Object recognition using Geometric Hashing
extensions

Gil Gattegno
Object recognition using Geometric Hashing extensions

Research Thesis

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science

Gil Gattegno

Submitted to the Senate of the Technion - Israel Institute of Technology
Nisan, 5768 Haifa April 2008
The Research Thesis was done under the supervision of Prof. Michael Lindenbaum and Dr. Illan Shimshoni in the Faculty of Computer Science.

The generous financial help of the Technion is gratefully acknowledged.
# Contents

Abstract 1

List of Abbreviations 2

1 Introduction 3

2 Preliminaries 5
   2.1 The imaging model ........................................... 5
   2.2 Affine Coordinates and Invariance .......................... 6
       2.2.1 Affine Coordinates ..................................... 7
       2.2.2 3D Affine Coordinates .................................. 7
   2.3 The geometric hashing algorithm ........................... 8
   2.4 Shallow objects ............................................... 9

3 The Extended Geometric Hashing for non-flat objects 11
   3.1 The range of affine coordinates associated with a projected 3D point 11
   3.2 Good and bad bases ......................................... 15
   3.3 The Extended Geometric Hashing Algorithm .................. 18
   3.4 Results .................................................... 22

4 2.5D Geometric Hashing 23
   4.1 Introduction ................................................ 23
   4.2 A consistency constraint on the non-planarity error .......... 23
   4.3 Efficient algorithm for consistency check: Plane Sweep ....... 24
   4.4 Representing 3D objects for recognition using the modified hashing procedures .................................................. 31
# Synthetic Data Experiments

5.1 Data Generation .......................................................... 33
  5.1.1 Synthetic Model Acquisition .................................... 33
  5.1.2 Synthetic Image Generation ...................................... 33

5.2 Algorithm ................................................................. 33
  5.2.1 Pre-processing ...................................................... 33
  5.2.2 Hashing ............................................................ 34
  5.2.3 Verification ....................................................... 34

5.3 Results ................................................................. 34
  5.3.1 Success rate and run time comparison ......................... 35
  5.3.2 Performance dependence on signal/clutter ratio ............ 35
  5.3.3 Performance dependence on basis filtration ................. 36
  5.3.4 Performance dependence on different consistency tests (2.5 D algorithm) ..................................................... 37
  5.3.5 Performance dependence on termination condition ........ 38
  5.3.6 Performance dependence on candidate list length .......... 38
  5.3.7 2.5D Geometric Hashing vs. Alignment ..................... 39

# Real Images Experiments

6.1 Data Generation .......................................................... 43
  6.1.1 Non-Synthetic Model Acquisition ............................... 43
  6.1.2 Non-Synthetic Image Generation ............................... 43

6.2 Algorithm ................................................................. 44
  6.2.1 Pre-processing ...................................................... 44
  6.2.2 Hashing ............................................................ 44
  6.2.3 Verification ....................................................... 44

6.3 Results ................................................................. 45
  6.3.1 Success rate and run time comparison ......................... 45
  6.3.2 Performance dependence on signal/clutter ratio ............ 47
  6.3.3 Performance dependence on model features .................. 47
  6.3.4 Performance dependence on Occlusion ........................ 49
  6.3.5 Performance dependence on scaling ............................ 50
6.3.6 Performance dependence on verification’s image and model features

7 Conclusions and Future Work

A Affine coordinates properties

B Analyzing the range of affine coordinates associated with image errors

C Determining the required number of image basis selections

D Alter’s Pose Estimation Method

Bibliography
List of Figures

2.1 Any point $p$ on the plane may be represented in the basis specified by basis coordinates $p_0, p_1,$ and $p_2$. ......................................................... 7

2.2 1. A non-flat object, 2. bad selection of bounding box, 3. good selection of bounding box results in shallow object, 4. substantial part of the object considered shallow under several viewing directions. .................. 10

3.1 Error regions in the image. ......................................................... 14

3.2 Vote distribution in a hash table leading to the Informative Basis Filter .......................... 17

3.3 Basis Angle filter: Undesirable bases are associated with basis normal making a large angle with the dominant normal .......................... 17

4.1 Additional Consistency Check: vector length and angle .......................... 25

4.2 Solving Line Segment Intersection Problem with Plane Sweep .......................... 25

4.3 Plane Sweep Algorithm: Sweep line in progress .................................. 27

4.4 Consistency Calculation: 1. Image basis selected, 2. 4th point selected, 3. uncertainty region calculated in the hash table, 4. From each entry in the hash bin the model 4th point is taken and projected to the image, 5. Consistencies calculated with minimum and maximum margins, 6. Adding a region in parameters space associated with a specific model basis, 7. Most consistent entry for each model basis is found after sweep. ............... 29

4.5 Plane Sweep snapshot during consistency check .................................. 30

4.6 Constructing bounding boxes around portions of the model to minimize the angle between the dominant normal and the viewing direction. .......... 31

5.1 Comparing 2.5D GH vs. Extended GH ........................................ 35

5.2 Signal/Clutter Ratio Effect on Success Rate and run time ........................ 36
5.3 Filters Effect on Success Rate and run time ............................................. 37
5.4 Consistencies check effect on success rate and run time .......................... 38
5.5 Comparing Score bounded termination vs. Iteration bounded termination.
      Note that the range of the success rate axis is small ......................... 39
5.6 Candidate list length effects on success rate and runtime ...................... 39
5.7 Comparing 2.5D Geometric Hashing With Alignment .............................. 41

6.1 Model produced by a range camera - different depths are represented by
different colors. The black regions correspond non-valid depth measurement. 44
6.2 Glass & Box Models (18 feature points) ................................................. 45
6.3 Glass Images (50 image points) ............................................................ 46
6.4 Box Images (50 image points) ............................................................... 47
6.5 Signal/Clutter ratio effects (image points 50/100/200) ........................... 48
6.6 Models’ features dependency (9/18/30 feature points) .......................... 50
6.7 Down scaled object(200/366 feature points) ........................................ 51

A.1 System Undergoing an Affine Transformation ........................................ 56
A.2 Basis Size Filter (adapted from [3]) .................................................... 57

D.1 Model points \(m_0, m_1, \) and \(m_2\) undergoing orthographic projection plus scale
to produce image points \(i_0, i_1, \) and \(i_2\) (adapted from [1]) .................. 66
List of Tables

4.1 Plane Sweep Algorithm: Overlapping Rectangles ......................... 27
6.1 Success rate and run time comparison .................................... 47
6.2 Performance dependence on signal/clutter ratio ....................... 49
6.3 Performance dependence on model features ............................ 49
6.4 Performance dependence on Occlusion .................................. 51
6.5 Performance dependence on scaling ..................................... 51
6.6 Performance dependence on verification’s image and model features .... 52
Abstract

Geometric Hashing (GH) is a traditional object recognition technique, designed to recognize flat objects. The algorithm relies on affine coordinates, which, for flat objects, are pose invariant (under the weak perspective imaging model). The aim of this paper is to extend the GH method to recognizing non-flat objects in a scene.

Our approach is to preserve the original Geometric Hashing efficiency, while allowing recognition with various degrees of non-flatness. We propose two algorithms: Extended Geometric Hashing and 2.5D Geometric Hashing and demonstrate their ability to deal with non-flat 3D objects using simulation and real data experiments.
List of Abbreviations

GH - Geometric Hashing

EGH - Extended Geometric Hashing

Ib - Iteration bounded termination

Sb - Score bounded termination
Chapter 1

Introduction

Object Recognition is one of the ultimate goals for vision systems. It is a hard task because the same objects do not look the same in different images, due to changes in imaging geometry, illumination and to changes in the shape of the object itself.

Much of the research in object recognition is focused on methods for compensating the geometric change in appearance due to imaging geometry. An object point corresponds to an image location which depends on the relative camera/object orientation and distance and on internal camera parameters. The location is modeled by a 3D object rotation, perspective projection, and hidden (occluded) part removal. For 2D objects which are far from the camera the geometric change is modeled well by an affine transformation (see [4] section 2.3).

A popular algorithm for 2D object recognition is Geometric Hashing [9]. It is based on affine coordinates which are invariant to geometric changes and facilitate the recognition process.

Affine coordinates however are invariant only to 2D objects. Therefore the geometric hashing algorithm is limited, in principle, to 2D objects as well. The aim of this note is to extend the geometric hashing algorithm usage and to modify it so that it becomes useful for recognizing 3D objects as well.

We consider shallow objects for which one dimension is smaller than the other dimensions. The algorithms we propose work best on shallow objects for which the small dimension (denote depth) is, say, 20% of the other dimensions. They work good enough, however, for objects which are 50% shallow or more. (See section 2.4 for precise definitions.)
We propose two algorithms, *Extended Geometric Hashing* and *2.5D Geometric Hashing*. The first is simpler and faster but the second can deal better with higher depths (and less shallow objects). Like the original geometric hashing algorithm, the proposed algorithm are based on finding affine bases and on using a hash table to disqualify inadequate model bases. The ability of the proposed methods to deal with non-flat 3D objects is demonstrated using simulation and real data experiments.

The paper continues as follows. We start by a short review of affine coordinates, and geometric hashing, as well as some required definitions (section 2). Then we describe the two algorithms (sections 3, 4). The propensities of the algorithms are demonstrated empirically in sections 5, 6. Finally, we conclude our work and describe more recent method in acquiring local invariant features for object recognition (sections 7).
Chapter 2

Preliminaries

2.1 The imaging model

We consider objects and their images in a camera. Let \( P \) be an object point specified by its homogeneous coordinates \( P = (X, Y, Z, 1) \). Let \( p = (x, y) \) be the corresponding image point. If the 3D point is specified by coordinates \( P_C = (X_C, Y_C, Z_C, 1) \), specified in the camera coordinate system then \( p \) is given by the perspective projections

\[
p = \frac{1}{Z_C} M P_C,
\]

where \( M = (K \ 0) \) is a 3X4 matrix, and \( K \) is a \( 3 \times 3 \) matrix describing the intrinsic camera parameters. If the object is specified in a world coordinate system, distinct from the camera frame, the image is given by equation 2.1 as well, except that now \( M = K(R \ t) \), where \( R \) is a rotation matrix and \( t \) is a translation vector, representing the change from the world coordinate system to the camera frame [4].

In this work we make several simplification. First we assume that the camera is calibrated and normalized, meaning that most components in \( K \) are zero. We also assume that the variation of the depth between the viewed points is much smaller then the distance of the points from the image plane. With the later assumption, the perspective projection is approximated well by a weak perspective model, known also as scaled orthographic projection.

For the camera coordinate system, the weak perspective model specifies the projection simply as \( x = sX_C, y = sY_C \). The uniform scaling \( s \) is \( s = f/Z_r \) where \( f \) is the focal length and \( Z_r \) is the distance from some reference point in the object to the camera. In
the more general case, the weak perspective projection is given simply by

\[ p = \frac{1}{Z_r} M P C, \tag{2.2} \]

where \( Z_r \) is the \( Z_C \) coordinate of a reference point in the object. Note that unlike the perspective model, this model is linear: the image coordinates are linear functions of the 3D object coordinates.

Flat objects, undergoing a weak-perspective projection can be represented by a 2D affine transformation. For these objects, each point’s third coordinate can be described as a linear combination of the first two \( (Z = cX + dY) \). Let \( \mathcal{R}_2, t_2 \) be the first two rows of the rotation matrix and the translation vector, respectively. Then, the weak-perspective may be represented as an affine transformation. Recall that \( p = (x, y)^T = (sX_C, sY_C)^T \). Then,

\[
p = s \begin{pmatrix} \mathcal{R}_2 & t_2 \end{pmatrix} \begin{pmatrix} X \\ Y \\ cX + dY \\ 1 \end{pmatrix} = s \begin{pmatrix} r_{11} + cr_{13} & r_{12} + dr_{13} \\ r_{21} + cr_{23} & r_{22} + dr_{23} \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} + \begin{pmatrix} st_x \\ st_y \end{pmatrix}
\]

Usually we ignore the particular imaging parameters and just refer to the general affine transformation specified either in homogeneous coordinates or by the usual coordinates (referred to also as affine coordinates):

\[
\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} X \\ Y \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}
\]

### 2.2 Affine Coordinates and Invariance

A geometric invariant usually refers to an image feature which does not depend on the relative position of the camera and image object, or on other properties (like illumination, reflectance, etc.) of the imaging process. For an unconstrained 3D point set, no invariants can be derived from a single view, and should such a function exist, it would have the same value for all sets with the same number of points [12].
2.2.1 Affine Coordinates

Let $p_0, p_1,$ and $p_2$ be three non-collinear points in 2D space, then $(p_1 - p_0, p_2 - p_0)$ are two non-collinear vectors which span the 2D space. Consider any other point $p$. This point can be represented using these two vectors as a basis (see Figure 2.1). Let the coordinates of a fourth point $p$ with respect to the chosen basis be $(\alpha, \beta)$ satisfying,

\[ p = p_0 + \alpha(p_1 - p_0) + \beta(p_2 - p_0). \]

These coordinates are called Affine Coordinates. They are unique and invariant under weak perspective transformation (see Appendix A).

![Figure 2.1: Any point $p$ on the plane may be represented in the basis specified by basis coordinates $p_0, p_1,$ and $p_2$.](image)

2.2.2 3D Affine Coordinates

For non-flat data sets, a third affine coordinate may be specified using the normal to the basis plane. Let $p_0, p_1,$ and $p_2$ be three non-collinear points in 3D space and let $(p_1 - p_0), (p_2 - p_0)$ and $(p_2 - p_0) \times (p_1 - p_0)$ be a basis spanning the 3D space with $p_0$ as origin. Let the coordinates of a fourth point $p$ with respect to the chosen basis be $(\alpha, \beta, \gamma)$ where

\[ p - p_0 = \alpha(p_1 - p_0) + \beta(p_2 - p_0) + \gamma(p_2 - p_0) \times (p_1 - p_0). \] (2.3)

While unique, these coordinates are not measurable from 2D projections and therefore cannot serve as invariant properties. We shall see that they are still useful for our purpose.
2.3 The geometric hashing algorithm

*Geometric Hashing* is a general *Model-Based* recognition technique introduced by Lamdan and Wolfson [9]. It relies on affine invariants and can handle 2D objects and their images. The algorithm refers to both object and image as sets of points which may be obtained by, say, extracting corners or other feature points. This approach is fairly general but is non-optimal as it does not use the appearance associated with every feature point. Recent papers, showing that some representations of such appearances are relatively stale under pose and illumination changes indeed lead to more efficient algorithms [10]. However, when such information is not available, the GH algorithm is a relatively efficient approach to performing partially occluded object recognition.

The algorithm consists of two basic phases: (i) the construction of a model hash table (preprocessing phase) and (ii) the matching of the models to an image (recognition phase).

In the preprocessing phase the model information is encoded and stored in a large lookup table. This lookup table (hash table) is scene independent and thus can be computed offline. Each model is encoded into the hash table by scanning all ordered triples of model points, considering them as bases. For every basis the affine coordinates \((\alpha, \beta)\) are calculated for every additional model point, and are used to add an entry to the hash table, indexed by these coordinates, and containing the model identity and the model basis. Multiple objects may be represented by the same hash table.

During the recognition phase feature points are extracted and several ordered triples of them are considered as image bases. For every such image basis, the affine coordinate of every other image feature point is calculated and is used to index into the hash table. If indeed, the basis and the other point come from the same model, then due to affine coordinate invariance, the hash table cell will contain an entry specifying this model and the corresponding model basis. These entries are used to vote for the model and the basis, and those achieving a substantial number are considered as a good hypothesis.

This elegant process meets several difficulties in practice. In particular, the unavoidable positional uncertainty in the location of the image feature points makes the process more complex and less reliable. Due to such uncertainty, the affine coordinates extracted from the image are not exactly those calculated from the model (and stored in the hash table). To guarantee that the later are indexed the process votes for a region in the \((\alpha, \beta)\)
plane. When this region is large, many false votes are made and reliability decreases.

Quantitatively, Grimson, Huttenlocher and Jacobs [6] consider location uncertainties bounded by a constant $\epsilon$. They considered model point $m_0, m_1, m_2, m$ associated with affine coordinates $(\alpha, \beta)$ of $m$ relative to the basis $m_0, m_1, m_2$. In ideal condition, without uncertainty, the projection $p$ of $m$ should lie exactly in $p_0 + \alpha(p_1 - p_0) + \beta(p_2 - p_0)$. In uncertain condition however $p$ lies in a circle centered in the ideal location, with radius that can be as high as $2\epsilon(1+|\alpha|+|\beta|)$([6]). The incorrectly placed image point is associated with different affine coordinates relative to the same basis. It turned out that voting for an elliptical region in the hash table guarantees that the corrected model and basis are voted for.

2.4 Shallow objects

Consider a non-flat object, enclosed in a bounding box. We refer to the dimensions of this box as height, width and depth. An object is shallow if its depth is lower than the other dimensions. Specifically, an object is 50% shallow if its depth is less than 50% of its width and height. Note that shallowness, as defined above, depends on the choice of the specific bounding box. We shall be interested in objects for which there is a bounding box specifying them as shallow, and refer to the depth axis as the dominant normal of the bounding box. We found that for general 3D objects and for arbitrary viewing directions, there is usually a substantial part of the object which may be considered as a shallow object with a small viewing angle (see figure 2.2).

A particular imaging situation is characterized by the viewing angle between the camera's optical axis and the dominant normal. We shall be interested mostly in cases where the viewing angle is small. Note that unless this angle is small, a substantial part of the object may be occluded.
Figure 2.2: 1. A non-flat object, 2. bad selection of bounding box, 3. good selection of bounding box results in shallow object, 4. substantial part of the object considered shallow under several viewing directions.
Chapter 3

The Extended Geometric Hashing
for non-flat objects

The first algorithm we propose, denoted Extended Geometric Hashing, is designed to handle models which are relatively shallow (50% shallow or lower). It essentially follows the original Geometric Hashing algorithm with minor modification. Much like the original one, it is efficient, and reaches a success rate of over 90% within the described depth limitation.

3.1 The range of affine coordinates associated with a projected 3D point

A point on a 3D object may be described, relative of a three point basis, by 3D affine coordinates \((\alpha, \beta, \gamma)\):

\[
P = P_0 + \alpha(P_1 - P_0) + \beta(P_2 - P_0) + \gamma((P_1 - P_0) \times (P_2 - P_0)).
\]

These coordinates are invariant to rigid transformations but, in general, not to projections. In the particular case, when \(P\) lies in the plane specified by \((P_0, P_1, P_2)\), \(\gamma = 0\) and then, the 2D affine coordinates \((\alpha, \beta)\), are invariant to weak perspective projection [6]. That is, letting \(p, p_0, p_1, p_2\) be the projections of the coplanar points \(P, P_0, P_1, P_2\), respectively, the relation

\[
p = p_0 + \alpha(p_1 - p_0) + \beta(p_2 - p_0)
\]
is satisfied for the same $\alpha, \beta$ specified between the 3D points. Although the non-flat object’s affine coordinates in the image vary from the ones calculated for the 3D object (model), their deviation is limited when the object’s depth is limited. The Extended Geometric Hashing algorithm takes into account the object’s depth to calculate the range for the variation of the affine coordinates, and uses this range to extend the traditional GH algorithm.

Consider a particular model basis $(P_0, P_1, P_2)$ and denote the plane specified by the basis points as the basis plane. We refer to the coordinate system defined by the basis as the object system. Let $P'$ be the projection of the point $P$ on the basis plane. The affine coordinates $(\alpha, \beta)$ of $P$ are shared by $P'$ and are invariant descriptors of $P'$ relative to the basis $(P_0, P_1, P_2)$, under weak perspective projection. These coordinates are used to store $P$ in the hash table. Note that the projected point $P'$ isn’t a real point of the object and is not visible in any image of it. However if the camera axis is roughly perpendicular to this plane, $p$ is relatively close to $p'$ and the affine coordinates of $p$ measured in the image, relative to the image basis $(p_0, p_1, p_2)$, do not deviate significantly from the (invariant) coordinates of $p'$.

More quantitatively, the relation between (the imaginary) $p'$ and $p$ can be bounded using the feature points measurement error, the height of $P$ above the basis plane, and the viewing angle.

**Claim 1.** Consider a 3D model basis $(P_0, P_1, P_2)$ and a fourth point $P$, associated with the affine coordinates $(\alpha, \beta, \gamma)$. Let $N = \frac{(P_0 - P_1) \times (P_1 - P_0)}{\|P_1 - P_0\| \times \|P_1 - P_0\|}$ be the normal to the basis plane. Consider a weak perspective projection specified by a viewing direction $V$, and scaling $s$. Let

$$p^* = p_0 + \alpha(p_1 - p_0) + \beta(p_2 - p_0).$$

and

$$e_N = (V \times N) \times V.$$ 

Then,

$$p - p^* = s \cdot \gamma \cdot \sqrt{(1 - (V \cdot N)^2) \| (P_1 - P_0) \times (P_2 - P_0) \|} \cdot e_N.$$

Using an alternative notation, let $\phi$ be the angle between $N$ and the viewing direction $V$. Let $h$ be the (3D) distance of $P$ from the basis plane. Then,

$$p - p^* = s \cdot h \cdot \sin \phi \cdot e_N.$$
Proof. A general vector $\mathbf{P}$ may be decomposed into a sum of two perpendicular components. One lying along $\mathbf{V}$ and another lying along the unit vector $\mathbf{e}_p = (\mathbf{V} \times \mathbf{P} / \|\mathbf{P}\|) \times \mathbf{V}$. Therefore, the point $P$ is projected, by the weak perspective model to the point

$$p = s(P \cdot \mathbf{e}_p)\mathbf{e}_p.$$ 

In particular, the difference vector, $\mathbf{P} - \mathbf{P'} = h \cdot \mathbf{N}$ is projected to $p - p' = s \cdot h \cdot \sin \phi \cdot \mathbf{e}_n$. By the affine coordinates invariance, the point $P'$ on the basis plane is projected to $p' = p'$. The alternative formulation with the different notation results by observing that $P - P' = \gamma(P_1 - P_0) \times (P_2 - P_0)$, and by noting that the angle between $\mathbf{V}$ and $\mathbf{e}_N$ is $\pi/2 - \phi$.

\[\square\]

We consider the absolute difference between the location of the projected point and the imaginary location of the corresponding “depth-less” point $p'$, as an “error” induced by the non-planarity and refer to it as $\epsilon_{non-planarity}$. In the context of the extended GH algorithm we use a uniform bound on the non-planarity error associated with all points on the object: $\epsilon_{non-planarity} \leq s \cdot h_{\max} \cdot \sin \phi_{\max}$. Working with shallow objects and limited viewing angles, allows us to bound $\epsilon_{non-planarity}$ by a reasonable value.

Given the image basis and the point $p$, we now know that some point in the disk of radius $\epsilon_{non-planarity}$ around $p$, is associated with affine coordinates (relative to the image basis), which are equal to those of the point $P$ (and $P'$). This enables us to proceed with the usual GH process. Before proceeding, we should add the other error component, related to image measurement inaccuracy.

The noise in the image basis points measurements is propagated and changes in the location of $p'$. The common model, adopted also here is that every feature point is measured with an added unknown noise, of arbitrary direction and size bounded by $\epsilon$. As discussed in [6], the error associated with measuring the basis points can move $p'$ to any location within a disk of radius $\epsilon(1 + 2|\alpha| + 2|\beta|)$, centered around $p'$. The additional error in measuring $p$ may move it from its true location by $\epsilon$. Therefore, the inaccuracy induced errors, may increase the distance between $p$ and $p'$ by a total of $2\epsilon(1 + |\alpha| + |\beta|)$.

The total error, including measurement based and non-planarity induced errors, is thus given as follows

**Claim 2.** Consider the imaging situation described in Claim 1, a viewing angle limited by $\phi_{\max}$ and an object with height limited to $h_{\max}$. Let $q (q_0, q_1, q_2)$ be the noisy measurement
of the point \( p \ (p_0, p_1, p_2) \), satisfying \( \|p - q\| \leq \epsilon \). Then, the distance between \( q \), and

\[
q^* = q_0 + \alpha(q_1 - q_0) + \beta(q_2 - q_0).
\]

is at most \( 2\epsilon(1 + |\alpha| + |\beta|) + \epsilon_{\text{non-planarity}} \) \( \epsilon_{\text{non-planarity}} = s \cdot h_{\text{max}} \cdot \sin\phi_{\text{max}} \).

Again, Claim 2 imply that there is a point with the correct affine coordinates within a specified disk around \( q \) (see Figure 3.1). To use the hash table efficiently we would like to know the range of affine coordinates associated with all the points in this disk.

![Figure 3.1: Error regions in the image.](image)

**Claim 3.** Consider an image basis \( (p_0, p_1, p_2) \) and a fourth point \( p \). Let \( q \ (q_0, q_1, q_2) \) be the noisy measurement of \( p \ (p_0, p_1, p_2) \), satisfying \( \|p - q\| \leq \epsilon \). Then, unless the basis
is very small or nearly collinear, the range of affine coordinates of \( p \) relative to the basis \( p_0, p_1, p_2 \) is an ellipse.

For proof and ellipse characteristics see Appendix B

To make the additional non-planarity component of the error small, we require that both \( s \) and \( \phi \) are small. These requirements are often not hard to satisfy. Note that having a very small \( s \) value, will make the object size close to zero, resulting in undesirable large uncertainty regions and much false voting. In our context which is common, we assume \( Z_0 \) is large enough to make the weak perspective projection hold, yet small enough so that the down scaling won’t be too harsh. The later requirement that \( \phi \) is small, is also satisfied in many situations and can be guaranteed using multiple representation of the model in the hash table (see Section 7).

3.2 Good and bad bases

For an efficient usage of the GH algorithm, the error regions associated with each point must be small. Large error regions imply that in the voting stage, many bases are voted for, causing a large computational expense as well as a substantially higher chance for false voting.

The error regions stay small if both the measurement inaccuracy induced error is small and the non-planarity error is small. The first condition is satisfied (for reasonable measurement errors) if the affine coordinates associated with \( p \) are both small. This happens for bases which are not small themselves (in terms of the area of the corresponding triangle); see equation A.1.

The second, non-planarity, component of the error is small if many of the object points are close to the basis plane. This happens for many bases when the object is indeed shallow. Note however that this sufficient condition is not always necessary; For every basis there is at least a small range of viewing angles for which the non-planarity errors are small. The range of viewing angles for which the errors associated with a large fraction of the object’s points are small, is wide for many bases. On the other hand, bases for which the normal to the basis plane makes a large angle with the dominant normal of a very shallow object, are likely to be bad for the GH algorithm: the range of viewing
angle for which the errors are small is small as well, and much of the object is likely to be occluded in these angles.

Therefore, in both the preprocessing and recognition parts of the algorithm, we use basis filtering to remove bases which are not likely to be good. Such filtering implies that the number of image bases selected until a good image basis composed of projected model points from good model basis is found increases. The recognition time does not necessarily increase because the sparser hash table is easier to maintain and access and no time is wasted in the verification of matches which are likely to be wrong.

We considered four basis filtration processes

- **Basis size filter** - Large affine coordinates result in undesirable large uncertainty regions and much false voting. The affine coordinates absolute values may be expressed as a ratio of two triangles areas (appendix A), e.g. \(|\alpha| = \frac{1}{2} \sum \triangle P_0^P P_2^P P_3^P \). To reduce the fraction of large affine coordinates, bases associated with small triangles whose area is smaller than some fraction (usually 0.1) of the area of the bounding box of the model, are disqualified.

- **Informative basis filter** - The distribution of indices over the space of invariants is non-uniform and causes non-uniform occupancy of the hash table ([15], [13]). Some bins in the hash table, associated with small affine coordinate, contain a large number of entries and therefore provide poor discrimination between bases and models. Voting according to the entries from such hash bin, wastes times and may even degrade performance. Therefore, we consider a basis for which the majority of the remaining feature points are associated with small affine coordinates as a non-informative basis. Such a basis is discarded by this filter. In our work we related to an affine coordinate between \(-0.5\) and \(0.5\) as a small one, and filtered out a basis if more than 50% of its 4th points had small affine coordinates (see figure 3.2). This is a simplified form of the suggestion made by [14], which develops a bayesian interpretation of geometric hashing.

- **Basis Angle filter** - To decrease the non-planarity component of the error, the angle \(\phi\), between the viewing direction and the basis plane normal should be small. Limiting the angle \(\phi_{Camera-BB}\) between the viewing direction and the dominant normal of the bounding box, does not help directly, because the basis plane normal does
Figure 3.2: Vote distribution in a hash table leading to the Informative Basis Filter

not coincide with the dominant normal. However by limiting the angle \( \phi_{Basis-BB} \) between the basis and the dominant normal, the angle \( \phi \leq \phi_{Basis-BB} + \phi_{Camera-BB} \) is limited as well (In our experiments we rejected bases associated with \( \phi_{Basis-BB} \) larger than 30°) (see figure 3.3).

Figure 3.3: Basis Angle filter: Undesirable bases are associated with basis normal making a large angle with the dominant normal
• **Depth Consistency filter** - since the depth of the model is bounded, we rule out a basis if a large percentage of points, specified in the object system defined by it, exceed a depth threshold. It is convenient to relate to the depth in terms of the third affine coordinate $\gamma$, therefore, we compare each points' $\gamma$ coordinate with the maximum possible $\gamma$ (denoted $\gamma_{\text{max}}$), which is the maximum depth specified in the object system defined by the current model basis. If for a large percentage of points, the $\gamma$ coordinate exceeds the $\gamma_{\text{max}}$ value, the basis is disqualified. In our experiments the percentage is 50%. Note that for shallow objects this filter is related to the basis angle filter.

The filters may be used both in the preprocessing and in the recognition stages of the algorithm (see below). They may be used alone or in combination. In the experiments section we investigate their influence and make some recommendations.

### 3.3 The Extended Geometric Hashing Algorithm

Like the traditional GH algorithm, the proposed algorithm is made of two phases: preprocessing and recognition. The preprocessing phase stores the 3D model information in memory (hash table). The hash table is scene independent and thus the preprocessing phase is done off-line without affecting the throughput. Every model triplet is considered as a candidate to be a basis. If it passes a filtration process (see above) then it becomes a basis, the affine coordinates $(\alpha, \beta)$ are calculated for every feature point ("4th point"), and used, after quantization, as an index to the hash table, and a hash table entry (composed of the basis, $\alpha, \beta$ and the 4th point) is recorded.

The recognition phase selects image bases and aim to match them against the model bases. Each selected image basis passes a filtration process also (the only applicable filters in this stage are the **basis size filter** and the **Informative basis filter**). If successful, the range of affine coordinates associated with every other image feature point is calculated and used to access the hash table. Every entry found in the hash table casts a vote for image-model basis match. A candidate list for each image basis consists of the top $N_c$ scorers. These candidates are verified to make the final decision.

The proposed algorithm is similar to the traditional (planar) **Geometric Hashing algorithm**. The two main differences are:
1. Not all bases are used and some are removed, both during preprocessing and recognition. Such processes are carried out in most practical versions of the traditional GH algorithm [3] but here, due to the non-planarity, we filter the bases using other criteria.

2. The calculation of the affine coordinate range used to index the hash table, takes into account non-planarity errors as well (claim 2 and 3).

\begin{procedure}
\caption{EGH Preprocessing(S)}
\begin{algorithmic}
\State (S is the set of 3D feature points extracted from a model.)
\Repeat
\State Select feature points triplet $P_0, P_1, P_2 \in S$
\If{filtration succeeds}
\State storeModelBasis(S, $P_0, P_1, P_2$)
\EndIf
\Until{all possible ordered triplets have been selected.}
\State Return a hash table.
\end{algorithmic}
\end{procedure}

\begin{procedure}
\caption{storeModelBasis(S, $P_0, P_1, P_2$)}
\begin{algorithmic}
\State $S' \leftarrow S \setminus \{P_0, P_1, P_2\}$
\Repeat
\State Select $P \in S'$
\State calculate the affine coordinates $(\alpha, \beta)$
\State $(i, j) = \text{Quantization}(\alpha, \beta)$
\State insert into hash table bin $(i, j): P, P_0, P_1, P_2, \alpha, \beta$
\State $S' \leftarrow S' \setminus \{P\}$
\Until{$S' = \{\}$}
\end{algorithmic}
\end{procedure}

Notes: The function $\text{Quantization}(\alpha, \beta)$ converts $(\alpha, \beta)$ to hash table indices.
procedure EGH Recognition($I, S, \text{Hash Table}$)

($I$ is a set of interest points taken from the input image,
$S$ is a set of 3D feature points extracted from a model.)

$L \leftarrow \{\}$, empty image basis Candidate List.
$G \leftarrow \{\}$, empty Global List of basis matches.

Compute the number of image bases selections needed, $N_o$.

repeat

Randomly choose an image basis ($p_0, p_1, p_2 \in I$).

If the basis passes the filtration criteria, then

$L \leftarrow \text{EGH voteCasting} (p_0, p_1, p_2, I, \text{Hash Table})$

Verify each match in $L$ and provide a verification score

select highest scored match $l \in L$

$G \leftarrow G \cup \{l\}$, $L \leftarrow \{\}$

until $N_o$ image bases have been selected.

Select match $m \in G$ with highest verification score

Return Match $m$.

procedure EGH voteCasting($p_0, p_1, p_2, I, \text{Hash Table}$)

$I' \leftarrow I \setminus \{p_0, p_1, p_2\}$

repeat

Select $p \in I'$

Calculate the compatible affine coordinates range

Cast a vote for the indexed basis in all bins in range:

$I' \leftarrow I' \setminus \{p\}$

until $I' = \{\}$

Create a candidate list with $N_c$ matches received the highest votes.

Return candidate list.
Notes

1. To succeed, the recognition phase must select at least one image basis which is the projection of a model basis, and which gets a high score both in voting and in verification. The number of randomly selected image bases, which guarantee this event with high probability $1 - \delta$ is

$$N_0 = Pr_A(-\log\delta\left(\frac{m}{n}\right)^3),$$

where $n, m$ are the number of image and model points respectively and $Pr_A$ stands for the probability for a correct basis to receive the highest score relative to a correctly chosen image basis (See Appendix C for the derivation.)

2. As observed in [6], the uncertainty in the measurement of the image interest points, depends on the particular imaging parameters and therefore cannot be handled in preprocessing, by say pre-computing the range of affine-invariant coordinates associated with every point. The effect of uncertainty must be accounted for at recognition. Note however that if the measurement errors are negligible, the affine coordinates uncertainty region associated with non-planarity error could be pre-calculated (see Appendix B.

3. Alter’s pose estimation method was used for verification. it essentially project every model point on the image and sums the number of model point matched by an image point in its error region. Note that here there is no non-planarity problem and the error regions are much smaller, yielding high accuracy. The verified match that received the highest score is added to a global match list.

4. The candidate list allows us to verify more than one match for each image basis, a variation on the algorithm which reduces the probability of false answers with the expense of additional computational load.

5. The recognition process stops after $N_0$ image bases were tested. Then, the top scorer from the global match list is selected as the match. Alternatively it may stop (earlier but somewhat less reliably) when some basis gets a score higher than a threshold. We refer to these two versions as Iteration bounded (Ib) termination and Score bounded (Sb) termination.
3.4 Results

The Extended Geometric Hashing algorithm was tested on synthetic and real data; see a detailed description of the experiments in sections 5,6. We found that while the traditional 2D Geometric Hashing, naturally deteriorates as the depth of the model increases, the Extended Geometric Hashing remained stable with a success rate of over 90% on models that are 40% shallow (see Experiment 5.3.1).

As for time consumption, the extended algorithm completes the recognition task in a few seconds (see Experiment 5.3.1).

Thus, the Extended Geometric Hashing algorithm is fast and effective for shallow objects (up 50%). For higher depths, the success rate declines. The second algorithm, denoted the 2.5D Geometric Hashing algorithm, can handle higher depths better.
Chapter 4

2.5D Geometric Hashing

4.1 Introduction

We now propose another algorithm, denoted the 2.5D Geometric Hashing. It is designed to overcome the limitation of the Extended Geometric Hashing and to recognize reliably deeper (less shallow) objects, but, on the other hand, is more computationally expensive.

The 2.5 Geometric Hashing algorithm relies on a more accurate model of the feature locations. While the Extended Geometric Hashing algorithm builds on a general bound on the location error size, the 2.5 Geometric Hashing uses a more detailed prediction, which specifies a consistency constraint.

4.2 A consistency constraint on the non-planarity error

The more detailed error characterization relies on Claim 1. Note that the non-planarity error vector associated with one basis and all other model points are directed along the same vector. Moreover,

Claim 4. Consider a 3D model basis \((P_0, P_1, P_2)\) and a fourth point \(P\), associated with the affine coordinate \((\alpha, \beta, \gamma)\). Let \(p_0, p_1, p_2, p\) be the projections of these points under the weak perspective imaging model, and let

\[ p^* = p_0 + \alpha(p_1 - p_0) + \beta(p_2 - p_0). \]

Then, \((p - p^*)/\gamma\) is the same vector for all points \(P\).
Proof. Recall that, by Claim 1, \( \frac{e_p}{\gamma} = s \cdot \sqrt{(1 - (V \cdot N)^2)} \|(P_1 - P_0) \times (P_2 - P_0)\| \cdot e_N. \)

The scale \( s \) and the viewing direction \( V \) are properties of the imaging process and all other parameters \( (P_0, P_1, P_2, N) \) are properties of the basis.

The claim implies that both the length of the vector \( (p - p^*)/\gamma \) and its angle are common to all projected model points. Therefore, it may be used to identify a genuine match between an image point and a corresponding model point: the model point parameters \( (\alpha, \beta, \gamma) \) and the basis may be used to calculate \( (p' - p^*)/\gamma \). A group of genuine votes for a match are consistent if they share this vector.

To test this vote consistency robustly we should consider the inaccuracy involved in the location measurements. By the common model, assuming isotropic bounded error, the real projections \( p_0, p_1, p_2, p \) lie in disks of radius \( \epsilon \) about the observed locations, denoted \( q_0, q_1, q_2, q \). The projection of \( P^* \), denoted \( p' \), lies in a disk of radius \( \epsilon(1 + 2|\alpha| + 2|\beta|) \) about

\[ q^* = q_0 + \alpha(q_1 - q_0) + \beta(q_2 - q_0). \]

Denote these disks \( D_{q_0}, D_{q_1}, D_{q_2}, D_q, D_{q^*} \). One vector between a point in \( D_{q^*} \) and a point in \( D_q \) is the projection of \( P - P' \), if indeed the points \( q_0, q_1, q_2 \) correspond to a basis.

To test the consistency we describe the vector \( E_p = (p - p^*)/\gamma \) by its length \( ||E_p|| \) and its angle (relative to the x axis) \( \angle E_p \). For every entry found in the hash table, we vote not only for the corresponding basis but also for the minimal rectangular range of \( ||E_p|| \times \angle E_p \) including all \( (||E_p||, \angle E_p) \) pairs consistent with the corresponding point (see figure 4.1).

After all the voting associated with a single image basis is done, we look for the maximally consistent \( (||E_p||, \angle E_p) \) region. Every \( E_p \) vector in this region is consistent with several matches between image and model points. Finding the region consistent with a maximal number of matches, is carried out efficiently using a plane sweep algorithm [7].

4.3 Efficient algorithm for consistency check: Plane Sweep

Plane Sweep is a powerful approach for solving problems involving geometric objects in the plane. The best examples for such problems are line segments intersection, finding the contour of the union of rectangles and Voronoi diagrams. In sweeping, an imaginary
vertical sweep line passes through the geometrical objects, from, say, left to right. The status of the sweep line changes at particular points which are called event points. An example of event points is the end points of the segments in line segments intersection problem (see figure 4.2).

Figure 4.1: Additional Consistency Check: vector length and angle

Figure 4.2: Solving Line Segment Intersection Problem with Plane Sweep
As the sweep line moves through an event point, the problem, which has been solved for the data to the left of the sweep line, is currently now solved for the data at or near the sweep line. Eventually the problem is solved for all the plane. Essentially, the plane sweep technique reduces a problem in two-dimensional space to a series of problems in one-dimensional space (along the sweep line), which are easier to solve.

Our problems revolves around axis parallel rectangles intersection. Each rectangle represent the uncertainty in a $\| E_p \| \times \mathcal{L} E_p$ 2D space. The event points in our case are the corners of the rectangles and the intersection points between their segments, which are computed on the fly. Our goal is finding a region where a maximal number of rectangles overlap. That is, a maximal set of points consistent with the error vector constraint.

We shall illustrate the plane sweep algorithm using an example. We start the plane sweep algorithm with the sweep line to the left of all rectangles and an empty overlapping rectangle list (see figure 4.3, position 1). The moments at which the sweep line reaches an event point it updates the overlapping rectangle list according to the event (see table 4.1):

- Entering a new rectangle (e.g. event points $(y_{1,2}, y_{2,2})$): the segment $(y_{1,2} - y_{2,2})$ is added to the list and the number of overlapping rectangles in that section increased (figure 4.3, position 2 and 3).

- Entering a new rectangle that intersects with previous ones (e.g. event points $(y_{5,4}, y_{4,4}, y_{1,4}, y_{6,4})$): the new segment $(\overline{y_{5,4} - y_{6,4}})$ is intersected and the new sub segments are added to the list and updated (figure 4.3, position 4).

- Exiting a rectangle: the segment is deleted from the list and the number of overlapping rectangles in that section decreased.

- Exiting a rectangle that intersects with previous ones (e.g. event points $(y_{1,5}, y_{6,5}, y_{2,5})$): the new segment $(\overline{y_{1,5} - y_{2,5}})$ is intersected, removing only the portion of the segment that does not overlap $(\overline{y_{6,5} - y_{2,5}})$ and joining adjacent segments $(\overline{y_{4,5} - y_{1,5}})$ and $(\overline{y_{1,5} - y_{6,5}})$ (figure 4.3, position 5).

The 2.5D Geometric Hashing algorithm consists of two phases: pre-processing and recognition. The pre-processing phase is similar to Extended Geometric Hashing pre-processing stage, with the addition that now we store the third affine coordinate $\gamma$ in the hash table as well.
Table 4.1: Plane Sweep Algorithm: Overlapping Rectangles

<table>
<thead>
<tr>
<th>Sweep Line Position</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overlapping Rectangles by Segments</td>
<td>-</td>
<td>-</td>
<td>$y_{1,2} - y_{2,2}$</td>
<td>$y_{1,3} - y_{2,3}$</td>
<td>$y_{1,4} - y_{2,4}$</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>$y_{3,5} - y_{4,5}$</td>
<td>$y_{5,5} - y_{6,5}$</td>
<td></td>
</tr>
</tbody>
</table>

Sweep Line progress

![Diagram of Plane Sweep Algorithm](image)

Figure 4.3: Plane Sweep Algorithm: Sweep line in progress

The recognition phase starts, Similarly to the Extended Geometric Hashing, by randomly selecting image bases (i.e., triplets of interest points). Each selected image basis is being filtered. If it passes the filtering criterion, the process continues by calculating the affine coordinate range for the remaining image interest points. Each hash table bin in this affine coordinate range is accessed to cast a vote for image-model basis match. Recall that the bins hold records of the model bases, the $4^{th}$ model point, and the corresponding 3D affine coordinates ($\alpha$, $\beta$, $\gamma$).

Every model point found is projected to the image plane (creating $q_*$). The range of the feasible $\|E_p\|$ and $\angle E_p$ values is determined (separately), and the combined rectangle range
is specified as the rectangle bounding all feasible $\|E_p\|$ and $\angle E_p$ values; see Figure 4.4.

After casting all votes for a given image basis we sweep the plane, locate the most consistent entry (see Figure 4.5) and add it to the candidate list with the number of image points supporting it. This number is equal to the maximal number of intersecting rectangles. After the required number of image bases has been reached we verify the basis associated with the top $N_c$ scorers from the candidate list and select the most consistent one.

To summarize, the main difference is in the vote accumulation. The 2.5D Geometric Hashing algorithm requires not only that the number of votes is large but also how many of them are consistent with one projection.

The recognition phase is summarized by the same algorithm described in section 3.3, except that the voting is replaced by the 2.5D voteCasting procedure.
Figure 4.4: Consistency Calculation: 1. Image basis selected, 2. 4\textsuperscript{th} point selected, 3. uncertainty region calculated in the hash table, 4. From each entry in the hash bin the model 4\textsuperscript{th} point is taken and projected to the image, 5. Consistencies calculated with minimum and maximum margins, 6. Adding a region in parameters space associated with a specific model basis, 7. Most consistent entry for each model basis is found after sweep.
procedure **2.5D voteCasting**($p_0, p_1, p_2, I$, Hash Table)

$I' \leftarrow I \setminus \{p_0, p_1, p_2\}$

repeat

	Select $p \in I'$

	Calculate the compatible affine coordinates range

repeat

	Select an entry ($P_i, P_j, P_k, P$) from bin

	Project model point's projection ($P$) to image.

	Compute the ranges of $\angle E_p$ and $\|E_p\|$ (denoted $R_{\angle E_p}, R_{\|E_p\|}$).

	Add an entry $R_{\angle E_p} \times R_{\|E_p\|}$ to the plane sweep table with reference to $P_i, P_j, P_k$

for all entries inside bins in range

$I' \leftarrow I' \setminus \{p\}$

until $I' = \{\}$

Sweep through the plane sweep table and locate the most consistent entry for each model basis

Create a candidate list with top $N_c$ matches.

Return candidate list.
4.4 Representing 3D objects for recognition using the modified hashing procedures

Since our hashing procedures are capable of handling shallow objects rather than full 3D ones, we need to represent the later in such a way that will allow recognition. The object should be divided as such, that from any possible viewing angle a representation of it, as a shallow object suitable for recognition using our algorithm, will exists.

Any object can be encoded from several viewing directions, each within the viewing angle limitation imposed by our recognition algorithm. For each viewing direction we can construct a bounding box that bounds the portion of the interest area that faces the camera, creating a small angle between the dominant normal and the viewing direction (see figure 4.6).

![Diagram of Viewing direction and normal vectors]

Figure 4.6: Constructing bounding boxes around portions of the model to minimize the angle between the dominant normal and the viewing direction.

By doing so, the object is actually partitioned into semi-flat parts, each stored in the hash table as independent objects and will be treated as such throughout the algorithm. Upon decision, results of different parts of the same object are factored in.

However, representing an object using a collection of shallow objects effects the algorithm throughput. The objects are now encoded and stored several times (consider a sphere around the object spanned by cones each within angle limitation) in the hash table. The hash bins become highly populated causing a large amount of votes for each
image point.

Creating a dense grid of viewing angles spanning all possible viewing angles around the object with overlapping, results in large amount of shallow object, highly populated hash table and low throughput. Creating a grid with a fixed spanning angle can results in viewing angle for which no shallow object exists. Therefore an adaptive procedure will be more suitable: starting with a dense grid and dilate each grid area based on the depth of the object points, as seen from that specific view.
Chapter 5

Synthetic Data Experiments

5.1 Data Generation

5.1.1 Synthetic Model Acquisition

Our synthetic objects were composed of $m$ feature points randomized inside a 3D box with a variable depth $\sigma$.

5.1.2 Synthetic Image Generation

The image is simulated by projecting the object’s 3D points using a parallel projection in a randomly chosen direction making a $30^\circ$ angle with the depth axis. (i.e. all projections directions form a cone about the depth axis.) This is, in a sense, the worst case projections set for which the projection angle is not more than $30^\circ$. A random location noise, modeled by an isotropic uniform distribution in a disk of radius 3 pixels is added to the projected model points. Random additional dots, simulating clutter, are added so that the total number of points in the image is $n$.

5.2 Algorithm

5.2.1 Pre-processing

For the synthetic experiments the features points were directly generated for the model and projected to create the image. We than randomly add clutter to the image.
5.2.2 Hashing

The hashing mechanism was implemented as follows: A two dimensional array was defined representing the hash table, containing 100 cells (hash bins) in each dimension. Each cell was comprised of a linked list of hash entries that holds the basis triplet, 4th point and the affine coordinates (α, β and γ).

The voting was performed using a plane sweep structure that was implemented as a double linked list. Each chain is equivalent to a region in the parameter space ($R_{\perp E_p} \times R_{[E_p]}$) associated with a specific model basis.

5.2.3 Verification

For the synthetic data experiments a verification score was assigned to every match in the candidate list using Alter’s pose estimation method (see Appendix D).

5.3 Results

A recognition task was considered successful when the match that received the highest score, after verification, was between an image triplet and a model triplet. This criterion is conservative because in many cases, the pose associated with a partial match cannot be told apart from the real one. Such partial matches are as good as the correct ones for recognition and pose estimation, and yet are considered as “failures” in our success rate calculations.

Every experiment is a function of several parameters

**Algorithm type** Geometric Hashing (GH), Extended Geometric Hashing (EGH), 2.5D Geometric Hashing (2.5DGH) and Alignment (A)- added for comparison.

**Model size** - m

**data size** - n

**Model depth** - σ

**Types of filters used**

**Termination condition** Iteration bounded (Ib) or Score bounded ((Sb)
5.3.1 Success rate and run time comparison

(This experiment was carried out for $m = 18, n = 25, \epsilon = 3, \sigma =$ variable, filters = All, termination = Ib, algorithm = GH/ EGH/2.5DGH)

The first experiment illustrates the difference in reliability and the running time for the proposed algorithms as a function of the object depth $\sigma$. This experiment illustrates the difference between the two approaches. The extended Geometric Hashing algorithm is fast and effective on relatively shallow objects whose depth is less than 50%, but declines in success rate when depth increases above that. The 2.5D Geometric Hashing algorithm, on the other hand, remains stable, reaching a success rate of above 90% even for depth as high as 70%. On the other hand, as the depth increases, it becomes less efficient. Since the original Geometric Hashing algorithm is not designed to handle non-flat objects, it quickly deteriorates and fails even for moderate depths.

![Graph](image1.png)

Figure 5.1: Comparing 2.5D GH vs. Extended GH

5.3.2 Performance dependence on signal/clutter ratio

(This experiment was carried out for $m = 18, n = 25/50/75, \epsilon = 3, \sigma =$ variable, filters = All, termination = Ib/Sb , algorithm = EGH/2.5DGH)

More clutter in the form of image points which are not model projections, make the task harder. The reliability decreases with more clutter because there are more chances for accidental false alignments. The runtime increases because it obviously takes more
time to find a good image basis. In this experiment we set the number of image point as $n = 25, 50$ or $75$. The number of model points remains the same ($m = 18$).

The results are in agreement with these expectations. We found that score bounded termination is faster but less reliable (see experiment 5.3.5). These characteristics are emphasized when the signal to noise ratio is lower (see Figure 5.2).

Thus, for shallow objects the extended hashing is the best choice. For deeper objects the 2.5D hashing is better, especially when using iteration bounded termination.

5.3.3 Performance dependence on basis filtration

(This experiment was carried out for $m = 18, n = 25, \epsilon = 3, \sigma$ = variable, filters = None/Basis angle/Basis size/Depth consistency/All, termination = Ib, algorithm = 2.5DGH)

We use filters throughout the algorithm for ruling out undesirable model and image bases, thus enabling the algorithm to achieve the success rate seen in this experiment. Undesirable bases are ones that do not meet with our pre-defined restriction (the purpose of which is to bound the error) or fail to contribute useful information. Each filter improves the success rate, but it takes their combined effort to reach higher success rate while depth increases (see Figure 5.3).

We use filtration in the pre-processing and in the recognition phases. The Basis Angle Filter and the Depth Consistency Filter are being used only in the pre-processing phase. The Basis Size Filter is in use in both phases.
Figure 5.3: Filters Effect on Success Rate and run time

As for throughput, using filters results in a more stable and accurate process but a slower one. In the pre-processing phase filtration is used to keep undesirable bases from being stored in the hash table. Scars table reduces access time which should have resulted in a higher throughput, however, it effects the number of image bases selection needed to reach the desired success rate, which is now higher. Therefore, the pre-processing phase filters has almost no effect on the throughput. On the other hand, once the number of selected image bases was calculated, each filtration being done in the recognition phase will effect the throughput, because each selected image basis is tested, in runtime, to see if it meets with the filter condition (the Basis Size Filter in our case).

5.3.4 Performance dependence on different consistency tests (2.5 D algorithm)

(This experiment was carried out for \( m = 18, n = 25, \epsilon = 3, \sigma = \) variable, filters = All, termination = Ib, algorithm = 2.5DGH)

The consistency check, which is the essence of the 2.5D Geometric Hashing is effective but computationally expensive. Therefore, we tested whether partial consistency, which check the consistency of either the angle \( \angle E_p \) or the size \( \|E_p\| \) of \( E_p \) suffice. We found that while these partial consistent checks are effective and indeed reduce the time, they are not as effective as the 2D combined consistency check (see Figure 5.4).
5.3.5 Performance dependence on termination condition

(This experiment was carried out for \( m = 18, n = 25, \epsilon = 3, \sigma = \) variable, filters = All, termination = Ib/Sb, algorithm = 2.5DGH)

The basic version of algorithm checks a pre-specified number of image bases, depending on the expected model/clutter ratio in the image and the required reliability. Both algorithms are accelerated by stopping to search for the best basis when some basis gets a high verification score. This acceleration comes with a price of reduced reliability. We tested this tradeoff by checking several thresholds on the score bounded termination version of the algorithm; see Figure 5.5.

We found that while the success rate significantly dropped as we lowered the threshold beyond a certain limit, the difference in success rate was not high when the threshold was high. The run time, however, was dramatically improved.

5.3.6 Performance dependence on candidate list length

(This experiment was carried out for \( m = 18, n = 25, \epsilon = 3, \sigma = \) variable, filters = All, termination = Ib, algorithm = 2.5DGH)

A candidate list is assigned to each image basis. During the recognition phase the votes each image basis received are stored in its candidate list. When the voting ends, we verify the top \( N_c \) scorers from the candidate list. We tested the effect of the number of matches being verified for each image basis on the success rate and runtime. As
Figure 5.5: Comparing Score bounded termination vs. Iteration bounded termination. Note that the range of the success rate axis is small.

expected, the more bases being verified the higher the success rate was, especially when the depth of the object increases. However, performing $N_c$ times more verifications effects the throughput (figure 5.6).

Figure 5.6: Candidate list length effects on success rate and runtime

5.3.7 2.5D Geometric Hashing vs. Alignment

(This experiment was carried out for $m = 18$, $n = 25$, $\epsilon = 3$, $\sigma = \text{variable}$, filters = All, termination = Ib, algorithm = 2.5DGH/A)

The common method of recognizing 3D Object Recognition from 2D images is called Alignment [8]. The Alignment algorithm randomly match model points with their image
representation and calculate the transformation between them. It continues with verifying it, by checking the consistency with the rest of the points. This experiment shows that our approach is more efficient with respect to Alignment. The complexity of the Alignment algorithm is vast since the number of possibilities is huge (for an $m$ point model and $n$ point image, it is $\binom{n}{3}\binom{m}{3}! \approx n^3m^3$). Moreover, it is difficult to decide whether the match was correct since numerous matches are eligible.

A straightforward comparison with our 2.5D Geometric Hashing proved to be inappropriate since our approach uses a statistical computation to determine the amount of experiments required to achieve the desired success rate. When using the Alignment algorithm as is, it runs through the entire possible image bases triplets, thus consuming a large amount of time and is irrelevant for any practical use. Therefore, we applied the same statistical computation on the Alignment algorithm, so both iterate on the same number of image bases. Still, although now the Alignment execution time took approximately 100 seconds, the success rate achieved was unsatisfactory. In order to even up the odds, we added filtration on the model bases in the Alignment algorithm. Since any affine coordinate based filtration is not appropriate, we applied the basis size and basis angle filters. The improvement in the success rate achieved by the filtration can be seen in Figure 5.7, and we match it against the 2.5D Geometric Hashing.

Since not all filters are applicable the success rate achieved by the Alignment is lower than the one achieved by the 2.5D Geometric Hashing algorithm. However, these results coincide with the ones achieved by the later when not all filters were in use (see experiment 5.3.3). As for throughput, the Alignment algorithm consumes more time since we run the verification procedure for each match and not only for the top matches in the candidate list.
Figure 5.7: Comparing 2.5D Geometric Hashing With Alignment
Chapter 6

Real Images Experiments

6.1 Data Generation

6.1.1 Non-Synthetic Model Acquisition

For non-synthetic model acquisition we used a range camera. It is based on projecting a pattern on the object surface and extracting the object geometry from the deformations of the pattern. A series of black and white stripes (known as coded light) is projected sequentially on the object. The patterns form a binary code that allows the reconstruction of the angle of each point on the surface with respect to the optical axis of the camera. Then one can compute the depth using triangulation. This produced a topographical-like image of the object, in which each color represent different depth (see Figure 6.1).

We used SUSAN corner detector on the images that were provided by the range camera to construct the model that was used in the hashing procedure and Canny edge detector [2] to add feature points to the model during verification (section 6.2.3)

6.1.2 Non-Synthetic Image Generation

The images were taken using a camera located at a large distance from the object, so that the parallel projection model is appropriate, in a direction making up to 30° angle with the object’s depth axis as specified in the model acquisition phase.

Extracting the feature points from the image was implemented similar to the extraction of the model features (using both detectors).
Figure 6.1: Model produced by a range camera - different depths are represented by different colors. The black regions correspond non-valid depth measurement.

6.2 Algorithm

6.2.1 Pre-processing

For the non-synthetic experiments both model and image feature were extracted directly from the images. The same goes for clutter.

6.2.2 Hashing

The hashing mechanism is the same as for the non-synthetic experiments.

6.2.3 Verification

Non-synthetic object recognition involves handling of a large amount of feature points. Increasing the number of model features improves reliability, however, it also increases the run time, and is not practical for really large detailed model. The common solution which we test is to use a small number of model features (extracted using the SUSAN corner detector) for hypothesizing the match (or equivalently the pose), and to make the final decision using a verification process which uses a much large number of features (edges, extracted by Canny’s operator).
Such a procedure reduces the throughput, making the nearest neighbor search for each projected model point in the image exhaustive. Therefore, for each match we projected the model and created an image-like structure which holds, for each projected point, its nearest neighbors. Then we did a dot-product between our structure and the real image to receive the verification score. This reduced the amount of time of searching for close image point to a constant.

We used $m_o = 56$ model points and $n_o = 1200$ image points in all the experiments except the one in which we show the influence of the amount of points being used in the verification on the success rate. Each experiment’s description relates to the amount of points being used in the recognition stage.

6.3 Results

Our non-synthetic models were a glass and a box models with texture on them (see figure 6.2). The glass model is 20% shallow while the box model is 50% shallow.

![Figure 6.2: Glass & Box Models (18 feature points)](image)

6.3.1 Success rate and run time comparison

(This experiment was carried out for $m = 18$, $n = 50$, filters = All, termination = 1b, algorithm = GH/EGH/2.5DGH)
The first experiment illustrates the difference in reliability and the running time for the proposed algorithms (see table 6.1). Much like the synthetic experiment, the *Extended Geometric Hashing* algorithm proven to be fast and effective enough on a shallow object. The *2.5D Geometric Hashing* algorithm, although time consuming, remains stable. The *original Geometric Hashing* algorithm fails handling a non-flat object.

Figure 6.3: Glass Images (50 image points)
Table 6.1: Success rate and run time comparison

<table>
<thead>
<tr>
<th>Glass Results</th>
<th>GH</th>
<th>EGH</th>
<th>2.5D GH</th>
<th>Box Results</th>
<th>GH</th>
<th>EGH</th>
<th>2.5D GH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Success Rate</td>
<td>63%</td>
<td>94%</td>
<td>98%</td>
<td>Success Rate</td>
<td>52%</td>
<td>91%</td>
<td>96%</td>
</tr>
<tr>
<td>Time (seconds)</td>
<td>3</td>
<td>17</td>
<td>23</td>
<td>Time (seconds)</td>
<td>3</td>
<td>17</td>
<td>24</td>
</tr>
</tbody>
</table>

6.3.2 Performance dependence on signal/clutter ratio

(This experiment was carried out for $m = 18$, $n = 50/100/200$, filters = All, termination = Ib, algorithm = EGH/2.5DGH)

More clutter (non-model projections image points) was added by lowering the corner detector threshold. In this experiment we set the number of image points as $n = 50, 100$ or 200, while the number of model points remain the same ($m = 18$). The more clutter there is, both algorithms’ reliability decreases and runtime increases (see table 6.2).

6.3.3 Performance dependence on model features

(This experiment was carried out for $m = 9, 18, 30$, $n = 50$, filters = All, termination = Ib, algorithm = EGH/2.5DGH)
By increasing the number of the model feature points the hash table becomes denser. Each bin contains a large number of hash entries which are being voted each time it is accessed. However, since the number of image points remains unchanged, the number of image basis selections needed to achieve the desired success rate diminishes. Increasing the model feature points affects the signal/clutter ratio because some of the clutter in the image becomes model points projection originated. In this experiment we set the number of model points as \( m = 9, 18 \) or \( 30 \), while the number of image points remains the same \( (n = 50) \). The more model feature points there are, both algorithms’ reliability
Table 6.2: Performance dependence on signal/clutter ratio

<table>
<thead>
<tr>
<th>Glass Results</th>
<th>Extended Geometric Hashing</th>
<th>2.5D Geometric Hashing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n=50 n=100 n=200</td>
<td>n=50 n=100 n=200</td>
</tr>
<tr>
<td>Success Rate</td>
<td>94% 92% 91%</td>
<td>98% 97% 95%</td>
</tr>
<tr>
<td>Time (seconds)</td>
<td>17 29 54</td>
<td>23 43 76</td>
</tr>
<tr>
<td>Box Results</td>
<td>Extended Geometric Hashing</td>
<td>2.5D Geometric Hashing</td>
</tr>
<tr>
<td></td>
<td>n=50 n=100 n=200</td>
<td>n=50 n=100 n=200</td>
</tr>
<tr>
<td>Success Rate</td>
<td>91% 88% 83%</td>
<td>96% 94% 91%</td>
</tr>
<tr>
<td>Time (seconds)</td>
<td>17 30 55</td>
<td>24 43 78</td>
</tr>
</tbody>
</table>

and runtime increase (see table 6.3).

Table 6.3: Performance dependence on model features

<table>
<thead>
<tr>
<th>Glass Results</th>
<th>Extended Geometric Hashing</th>
<th>2.5D Geometric Hashing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>m=9 m=18 m=30</td>
<td>m=9 m=18 m=30</td>
</tr>
<tr>
<td>Success Rate</td>
<td>93% 94% 95%</td>
<td>95% 98% 98%</td>
</tr>
<tr>
<td>Time (seconds)</td>
<td>15 18 23</td>
<td>18 23 30</td>
</tr>
<tr>
<td>Box Results</td>
<td>Extended Geometric Hashing</td>
<td>2.5D Geometric Hashing</td>
</tr>
<tr>
<td></td>
<td>m=9 m=18 m=30</td>
<td>m=9 m=18 m=30</td>
</tr>
<tr>
<td>Success Rate</td>
<td>89% 91% 92%</td>
<td>90% 96% 97%</td>
</tr>
<tr>
<td>Time (seconds)</td>
<td>14 18 23</td>
<td>19 24 32</td>
</tr>
</tbody>
</table>

6.3.4 Performance dependence on Occlusion

(This experiment was carried out for $m = 18$, $n = 50$, filters = All, termination = 1b, algorithm = EGH/2.5DGH)

This experiment tests the durability of the Extended Geometric Hashing and the 2.5D Geometric Hashing to occlusion. In the first two images we used, the object was placed so that part of it was occluded, $1/6$ of the feature points in the first image and $1/3$ in the second (see figure 6.3). In both algorithms the success rate decreases as the occlusion is raised. However the 2.5D Geometric Hashing is less influenced by occlusion since it relays on the geometric consistencies besides the number of voters (see table 6.4).
Figure 6.6: Models’ features dependency (9/18/30 feature points)

6.3.5 Performance dependence on scaling

(This experiment was carried out for $m = 18$, $n = 200/366$, filters = All, termination = Ib, algorithm = EGH/2.5DGH)
Table 6.4: Performance dependence on Occlusion

<table>
<thead>
<tr>
<th>Success Rate</th>
<th>Extended Geometric Hashing</th>
<th>2.5D Geometric Hashing</th>
</tr>
</thead>
<tbody>
<tr>
<td>No occlusion</td>
<td>92%</td>
<td>97%</td>
</tr>
<tr>
<td>1/6 of features occluded</td>
<td>89%</td>
<td>95%</td>
</tr>
<tr>
<td>1/3 of features occluded</td>
<td>82%</td>
<td>91%</td>
</tr>
</tbody>
</table>

We examined the algorithm performance on image in which the projected model occupied 10% of the image area. From the results we can see that although the signal/clutter ratio remains almost as before (see Experiment 6.3.2) the success rate is lower since a substantial part of the projected model image points remained undetected and the error regions created by actual model related bases (which are small), were large resulting in many false votes (see table 6.5).

Figure 6.7: Down scaled object (200/366 feature points)

Table 6.5: Performance dependence on scaling

<table>
<thead>
<tr>
<th></th>
<th>Extended Geometric Hashing</th>
<th>2.5D Geometric Hashing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>n=200</td>
<td>n=366</td>
</tr>
<tr>
<td>Success Rate</td>
<td>87%</td>
<td>90%</td>
</tr>
<tr>
<td>Time (seconds)</td>
<td>54</td>
<td>92</td>
</tr>
<tr>
<td></td>
<td>n=200</td>
<td>n=366</td>
</tr>
<tr>
<td>Success Rate</td>
<td>91%</td>
<td>93%</td>
</tr>
<tr>
<td>Time (seconds)</td>
<td>73</td>
<td>115</td>
</tr>
</tbody>
</table>

6.3.6 Performance dependence on verification’s image and model features

(This experiment was carried out for $m = 18$, $n = 50$, filters = All, termination = 1b, algorithm = EGH/2.5DGH)
This experiment illustrates the effects of the amount of additional points \((m_v, n_v)\), added for both model and image in the verification step of non-synthetic objects, on the success rate and the running time for the proposed algorithms. These points were extracted using the *Canny* edge detector with a different threshold (see table 6.6).

<table>
<thead>
<tr>
<th>Success Rate</th>
<th>92%</th>
<th>94%</th>
<th>95%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time (seconds)</td>
<td>8</td>
<td>17</td>
<td>39</td>
</tr>
</tbody>
</table>

Table 6.6: Performance dependence on verification’s image and model features
Chapter 7

Conclusions and Future Work

In this thesis we proposed two extensions of the Geometric Hashing algorithm that enable it to recognize objects which are not necessarily flat. The two algorithm enable a choice between a more reliable algorithm and a faster one. Their performance was evaluated using simulation and real data experiments.

Our extensions present two major enhancements to the traditional GH algorithm: (a) taking into account the object’s non-flatness in calculating the range for the variation of the affine coordinates (b) using basis filtering to remove bases which are not likely to be good. While the Extended Geometric Hashing algorithm uses a general bound on the location error, resulted by the affine coordinates variation, the 2.5 Geometric Hashing uses a more detailed prediction, which specifies a consistency constraint.

We evaluated the performance of these algorithms using simulation and real data experiments. The 2.5 Geometric Hashing reaches a success rate of above 90% even when object’s non-flatness increases. Its reliability is kept at high 94% values, even when the clutter was three times larger. For shallow objects the Extended Geometric Hashing algorithm gives the same reliability but is faster, and is therefore preferred.

The original GH algorithm as well as our extensions, represent the image and the model as a set of points (in 2D and 3D respectively). The only characteristics of these points are their location. As such, the model point corresponding to some image point cannot be identified by itself and should rely on its location relative to other points.

Recently however, several methods for characterizing feature points, extracted from real images, were proposed and shown to be highly effective [10, 11].

Such algorithms, in which feature points are selected not based on their location
only but on additional attributes, have the potential to dramatically improve the success rate when used in our suggested algorithms. Therefore the advantage of the proposed algorithms is mostly when such local characterizations cannot be applied.

As described here the algorithms are directly applicable for limited (but not necessarily small) viewing angle. Their applications to general 3D objects from an arbitrary point of view, is straightforward as well: viewing a 3D object we see only its non-occluded part. A large fraction of this non-occluded part can usually be considered as a shallow object (as defined here). Therefore the object’s surface should be divided into overlapping shallow sub-surfaces, which should serve as partial models to the object. Note that all of them may be represented together in the hash table. The design of such representation is left for future work.
Appendix A

Affine coordinates properties

Affine Coordinates are unique and invariant under weak perspective transformation.

Let \( p_0, p_1, \) and \( p_2 \) be three non-collinear points in 2D space, then \( (p_1 - p_0, p_2 - p_0) \) are two non-collinear vectors which span the 2D space. Consider any other point \( p \). This point can be represented using these two vectors as a basis. Let \( C \) be the coefficient matrix of the following coupled, linear algebraic equations set:

\[
\alpha(p_1^p - p_0^p) + \beta(p_2^p - p_0^p) = p - p_0^p \\
\alpha(p_1^p - p_0^p) + \beta(p_2^p - p_0^p) = p - p_0^p
\]

The linear set has a unique exact solution iff \( \det C \neq 0 \);

\[
\det C = \begin{vmatrix}
 p_1^p - p_0^p & p_2^p - p_0^p \\
 p_1^p - p_0^p & p_2^p - p_0^p
\end{vmatrix}
\]

given that the points are non-collinear, i.e.,

\[
\frac{p_1^p - p_0^p}{p_1^p - p_0^p} \neq \frac{p_2^p - p_0^p}{p_2^p - p_0^p}
\]

therefore, \( \det C \neq 0 \). Being unique Affine Coordinates can be used as invariants for affine transformation.

Consider an image undergoing an affine transformation. Let \( p_0, p_1, p_2, \) and \( p \) be a set of non-collinear points in that image, and let \( p_0', p_1', p_2', \) and \( p' \) be a set of points in another image obtained by an affine transformation, \( T = Ax + b \) where \( A \) is a general matrix, of
it (see Figure A.1) then:

\[ p' = Ap + b \]
\[ = A(p_0 + \alpha(p_1 - p_0) + \beta(p_2 - p_0)) + b \]
\[ = Ap_0 + b + \alpha(Ap_1 - Ap_0) + \beta(Ap_2 - Ap_0) \]
\[ = Ap_0 + b + \alpha(Ap_1 + b - (Ap_0 + b)) + \beta(Ap_2 + b - (Ap_0 + b)) \]
\[ = p'_0 + \alpha(p'_1 - p'_0) + \beta(p'_2 - p'_0) \]

that is, the coordinates of point \( p' \), with respect to the transformed basis \( p'_0, p'_1, \) and \( p'_2 \), will remain \((\alpha, \beta)\). This is the affine coordinate invariant property.

![Figure A.1: System Undergoing an Affine Transformation.](image)

The affine coordinate \( \alpha, \beta \) have an interesting geometric interpretation: their sizes depend on the sizes of the two triangles \( P \) creates with the basis’ points [3] (see figure A.2), and can be expressed as:

\[ |\alpha| = \frac{S \triangle P_0P_2P}{S \triangle P_0P_1P_2}, \quad |\beta| = \frac{S \triangle P_0P_1P}{S \triangle P_0P_1P_2}. \]  

(A.1)

Consequently, basis triplets which yield a small area relative to the triangles areas, formed by the basis points and the fourth model point, will result in a large affine coordinate and should not be used as bases [3].
Figure A.2: Basis Size Filter (adapted from [3])
Appendix B

Analyzing the range of affine coordinates associated with image errors

A correct voting in the Geometric Hashing algorithm requires an understanding of the effect of sensor’s uncertainty on the \((\alpha, \beta)\) coordinates. The following claim, which is an extension of the result proved in [6], specifies the uncertainty in the affine coordinates.

Claim. Consider an image basis \((p_0, p_1, p_2)\) and a fourth point \(p\). Let \(q = (q_0, q_1, q_2)\) be the noisy measurement of \(p = (p_0, p_1, p_2)\), satisfying \(\|p - q\| \leq \varepsilon\). Then, unless the basis is very small or nearly collinear, the range of affine coordinates of \(p\) relative to the basis \(p_0, p_1, p_2\) is, an ellipse.

Proof. By Claim 2 the Euclidean distance from the noisy point \(q\) to the point \(q^*\), specified by the true affine coordinates and the noisy basis, is upper bounded by

\[
r = 2\varepsilon(1 + |\alpha'| + |\beta'|) + \varepsilon_{non-planarity}
\]

Therefore, analyzing the range of values for the true affine coordinates \((\alpha', \beta')\) may be done by solving

\[
\|q - q^*\| < r,
\]

where

\[
q^* = q_0 + \alpha'(q_1 - q_0) + \beta'(q_2 - q_0)
\]
and

\[ q = q_0 + \alpha (q_1 - q_0) + \beta (q_2 - q_0). \]

Let \( u = q_1 - q_0 \) and \( v = q_2 - q_0 \) be the vectors specifying the image basis coordinate system and let \( \psi \) be the angle between them. Then, the boundary of the uncertainty region in the \((\alpha', \beta')\) plane is specified by

\[ r^2 = [(\alpha - \alpha')u]^2 + 2(\beta - \beta')(\alpha - \alpha')uv \cos \psi + [(\beta - \beta')v]^2. \quad (\text{B.1}) \]

That is, it is a conic in the \((\alpha', \beta')\) plane. Expanding it, we get

\[ a_{11}(\alpha')^2 + 2a_{12}\alpha'\beta' + a_{22}(\beta')^2 + 2a_{13}\alpha' + 2a_{23}\beta' + a_{33} = 0 \quad \text{(B.2)} \]

where

\[
\begin{align*}
a_{11} &= u^2 - 4\epsilon^2 \\
a_{22} &= v^2 - 4\epsilon^2 \\
a_{12} &= uv \cos \psi - 4s_\alpha s_\beta \epsilon^2 \\
a_{13} &= -u [\alpha u + \beta v \cos \psi] - 4s_\alpha \epsilon^2 - 2s_\alpha \epsilon_{\text{non-planarit}} \\
a_{23} &= -v [\beta v + \alpha u \cos \psi] - 4s_\beta \epsilon^2 - 2s_\beta \epsilon_{\text{non-planarit}} \\
a_{33} &= u^2 \alpha^2 + 2 \alpha \beta uv \cos \psi + v^2 \beta^2 - 4\epsilon^2 - 4\epsilon_{\text{non-planarit}} - 4\epsilon_{\text{non-planarit}}^2
\end{align*}
\]

where \( s_\alpha \) denotes the sign of \( \alpha \) \((s_0 = 1)\). The three quantities:

\[
I = a_{11} + a_{22} \\
J = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} \\
\Delta = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}
\]

with \( a_{ik} = a_{ki} \((i,k = 1, 2, 3)\), are invariants with respect to translation and rotation transformation \([5]\).

If \( \Delta \neq 0, J > 0 \) and \( \frac{J}{\Delta} < 0 \) then the conic defined by equation B.1 is an ellipse. For
the conic (B.2), the three quantities are:

\[ I = u^2 + v^2 - 8 \epsilon^2 \]
\[ J = u^2 v^2 \sin^2 \psi - 4 \epsilon^2 (u^2 - 2uv s_\alpha s_\beta \cos \psi + v^2) \]
\[ \Delta = -4u^2 v^2 \sin^2 \psi \epsilon^2 (1 + s_\alpha \alpha + s_\beta \beta)^2 - 4u^2 v^2 \sin^2 \psi \epsilon \epsilon_{\text{non-planarity}} (1 + s_\alpha \alpha + s_\beta \beta) \]
\[ + 12 \epsilon^2 \epsilon_{\text{non-planarity}} (u^2 - 2uv s_\alpha s_\beta \cos \psi + v^2) - 4u^2 v^2 \sin^2 \psi \epsilon_{\text{non-planarity}}^2 \]

Note that if \( J > 0 \), then

\[ \Delta < -4u^2 v^2 \sin^2 \psi \epsilon^2 (1 + s_\alpha \alpha + s_\beta \beta)^2 - 4u^2 v^2 \sin^2 \psi \epsilon \epsilon_{\text{non-planarity}} (1 + s_\alpha \alpha + s_\beta \beta) \]
\[ + 3u^2 v^2 \sin^2 \psi \epsilon_{\text{non-planarity}}^2 - 4u^2 v^2 \sin^2 \psi \epsilon_{\text{non-planarity}} \]
\[ = -u^2 v^2 \sin^2 \psi [4 \epsilon^2 (1 + s_\alpha \alpha + s_\beta \beta)^2 + 4 \epsilon \epsilon_{\text{non-planarity}} (1 + s_\alpha \alpha + s_\beta \beta) + \epsilon_{\text{non-planarity}}^2] \]
\[ = -u^2 v^2 \sin^2 \psi [2 \epsilon (1 + s_\alpha \alpha + s_\beta \beta) + \epsilon_{\text{non-planarity}}]^2 < 0 \]

Therefore, if \( u^2 + v^2 > 8 \epsilon^2 \) and \( J > 0 \), then the conic specified by equation B.2 is an ellipse. These conditions are not met only when the image basis points are very close or nearly collinear. Such bases yield high affine coordinate values and high sensitivity to measurement uncertainty. They are avoided (filtered out) as part of the filtering procedure.

We can now compute the invariant characteristics of the ellipse B.2.

The area of the ellipse is given by

\[ \pi [4u^2 v^2 \sin^2 \psi \epsilon^2 (1 + s_\alpha \alpha + s_\beta \beta)^2 - 4u^2 v^2 \sin^2 \psi \epsilon \epsilon_{\text{non-planarity}} (1 + s_\alpha \alpha + s_\beta \beta) \]
\[ + 12 \epsilon^2 \epsilon_{\text{non-planarity}} (u^2 - 2uv s_\alpha s_\beta \cos \psi + v^2) - 4u^2 v^2 \sin^2 \psi \epsilon_{\text{non-planarity}}^2 \]
\[ / [u^2 v^2 \sin^2 \psi - 4 \epsilon^2 (u^2 - 2uv s_\alpha s_\beta \cos \psi + v^2)]^{3/2} \] (B.3)

The center of the ellipse is at

\[ x_0 = -1 \left( \frac{\begin{vmatrix} a_{13} & a_{12} \\ a_{23} & a_{22} \end{vmatrix}}{J} \right) \]
\[ = -1 \left( \begin{vmatrix} a_{13} & a_{12} \\ a_{23} & a_{22} \end{vmatrix} \right) \]
\[ = -\frac{1}{J} [-\alpha \alpha^2 v^2 \sin^2 \psi + 4 \epsilon (\alpha \alpha^2 - s_\alpha (1 + \beta s_\beta) v^2 + uv \cos \psi (\beta - \alpha s_\alpha s_\beta + s_\beta)) \]
\[ + 12 \epsilon \epsilon_{\text{non-planarity}} (-v^2 s_\alpha + uv \cos \psi s_\beta)] \]
\[
y_0 = -\frac{1}{T} \begin{bmatrix}
  a_{11} & a_{13} \\
  a_{21} & a_{23}
\end{bmatrix}
\]
\[
= -\frac{1}{T}\left[-\beta u^2 v^2 \sin^2 \psi + 4\epsilon^2 (\beta v^2 - s_\beta (1 + \alpha s_\alpha) u^2 + u v \cos \psi (\alpha - \beta s_\alpha s_\beta + s_\alpha))
\right.
\]
\[
+ 2\epsilon \epsilon_{\text{non-planarity}} (-u^2 s_\beta + u v \cos \psi s_\alpha)]
\]

and the orientation is given by

\[
\tan 2\Phi = \frac{2a_{12}}{a_{11} - a_{22}}
\]
\[
= \frac{2(u v \cos \psi - 4s_\alpha s_\beta \epsilon^2)}{u^2 - v^2}.
\]

Note that with zero noise, the center of the ellipse is in the measured affine coordinates \((\alpha, \beta)\). Moreover, if the measure error \(\epsilon\) is small relative to the basis size the center of the ellipse remains close to \((\alpha, \beta)\). Therefore, by selecting large bases the location of the center of the ellipse becomes less sensitive to noise.

\(\square\)
Appendix C

Determining the required number of image basis selections

The required number of image basis selections determines the number of model-image matches we’ll check during the phase. When that number is reached (with the exception of using a threshold stop) the highest scored match is selected and the transformation between the model and the image is calculated according to it. Each selection consists of a randomized image basis. The algorithm chooses image bases and for each remaining image point votes for model bases stored in the hash table.

We compute the required number of bases selections needed to achieve the desired success rate as follows: Let \( k \) be the required number of basis selections needed and let \( n \), \( m \) be the number of image and model points respectively (clearly, we take into account only the model’s points that were not disqualified by filtration in the preprocessing phase). The probability that at least one correct image basis is chosen in \( k' \) random basis selections is:

\[
1 - \left(1 - \frac{m^3}{n}\right)^{k'}
\]

We want that this probability to exceed a desired success rate \((1 - \delta)\), where \( \delta \) is the probability for failure:

\[
1 - \left(1 - \frac{m^3}{n}\right)^{k'} > 1 - \delta
\]

\[
k' \log \left(1 - \frac{m^3}{n}\right) < \log \delta
\]

\[
k' > \frac{\log \delta}{\log (1 - \frac{m^3}{n})},
\]
using $\log(1 - \epsilon) \approx \epsilon$, the number of attempts needed to achieve the desired success rate is:

$$k' > -\log \delta \left( \frac{n}{m} \right)^{3}.$$  

Taking this number of bases guarantees with probability $(1 - \delta)$ that at least one correct image basis is selected. However, this does not assure us that this basis will receive the highest score by the algorithm. Hence, we use an empirically calculated probability $Pr_A$ stands for the probability for a correct basis to receive the highest score. The required number of matches needed is $k = Pr_A k'$. Note, we also take into account that some model's points were disqualified by filtration in the preprocessing phase by using $m = m - \# disqualified points$. 
Appendix D

Alter’s Pose Estimation Method

When the candidate list was constructed we verify each of its elements using Alter’s geometric pose estimation method [1]. Alter presents a method to compute the pose of a model from three corresponding point under weak-perspective projection. Figure D.1 shows the three model points $m_0, m_1$, and $m_2$ being projected orthographically to the plane that contains $m_0$ and is parallel to the image plane, and scaled down by scale factor $s$ into the image. All that is pertinent to recovering the 3D pose of the model are the distances between the model and image points.

Let the distance between model points be $(R_{01}, R_{02}, R_{12})$, and the corresponding distance between the image points be $(d_{01}, d_{02}, d_{12})$. Also let

\[
\begin{align*}
\alpha &= (R_{01} + R_{02} + R_{12})(-R_{01} + R_{02} + R_{12})(R_{01} - R_{02} + R_{12})(R_{01} + R_{02} - R_{12}) \\
\beta &= d_{01}^2(-R_{01}^2 + R_{02}^2 + R_{12}^2) + d_{02}^2(R_{01}^2 - R_{02}^2 + R_{12}^2) + d_{12}^2(R_{01}^2 + R_{02}^2 - R_{12}^2) \\
\gamma &= (d_{01} + d_{02} + d_{12})(-d_{01} + d_{02} + d_{12})(d_{01} - d_{02} + d_{12})(d_{01} + d_{02} - d_{12}) \\
\sigma &= \begin{cases} 
1 & \text{if } d_{01}^2 + d_{02}^2 + d_{12}^2 \leq s^2(R_{01}^2 + R_{02}^2 + R_{12}^2) \\
-1 & \text{otherwise}
\end{cases}
\end{align*}
\]

Then the unknown parameters in Figure D.1 are

\[
\begin{align*}
s &= \sqrt{\frac{b + \sqrt{b^2 - ac}}{a}} \\
(h_1, h_2) &= \pm \left(\sqrt{(s R_{01})^2 - d_{01}^2}, \sigma \sqrt{(s R_{02})^2 - d_{02}^2}\right) \\
(H_1, H_2) &= \frac{1}{s}(h_1, h_2)
\end{align*}
\]

(D.1)
Figure D.1: Model points \(m_0, m_1,\) and \(m_2\) undergoing orthographic projection plus scale to produce image points \(i_0, i_1,\) and \(i_2\) (adapted from \([1]\))

For each image of three points there are two poses which yield that image. Equation D.1 yields two pairs of values for \(h_1\) and \(h_2\) which corresponds to those two poses. Given image points \(i_0 = (x_0, y_0), i_1 = (x_1, y_1),\) and \(i_2 = (x_2, y_2),\) the 3D location of the model points in camera-centered coordinates are:

\[
\begin{align*}
m_0 &= \frac{1}{s} (x_0, y_0, w) \\
m_1 &= \frac{1}{s} (x_1, y_1, h_1 + w) \\
m_2 &= \frac{1}{s} (x_2, y_2, h_2 + w),
\end{align*}
\]

where \(w\) could be any number.
Using the above allows us to express the position of any model point in the image using the 3D affine coordinates and \((H_1, H_2)\). Given a model point \(m_3\),

\[
m_3 = \alpha (m_1 - m_0) + \beta (m_2 - m_0) + \gamma (m_1 - m_0) \times (m_2 - m_0) + m_0,
\]

the image location of \(m_3\) is

\[
\begin{align*}
(\alpha (x_1 - x_0) + \beta (x_2 - x_0)) + \\
\gamma ((y_1 - y_0)H_2 - (y_2 - y_0)H_1) + x_0, \\
\alpha (y_1 - y_0) + \beta (y_2 - y_0) + \\
\gamma ((x_1 - x_0)H_2 + (x_2 - x_0)H_1) + y_0.
\end{align*}
\]
Bibliography


גילה גטיין

יזורזזיעזימע הערבות לאלגוריתמים ורבדול היגיאומטרית
지요 עכמה בערות החבחת לאלוגיתם הערבל הגיאומטריה

總的而於紀錄

לשם המילוי חלקי של הדרישות לכבדת התдоров
מגיסטר למידעём במנועי המושב

יגל גטני

הנה לכנס החככינים - מכון טכנולוגי לישראל
נכנס תשלימה בחיפה אפריל 2008

Technion - Computer Science Department - M.Sc. Thesis MSC-2008-11 - 2008
המҳErrorResponse}, מיכאל לינדנברג ודרי על שימשנים
בפользоватל למודי המסהב.

אני מודה לטכניון על התמיכת הכספית והדיבת בחש聯絡ית.
תוכן העניינים

1
tкупיר

2
רשימת קצורים

3
מקודם

2.1
מודול ייעוץ תומנה

2.2
קואורדינטאות אפיניות ואוטוונרואטימיות

2.2.1
קואורדינטאות אפיניות

2.2.2
קואורדינטאות אפיניות תלת-ממדיות

2.3
אלגוריתמיםعرفוליםיגהואוטיים

2.4
עיצמים "רדידים"

3
אלגוריתמיםعرفולגואומטרים多项億 עצמים לא שטח

3.1
תহוש הקואורדינטאות האפיניות והמשויכות לקודד תהלת

3.2
סיפיסים "╁╊.FloatTensor" "╉▂商用"

3.3
אלגוריתמיםعرفוליםיגהואומטרים多项億

3.4
תוראות

4
אלגוריתמיםعرفולגואומטרים多项億 מיומד

4.1
מקבץ

4.2
מקבלי עבקיםעל舲וניאתתבניתמודסרמייהוים

4.3
משימהיעיללבדוקאקצבתformularioאלגוריתמסיכרה

4.4
יזוגעיצמיםתלת-ממדייםלאsword 우리의

4.5
הערוליםификаואוטיים多项億עיצמע
תוכן העניינים (המשך)

5 ניסיון עם תילונים סינתטיים
5.1 יצרת התילונים
5.1.1 יצרת מודל סינתטי
5.2 הפکת תילונים סינתטי
5.2.1 ערבול
5.3 אלגוריתם
5.3.1 שיווק התילונים וחברות
5.3.2 בניגוד לתילונים חיסול ואוזן / ער
5.3.3 בניגוד לתילונים חיבור
5.3.4 בניגוד לתילונים בידידות
5.3.5 בניגוד לתילונים דיבור
5.3.6 בניגוד לתילונים בידידות
5.3.7 בניגוד לתילונים דיבור
6 ניסיון בחומוט מאמית
6.1 יצרת התילונים
6.1.1 חישת מודל אמתי
6.2 הפקת תילונים אמתי
6.2.1 ערבול
6.2.2 שיווק התילונים וחברות
6.3 אלגוריתם
6.3.1 שיווק התילונים וחברות
6.3.2 בניגוד לתילונים חיסול ואוזן / ער
6.3.3 בניגוד לתילונים חיבור
6.3.4 בניגוד לתילונים בידידות
6.3.5 בניגוד לתילונים דיבור
6.3.6 בניגוד לתילונים בידידות
7 דיוון והסметות מסקנות

Technion - Computer Science Department - M.Sc. Thesis MSC-2008-11 - 2008
<table>
<thead>
<tr>
<th>מספר</th>
<th>תוכן</th>
</tr>
</thead>
<tbody>
<tr>
<td>55</td>
<td>A: תכונות הקואורדינטות האפיגות</td>
</tr>
<tr>
<td>59</td>
<td>B: גישות חונות הקואורדינטות האפיגות המשויכות לאות שירות</td>
</tr>
<tr>
<td>63</td>
<td>C: קבוצת מספר בסיים התמונה הנדרשת ליזום</td>
</tr>
<tr>
<td>65</td>
<td>D: שיטה מצוינת של אולטיר</td>
</tr>
<tr>
<td>68</td>
<td>רשימה מקורות</td>
</tr>
</tbody>
</table>
רשימת הערבים

2.1 כל הקודה ש לע המישור היינודי ליוגנ על ידי ביסיס המורכב

2.2 מנוון פִון בּוסינ

1.1 ענן אל פּוינון בּוסינ

3. בחירהENCE תשב טנקב והוספת התוכנה הלאשה "קרדיט" הזונות התוכנות שותקות

4. חלק וניהם מעוצב נושה "קרדיט" הזונות התוכנות שותקות

3.1 סימן על פון הזונות הכנס: ביסיס אל רופים והאל בוח

17. קורס לע זון הנסיך: "אפים דהומניטי" הזונות הנסיך

4.1 בדיקת עבוקת נספח: זון הוזה והוקטור

4.2 סדרון ביצוע הזונות הקטניא: סריקה במשרור

4.3 סדרון מברוק למיס ערמות: הקדמונות של סריקה

4.4 חישוב עבוקית

4.5 מותיאת סריקה באמר בדיקת עבוקית

4.6 בניית הורית והוספת סיבר חלב מהמודל על צדיל פלוריאל

5.1 השעון בך הארגונית ת-2.5 ממידי לבן המוכפל

5.2 השפעת יח צאצא/רחש על ית חלצהלות ומימirates

5.3 השפעת חסין על ית חלצהלות ומימirates

5.4 השפעת בדיקת הענקוב על ית חלצהלות ומימirates

5.5 השעון בך קריטריו סים מובנים צו לבן קריטרו

5.6 ביסיס מספור תורות קודוב

5.7 השפעת איטר רשם מעמדים אילמיד על ית חלצהלות

5.8 יימרי

6.1 מודל המטר見る ממולכת מרחק – האביגת חונס

6.2 ניסיונות את עונם של יוזמה: הארגונים השחרורים מציינו ממידת

6.3 גמישלא עمهرجان בר הנסיך (18 קורדונים מיצוגים)

6.4 מודל חלצהלות הספל (50 קורדונים מיצוגים)

6.5 מודל חלצהלות הנספל (50 קורדונים מיצוגים)

6.6 איניסים מוקטן (200/366 קורדון מיצוג)

6.7 איניסים מוקטן (200/366 קורדון מיצוג)

A.1 סינון מבוסס ג Deadpool

A.2 סינון מבוסס Deadpool

D.1 מודל מריך י信息技术 רכיבת סיסמה קודידת MO,Μ1,Μ2

D.2 מודל מריך י信息技术 רכיבת סיסמה קודידתMO,Μ1,Μ2
## רצימת הטבלאות

<table>
<thead>
<tr>
<th>אלגוריתם סירקה במיוחד: מלבגוס חפיפה</th>
<th>4.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>השואאת ייח העלאה זמני ייצוג</td>
<td>6.1</td>
</tr>
<tr>
<td>בט HWND כתר חית ביחס אזות/ראש</td>
<td>6.2</td>
</tr>
<tr>
<td>בט HWND כתר במנוע שקודם ו البعיד מודל</td>
<td>6.3</td>
</tr>
<tr>
<td>בט(HWND כתר בחרת)</td>
<td>6.4</td>
</tr>
<tr>
<td>בט(HWND כתר בחרת / הקטנה)</td>
<td>6.5</td>
</tr>
<tr>
<td>בט(HWND כתר במנוע שקודם ו البعיד מודל / תופעות הפעלה)</td>
<td>6.6</td>
</tr>
<tr>
<td>אלימינציה</td>
<td>52</td>
</tr>
</tbody>
</table>

Technion - Computer Science Department - M.Sc. Thesis  MSC-2008-11 - 2008
תקציר

зорחי עמות בתמונהشهر מתמוסס רבד של יישומי מעשיים בחרויות ריב.

משולחת זו רוחי ביטיה של מיקום על העתקי אלב בת HCI הפיקח התמונה.

בתיקון זה העתקי עזרו ترامפרספקטרלת מושפע משלומי חורואא פרפרטיר המצלמה.

התחלה. התמונה מסתיימת бюджет לחית התצוגה של מספי האינפתי של עצמים.

зорחי מתמוסס מתמוסס מציינת נוכחות של עצמ (מודל) יהודי מרחב התמונה.

משולחת זו ביכר כל מרכבות משני שלבים: אי שלב הריכוז ובוינו את ספיטית המודלים או התמונותلاحグループ מקבמה. ב. שלב ההיבים, בוינו את ספיטית המעם התמונה והלומד קביעה את מיקומם של המודלים הזוהים.

אלגוריתם "הסרבללה האימות" Geometric Hashing – העיון שלを持פי הסוריתון לחוזה

말וס מתמוסס האלגוריתם עיגל בך שהו מإزالة ואמק החית צו פרט מוע תומד

בתמונה Trầnו מזרם דו-ממדי אלגוריתם מוטס על הקואדינטים

אפקיטו השכונתנית אשר מחלאת את הצורעד עצם דו-ממדי יצר träպונטつき המתר.

ועוזדו היא הרחבת האלגוריתם כ長い עצמים שיים להזון דו-ממדי.

גישות היא_sampler על עיגול האלגוריתם המוקדם במקביל לאפרוע זורי עצמים

בורגנות שונת של עצמים. אמט מרהכולות כי התחלת של העתק למידה התמונה היא

(קורות החלפת פרפסקורטסיביים ב mulher לא מגד הגד מועש תומד

המצלה חית על מוקדים). ישכ כ. אם מרצים שני אלגוריתמים: המגדל

המודלים או יכלות להזהדום עם (2.5D GH) 2.5-ה מיימי (Extended GH)

ערוצני שיאים דו-ממדיים בא mongעות ניסיון ונתונים מטריים踒יוסים.

אלגוריתם h ה מתמוסס על הקואדינטים האפינית. בהינתן שלוש קודות

אלגוריתם h מתמוסס על הקואדינטים האפינית. בהינתן שלוש קודות פאunist ה

$\alpha,\beta$ - ו

$\alpha(p_2 - p_0) + \beta(p_1 - p_0)$

הקודינים האפינית של p בסיסו זה. התכתות על הקואדינטים האפינית

אותו שיאי מושнатות את בוערה תרגספורמצית אפינית. איגוריתטיות זה

אותו הבסיסי האלגוריתם.
האלגוריתמים מורכבים מעניינים:

א. שלב תוריםapeutics, בו מחוללים את הקודדים המאפיינים את המודל והורכים השרטטים בשיסים, לכל ביס מחשבים את הקואורדינטאות המאפיינים עבר הקודדים המעוררים, עבר כל קוד מקודס בבובות ערבול (hash) את המודל.

הבטיס בוהוט את הקואורדינטאות של הקודדים מהובא בבחס.

ב. שלב היזור, בו מחוללים את הקודדים המאפיינים המחלקות המностעג, בוחרים בבטיס שלושה עבעים מחשבים קואורדינטאות אפיין של הקודוד המורדים, נגיסים לצה המטמאים בּבל ת he - עבך כל קוד קואורדינטאות, מרצבים את ביטס המחלקות המתחנים ובמותעותimas תקחים את ביטס מחומר השיקולים ב ims. (verification) של האמונות המרביית הבולKHRי (הש正しい) ללשבר לפי את השינוע האנטרופיר (Hamsters יונק בערב הבחריל החלקה).

היאוניטיטים מצטיינים שאכNonNullים במורס ניסן שות הקולות שמס תליין של הקודודים המודל. הם בוחנו את ביטס המתחים במיקומי ( CONCAT) והשלב לפי את השנייה האנטרופי (Hamsters יונק בערב הבחריל החלקה).

האימונים התבצעו על ידי חיבוב הטרנספורמציה של ביטס התהלות לסיס המודל, הלולו השערועת התהלות ית לקודוד המודל לתחלוף על פי הטרנספורמציה שהושבה.

בידיקה הימיוותות של קודוד המתחים בשיבוב הקודודים המודלים.

האלגוריתמים מבוקשים ב שוער. בשילーション מדויקיות ומג稞ית בוז הקודודים ינות השנין הבוער לפי אוזן שיאמה בשל בתום המתרחש בברוקור, המבוקש המקסימלי המורדים את תכנית (shallow) הנטייה懂 תוף שלמיים האנטרופי.

れたופה,ורים ב - אוזן (alpha, beta) הימיוות על ידי טבלת he - את בוז החבל במקודס הקודודים בתום. התרחיבות בכל אוזן.getSharedPreferences את בובות, תועדו למספר את האימיוות

באורי ענישה המתחה עבך כל אוזן שיאמה המתחה הבוער של.

האלגוריתמים הממוכליים מספר על yaşam דימויים ומגتكوיה תחת הרחבון "Clouds Shallow" (Shallow). הבנויים בتصم תולים-מדויים הממקק בתיבת מים. האלגוריתמים המבוקשים מותר עניבת הכתר החבל הבוער בוז kodק או על 40% aoz שיאמה יופים אנימאים וממקק של העבש ביצר דימויים האנטרופי. האלגוריתמים ענישת פוחת או יופים דימויים על העבש

גודל_algorihm צ-2.5 ממידי יופי תור או מבוקשים בתחלן. עד שיים עבשים (70% Shallow) תור.

II

Technion - Computer Science Department - M.Sc. Thesis MSC-2008-11 - 2008
האלגוריתמים המוכלים (Extended GH) מתוכנסים לעשייה הרצונית על ידי השナルים העונים.

ההתטואת הביצוע בערך מרחוק(age) הוא שינון של הצוותים המוכנים אל השינון מספר פעמים בשינויים תכניים ופרטים פיקטיביים. האלגוריתמים המוכלים are מאני 40% Shallow 5.4 מימי שיפור של יחס השיטה.

של מעלי 90% אפיל עוצר עциально הם.

ככ שימוכוסות של השעמים (מספר הקודקוד המאפילים את המודליים) משמשות
הויתות הפקת מרכבות ייחור. revelations מיציאת התאמה של ביסיס מודלי במענה הפגカフェ.
קולת ויתר, טבלת ה-опטיה הופכת עמוסה יוטר ותחלייל התכבות הופך ארך יוטר.

למרות זאת, הגיוון ההופך אמין יוטר. נוכי ואשר הוצגו פעטי האלוגריהים המזוהים
בחל החוסטי ה-פעמים בחוק חלוק, כשיר יוסח התכלה סטיב-ה-85% באלוגריהים
המוכולים למעל 90% ב-2.5 מימד.

ההרחבת האלוגריהים החדשות (כמו גם האלוגריהים המଓ) עונים על ה erad בנוי
פעמים (זר-פרידיווא וא תלול-פרידיווא) המינויים או הוספ את קודין התיאנוגנוז
על דיו מיקוטוב בלב. לאחרונה הוצא מיצוג בחוק הקודינים המרכזים את המודל
לא מאופיינת על פי מיקוטוב בלב אלא על פי התכוניות מסוף, דבר המקרה יושב
הכלול בגוות יוטר או הוראות בודר בורה.

ההרצבה שצנג ימכブפעים "commended שוטו" בימדה כלשהי או ינות
לשים שג ענבר או פיים כליל יוטר על ידי התלול עני-פעמים והופיס הועים על
ה�יתורין שדרש. הת-פעמים של או הוראות בנפרד טבלת ה-החלשה
לצב היוצרות של מצעים בתכוזה תשכלל אתındaki שיניף ענבר חלקי.