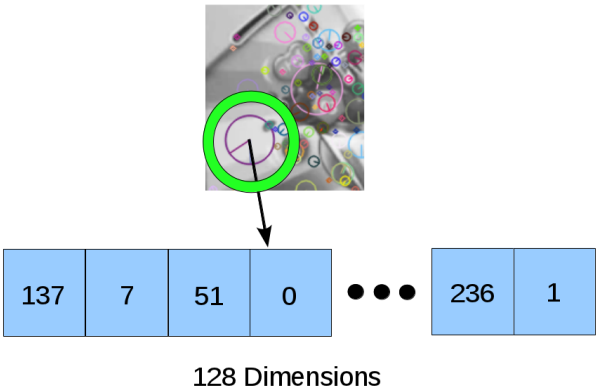

Figure 7: Visualization of a SIFT Descriptor

Finally, the feature vector is modified to increase robustness to illumination changes. The vector is normalized, and a threshold is used which creates a ceiling for the maximum value, removing the influence of large gradient magnitude values, and then normalizing again to unit length.

2.5.6 Feature Matching

Results of SIFT feature matching can be seen in Figure 8. The objects have previously had feature extraction and description per-

formed, and stored into a feature database. Test image analysis scans the feature database for best matches, and uses a voting system to decide the correct object and its most likely pose.

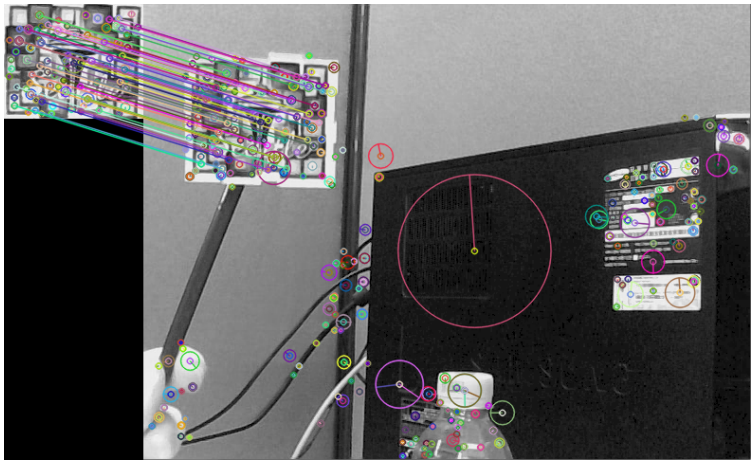

Figure 8: SIFT Matching Results

Using these matching features, their geometry can be compared to determine a mapping between the reference object and its location in a test frame. Seen in the figure above, a large number of strong, positive matches have been found, and by using them the object is localized as shown in shown in Figure 9. The details of this will be discussed in 2.6.3 using the robust outlier estimator, RANSAC.

SIFT matching is also robust to partial occlusion. Shown in Figure 10, only a small subset of the original features of the objects are visible, as they occluded by other objects in the scene. As well, even under rotation and heavy occlusion, correct detections can be made.

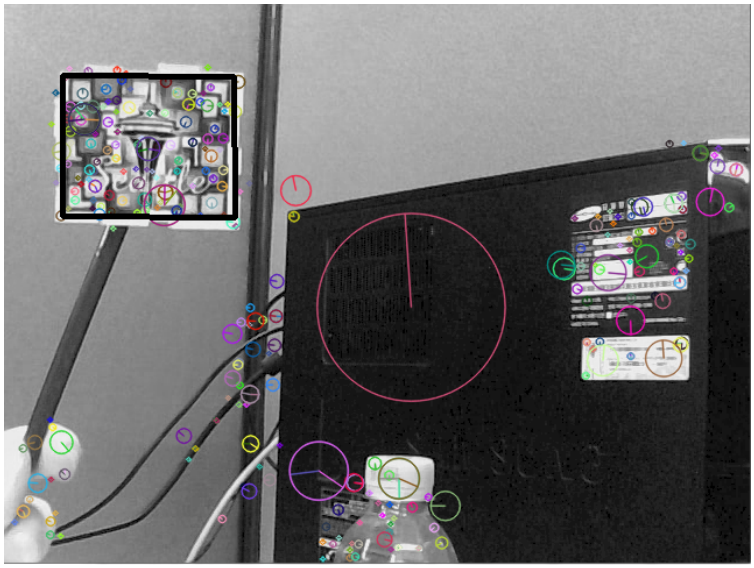

Figure 9: Object Localization in Test Image

2.6 Verification and Localization

Interest point detectors and descriptors give us a method for matching features between sets of images. It may be unclear, however, how exactly individual matches contribute to the location of an object. Moreover, a group of matches should confer more information about the object's location than a single match. The goal is to be able to accurately localize the reference image in the test frame; finding a transformation matrix such as the *fundamental matrix* or a *homography* are ways to do.

The goal is to solve for a transformation which exactly localizes the object in the test frame. Various methods exist such as geometrical analysis, least squares fits, and voting methods. The methods vary from their stability and complexity.

Undoubtedly, among a set of matches there will be incorrect

Figure 10: Object Localization Under Partial Occlusion

matches. A method must be robust, at least to some degree, to poorly matched features, henceforth referred to as *outliers*.

2.6.1 Fundamental Matrix

Robust methods for outlier detection have been developed, and one application of them is for detection of transformation matrices that exist between images. Torr[20] makes the case for finding this transformation in the feature matching case.

To state formally, there exists a set of linear equations describing the relationship between two different views of the same world space. Given a set of feature points, $\vec{P}$, in the first view, we can transform them with an arbitrary rotation about the origin, followed by an arbitrary translation. This transformed set of points, $\vec{P'}$, in the second

image is related by a `3x3` matrix, $\mathbf{F}$, which satisfies

$$(x'_i, y'_i, 1)^T \left[\mathbf{F}\right] (x_i, y_i, 1) = 0 \tag{4}$$

where $\vec{p_i} = (x\_i, y\_i, 1)$ are the homogeneous coordinates of the first frame, and $\vec{p'_i} = (x\_i, y\_i, 1)$ are the inhomogeneous coordinates located in the transformed frame. Using the constraint

$$|[\mathbf{F}]| = 0 \tag{5}$$

each of the pairs of points satisfies the equality

$$f_1 x'_i x_i + f_2 x'_i y_i + f_3 x'_i + f_4 y'_i x_i + f_5 y'_i y_i + f_6 y'_i + f_7 x_i + f_8 y_i + f_9 = 0 \tag{6}$$

where the fundamental matrix, $\mathbf{F}$ is

$$[\mathbf{F}] = \begin{bmatrix} f_1 & f_2 & f_3 \\ f_4 & f_5 & f_6 \\ f_7 & f_8 & f_9 \end{bmatrix} \tag{7}$$

Since this fundamental matrix exists which describes the transformation of one set of points to another, then a method for solving for or estimating the fundamental matrix will solve the problem of object localization from a setup of matches.

2.6.2 Homography Matrix

The homography matrix is a simplification of the fundamental matrix, having only eight degrees of freedom; it is a linear transformation

that relates two views of a planar scene under an ideal pin-hole camera assumption[17].

To keep the discussion simple, we will use a robust outlier estimator to generate a model for our data. Our data, remember, is a set of matches or correspondences between features of two separate frames. The model, in this case, is the homography matrix which gives the transformation of the object. This transformation not only gives us an exact mapping of the geometry of the object between scenes, but also key information pertaining to the number of **strong** correspondences, or *inliers*, and the number of **poor** matches, or *outliers*.

2.6.3 RANSAC

RANdom SAmple Consensus (RANSAC)[4] is a method for fitting a model to experimental data. In particular, the authors show its usage in the Location Determination Problem (LDP) associated with image analysis and computer vision for estimation of the fundamental matrix given a set of matched feature points.

Classical model fitting techniques attempt to fit a model to *all* the data; they operate under the assumption that the maximum deviation between datum will be smoothed by having a large enough data set. For many cases, this assumption does not hold, and the data set is said to contain uncompensated errors - outliers.

Common heuristics for handling gross error involved iterations of estimating the model, and eliminating the datum with the highest disagreement; termination was handled by a preset threshold or data

availability.

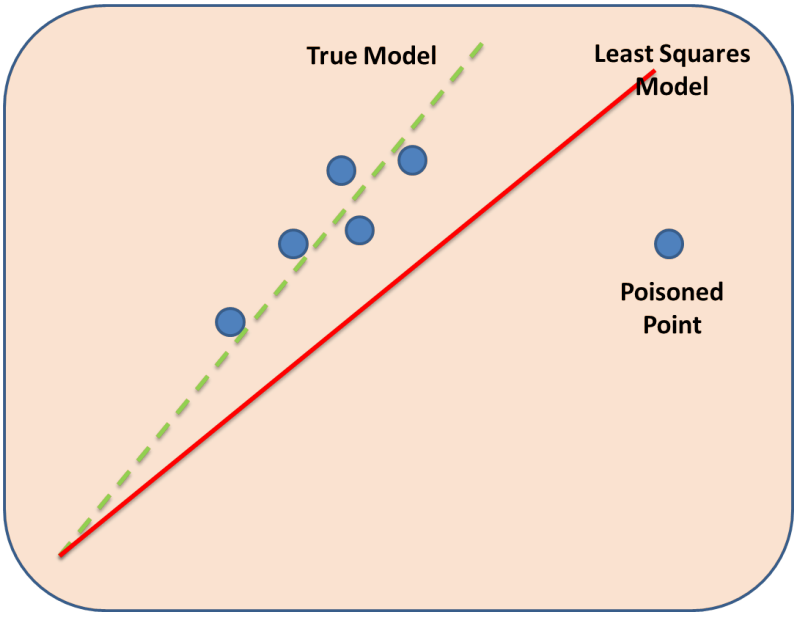

Figure 11: An Outlier Disrupting Less Robust Modeling Methods

It has been shown that even one outlier is enough to disrupt the above heuristics. As shown in Figure 11 from [4], even in a simple two dimensional case, the extreme outlier ("poisoned point") fools a highest disagreement elimination iterative least squares method.

RANSAC was created to be robust in the presence of outliers. Rather than attempting to operate on as much of the data as possible to find an initial solution, RANSAC begins with the smallest solution set possible for the task. From there, it establishes the agreement of the remaining data with the solution, and adjusts the model in this fashion until termination.

The basic, simplified form of the original RANSAC algorithm can be stated in psuedocode as follows. For detailed description of the

background and the complete RANSAC algorithm, see [4], [17].

```
struct RANSACOutput {
  // Homography Matrix: init -> nil
  Mat model;

  // Model Error w/Match Data: init -> inf
  float error;

  // Inlier Status (0/1)
  vector<unsigned char> is_inlier;
};

RANSACOutput RANSAC( matches, max_iters, thresh )
{
  RANSACOutput r;
  int n = 4;  // Four correspondences required

  while ( --max_iters >= 0 && matches.size() >= n ) {
    // Choose N random matches for model generation
    vector<int> test_model_indices = random_selection(n, matches.size());

    // Estimate a transform homography matrix
    Mat H_test = generate_homography( matches, test_model_indices );

    // Calculate error using all match data
    float test_error = estimate_error( matches, H_test );

    // Update if we have a better model
    if ( test_error < model_error ) {
      r.model = H_test;
      r.error = test_error;
      r.is_inlier = check_inliers( matches, H_test, test_error );
  }

  return routput;
}
```

2.6.4 Hough Transform

The Hough Transform [8] is a method for establishing the presence of a certain class of objects in an image using a voting procedure. Of the various methods and implementations, the Adaptive Hough Transform [10] uses a small accumulator and a coarse to fine search strategy to identify significant peaks in the Hough parameter spaces.

In general, while the simplest implementation of the Hough transform is conceptually simple, it also suffers from high computational complexity and memory requirements. For a thorough survey of the Hough transform, see [9].

Comparing with RANSAC, a plane detection scenario using both a 3D Hough Transform and a RANSAC implementation was done in [19]. It was the case here, and generally is, that RANSAC is faster to and preferred over the Hough Transform - and extensions to the traditional RANSAC algorithm are equally interesting and powerful (e.g. PROSAC[3]).

2.7 Motion Analysis

Temporal correspondences between two successive images in a video sequence can be used to simplify the computations for object tracking. Under the assumption that any two successive frames of a video sequence are minimally different, much of the computation for detection is redundant

2.7.1 Optical Flow

Similar in respect to the kinematics equations of basic physics, the motion of an object or its components - for our discussion let us say pixels - can be described by 2D motion with a delta time component. Using $I(x, y, t)$ as the intensity values of the image in the $\mathbb{R}^2$ coordinate space at time t, we can state the following equality between two different frames: [7], [1]

$$I(x, y, t) = I(x + \triangle x, y + \triangle y, t + \triangle t) \tag{8}$$

In a video sequence, the following *small motion* assumption can be made to simplify the task of solving for the flow. With this assumption and rewriting a using Taylor expansion we then have:

$$I(x+\triangle x, y+\triangle y, t+\triangle t) = I(x, y, t)+\frac{\partial I}{\partial x}\triangle x+\frac{\partial I}{\partial y}\triangle y+\frac{\partial I}{\partial t}\triangle t+... \tag{9}$$

By the *small motion* assumption it follows that:

$$I(x + \triangle x, y + \triangle y, t + \triangle t) - I(x, y, t) approx 0 \tag{10}$$

And, therefore:

$$\frac{\partial I}{\partial x}\frac{\triangle x}{\triangle t} + \frac{\partial I}{\partial y}\frac{\triangle y}{\triangle t} + \frac{\partial I}{\partial t}\frac{\triangle t}{\triangle t} = 0 \tag{11}$$

Where $\frac{\triangle x}{\triangle t}$ and $\frac{\triangle y}{\triangle t}$ are our optical flow motion vectors in the x and y directions. Using the derivatives of the image as required by the

equation above, we have to solve:

$$I_x V_x + I_y V_y = -I_t \tag{12}$$

which is an equation which two unknowns, and not solvable directly. Optical flow algorithms such as the Lucas-Kanade [15], [16] method.

2.7.2 Block Based Motion Vectors

Original popular as a method of *motion compensation* for image prediction in a video sequence as a way of reducing bit-rates for compression, block based matching motion algorithms use a search based method to find the location of a MxN sub-window (block) in the previous frame.

There are a plentiful number of variants on the basic algorithm to the search pattern or the metric used, but the basic idea is to use a coarse to fine sweep, and maintain some error metric for the difference between blocks. The goal is to attempt to find the best matching block between two frames. Figure 12 outlines an example of a typical coarse the fine scan for the block originally located at position 1 in the current frame, and the scan technique used to find the matching block in the previous frame. See [11] for an introduction to motion compensation and inter-frame image prediction.

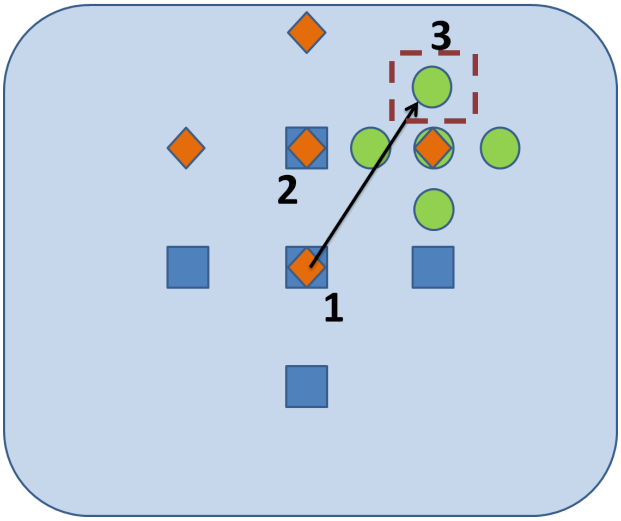

Figure 12: Coarse to Fine Block Based Motion Search.

There are a range of error metrics that can be used such as the SAD (*Sum of Absolute Differences*) or the CCF (*Cross Correlation Function*) to determine the similarity between blocks. For an image intensity function $I(b)$ where b is a mapping of coordinates that create a block, we want to find the minimum SAD score of all blocks in the window, w.

$$SAD_{score} = \arg\min_{b} \left(\sum_{b \in w} \left| I(b) - I'(b) \right| \right) \tag{13}$$

The problem with typical block based motion methods is the computation time. Matching every block in an image is a relatively expensive process that is made more expensive by smaller block sizes and more fine-grain searches. The trade-off being that for faster searches and larger block sizes the accuracy and granularity of motion is reduced.

Though optical flow approaches are more typically used for computer vision purposes, much of the rest of this document will use a less accurate, block based method to demonstrate the system is not dependent on a strong motion analysis algorithm.

3 Reduced Complexity SIFT Matching System

3.1 System Overview

A diagram of the system is shown in Figure 13; the system consists of a group of core modules with a number of optional modules that improve performance and detection accuracy.

The core modules consist of the following:

- SIFT Detection and Description
- Motion Analysis
- Adaptive Extraction Window
- Radial **OR** Descriptor-Less matching
- Localization (RANSAC)

The optional modules consist of:

- Growing Unit
- Pruning Unit
- Binary Descriptor Unit

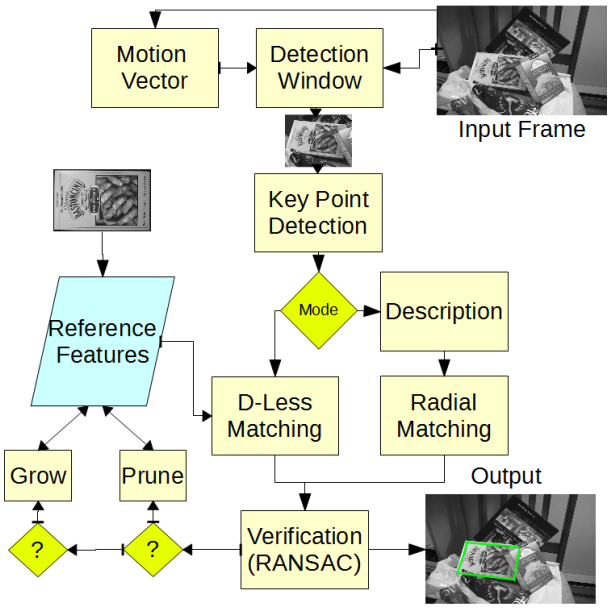

Figure 13: Reduced Complexity SIFT Tracking System Diagram

To keep in line with our focus on real-time execution and low complexity, the training database of SIFT features and descriptors for the object to be tracked are collected only from one image, and then used for tracking during a test sequence.

3.2 Motion Analysis Unit

The motion analysis unit handles requests for movement between the current frame and the previous frame. It provides motion vectors given an input point. Though it is generalized to support various flow systems, the block based motion vector used in H.264 has been chosen for demonstration.

The unit acquires the motion vector matrix for the current frame. This can be acquired live (i.e. directly from an encoder such as H.264)

or loaded from disk for previously encoded, test sequences.

```
MotionMap mmap;

if (live) {
  mmap = system.acquire_mmap();
} else {
  mmap = mmreader.next_map();
}
```

Once loaded, depending on the setup, the motion map is able to service calls to predict the new location of a point given the motion vector. For the block based method, the motion map is a down-scaled compared to the image frame size. By default the system uses a `4x4` block size for motion vectors. Therefore, a `640x480` image frame would correspond to a `160x120` motion map.

```
MotionVector MotionMap::get_mv(size_t const x, size_t const y) const
{
  MotionVector mv;
  // x motion
  mv[0] = mmap_.at(y / blocksize_y, x / blocksize_x)[0];
  // y motion
  mv[1] = mmap_.at(y / blocksize_y, x / blocksize_x)[1];

  return mv;
}
```

For the case of the backwards motion vector (the default), key-points located at $\vec{P}$ in the current frame, `N`, would be predicted to have been located at $\vec{P'}$ in the previous frame.

$$\vec{P'} = \vec{P} + \vec{m} \tag{14}$$

Though the system is not restricted to a particular form of motion analysis, motion vectors are used from block based matching in the

H.264 video encoder's motion estimation system. As a block based system, these motion vectors are notorious for high noise and only serve to approximate the locations of pixel sized keypoints. We will show using our reduced complexity system even with block based motion vectors, highly accurate results can be obtained in real-time.

3.3 Adaptive Extraction Window

In order to reduce detection of irrelevant keypoints the search region control unit reduces the amount of the frame to be searched by the keypoint detector. It uses previous object location information along with current motion vector information to determine the region to be searched.

This is a dynamic extension of previous work of [14] where they setup a static threshold search region around the previously localized object region. An area 10% larger than the object was chosen as the search region.

In this work, the search region is adaptively determined using motion information between consecutive frames. In general, when the region around the object experiences *small motion* with low valued motion vectors, there is no need to search a large region around the object. Correspondingly, when there is significant motion, especially in the case of block based motion vectors, the case for larger error is greater, and we extend the search region to account for this potential error.

Rather than establish some sophisticated metric, this adaptive

search region is created relative to the dominant motion vector present between frames. Histograms of motion vectors in the area containing and bordering the object are created, and the dominant X and Y directions are used to generate a sub-window extraction of the current frame for analysis. Figures 14 and 15 show the case for finding the dominant motion vectors in both a quiet (motion-less) frame, and a moving frame. The center of each histogram represents zero motion, with left and right directions indicating more motion towards the negative and positive axis directions. The dominant motion vector is composed on the X and Y MV histograms values with the most *votes*.

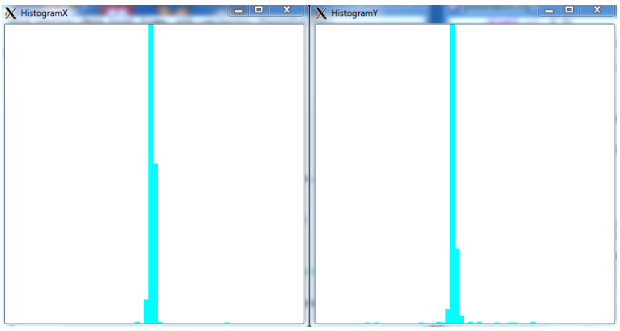

Figure 14: MV Histograms in a quiet frame.

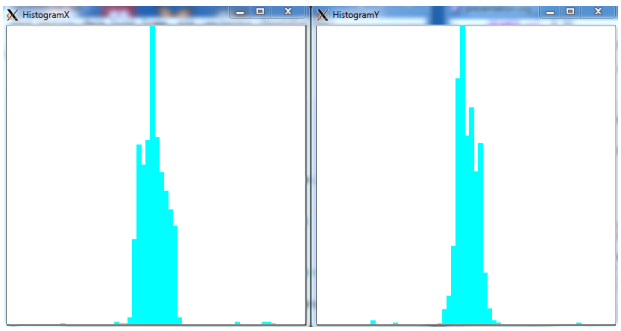

Figure 15: MV Histograms of an object moving in a frame.

A search region based on the dominant motion vector is extracted

and SIFT detection is performed on this sub-window alone. Effectively, irrelevant regions of the image are removed from analysis to reduce computational complexity. Figure 16 shows how this adaptive search region is used to forego analysis on irrelevant SIFT features in the frame.

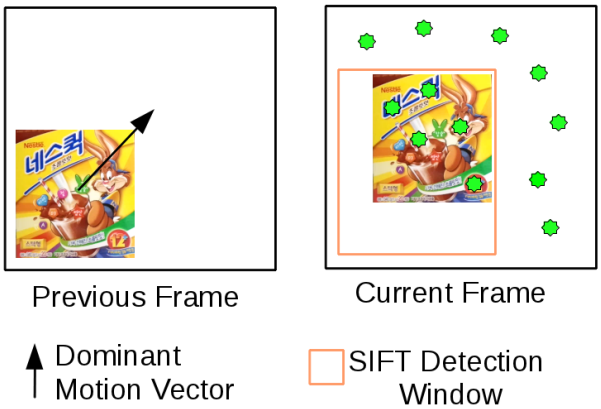

Figure 16: Adaptive Detection Sub-Window of the Frame.

Examples of the adaptive extraction window in the real test sequence, *flowers*, is show in Figure 17 below. The reduction to a small search region in the frame immediately discards uninteresting features.

Figure 17: AEW Extractions in the Flowers Sequence

3.4 Keypoint Detection

Keypoints are detected using a SIFT class in the image according to the search region. The Rob Hess open source SIFT implementation [6] inside of `OpenCV 2.4` is used for both detection and description. The parameters used for detection are described in the table.

Table 1: SIFT Parameters

Parameter	Value
Octave Layers	3
Contrast Threshold	0.04
Edge Threshold	10
Sigma	1.6

3.5 Keypoint Description

The standard SIFT description unit is used along with our optimizations to improve run-time efficiency. As discussed in the Radial Matching section, when candidates are enumerated for matching, the descriptors for a test feature will be computed only when reference features have *two or more* test features as candidates. When there is only one feature inside the matching radius, it is assumed to be a match, and its validity as an inlier or outlier is computed through RANSAC.

An optional module, a SIFT binary descriptor, is described in section 3.12, which alleviates problems due to the *curse of dimensionality.* In addition, the expensive Euclidean distance calculation can be reduced to simple bit manipulation instructions and bit-set counting. It

is closely based off the work done in [18].

3.6 Radial Matcher

The radial matcher performs feature based matching using a distance metric inside of a predefined search region. This search region is not to be confused with the search region control unit, which hints at the region of the image for SIFT keypoint extraction and description.

The radial matcher searches a neighborhood around the current location of a tracked, reference feature for matching features in the current test frame. It does this based on a maximal separation distance; for each reference feature, test image keypoints located inside of a reference feature's separation distance are marked as candidates for potential matches. After candidates are enumerated, those test features that are not single candidate correspondences to a test feature have their descriptor computed, and perform the distance metric calculation to determine the best match between these test features.

This reduces the number of required [and often] expensive distance calculations (e.g. for the Euclidean distance) in addition to only computing SIFT descriptors when absolutely necessary.

The separation distance, which describes the maximum distance between the reference point and the test point in pixels, is an adaptively determined system parameter. It is also based off of the dominant motion vector; lower values of the separation distance correspond to a reduced number of distance calculations and comparisons, but inherently require more precision from the motion analysis system than

would larger separation distance values.

Even with approximate motion analysis systems such as H.264's motion vector, a low threshold for separation was found to be suitable. Experimentation shows a separation distance as low as 4 pixels (one block size) could produce very highly accurate results at reduced computational rates. Therefore, the separation distance is determined adaptively using a limiting ratio and upper bound.

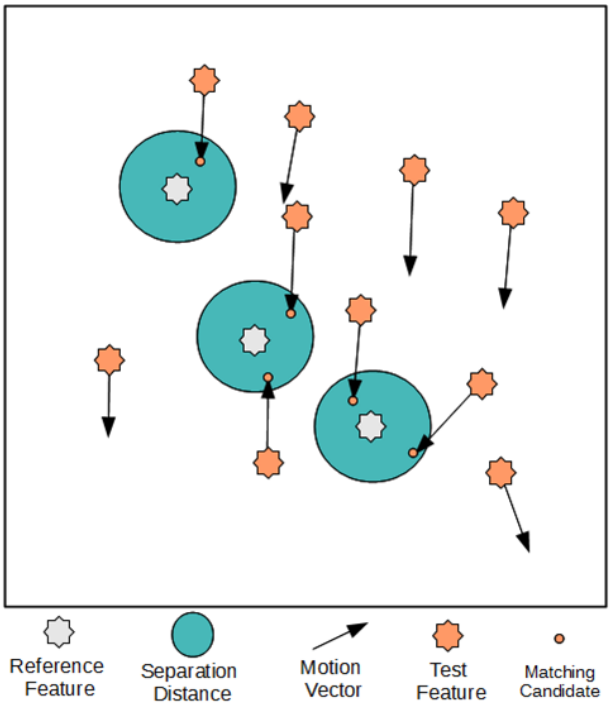

Figure 18: Radial Matching using a Backwards Motion Vector. Test features from the current frame are matched with reference features when their adjusted position lies within the separation distance of a reference feature.

Traditional SIFT matching operates using a ratio test between the two nearest neighbors. During matching, the two keypoints with the lowest distances are compared, and ratios above a threshold (com-

monly `0.6-0.8`) are rejected as poor matches. Though this is one method to reduce the likelihood of a poor match, it is really only effective for the case of many-to-one correspondences. Using temporal locality, the radial matcher system eliminates many erroneously matched features, and therefore does not employ this nearest neighbor ratio test. Requiring it would risk eliminating correct matches in the case of only one neighboring test point.

3.7 Localization

The localization of an object is performed after a set of matches has been found. As discussed earlier in 2.6, common methods for determining an object pose and location from feature points are RANSAC and a Hough Transform. In our system, RANSAC is used with a reprojection threshold of `0.1`. This tightens the constraints on what is considered an inlier versus an outlier. This variable also represents a potential for analysis and optimization, but is left as future work. See [4] for more information.

RANSAC is used in two ways for the object localization process. The first is to find the homography for identification of the bounding polygon of the object. By finding this polygon, simple, sanity tests can be performed to verify the likelihood of a correct match. For example, large area changes from the previous frame to the current indicate an increased probability of a false detection.

RANSAC is also used to find the homography between the previous frame and the current. This homography is used to update all refer-

ence points to their location in the current frame. Through matching, we expect to have found a subset of correct, matching reference points. But by using the homography generated by RANSAC, the transformation for all reference points, including unmatched or outliers, can be performed. This transformation also improves the location accuracy to sub-pixel precision.

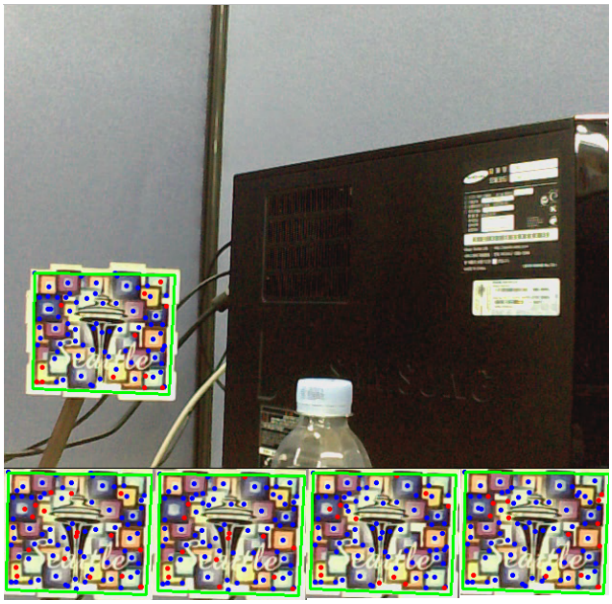

Figure 19: Object Detection Results with RANSAC.

The results of localization using the RANSAC algorithm are shown in Figure 19. The blue points correspond to inliers and the red points to outliers. The green bounding box is the result of the transformation from the original object coordinates to the new world coordinates in the test frame. Even with a large percentage of outliers the RANSAC algorithm is able to correctly determine the homography which localizes the reference object in the test frame with high accuracy and low error.

```
void homography_calculation( vector<Point2f> src_pts,
                             vector<Point2f> dst_pts )
{
  // Use RANSAC to find the model (H), and the error, inliers, etc.
  RANSACOutput r = RANSAC(matches, MAX_ITERS, RPROJ_THRESH );

  // Perform sanity checks, verify inlier % accuracy, ...
  // If pass, FOUND object and update status
  if (verify_result( r )) {
    auto new_roi = object_localization(r.H, object_roi_pts);
    update_object_roi( new_roi );
  }
}

vector<Point2f> object_localization( Mat H, vector<Point2f> roi_pts )
{
  // Perspective Transform the Bounding Region to locate object
  capp::PerspectiveTransform ptransformer(H);

  // Find the new location of the object roi points using
  // a transform
  vector<Point2f> new_roi = ptransformer(roi);

  return new_roi;
}
```

3.8 Failsafe Mode

Detection failures can still occur due to feature mismatch, object localization error, or disappearance of the object of interest from the scene, to name a few. There is a need for system recalibration in the event of a detection failure. To keep the complexity of the system low, but resilient to failure, an object localization error puts the system into a failsafe mode.

In failsafe mode, the system alternates between scanning the entire frame using a) the current reference set and b) the original reference

features. In this case, the Radial Matcher uses an infinite separation and becomes a Naive Matcher.

When a redetection is made from failsafe mode, the radial matching system is reinitialized with the detected feature set and updated feature point locations to continue tracking from the next frame.

3.9 Pruning

In order to improve matching speed and accuracy, a keypoint reduction step is introduced called pruning. Pruning eliminates consistently poorly matched features based on a lifetime parameter to the system. In each frame, a keypoint that is successfully matched and verified holds its lifetime counter's value, while keypoints that are not matched or do not pass inlier verification have their counters decremented. When a lifetime counter reaches zero, the pruning phase discards the feature from the reference set - eliminating it from future matching comparisons and computations.

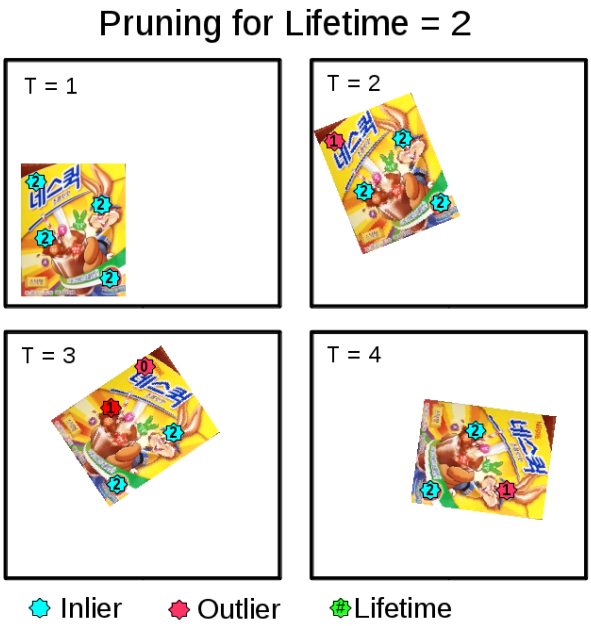

Figure 20: Visualizing the Pruning Process.

The pruning process can be visualized as shown in Figure 20. The consistently unmatched features or features identified as outliers for a period greater than the *lifetime* of the system are pruned. Also shown in Figure 21 is an example of features being pruned away in the *flowers* sequence; the two features on the left disappear after consecutive identifications as outliers equal to their lifetime of 2.

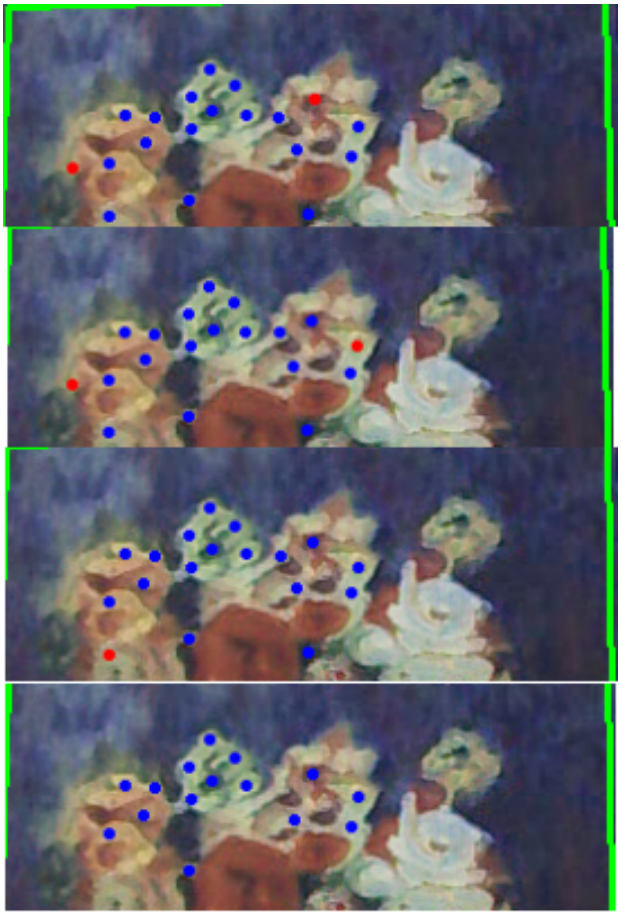

Figure 21: Pruning Flowers for a lifetime of 2.

The pruning algorithm can be described by the following pseudocode:

```
void pruning()
{
  auto rf     = ref_features.begin();
  auto rfend = ref_features.end();

  auto last =
    std::remove_if( rf, rfend, [](TrackedFeature& rf)
    {
      if (rf.counter <= 0)
        return true;
```

```
      return false;
    });

  // Removed features have been swap()'d towards the end and
  // last is the position of the last, valid feature.
  ref_features.resize( last - ref_features.begin() );
}
```

3.10 Growing

Though the objective of the system is to reduce complexity for real-time object tracking, a growing system is in place to counteract the pruning process. Ideally, the pruning process should eliminate keypoints which are not repeatably detectable or correctly matched, leaving only a subset of strong interest points that describe the tracked object. In experimentation, feature matching does not always yield highly repeatable matching of individual features.

Likewise, the appearance of an object in a video sequence may change over time; this is especially true when variations in lighting are introduced, along with various obstructions and rotations. Keypoints may disappear or appear as well with scale changes relative to the camera. For these reasons, the inclusion of new object features over time gives rise to more stable tracking.

The growing module acquires keypoints detected in the test image but not matched during the matching process. Test keypoints that were matched but rejected as poor matches (outliers) are not acquired during the growing phase. The idea is to acquire new, distinct feature points that have recently appeared in the current viewpoint of the object. But, this growing process is only performed when the trend

of the tracker's matching accuracy is **decreasing**. This allows us to perform maximal pruning operations and minimal growing operations. An implementation of the growing algorithm is shown:

```
void growing()
{
  auto tf    = test_features.begin();
  auto tfend = test_features.end();

  // Only Grow when history of accuracy is suffering
  if (!tracking_accuracy_diminishing())
    return;

  for( ; tf != tfend; ++tf ) {

    // feature status is maintained for the current
    // operational frame: INLIER, OUTLIER, UNMATCHED
    if (tf->status == UNMATCHED)
      prospective_features.push_back( *tf );
  }
}
```

Since an increased number of keypoints increases the complexity of matching, the growing module provides a lifetime initialization parameter which helps to quickly reject keypoints that have been grown but do not match well in subsequent frames.

For example, using an initialization of one, this enforces that a new feature be matched in the next frame where its lifetime counter will be reinitialized. The choice of the lifetime parameter represents another chance for significant optimization future work. Currently, through parametric analysis, the lifetime parameter was determined to be ideal in the 2 to 4 range.

3.11 Binary Descriptors

As an extension to the standard feature description process, the use of a binary descriptor for matching is implemented. The process is based on earlier work from [18]; the binary descriptor combats problems with high dimensionality vectors as well as providing a cheaper matching metric than the Euclidean distance. The binary descriptor module requires a one-time pre-processing stage to calculate the medians of the training set. As shown in [18], the variations of the medians with different partitions of training data do not vary significantly, where for 99% of features, maximally, three bits change due to different median threshold values. This allows us to generate the medians on the original training set only once, a small computational cost (relative to training set size) that runs in a few milliseconds on the test machine.

3.11.1 Median Calculations

From the training data, the estimated density function is:

$$f(v) = \frac{1}{n} \sum_{j=1}^{n} \delta(v - d_{j,i}) \tag{15}$$

where δ is the Dirac delta function. The medians are calculated in the previous work according to:

$$m_i = \inf\left[w : F(w) >= 0.5\right] \tag{16}$$

but, for the sake of simplicity in this paper - the medians are chosen

through a sort, and selection of the median value in each dimension.

3.11.2 Binary Description

Using the medians, each dimension of a feature vector is compared with the median corresponding to its dimension. Values greater than the median are converted to binary `1` and values less than or equal to the median are converted to binary `0`.

$$
\begin{aligned}
&Given: \\
&\quad \vec{D} = \{d_1, d_2, ..., d_n\} \\
&\quad \vec{M} = \{m_1, m_2, ..., m_n\} \\
&\quad \vec{B} = \{b_1, b_2, ..., b_n\}
\end{aligned}
$$

$$
b_j = \begin{cases} 0 & \text{if } b_j \leq m_j, \\ 1 & \text{else} \end{cases} \tag{17}
$$

An example of the binary description process is shown in Figure 22. For each dimension of the descriptor, the value is compared and replaced with the precalculated median value for that dimension as in equation (17).

Medians

5	5	5	5	5	5	5	...

SIFT Descriptors

2	5	7	9	5	14	30	...
8	17	55	128	58	22	17	...
24	7	6	11	5	6	14	...
11	32	20	83	44	2	0	...

Binary Descriptors

0	0	1	1	0	1	1	...
1	1	1	1	1	1	1	...
1	1	1	1	0	1	1	...
1	1	1	1	1	0	0	...

Figure 22: Binary Descriptor Conversion

3.11.3 Hamming Distance Metric

Two binary descriptors' distances are compared using the Hamming distance, which finds the number of differing bits between two descriptors. The best match between two binary descriptors is thus determined by the minimum number of differing bits.

$$\sum_i |b_{k,i} - b_{l,i}| \tag{18}$$

The Hamming distance can be computed efficiently using XOR instructions and a set-bit count instruction[4], or by using a look-up table. The binary descriptor is represented in a succinct representation requiring only as many bits as dimensions in the descriptor. Even for previous, optimized versions which use integer based representation of the SIFT descriptor, the `128` dimensional vector would require at least

[4]On Intel x86 with GNU gcc see: `__builtin_popcount(unsigned int x)`

128 bytes of storage. The binary version of the descriptor uses only 128 bits of storage (16 bytes). This results in an 8X memory savings.

The reduction in computational effort is also greatly reduced. As stated, the Hamming distance can be computed using relatively cheaper and fewer XOR instructions. For the 128 dimension SIFT descriptor on a 32-bit processor, 4 XOR instructions, 4 set-bit count instructions, and one addition instruction are needed versus the common Euclidean distance which uses 128 subtractions, 127 additions and 128 multiplications and a square root operation. The speed up results for matching on a common Intel series processor are shown during experimentation in 4.5.

3.11.4 Implementation

```
// Calculate medians in each dimension (column wise)
MedianVector calculate_medians(Mat descriptors)
{
  // Vector of median values for each dimension
  MedianVector mv;

  // For each column, sort column wise then grab the median
  // Simplification of the probability density.
  for ( auto col = descriptors.col_begin();
            col!= descriptors.col_end();
            ++col )
  {
    std::sort( col->begin(), col->end() );

    size_t len = col->length;
    size_t median_index;

    // If length is even, prefer the leftmost median
    if ( is_odd(len) )
      median_index = len / 2;
    else
```

```
      median_index = len / 2 - 1;

    // Push the median for dimension N into median vector
    mv.push_back( (*col)[median_index] );
  }

  return mv;
}

BinaryDescriptor binarize_descriptor(Descriptor a, MedianVector mv)
{
  // Initialize a binary descriptor to match length of 'a'.
  BinaryDescriptor bd;

  // Loop over every dimension, checking with the median value
  // for that dimension; one if greater than, else zero
  for ( auto val = a.begin(), m = mv.begin();
            val != a.end();
            ++val,++mv ) {

    if ( *val <= *m )
      bd.push_back( 0 );
    else
      bd.push_back( 1 );
  }

  return bd;
}
```

3.12 Descriptor-Less Matching

The radial matching system represents a significant reduction in both descriptor computations, and descriptor distance comparisons. This savings is possible in part to two major factors:

- Temporal [Motion] information between frames
- Strong Outlier Estimation from RANSAC

A further, extreme simplification of the radial matching system is to simply ignore similarity measures between descriptors entirely, and base all information off of repeatable features, temporal information, and model estimation through RANSAC. This has been dubbed the Descriptor-Less Matching system, or D-Less Matching.

In D-Less matching, a similar candidate feature process is performed, except rather than enumerating multiple candidates, and then comparing to find the **best** using a descriptor metric, the **best** candidate is simply considered to be the closest in 2D space.

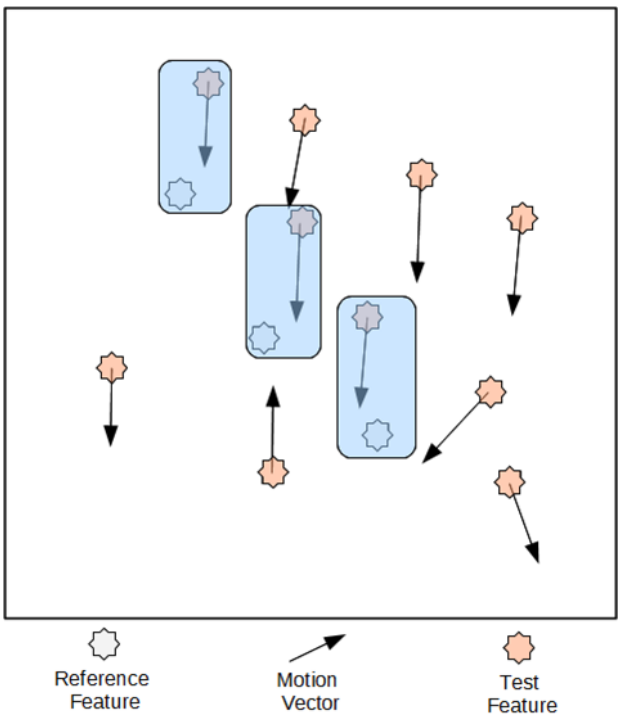

Figure 23: Choosing the Closest Feature in Descriptor-Less Matching

For example, as shown in Figure 23, there are several candidate features falling inside of the reference feature's separation distance -

but, for D-Less matching, the closest feature by 2D Euclidean distance is selected as a match - which will be verified during the RANSAC model and object localization phases.

3.12.1 Implications of D-Less Matching

The benefits of Descriptor-Less Matching in terms of computation time are unarguably strong. However, the success of D-Less matching is contingent upon several factors.

- Feature Repeatability
- Keypoint Density
- Motion Vector Accuracy
- Outlier Estimation

For success of feature matching with the D-Less system, there must be a high percentage of near-exact features present on the object between consecutive frames. This can be alleviated somewhat by using the growing and pruning systems along with updating all reference features' positions so that even features that are only repeatable with a certain frequency can be tracked.

The density of feature points on an object is of relevance as well. For a high density or concentration of features in a single region of an object, and using a block-based, inaccurate motion vector, there is the risk of many erroneous matches. An ideal feature density is hard to determine precisely, though it should be the case that, in the common

case, there are not more than two or three features within a separation distance of a reference feature.

Finally, accurate outlier estimation with RANSAC is vital to the generation of a strong model to represent the transformation of the object from reference feature space to the current feature space in the test image. It can be expected that error will accumulate over time, and as such, we must strive to minimize this error through accurate outlier estimation. Through RANSAC is strong to even high percentages of outliers, one inaccurate model prediction can send the tracker into a miss-detection, and cause for a reset.

It will be shown in the experimentation sections that follow that D-Less matching, for the system described thus far, is a strong alternative to traditional SIFT matching and tracking. It will also be shown that the speed-up between radial and d-less matching is also significantly better without loss of accuracy.

4 Experimentation

For a standard of comparison, test sequences in both VGA (640x480) and 720x480 formats have been used to a) compare the reduced complexity system to a traditional, SIFT matching and tracking setup b) establish strong system parameter values for optimal matching accuracy and speed.

To this end, the primary variables examined are error distance (ErrorD) from ground truth, and processing time per frame. The error distance from ground truth is the pixel distance between real and measured point values bounding the object - i.e. the points whose convex hull represents the object's boundary. Processing time is in seconds. All test sequences were performed on an Intel core 2 duo using a debug build (no optimizations).

4.1 Test Sequences

Table 2: Test Sequences Used for Evaluation

Sequence	Resolution	Notes
Seattle	640x480	H & V Motion, Blur
Nesquik	720x480	H Motion, Box, High Keypt. Density
Bottle	640x480	Reflective, Rotation, Motion, Blur
Box	640x480	Fixed Camera, Object Occlusion
Flowers	720x480	Multi Plane Rotation, Scale Changes
Firesign	720x480	Camera Motion, Out-of-Plane Rotation
Bathroom	720x480	Camera Motion, Rot, Lighting Changes

4.2 Naive Matcher

A naive matching system based off of Lowe [13] modified for tracking has been used as a baseline for testing accuracy and performance to the reduced complexity SIFT matching tracking system.

The naive matching system uses a static, reference feature database, and, for each frame, extracts and describes all features. It performs full matching between all reference features and the frame's current features. It uses the same robust localization methods (RANSAC) as the reduced complexity system. Refer to 2.5.6 for more information on the naive SIFT matching system.

The naive tracker experiences failures as seen in Figure 24 during difficult frames in a test sequence. Blurring, occlusion, lighting changes, and more can adversely affect the success of the tracker. In addition to the presence of detection failures, the naive tracker runs just under real-time operation with operational times highlighted in Table 3.

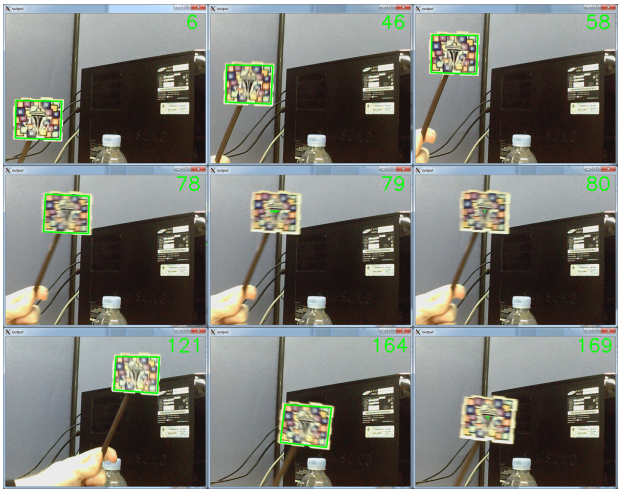

Figure 24: Naive Tracker for Sequence Seattle.

Table 3: Naive Tracker Processing Time Breakdown

	Detection (ms)	Description (ms)	Matching (ms)	Homography	FPS	1-Prec (%)
Bottle	34.64	50.92	4.44	1.99	9.09	0.65
Nesquik	39.99	54.84	1.88	1.96	7.50	0.62
Firesign	36.89	39.91	4.20	1.55	10.36	0.60
Flowers	36.66	57.68	15.82	9.31	8.37	0.57
Beatles	40.25	54.70	18.77	19.64	7.50	0.62

4.3 Low Complexity SIFT Tracker

4.3.1 Graphical Results

Bounding boxes for object detection are shown for several test sequences in Figures 25 and 26. As will be shown in the following sections, both the radial matcher and the d-less matcher give similar, excellent results in terms of accuracy.

Of particular note, recall from Figure 24, that the naive tracker

failed to detect the object due to blurring. This is not the case for the proposed system, as both exploitation of temporal locality along with growing and pruning reduce failure detections to nearly zero.

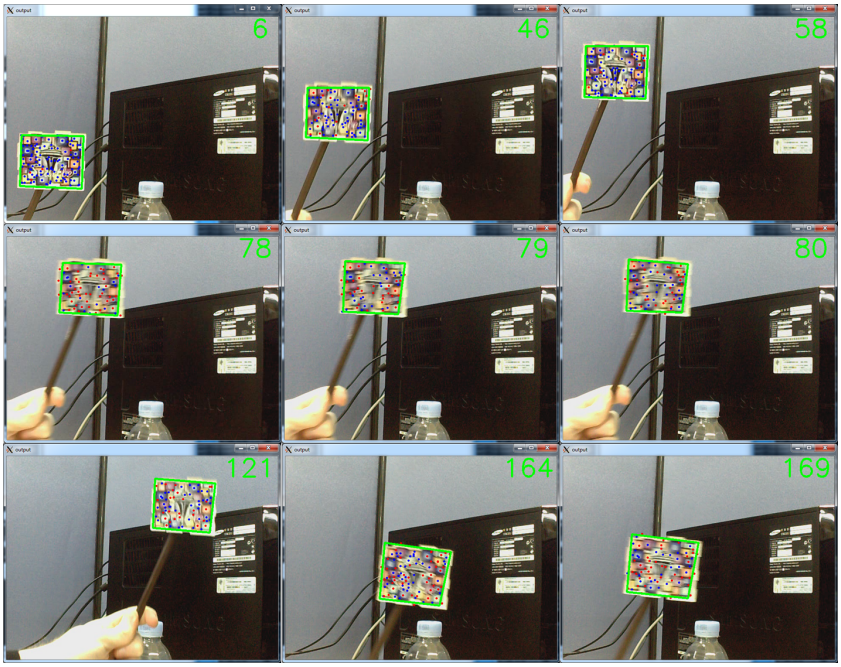

Figure 25: Low Complexity Tracker Seattle Sequence Results

In the following sections, the various modules of the Low Complexity SIFT Tracker are discussed with their contributions analyzed in terms of performance.

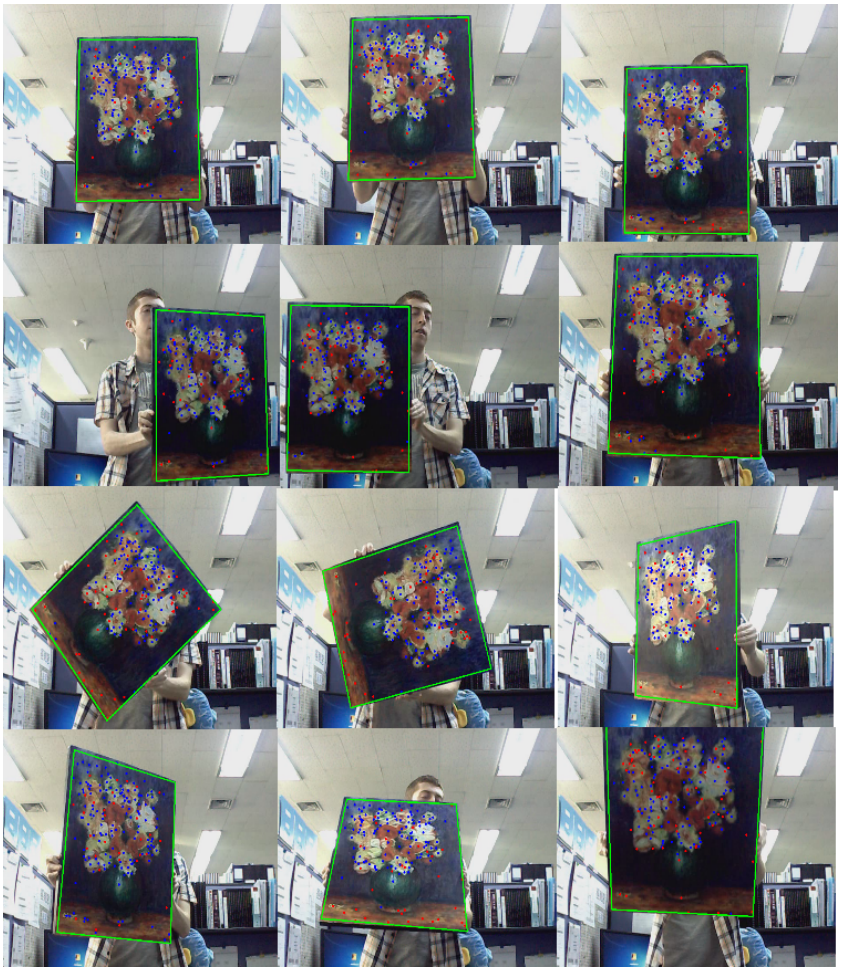

Figure 26: Flowers Test Sequence Results

4.4 Adaptive Extraction Window

The performance of the adaptive extraction window described in 3.3 is compared in Table 4 and Figure 27. Its largest contributions are to *description* and *matching* phases. For this test, the radial matching system is used without growing, pruning or other special optimizations.

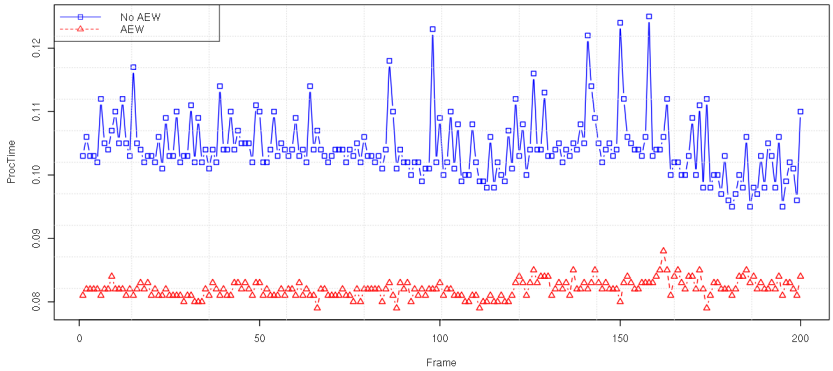

Figure 27: Adaptive Extraction Processing Time Comparison

Table 4: Run-time and Accuracy Comparison for AEW

		Description (ms)	Matching (ms)	FPS	1-Prec (%)
Bottle	No AEW	52.93	1.72	10.76	0.36
	AEW	27.03	0.45	15.21	0.36
Firesign	No AEW	40.33	1.72	11.38	0.39
	AEW	34.84	1.20	12.32	0.39
Flowers	No AEW	59.85	59.67	9.51	0.32
	AEW	42.83	35.66	11.68	0.32
Beatles	No AEW	52.93	1.72	10.76	0.36
	AEW	27.03	0.45	15.21	0.36

4.5 Binary Descriptors

Similar to the work in [18], and as discussed previously in 3.12, binary descriptors were proposed to improve the performance of descriptor matching. By modifying the original SIFT code concerning descriptors, and paying a one-time penalty (on the order of milliseconds) to calculate the medians, binary descriptors greatly improve the matching time.

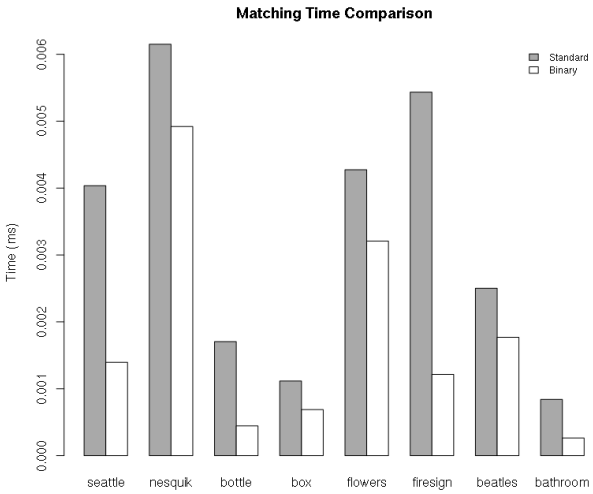

Figure 28: Improved Matching Time with Binary Descriptors

Figure 28 shows the results for the binarized descriptors *matching time* phase versus the standard SIFT descriptors. Significant matching time improvements were realized, without loss of accuracy, as observed by the outlier estimation via RANSAC, and reported in Figure 29, showing the inlier percentage and the ground truth error distance.

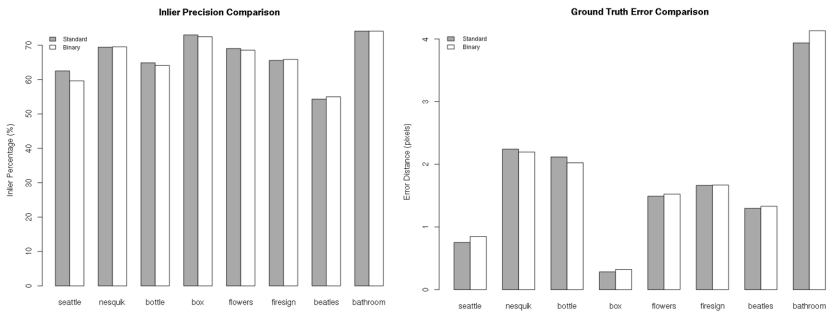

Figure 29: Accuracy w/Binary Descriptors

4.6 Radial Matching

The Radial Matcher of 3.7 is proposed as a drop-in replacement for the naive SIFT matcher. It uses the dominant motion vector to adaptively determine a maximum separation distance for candidate, test features.

The radial matcher exhibits performance improvements in both matching time and homography estimation time (RANSAC). While the former is immediately obvious, the latter may not be. As a result of radial matching, the percentage of outliers from matching is reduced - directly improving RANSAC's ability to quickly converge to a correct model.

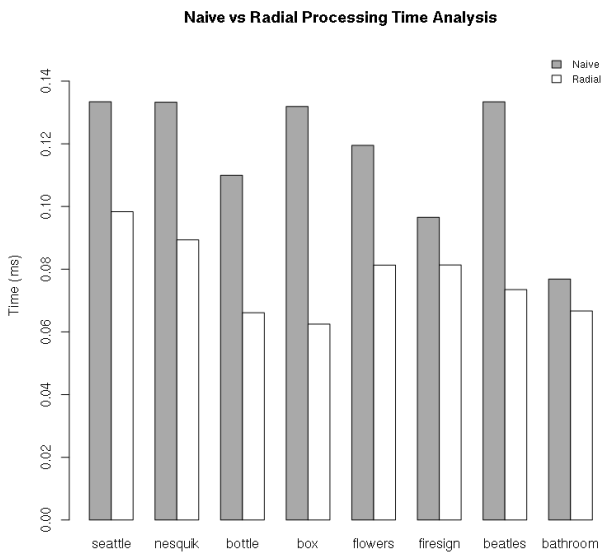

Figure 30: Radial vs Naive Matching Processing Time

4.7 Descriptor-Less Matching

Result of Descriptor-Less Matching are examined in Figure 31 and 32 below. The analysis is performed against the naive tracker and the radial matcher. The processing time improvements over the naive tracker are dramatic, and most of the time improvement over the radial matcher is a result of not performing the *description* phase.

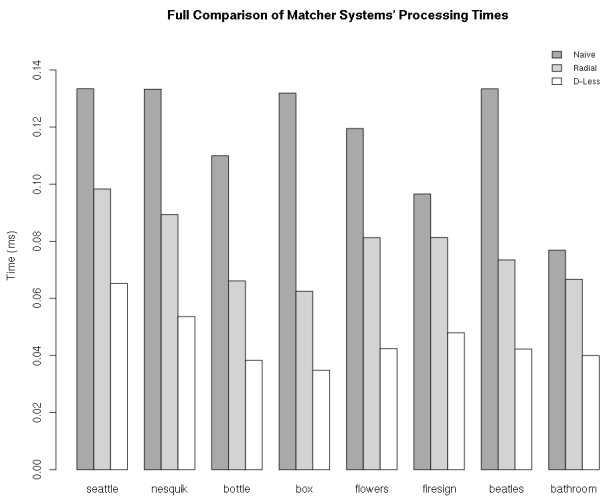

Figure 31: D-Less, Radial, and Naive Processing Time Analysis

The inlier percentage, shown in Figure 32, shows the increase in matching accuracy experienced over the naive tracker. Correspondingly, both the radial and d-less matcher exhibit very similar inlier ratios. The takeaway point is that without considerably loss of accuracy, we can use temporal locality and motion information in place of descriptor calculations and comparisons to achieve the same effect (object tracking).

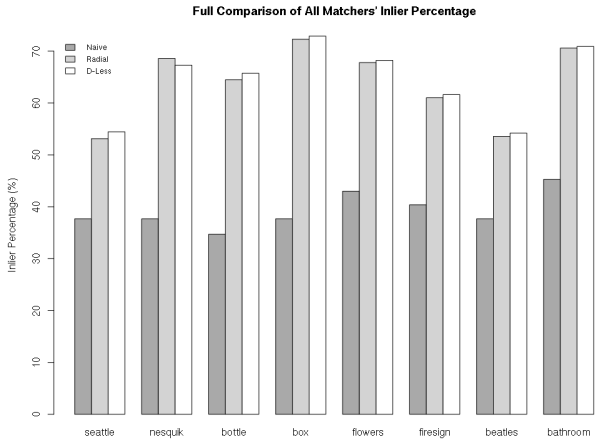

Figure 32: D-Less, Radial, and Naive Inlier Percentage Analysis

4.8 Growing & Pruning

The growing and pruning facilities were evaluated together, and compared to their corresponding GP-less counterparts in Figure 33 and 34. In most cases, we observe a significant improvement in the inlier ratio seen during matching - especially for the video sequences with significant dynamic change to the appearance of the object being tracked.

Also, as expected, growing and pruning does not have significant benefit in terms of accuracy for the *box* sequence, as the object and camera are fixed and only an intercepting, object partially occludes the box during tracking.

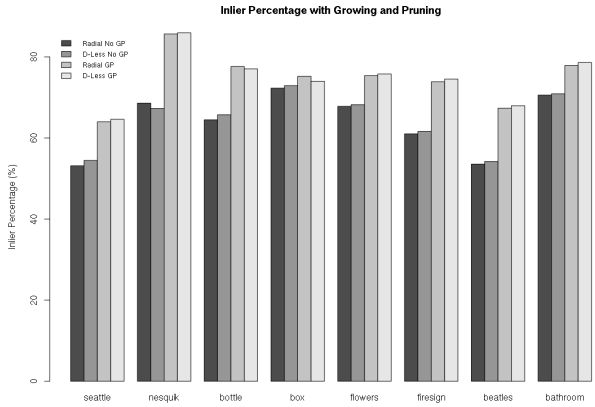

Figure 33: Growing and Pruning Effects on Inlier Ratio.

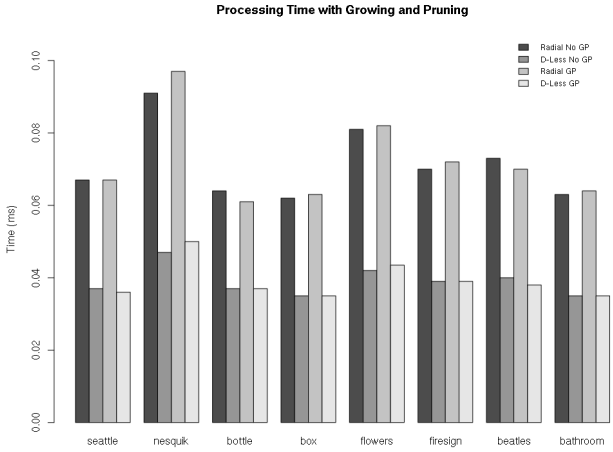

Figure 34: Growing and Pruning Effects on Processing Time.

The processing time changes seen in Figure 34 generally show that there are little-to-no adverse effects from the use of growing and pruning. In the general case there is either no improvement or minute changes in either direction for processing time.

4.9 Final System and Results

The preceding sections discussed the improvements of the proposed modules independent of each others' contribution. Here, we combine and present the results for the complete low complexity SIFT tracking system.

Suggested Operation:

- Adaptive Extraction Window
- D-Less Matching
- Growing & Pruning

Though the radial matcher and the reduced descriptor computation provides considerable improvement over the naive tracker, the d-less tracker without any descriptor computations and using purely motion information with temporal locality has been shown to achieve equally accurate results at a significantly higher frame rate.

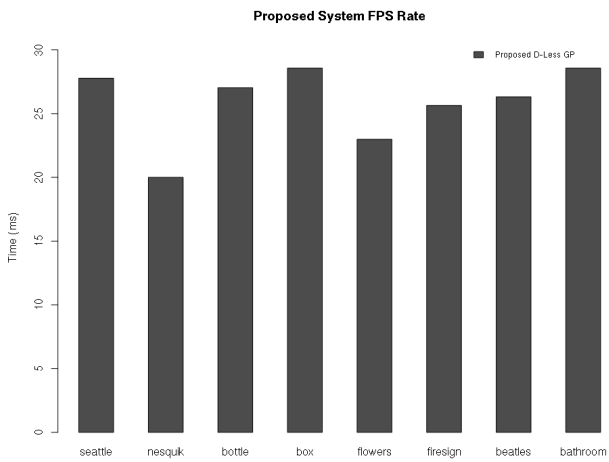

Figure 35: Proposed System FPS Results on Test Sequences

The final design achieves real-time results, on average 25-26 FPS, which represents a significant improvement over the naive SIFT tracker. It builds upon motion tracking by utilizing repeatable, local features, like SIFT with aggressive optimizations to exploit temporal locality.

5 Conclusion

An adaptive, radial matching system used for SIFT tracking is able to achieve near real-time performance at 15 frames per second. In addition to speed-up improvements, accuracy is increased by filtering the potential matches and eliminating poor candidates; this is reflected in the improvement of the 1-precision and inlier percentages in the Experimentation section.

Further, we have demonstrated that a descriptor-less matching forgoes SIFT descriptor computations, which results in a direct time savings without loss of matching accuracy. Motion vectors used can be as simple as block-based (e.g. H.264), or even as complex as optical flow (e.g. LK Tracker). The choice of motion vector is relevant, but flexible in that block-based or flow both demonstrate excellent results.

Optical flow was tested to run on VGA and 720x480 test sequences on average in 3-4 ms per frame, and does not contribute significantly to the overall processing time. The descriptor-less SIFT tracking system achieves real-time performance, running at average speeds over 25 FPS.

The growing and pruning modules have been shown to account for dynamics in scene changes and local features on the object as its scale and pose change through a test sequence. It does this at approximately the same processing rate. The method by which growing and pruning works is a very simplified procedure, and is a subject for future research. More complex interactions to minimize grow operations while maximizing prune operations without accuracy loss are

potentially time-saving optimizations.

The complete, proposed system brings reliable SIFT tracking via reduced descriptor extraction and matching computations to a near real-time domain using the radial matcher (>15 FPS), and d-less tracking at over 25 FPS. The overhead for the required motion vectors is minimal (<4 ms), dependent on the choice of motion vectors; motion vectors can be precalculated for added benefit, or acquired through various video encoders such as the H.264 encoder. This work acquires them directly from a hardware encoder by CAPP Lab[5]. Correspondingly, the simplicity of the system when compared to other, state-of-the-art machine learning techniques is great, and its application for direct implementation in hardware promises for an economic solution to a hardware-based, generic object tracking system.

[5]http://capp.snu.ac.kr

Bibliography

[1] S. Baker and I. Matthews. "Lucas-kanade 20 years on: A unifying framework". In: *International Journal of Computer Vision* 56.3 (2004), pp. 221–255.

[2] I. Biederman. "Recognition-by-components: a theory of human image understanding." In: *Psychological review* 94.2 (1987), p. 115.

[3] O. Chum and J. Matas. "Matching with PROSAC-progressive sample consensus". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on.* Vol. 1. Ieee. 2005, pp. 220–226.

[4] Martin A. Fischler and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: http://doi.acm.org/10.1145/358669.358692.

[5] C. Harris and M. Stephens. "A combined corner and edge detector". In: *Alvey vision conference.* Vol. 15. Manchester, UK. 1988, p. 50.

[6] Rob Hess. "An open-source SIFTLibrary". In: *Proceedings of the international conference on Multimedia.* MM '10. Firenze, Italy: ACM, 2010, pp. 1493–1496. ISBN: 978-1-60558-933-6. DOI: 10.

1145/1873951.1874256. URL: http://doi.acm.org/10.1145/1873951.1874256.

[7] B.K.P. Horn and B.G. Schunck. "Determining optical flow". In: *Artificial intelligence* 17.1-3 (1981), pp. 185–203.

[8] P.V.C. Hough. *Method and means for recognizing complex patterns*. Tech. rep. 1962.

[9] J. Illingworth and J. Kittler. "A survey of the Hough transform". In: *Computer vision, graphics, and image processing* 44.1 (1988), pp. 87–116.

[10] J. Illingworth and J. Kittler. "The adaptive Hough transform". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 5 (1987), pp. 690–698.

[11] S. Kappagantula and K. Rao. "Motion compensated interframe image prediction". In: *Communications, IEEE Transactions on* 33.9 (1985), pp. 1011–1015.

[12] D.G. Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[13] D.G. Lowe. "Object recognition from local scale-invariant features". In: *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*. Vol. 2. Ieee. 1999, pp. 1150–1157.

[14] Y. Lu et al. "Low Complexity Homography Matrix Based SIFT for Real-Time 2D Rigid Object Tracking". In: *Wireless Commu-*

nications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on. IEEE. 2010, pp. 1–4.

[15] B.D. Lucas and T. Kanade. "An iterative image registration technique with an application to stereo vision". In: 1981.

[16] B.D. Lucas and T. Kanade. "Optical navigation by the method of differences". In: *International Joint Conference on Artificial Intelligence.* Vol. 2. Citeseer. 1985, pp. 981–984.

[17] "RANSAC for Dummies". In: 2009.

[18] M. Stommel et al. "A fast, robust and low bit-rate representation for SIFT and SURF features". In: *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on.* 2011, pp. 278 –283. DOI: 10.1109/SSRR.2011.6106749.

[19] F. Tarsha-Kurdi, T. Landes, P. Grussenmeyer, et al. "Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data". In: (2007).

[20] P.H.S. Torr and D.W. Murray. "Outlier detection and motion segmentation". In: *Sensor Fusion VI* 2059 (1993), pp. 432–443.

[21] A. Witkin. "Scale-space filtering: A new approach to multi-scale description". In: *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP'84.* Vol. 9. IEEE. 1984, pp. 150–153.

A Source Notations

Throughout, a C++ style pseudocode is used to describe algorithms and system operation. While most of the syntax should be understandable to readers with any programming background, there are some C++0x (C++11) specific operators and idioms used which are briefly explained here. For more information see the freely available, latest C++ standard working draft, N3376.

`auto` Keyword

Readers coming from a C background may be confused at the use of `auto` without an identifier following. `auto` is C++11's reuse of the keyword to tell the compiler to automatically determine the type of the variable being declared. This only works if type deduction is possible.

```
int  x = 1;
auto y = 2;
auto z = &z;
```

is equivalent to

```
int  x = 1;
int  y = 2;
int* z = &z;
```

Lambda Expressions

Lambda expressions are inline, anonymous functions similar to what's found in Common Lisp. They are primarily used in place of function objects (functors) that are used only once whose overhead

pollutes the code space. Their most basic usage can be represented in an example.

```
struct Match {
  Feature ref;
  Feature test;
  float distance;
};

struct MatchDistComp {
  float threshold;

  MatchDistComp(float Threshold) : threshold(Threshold)
  {  }

  bool operator() (Match const& m)
  {  return  m.distance > threshold; }
};

// ...
float threshold = 150.0;
auto last = std::remove_if(matches.begin(), matches.end(),
                           MatchDistComp(threshold));

matches.resize( last - matches.begin() );
```

As opposed to the lambda version:

```
float threshold = 150.0;

auto last = std::remove_if(matches.begin(), matches.end(),
                           [&threshold](Match const& m)
                           {
                             return m.distance > threshold;
                           });
```

Which eliminates the need for MatchDistComp from the namespace. As well, lambda expressions can also capture variables, which bring them into scope inside the expression. This can make the initialization of traditional functors with required data much less painful, as can be the case when several constants or variables are involved.

STL Iterators

The C++ Standard Template Library sports a powerful technique for creating general-purpose algorithms that operate on a range of different containers and data types. While too complex to explain in detail in this appendix, the premise is that a container (such as a **vector**) provides certain guarantees on how its data is accessed. The job of an *iterator* is to point to a location inside a *container*, and provide a means of traversal through the container (e.g. forward, backward, random-access). An iterator can be thought of as a sophisticated pointer - having all of its powers with some of C++'s type-safety. Not all containers support all directions and iterator types, but for the purpose of this work, this is not an issue. *The C++ Standard Template Library: A Tutorial and Reference* is an excellent resource for the C++ STL.

In general, our use of iterators is for traversing a data container, such as:

```
std::vector<Point2f> points;
// ... Fill in points

// Iterators, itb and ite, point to the beginning and end of our
// container.
auto it  = points.begin();
auto ite = points.end();
for ( ; it != ite; ++it ) {
  // ... Process Point
  it->x += 1;
  it->y -= 2;
}
```

Similarly, without knowing [much] about the underlying datatype

and specifically handcrafting a function, we can operate on containers of our datatypes for various algorithms such as sorting or conditional removal.

```
std::vector<KeyPoint> keypoints;
// ...
// Possible if KeyPoint has an operator<() member function
std::sort( keypoints.begin(), keypoints.end() );
```

With minimal modifications and lines of code, we have available a library of general-purpose algorithms at our disposal.

Abstract

컴퓨터 비전분야에서는 지역 불변 특징(local invariant features)들을 이용하여 물 체를 인식 하는 연구가 다방면으로 진행되고 있다.

인식을 위한 크기, 회전과 관점에 대하여 불변하는 특징을 얻기 위해서는 가우시안 컨볼루션(Gaussian convolution)이나 이것을 근사화 한 방법을 이용하는데, 이는 수 많은 크기에 대한 연산이 필요하다. 게다가 강하고 뚜렷한 특징들을 표현하는 것과 매칭 과정 또한 연산 량을 증가시킨다.

SIFT는 강력하고 지역 불변한 특징들을 이용하여 스테레오 비전과 물체 인식에 사용 되는 영상 매칭에 많이 쓰이는 방법 중 하나 이다. SIFT 또한 "curse of dimensionality"에 의해 차원이 증가함에 따라 연산 량이 지수적으로 증가하는 단점이 존재한다.

따라서 SIFT와 같이 크기와 회전에 불변하는 특징들을 구하는데에는 많은 연산이 필요하며 이 많은 연산 량은 실시간 시스템 구현시 간과할 수 없는 부분이다. 이 논문은 매칭의 정확성을 향상 시키면서도 빠르게 수행하는 방법들을 제시한다.

물체 추적에서 SIFT와 같은 지역 불변 특징들의 장점을 이용하기 위하여, 이 논문에 서는 잠정 구역(temporal locality)성과 연속하는 프레임간 일정한 매칭을 위해 radial metric을 이용한 감소된 연산량 추적 시스템(Reduced Complexity Tracking System)을 소개한다. 게다 가 descriptor 변형과정(descriptor modification)이 수행되어 다차원수의 오류(high dimensionality errors)를 감소시키고, 정확성과 적은 연산 량을 위하여 틀리게 추적된 특징들은 필요에 따라 제거되어 새로운 특징들로 대체된다. 이 시스템은 기존 SIFT를 뛰어넘어 비해 더 빠른 수행시간과 낮은 오차율을 갖는다.

이 Radial Matching의 개념을 확장하여 descriptor없이 연속된 프레임간의 움직임 벡터(motion vector)를 이용하여 참고 특징(reference feature)과 기하학적으로 가장 가까운 특징 후보(candidate feature)를 찾는 Descriptor-Less 매칭 알고리즘을 제안한다 . Descriptor-Less 매칭 알고리즘은 SIFT 추적에서 가장 연산이 많은 부분인 descriptor 추출을 하지 않으면서도 매칭 정확성을 저하시키지 않으며, 기존의 SIFT 에 비해 엄청난 매칭 속도 향상을 가져온다.

제안된 Radial Matching 시스템은 로 엔드(low-end)급 Intel Dual Core 환경에서 14-15 FPS로 동작하며, Descriptor-less SIFT 추적 알고리즘(D-Less SIFT tracking) 은 같은 환경에서 평균 25 FPS로 동작한다.

Keywords : SIFT, SURF, Object Tracking, Computer Vision

Student# : 2010-24082

Acknowledgments

As a beginner into the realm of computer vision, much of my self-study and exploration was met with exceeding difficulties in understanding the current state-of-the-art literature. A field such as this requires an intricate understanding of several branches of mathematics, algorithms, computing, and even psychology and neural sciences. There were many who helped along the way, and it is impossible to mention everyone and their contribution, but know that you are in my thoughts and have my gratitude.

I must start by thanking both Samsung Electronics and Seoul National University, who awarded me with the Samsung Global Scholarship Program and this opportunity for master's study. Under their excellent care I have been able to work without distraction and comfortably.

I also would like to thank the professors and faculty at SNU. They have made many allowances to allow a foreign student to be able to comfortably study and work in a positive environment.

I must mention that the students in CAPP Lab are among the finest specimen of young men and women that I have ever had the pleasure of working with, and you will be sorely missed as I move onto the workplace. I will also cherish the friendships, and remember the excellent times we've shared. I cannot mention everyone, but I must mention those who have tirelessly assisted me in my efforts towards graduation.

To ChulHee, DongHyeon, and SunWoong - you guys helped me

through my master's defense presentation during a time when my thoughts were somewhere lost in Andromeda. Thank you so much for taking care of the minor things, and helping me make it to the finish. And thank you to DongHyeon for taking my garbled mess of an abstract and making sense of it the day before submission.

To KyungHun - Thank you for keeping me in the loop on both lab affairs as well as interesting things happening outside of it. Our chats about the latest new gadgets and movies were just the periods of relaxation I needed to keep from blasting my computer and all the research with it to smithereens. Our conference trip with HanSoo to "Glorious, JeJu Island" will always be one of my fondest memories.

To Mr. Jang - Thank you for your inspiring words, and helping me understand how to focus my research. Without your guidance, I doubt I would have been able to concentrate on a specific topic, and find a way to make a contribution. I am also grateful for the culture and language lessons by your hand. I wish you, your wife, and your two, new, beautiful babies all the best.

To KyuDong - Thank you for your tough love, and helping me understand those difficult concepts of both a personal and professional nature. Thank you especially for the recent assistance, analysis, and thorough explanation (that I could understand) of the professional world and my interactions with it. Your personality and leadership in our lab make you an invaluable asset, as I know everyone appreciates and respects your opinion.

To HanSoo - Thank you for your patience with me during all the

Skype projects and demos. I know I wasn't always available when you needed me, but thankfully we were able to put together a working demo when required. The wisdom you shared over these two years is much appreciated; and the fun times in JeJu and the exciting car chases are memories that will last me a lifetime.

To SunWoong - My **Graduation Manager** - I cannot say thank you enough for all the things you have helped me with over these past two years. I must have bothered you for help with forms, deadlines, research, and who knows what else over a thousand times - but each and every time you instantly put everything aside to help me. Your kindness borders on the point of criminal, matched only by your intellect and your steadfastness and work ethic. You are arguably the MVP of CAPP Lab, together with the other titles you've acquired along the way as a USB Master, Kim VIDAN, and the Matrix's Neo. I know you will be successful in any endeavor you undertake, and your name will be one heard often in the years to come.

To my professor, Dr. Hyuk-Jae Lee - Thank you for this opportunity to work and interact with you and your students at CAPP Lab. Your procurement of talented individuals is matched only by your ability to develop their potential. I am grateful for your guidance during these two years, and allowing me the freedom to work at my own comfort zone. You guided me into this computer vision and machine learning realm, an area which has fascinated me, and sparked an interest that I see continuing throughout the rest of my career. You are an excellent friend and mentor, and I hope to be able to continue

working with you in the years to come.

To my Sister - You have so much potential and are gifted in so many ways. I am so glad to see you working hard to realize this potential; I know the path is not easy, but with your strength I know you will have no problem following it to conclusion. You are a hard worker, much more so than myself, and I know that your success is inevitable. Thank you for always remembering your big brother, and being strong for all of us.

To my Mother - Thank you for supporting me and my decisions even when I am sure they are not what you would have chosen for me. Under your constant, steadfast watch, I have passed unscathed through the trials I have undergone. With all the many jobs and tasks you perform, I am not sure how you find time in a single day to manage them all while keeping watch over all your darlings. You have always put us first, even over yourself, and there is no way that my words or actions can thank you enough. Thank you for always letting me have the various titles as your "Number One Baby" or "Best Baby Ever." You have my undying love and affection.

I dedicate this and all that I do to the memory of my Father. His nurturing words and his guidance are sorely missed. His ability to overlook my faults, and praise my successes are undoubtedly the reason I have the will of fortitude to continue, and to achieve in all things that I do. I hope you continue to overlook all of my many faults, in the midst of my past and, hopefully, future success. I strive to be a portion of that which you embody - a powerful intellect, a

polished writer, a loving father, a positive thinker - a renaissance man with innumerable talents whose memory lives on in all that knew him. I hope this is the first of many opportunities I have to acknowledge you and all those who have guided me throughout my life.