ABSTRACT

CROOK, SUSAN BAILEY. Automated Shape Recognition and Curve Matching using Discrete Invariants. (Under the direction of Pierre Gremaud.)

We propose a new type of algorithm for curve matching. Our approach is based on recent theoretical advances regarding integral quantities that are invariant under certain group actions. Concentrating on rigid motions in $\mathbb{R}^2$, we construct discrete integral invariants. As the direct application of numerical quadratures to integral invariants does not result in invariant quantities, we show how to "invariantize" discrete quantities that only depend on samplings of the curves.

The significance of these new discrete invariants is threefold. First, these invariants provide a way to compute invariants for curves given discretely. Second, our approach is not limited to the Special Euclidean group. Third, and most importantly, our discrete invariants are robust with respect to curve samplings.

The performance of the proposed approach is successfully tested on two applications: character recognition and jigsaw puzzle assembly.

Automated Shape Recognition and Curve Matching using Discrete Invariants

by
Susan Bailey Crook

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Applied Mathematics

Raleigh, North Carolina

2013

APPROVED BY:

Stephen Campbell	Irina Kogan
Zhilin Li	Pierre Gremaud Chair of Advisory Committee

DEDICATION

To the hosts, guests, and panelists of NPR's "Wait Wait...Don't Tell Me" and the "Pop Culture Happy Hour" gang who have been my constant companions and friends during my hours of research and writing.

And, as with everything I do, in memory of my mom, Bettie Lee Mason Crook.

BIOGRAPHY

Susan Bailey Crook was born May 21, 1985 to Jim and Bettie Lee Crook. Susan was raised, along with her younger sister Sarah, in Oak Ridge, Tennessee where she was surrounded by a love of education, math, science, and creativity. At home her parents encouraged her curious nature and helped her become a life long learner. She was extraordinarily lucky to have a close group of friends who also loved to learn and did not tease her too much for being a nerd.

After graduating from Oak Ridge High School, Susan attended the University of South Carolina as a McNair Scholar. At the **real** Carolina, Susan was able to explore her various interests and eventually settled on pursuing a BA in French and a BS in Mathematics. She graduated Summa cum Laude with Honors from the South Carolina Honors College in May 2007 after four wonderful years of classes, friends, and several trips abroad.

Susan decided to attend North Carolina State University in pursuit of her M.S. in Applied Mathematics. At the end of her first semester, she realized she was not going to learn all the math she wanted to in only two years and switched to the Ph.D. track. During her time at NCSU, she has had the opportunity to work with many skilled mathematicians and educators. Susan is now a tenure track Assistant Professor of Mathematics at Loras College in Dubuque, IA where she is part of an energetic, supportive, and knowledgeable faculty. She is dedicated to teaching and is excited about her future career in academia.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Dr. Pierre Gremaud, for helping me through this journey. It wasn't always smooth sailing, but I always appreciated having an experienced mentor to guide and encourage me. Second, thanks to Dr. Irina Kogan for her additional help and expertise. Thanks to my committee members, Dr. Stephen Campbell, Dr. Zhilin Li, and Dr. Stephen Peretti for their time. Also, thanks to Dr. Mark Hoefer for serving as a substitute on my committee.

My family has been crucial in the Ph.D. process. My sister and father have listened to my joys and sorrows (sometimes at the same time) and I owe them endless gratitude. My extended family (Suzanne, Houston, and Charly), I thank you for helping take care of things so that I could be here doing this. And, Mom, I know without a doubt that you've been watching and cheering me on.

The list of friends who have been a part of my life the past 5 years is much too long to list, but I thank you all for everything you've done. My officemates, for helping brighten the day after bad meetings or telling me to stop talking and get to work. Jeb and Rachael, for commiserating and reminding me that sometimes a grumble isn't a bad response. Karen, Kristen, J.T., Matt, Sarah, Adam, Kate, and Shana - thank you for being in my loving entourage. A hearty thank you to my Oak Ridge girls - Swing, Bauman, Sam, Sariti, Alex, and Rossiepoo - for always supporting me in everything I do!

While I owe a huge amount of thanks to every teacher I've ever had, a few deserve name recognition. Mrs. Leavy, for being my school mom from first grade to senior year. Mr. Goforth, for being the first math teacher who wouldn't let me give up just because I was frustrated. You were the first person to show me that math could be fun and that I could do it on my own. Mrs. Reed was the first one to put the idea of math graduate school in my head. Mrs. Albert has long been my math idol and if I am ever half as good a teacher as she is, I will be ecstatic. Dr. Sumner and Dr. Roberts, thanks for helping me realize that the college level is where I belong.

Lastly, I have had multiple mentors through my education. Denise, without you, nothing would ever get done in the math department! Barbi and Melissa have been so supportive of me as I participated in every teaching program at NCSU. I will miss discussing my teaching ideas with you. Molly has given me invaluable teaching advice over the past years and, without her, I would have been lost during the job search process. And, finally, many thanks and much love to my second father, Mr. Burns, for always giving me just enough whining time before telling me to shut up and work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Objective of Curve Matching

Our goal is to construct, analyze, implement, and test efficient and reliable numerical methods to determine whether two or more curves are equivalent under some geometrical transformation. Central in our work is the case of the special Euclidean group (translations and rotations in $\mathbb{R}^2$). Two curves will thus be equivalent if one can be mapped to the other by a composition of rotation and translation.

In practical situations such as character recognition or medical applications, only sampled nodes are available to describe any given curve; this is very different from having, say, a parametric representation at our disposal. We propose new families of discrete invariants which can be directly applied to discretized curves. These invariants are left unchanged by the action of the geometric transformations and can be used in a robust way to assess equivalency.

1.2 Applications of Curve Matching

In this thesis we focus on the applications of curve matching to character identification and automated jigsaw puzzle solving. The development of automated character recognition has been spurred by recent technological developments, such as the increasing popularity of tablet

computers. Curve matching is one of the methods used in character recognition as detailed in [61], [60], [98], [37], [42], [96], and [35].

While jigsaw puzzle assembly may seem a frivolous problem, it has far reaching applications and has been widely studied, [41], [16], [54], [110], [38], [67], [90]. Computer algorithms developed to solve jigsaw puzzles may be used in object assembly and object recognition. This application is commonly seen in factory work where assembly lines are automated for faster and cheaper assembly of products. Depending on what is being assembled and how the assembly line is arranged, the machines may have to decide how best to match two pieces (similar to fitting two puzzle pieces together when only given the two pieces) or they may have to find a best match for a given piece out of several options presented (similar to finding a fit for a puzzle piece given all the other pieces). More detailed descriptions of the use of curve matching in object recognition and identification may be found in [94], [55], [10], [105], [18], [75], [109], [68], and [39].

Though we focus on these two applications, curve matching is also widely used in scientific applications and research. Researchers studying brain functions may use curve matching for brain image mapping, as in [106], [72], and [8]. Facial recognition algorithms, such as those detailed in [101] and [92], simplify faces to planar curves and use curve matching. Curve matching may be used to help doctors identify if patients are suffering from osteophytes, [70], by comparing x-rays of spines to a database of templates.

Marine researchers are able to save themselves a significant amount of time and money by using curve matching methods to track animal movement, [48] and [46]. Rather than tagging individual animals with electronic trackers and recording their movements via satellite, researchers now place underwater cameras in certain areas. Each animal that swims by is recorded. A computer then traces the unique fin outline of the animal and runs it through a database of fin outlines from animals who have previously swum past the camera. If there is a significant match to a previously known fin, then the researchers have an animal they have previously spotted. The upkeep cost of using cameras rather than trackers is less and there are less man hours

needed to install them. The researchers are also able to avoid the invasive procedures of having to capture animals to tag. Similar methods are used to identify elephants by their ear notches, [6].

Curve matching methods may also be used to help gain knowledge of previous civilizations. At archeological dig sites, historical ruins are often found in many pieces. It can take a human a significant amount of time to accurately reassemble a find since they are in effect having to solve a large scale, three dimensional jigsaw puzzle. Not only a hard task, but a delicate one as the pieces may be extremely fragile. Rather than solve these tasks manually, we may take a picture of each fragment and have a computer use curve matching techniques to virtually reassemble the whole piece. Once the piece is completed, the algorithm will also specify the transformations needed to assemble the pieces. This application of curve matching technology is further detailed in [78] and [104].

Curve matching also has a place in national and personal security. We have all seen a detective reassemble a shredded secret message in a crime solving television show. Computerized curve matching algorithms are making this a quicker and less involved task despite the fact that shredders are becoming more advanced in response to the heightened threat. In 2011, the Department of Defense's Defense Advanced Research Projects Agency (DARPA) held a shredder challenge, [1]. The challenge was composed of five separate shredded puzzles. Each puzzle had a varying number of shredded documents and the method of shredding was varied as well. Teams had to reassemble the documents to find an embedded clue which they had to solve in order to complete the puzzle. The winning team, "All Your Shreds Are Belong to U.S.", solved the five puzzles in slightly over one month to win the challenge. According to DARPA the goals of this challenge were "to identify and assess potential capabilities that could be used by our warfighters operating in war zones, but might also create vulnerabilities to sensitive information that is protected through our own shredding practices throughout the U.S. national security community." These same threats present themselves in our daily lives as well. As technology evolves to the point where personal computers are able to easily reconstruct

shredded documents, personal information becomes less secure even when precautions are taken.

1.3 Methods of Curve Matching

Many methods of curve matching and registration exist; general surveys of the field are given in [4] and [114]. These methods may be split into two categories - contour-based and region-based. Contour-based methods use only the outline of the curve to classify and group curves while region-based methods take into account all the pixels within the shape region. All open curve matching techniques will be contour-based by default. In both of these categories we may further split into structural versus global approaches. A global approach treats the curve as one entire piece whereas structural approaches represent the curve in segments (also called primitives). Our method is a contour-based technique. Depending on how it is used, it may be either a global or structural technique. In the case of character recognition it is a global technique, but in puzzle assembly it is used as a structural technique.

Some examples of each subcategory of techniques are listed in Figure 1.1, reproduced from [114]. We note that many curve matching algorithms use a combination of two or more of these techniques.

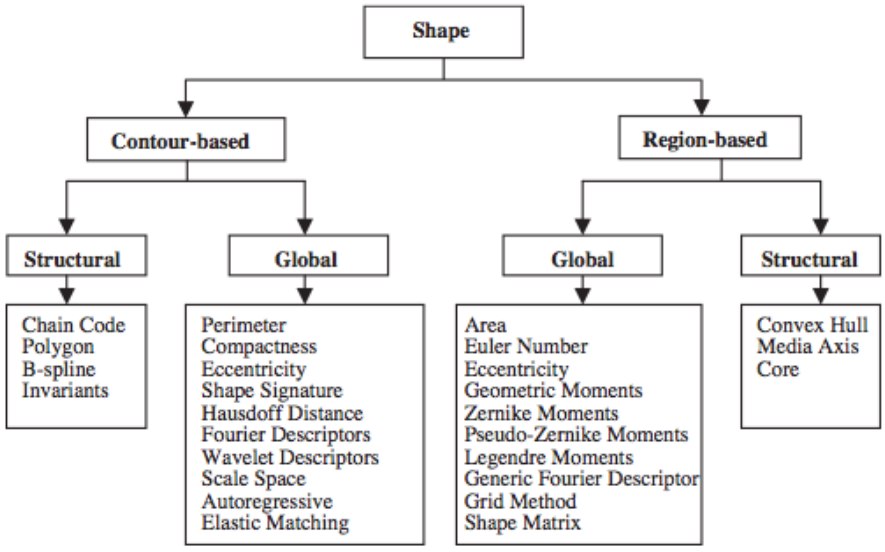


Figure 1.1: Classification of curve matching and registration methods (reprinted from [114]).

Selected Region-based Global Techniques

Moments are a common region-based global curve matching technique. We will provide examples of methods using geometric, Zernike, pseudo-Zernike, and Legendre moments. We choose to focus on moment based methods because we will compare the success rate of our character recognition method to that of several methods using moments. Moments are created by mapping a function onto a polynomial basis of choice. Geometric moments are found by projecting a function onto a standard power basis and are invariant to translation, scaling, and rotation. Curve matching methods employing geometric moments are proposed in [112], [31], [3], [97], and [80].

Legendre, Zernike, and pseudo-Zernike moments are also referred to as orthogonal moments since the polynomial bases used are orthogonal. Zernike moments are the mappings of an image onto a set of Zernike polynomials. These moments represent the properties of the image with no overlap of information since the Zernike polynomials are orthogonal to one another. The magnitudes of Zernike moments are invariant to rotation; however, they are dependent on scaling and translation of the image. Thus if the curve which we are attempting to classify

may have been rotated, but not translated or scaled, these moments provide a viable method. Pseudo-Zernike moments have been shown to be more robust and less sensitive to noise making them good choices for use in image classification and recognition. Legendre moments are created by mapping a function or image onto a set of Legendre polynomials. In the method to which we will compare our character recognition method ([42]), Legendre-Sobolev polynomials are used. Methods using orthogonal moments are given in [64], [25], [66], [86], [56], [85], [99], [100], and [108].

Selected Region-based Structural Techniques

Methods using the medial axis begin by creating a skeleton of an image. A skeleton is defined as a connected set of medial lines that represent the figure. This skeleton is then decomposed into segments which may be represented as a graph. Thus, the problem has been reduced to one of graph matching. The pitfall of methods using this technique is that the computation of the medial line is often difficult. In [87], [11], [81], curve matching techniques using medial lines are proposed.

The distance from a curve to a convex hull of other shapes may be used to classify and identify curves. The distance used to measure the distance may vary based on method, e.g. Manhattan and Euclidean distances ([42], [43], [44]), Fréchet distance ([15]), and Hausdorff distance ([47]). This technique is often used with other techniques to be more effective, for instance the method detailed in [42] combines Legendre-Sobolev moments, integral invariants, and distance to convex hull.

Selected Contour-based Global Techniques

The method of using Fourier descriptors, ([5], [63], [89], [5]) begins by describing the outline of a shape by its arc length from a declared origin. This parameter is then normalized so that its sum along the curve is 2π. A function giving the angular variation between the tangent at the origin and that tangent at a given position is then written in terms of a Fourier series. The coefficients of this series are invariant to translation, rotation, change of scale, and change of origin of the shape and, thus, are useful in classifying and identifying curves when transforma-

tions may have occurred. Similar techniques using wavelets to rewrite the function of angular variation rather than Fourier series exist, ([102], [74], [58]).

Elastic matching warps specified pixels of one curve to the other and attempts to optimize this matching to discover if two curves are the same. The pixels may be either on the edge of a given shape or inside the shape. This method has obvious uses in the applications of puzzle assembly and character recognition ([113], [7], [17]). A recent survey of elastic matching techniques and its application to character recognition is given in [103].

Both of these techniques work to identify and classify a given shape without segmenting it. Descriptors strive to do this without changing the curve or finding an exact transform while elastic matching tries to transform one shape to the other. Our curve matching method when used for global matching is closer to that of descriptors. We will measure certain aspects of the figure and use these to identify or classify our curve.

Selected Contour-based Structural Techniques

For contour-based structural methods, curves are first broken into smaller segments. The common methods for splitting a curve into primitives are based on polygonal approximation, curvature decomposition, and curve fitting, [88].

One such method relies on representing curves with B-splines. Further information on algorithms used to find B-spline representations of curves can be found in [29], [30], [79], and[9]. In [26] and [107], Cohen and Wang propose an algorithm for curve matching using a similarity measure based on B-spline knot points. While effective, this method cannot be used in cases where there may be an affine transformation between the curves or in cases of occlusion. A new algorithm that can handle curves related via affine transforms and occlusion was proposed in [57] by Huang and Cohen; this new curve matching algorithm relies on representing the curves with weighted B-spline curve moments. The application of this matching algorithm to the process of matching homologous histological sections of rat brains was studied in [27], [3]. A new curve matching algorithm using "super-curves" was developed by Xia and Liu in [111]. This approach uses a B-spline fusion technique to find a B-spline approximation of the super curve

created by considering both curves in one coordinate system. As a result rather than examining two B-spline curves, only one B-spline curve will be used in this technique. This method allows for affine transforms and occlusion when used for curve matching and registration.

The method we propose uses integral invariants for the purpose of curve matching. As mentioned previously, depending upon the application this method may be considered either a contour-based global type of technique or a contour-based structural technique. As with many of the techniques described above, invariants may be more effectively used when combined with other curve matching methods.

Invariants have long been used for curve matching, see, for instance, approaches based on the moving frame method proposed by Elie Cartan [[22], [51],[50], [84],[34], [33]]. Many types of invariants have been proposed and used - differential invariants ([28]), joint invariants ([45], [83], [12], [18]), and integral invariants ([93], [73], [76], [53], [77]). The circular area and cone area signatures in particular are examples of widely used integral invariants ([20], [36], [76]). Each type of invariant has its own benefits and pitfalls.

If we consider the special Euclidean group on curves in $\mathbb{R}^2$, then the most commonly used invariants are curvature and the derivative of curvature with respect to arc length. Assuming we have a parametric equation of a curve, $(x(t), y(t))$, $t \in [0, T]$, these can be written as

$$\kappa(t) = \frac{\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)}{(\dot{x}^2(t) + \dot{y}^2(t))^{3/2}}$$
$$\kappa_s(t) = \frac{d}{ds}\kappa(t) = \frac{(\dot{x}^2 + \dot{y}^2)(\dot{x}\dddot{y} - \dddot{x}\dot{y}) - 3(\dot{x}\ddot{x} + \dot{y}\ddot{y})(\dot{x}\ddot{y} - \ddot{x}\dot{y})}{(\dot{x}^2 + \dot{y}^2)^3}$$

where the dots denote differentiation with respect to t. In fact these are the basis for all invariants based on derivatives of parametric curves in the Euclidean plane. While it is known that two curves are equivalent under rigid motion if and only if their curvatures as a function of arc length are the same, it is not practical to classify curves in this manner for several reasons.

First, the above invariants assume a parametrization of the curve which might not be avail-

able or may be costly to find. Second, parametrizing with respect to arc length may be difficult in practice. Third, the above quantities are only defined for continuous (as opposed to discrete) curves. Further, numerical approximations of the quantities will in general not be invariant. Fourth, the differential nature of the above invariants assumes some degree of smoothness of the curves which may not be present in practice.

The invariants we propose in this thesis are different from the above construction in two fundamental ways

1. They are *integral*, not differential. This ensures a higher stability and lower sensitivity to perturbation and/or noise in the data.

2. They are *discrete*. In other words, they can be applied directly to discrete curves, i.e., curves described only at a finite number of nodes.

1.4 Overview of Thesis

In Chapter 2 we review previous work and situate our approach in respect to the relevant literature. We provide needed background material for the analysis of our method. Specifically, we discuss the development and use of differential and integral invariants. In Chapter 3 we detail the method that we have developed to generate curve invariants which allow for discrete input. We also describe and outline our algorithm utilizing discrete curve invariants for curve matching. Analysis of our method and the discrete invariants will also be included. Finally, in Chapter 4, we present two real world applications of our algorithm. To demonstrate the matching of open curves, we present the example of character recognition and for closed curve matching, we show an example finding the area of best match on pairs of puzzle pieces.

Chapter 2

Continuous Invariants and Signatures for Planar Curves

2.1 Curve Invariants

We review some necessary background that will be used in the development and analysis of discrete invariants. We begin with the concept of a group action defined below.

Definition 2.1.1. Let G be a group and let Ω be a set. The action of G on Ω is a map $\alpha : G\times \Omega\rightarrow \Omega$ satisfying the following:

identity: $\alpha(e, s) = s$, for all $s \in \Omega$ with e being the identity in G.

associativity: $\alpha(g_1, \alpha(g_2, s)) = \alpha(g_1 g_2, s)$ for all $g_1, g_2 \in G$ and $s \in \Omega$.

We denote the action by $\alpha\ (g, s) = gs$.

We focus on the special Euclidean group which acts on $\mathbb{R}^2$ in the following way

$$\begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

where $v_1, v_2, \phi \in \mathbb{R}$. Thus this group action consists of translations and rotations.

We define the concept of an invariant. As might be suggested by its English definition, an invariant is a function that is not altered when certain transformations are performed.

Definition 2.1.2. Let the group G act on a set Ω. A function $f : \Omega \to \mathbb{R}$ is **invariant** if it is unaltered by the action of G, i.e. $f(g\omega) = f(\omega)$, for all $g \in G$ and $\omega \in \Omega$.

A group action on $\mathbb{R}^2$ induces an action on curves in $\mathbb{R}^2$ in a natural way. Namely, to a parametrized curve γ in $\mathbb{R}^2$, with $\gamma(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$, $t \in (0, T)$, we associate the parametrized curve $\bar{\gamma}$ where

$$\bar{\gamma}(t) = \begin{bmatrix} \bar{x}(t) \\ \bar{y}(t) \end{bmatrix} = g \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}, \qquad t \in (0, T),$$

with $g \in G$. Similarly, if a discrete curve γ_N in $\mathbb{R}^2$ is given by N ordered nodes $\left\{ \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right\}_{i=1}^{N}$, then we define $\bar{\gamma}_N$ as being determined by the N nodes $g \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} \bar{x}_i \\ \bar{y}_i \end{bmatrix}$, $i = 1, \ldots, N$.

To generate invariants, we use the "invariantization" procedure introduced in [34] which follows from a generalization of Cartan's moving frame method, [21], [22]. The invariantization process is outlined below in the case of $\mathbb{R}^n$.

Invariantization Process

Let G be an r-parametric group action on $\mathbb{R}^n$ with coordinates $(x_1, \ldots, x_n)$.

1. The group transformation equations are written as

$$\overline{x}_i = \alpha_i(\lambda_1, \ldots, \lambda_r, x_1, \ldots, x_n), \quad i = 1, \ldots, n. \tag{2.1}$$

2. Choose constants $c_1, \ldots, c_r$ and set r of the transformed variables equal to these constants.

Up to possible relabeling, we have

$$\alpha_i(\lambda_1, \ldots, \lambda_r, x_1, \ldots, x_n) = c_i, \quad i = 1, \ldots, r.$$

3. Solve the previous r equations for $\lambda_1, \ldots, \lambda_r$ if possible. If it is not possible, we must try another cross-section $c_1, \ldots, c_r$. Otherwise, the corresponding solution $g = \rho(x) \in G$ is a moving frame.

4. Compute the action of the moving frame on the remaining coordinates. More precisely, substitute the values found for $\lambda_1, \ldots, \lambda_r$ into Eq. 2.1 to obtain $n - r$ non-constant invariants

$$\bar{x}_{i+r}|_{g=\rho(x)} = I_i(x_1, \ldots, x_n), \quad i = 1, \ldots, n - r.$$

It was proved in [21] and [22] that under some generic conditions on the group action step three of the process can be performed for some choice of a subset, $(i_1, \ldots, i_r) \in (i_1, \ldots, i_n)$, and almost all values of $c_1, \ldots, c_r$.

Definition 2.1.3. Two planar curves, γ_1 and γ_2, are **equivalent** under the actions of G if there exists $g \in G$ such that $\gamma_1 = g\gamma_2$ where $g\gamma_2$ denotes the transformed curve γ_2 under g.

We want to determine if two curves are equivalent without finding the explicit transform that relates them. To do this, it is most advantageous to use a signature constructed from invariants.

2.2 Integral Invariants and Signatures

We concentrate on *integral* invariants because they are better equipped to handle noisy/perturbed data than invariants based on differentiated quantities (such as curvature). In [76] and [77], integral invariants were proposed that are restricted to planar curves undergoing Euclidean transformations. In light of possible applications, we will use the type of integral invariants

presented in [53] and [35], which may be developed for use on planar curves undergoing group transformations other than Euclidean. We follow the general presentations given in these papers as we define continuous integral invariants for special Euclidean group actions on planar curves.

Suppose we have a parametric curve γ in $\mathbb{R}^2$ given by $(x(t), y(t))$, $t \in [0, T]$. The explicit construction of integral invariants is based on (i) prolonging the action of, say, the special Eulidean group, from curves in $\mathbb{R}^2$ to new *integral variables* and (ii) applying the general invariantization process outlined above.

By translating the initial point $\gamma(0)$ to the origin and defining

$$\begin{aligned} X(t) &= x(t) - x(0), \\ Y(t) &= y(t) - y(0), \end{aligned} \tag{2.2}$$

in the above integrals, we can reduce the problem of finding invariants for the special Euclidean group to finding invariants for SO(2).

The integral variables are defined as follows

$$\begin{aligned} X^{(ij)}(t) &= \int_0^t X^i(\tau) Y^j(\tau) dX(\tau), \\ Y^{(ij)}(t) &= \int_0^t X^i(\tau) Y^j(\tau) dY(\tau), \end{aligned} \tag{2.3}$$

where the integrals are taken along the curve γ and where i, j are nonnegative integers such that $i + j \neq 0$.

Using integration by parts, it is easy to find dependencies between some of the above integral variables. For instance, we have

$$\begin{aligned} Y^{(10)}(t) &= X(t)Y(t) - X^{(01)}(t) \\ Y^{(20)}(t) &= X^2(t)Y(t) - 2X^{(11)}(t) \\ X^{(02)}(t) &= X(t)Y^2(t) - 2Y^{(11)}(t). \end{aligned}$$

Following [53] and [35], it can be shown that the problem of finding integral invariants under the action of the special Eucliean group on curves in $\mathbb{R}^2$ reduces to finding invariant functions of the variables

$$\{X(t), Y(t), X^{(01)}(t), Y^{(11)}(t), X^{(11)}(t)\}$$

where

$$\begin{aligned} X^{(01)}(t) &= \int_0^t Y(\tau)dX(\tau) \\ Y^{(11)}(t) &= \int_0^t X(\tau)Y(\tau)dY(\tau) \\ X^{(11)}(t) &= \int_0^t X(\tau)Y(\tau)dX(\tau). \end{aligned} \tag{2.4}$$

In other words, we apply the invariantization process 2.1 in $\mathbb{R}^5$. The corresponding group action (in our case, rotations) becomes

$$\begin{aligned} \overline{X} &= \cos(\phi)X - \sin(\phi)Y \\ \overline{Y} &= \sin(\phi)X + \cos(\phi)Y \\ \overline{X}^{(01)} &= X^{(01)} + \frac{1}{2}\cos(\phi)\sin(\phi)(X^2 - Y^2) - \sin^2(\phi)XY \\ \overline{Y}^{(11)} &= \cos(\phi)Y^{(11)} - \sin(\phi)X^{(11)} \\ &\quad + \frac{1}{3}\cos(\phi)\sin(\phi)\left[\sin(\phi)X^3 + 3\cos(\phi)X^2Y - 3\sin(\phi)XY^2 - \cos(\phi)Y^3\right] \\ \overline{X}^{(11)} &= \cos(\phi)X^{(11)} + \sin(\phi)Y^{(11)} \\ &\quad + \frac{1}{3}\cos(\phi)\sin(\phi)\left[\cos(\phi)X^3 - 3\sin(\phi)X^2Y - 3\cos(\phi)XY^2 + \sin(\phi)Y^3\right]. \end{aligned} \tag{2.5}$$

By considering the cross-section (normalization) $\overline{Y} = 0$, we get

$$\begin{aligned} \cos(\phi) &= \frac{X}{\sqrt{X^2 + Y^2}} \\ \sin(\phi) &= \frac{-Y}{\sqrt{X^2 + Y^2}}. \end{aligned} \tag{2.6}$$

Substituting these values into Eq. 2.4, we find

$$
\begin{aligned}
\overline{X} &= \sqrt{X^2+Y^2} \\
\overline{Y} &= 0 \\
\overline{X}^{(01)} &= X^{(01)} - \frac{1}{2}XY \\
\overline{Y}^{(11)} &= Y^{(11)}X - \frac{1}{2}Y^{(20)}Y - \frac{1}{6}X^2Y^2. \\
\overline{X}^{(11)} &= XX^{(11)} - YY^{(11)} - \frac{1}{3}(X^3Y - XY^3)
\end{aligned}
\tag{2.7}
$$

Thus, we have four non constant invariants which we will denote as

$$
\begin{aligned}
R &= \sqrt{X^2+Y^2} \\
I_1 &= X^{(01)} - \frac{1}{2}XY \\
I_2 &= Y^{(11)}X - \frac{1}{2}Y^{(20)}Y - \frac{1}{6}X^2Y^2 \\
I_3 &= XX^{(11)} - YY^{(11)} - \frac{1}{3}(X^3Y - XY^3).
\end{aligned}
\tag{2.8}
$$

By construction, each of these is invariant under rotation and, after substitution (Eq. 2.2), rotation and translation.

We can also consider the effects of reflection on these invariants. We consider a reflection over the y-axis. This is equivalent to letting $\hat{X} = -X$ and $\hat{Y} = Y$. Performing this reflection will change the sign of $X^{(01)}$ and $Y^{(11)}$. If we consider the formulas of invariants in Eq. 2.8 with these new variables, we find

$$
\begin{aligned}
\hat{R} &= \sqrt{\hat{X}^2+\hat{Y}^2} = \sqrt{X^2+Y^2} = R \\
\hat{I}_1 &= \hat{X}^{(01)} - \frac{1}{2}\hat{X}\hat{Y} = -X^{(01)} + \frac{1}{2}XY = -I_1 \\
\hat{I}_2 &= \hat{Y}^{(11)}\hat{X} - \frac{1}{2}\hat{Y}^{(20)}\hat{Y} - \frac{1}{6}\hat{X}^2\hat{Y}^2 = Y^{(11)}X - \frac{1}{2}Y^{(20)}Y - \frac{1}{6}X^2Y^2 = I_2 \\
\hat{I}_3 &= \hat{X}\hat{X}^{(11)} - \hat{Y}\hat{Y}^{(11)} - \frac{1}{3}(\hat{X}^3\hat{Y} - \hat{X}\hat{Y}^3) = -XX^{(11)} + YY^{(11)} - \frac{1}{3}(-X^3Y + XY^3) = -I_3.
\end{aligned}
\tag{2.9}
$$

We see that R and I_2 are also invariant to reflection while I_1 and I_3 are not. We may square I_1 and I_3 to render them invariant to reflection.

Definition 2.2.1. If Γ denotes a set of curves in $\mathbb{R}^2$ and G acts on Γ, then $S : \Gamma \to \Gamma$ is a **signature map** if $S(\gamma) = S(g\gamma)$, for all $g \in G$ and for all $\gamma \in \Gamma$. The image, $S(\gamma)$ is called the **signature** of γ.

The main property of signatures is that the signatures of equivalent curves are equal.

Example 2.2.2. Consider the curve, given in polar coordinates, $r = \theta$ which produces a spiral. This can be rewritten in Cartesian coordinates as $(x(\theta), y(\theta)) = (\theta\cos(\theta), \theta\sin(\theta))$. A plot of this curve for $\theta \in [0, 4\pi]$ can be seen in Figure 2.1.

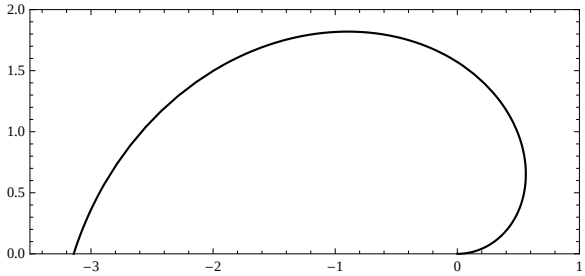


Figure 2.1: Plot of $(x(\theta), y(\theta)) = (\theta\cos(\theta), \theta\sin(\theta))$ over the interval $[0, 4\pi]$.

The invariants defined above in Eq. 2.8, take the form

$$\begin{aligned} R^\gamma(\theta) &= \theta \\ I_1^\gamma(\theta) &= \frac{-1}{6}\theta^3 \\ I_2^\gamma(\theta) &= 2\theta + \frac{-\theta^3}{3} - 2\sin(\theta) \\ I_3^\gamma(\theta) &= -4 + 4\cos(\theta) + 2\theta^2. \end{aligned} \tag{2.10}$$

Computing invariants on curves allows us to consider the geometric interpretation of the invariants.

The invariant R, which is invariant with respect to rotation, translation, and reflection, is the length of the secant line connecting the initial point, $(X(0), Y(0)) = (0, 0)$, to the current point, $(X(t), Y(t))$. This geometric interpretation can be seen in Figure 2.2a.

(a) Invariant R.

(b) Invariant I_1.

Figure 2.2: Geometric Interpretations of Integral Invariants.

The invariant I_1 is invariant with respect to rotation and translation. The invariant $I_1(t)$ for some t in our domain can be interpreted geometrically as the signed area between the secant line connecting the initial point $(X(0), Y(0)) = (0, 0)$ and the point $(X(t), Y(t))$ and the curve. This geometric interpretation can be seen in Figure 2.2b.

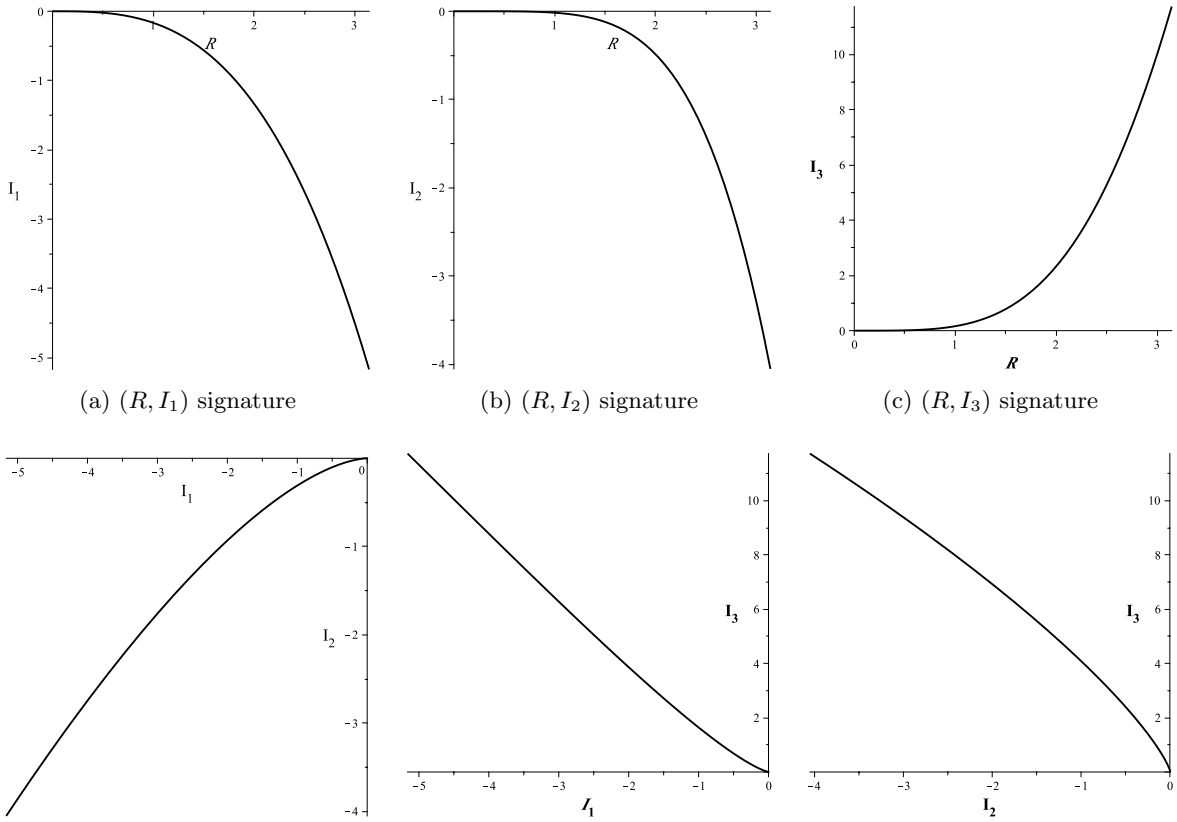
The invariant I_2 is invariant with respect to rotation, translation, and reflection. The invariant I_3 is invariant with respect to rotation and translation. Unfortunately, the invariants I_2 and I_3 do not have such easily explained geometric interpretations. Further details and a more involved geometric interpretation are given in [35].

While invariants themselves capture some important properties of curves, the true power of using invariants for curve matching lies in using them to create signatures. We have from the above considerations

Theorem 2.2.3. *Let $\gamma(t)$ be a planar curve. Then $(R^\gamma(t), I_1^\gamma(t)), (R^\gamma(t), I_2^\gamma(t))$, $(I_1^\gamma(t), I_2^\gamma(t))$,*

$((R^\gamma(t), I_3^\gamma(t)), (I_1^\gamma(t), I_3^\gamma(t))$, *and* $(I_2^\gamma(t), I_3^\gamma(t))$ *are special Euclidean signatures of* γ.

Example 2.2.4. We return to the example of the parametrically given spiral shown in Figure 2.1. We plot the signatures using the invariants given by Eq. 2.10 in Example 2.2.4 . In Figure 2.3, we illustrate the special Euclidean signatures for this spiral for $\theta \in [0, \pi]$.



(a) (R, I_1) signature (b) (R, I_2) signature (c) (R, I_3) signature

(d) (I_1, I_2) signature (e) (I_1, I_3) signature (f) (I_2, I_3) signature

Figure 2.3: Various signature curves for $(x(\theta), y(\theta)) = (\theta\cos(\theta), \theta\sin(\theta))$ over the interval $[0, \pi]$.

We may use these signatures for the purpose of identifying equivalent curves. It was shown in [2] that given two continuously differentiable curves, γ and $\overline{\gamma}$, that do not intersect a circle of radius r more than once for each $r > 0$, the two curves are equivalent with respect to special Euclidean actions if their $(R(t), I_1(t))$ signatures coincide. A similar result was shown for the affine group and $(R(t), I_2(t))$ signature in [59].

There are many advantages to using signature curves to classify and identify curves. By using signature curves to identify if two curves are equivalent we are able to avoid having to find an explicit group element under which one curve may be mapped to the other. Often our end goal is not to find the transformation, but rather just to find that such a transformation exists and employing signatures determines this without the work of finding the transform. Additionally, to compare two curves without using the pairing of invariants to create signatures, we would need to ensure that a uniform parameterization is used for the two curves. Using signature curves allows us to overcome this difficulty.

Unfortunately, signatures based on integral invariants are dependent upon our choice of initial point. In the case of open curves, we have two natural options for initial point and may avoid this dependence by computing two signatures, one associated with each choice of initial point, to use for curve matching purposes. For closed curves, we have infinite choices for initial point and this issue can greatly complicate the task of curve matching. In Chapter 3, we discuss how we address this dependence upon initial point in our curve matching processes.

Chapter 3

Discrete Invariants and Signatures for Planar Curves

3.1 Invariant Approximations of Integral Invariants

The approaches described so far rely heavily on curve parametrizations. In most practical cases, curves are given through nodal values (pixels for instance) and parametrizations are not available. There are essentially two ways to deal with this. First, one could construct curve approximations based polynomial interpolation, splines or other similar methods and derive parametrizations from these constructions. This approach is very sensitive to perturbation and noise and would require patching up several local reconstructions in order to guarantee an acceptable level of accuracy. Another approach is to consider numerical approximations to internal invariants. This was adopted in [35] where local approximations to integral invariants were implemented. Here, we take this approach further and construct *global discrete integral invariants*. The main challenge we have to solve is that is the invariance property is not preserved by numerical quadratures. We solve this problem by invariantizing discrete integral variables directly (as opposed to discretizing invariant integral variables). We thus construct a **new class of discrete invariants**. Let us also note that there exist invariant numerical approximations

to differential invariants ([19],[14], [12], [18])).

Our discrete invariants do not require a dense sampling of the curves to be matched. The discrete invariants which we propose are able to match curves regardless of how they were sampled. Provided two different samplings of the same curve our method will still identify a match. As a result of this independence of sampling method, our matching technique more accurately associates the underlying structure of the curve with the given set of points. The lesser requirements of curve sampling needed for our matching technique make it useful in varied applications.

3.1.1 Derivation of Discrete Invariants

We develop a method to derive discrete invariants. We derive discrete invariants by applying a quadrature rule to the integral variables previously introduced in Eq. 2.4 and then following the invariantization process described in [35] and outlined in 2.1. For our purposes, we apply two common quadrature rules to derive discrete invariants: the trapezoidal rule and Simpson's rule, both part of the Newton-Cotes family of quadrature rules.

The **trapezoidal rule** approximates the integral of a function, f, on an interval $[a, b]$ by the area of a trapezoid such that

$$\int_a^b f(x)\,dx \approx (b-a)\frac{f(a)+f(b)}{2}.$$

In practice, the domain of integration itself is discretized into N subdomains (x_0, x_1), (x_1, x_2), $\ldots$, (x_{N-1}, x_N) where the $N+1$ nodes x_i, $i = 0, \ldots, N$ satisfy $a = x_0 < x_1 < \cdots < x_N = b$. The corresponding quadrature formula, known as the **composite trapezoidal rule**, is

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{N} \frac{f(x_i)+f(x_{i-1})}{2}\,(x_i - x_{i-1}).$$

For a smooth enough integrand f ($\mathcal{C}^2$) and uniform meshes, the error is

$$\text{error} = -\frac{(b-a)^3}{12N^2} f"(\xi),$$

at some point ξ, $\xi \in (a, b)$.

Simpson's rule approximates the integral of a function, f, on an interval $[a, b]$ by

$$\int_a^b f(x)\,dx \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right].$$

Again, in practice, we use **composite Simpson's rule** to approximate the integral of f over an interval $[a, b]$ that is divided into N subintervals, N even, as

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{N} \frac{x_i - x_{i-1}}{6}\left[f(x_{i-1}) + 4f\left(\frac{x_i + x_{i-1}}{2}\right) + f(x_i)\right].$$

For f smooth enough ($\mathcal{C}^4$) and uniform meshes, the error is

$$\text{error} = -\frac{(b-a)^5}{180N^4} f^{(IV)}(\xi),$$

for some $\xi \in (a, b)$.

We use the composite trapezoidal and Simpson's rules to derive discrete invariants: $I_1^T(n)$, $I_2^T(n)$, $I_1^S(n)$, and $I_2^S(n)$. To do so, we first apply the quadrature rule to an integral variable and then use the invariantization process as described in 2.1. The steps of the derivation are presented below.

Discrete Invariant Derivation Process

1. Apply a quadrature rule to each of the expressions in Eq. 2.4 on a discrete mesh, (X_i, Y_i), $i = 0, \ldots, N-1$ with $(X_0, Y_0) = (0, 0)$.

2. Substitute $\overline{X}_i = \cos(\phi)X_i - \sin(\phi)Y_i$ and $\overline{Y}_i = \sin(\phi)X_i + \sin(\phi)Y_i$ in to the expressions obtained in step one.

3. Substitute the normalization $\cos(\phi) = \dfrac{X_n}{\sqrt{X_n^2 + Y_n^2}}$ and $\sin(\phi) = \dfrac{-Y_n}{\sqrt{X_n^2 + Y_n^2}}$ in to the expressions from step two.

First, we derive invariants using the trapezoidal rule.

To derive a numerical approximation to the first integral invariant, we begin with the integral variable, $X^{(01)}(t) = \int_0^t Y(s)\, dX(s)$. We approximate $X^{(01)}$ using the trapezoidal rule with n nodes, $1 \leq n \leq N$, to obtain

$$X^{(01)}(n) = \frac{1}{2}\sum_{i=1}^{n}(Y_i + Y_{i-1})(X_i - X_{i-1}).$$

Rotation on the plane by an angle ϕ induces the transformation

$$\begin{aligned} \overline{X}_i &= \cos(\phi)X_i - \sin(\phi)Y_i \\ \overline{Y}_i &= \sin(\phi)X_i + \cos(\phi)Y_i. \end{aligned} \tag{3.1}$$

We prolong this action to $X^{(01)}(t)$ by

$$\begin{aligned} \overline{X_T^{(01)}(n)} =& \frac{1}{2}\sum_{i=1}^{n}(\overline{Y}_i + \overline{Y}_{i-1})(\overline{X}_i - \overline{X}_{i-1}) \\ =& \frac{1}{2}\sum_{i=1}^{n}(Y_{i-1}X_i - Y_iX_{i-1}) + \cos(\phi)\sin(\phi)(Y_i^2 - Y_{i-1}^2 - X_i^2 + X_{i-1}^2) \\ &+ (\cos^2(\phi) - \sin^2(\phi))(Y_iX_i - Y_{i-1}X_{i-1}) \end{aligned}$$

Due to cancelation and the fact that $0 = X_0 = Y_0$, this expression simplifies to

$$\begin{aligned}\overline{X_T^{(01)}(n)} =& \frac{1}{2}\sum_{i=1}^{n}(Y_{i-1}X_i - Y_iX_{i-1}) + \cos(\phi)\sin(\phi)(Y_n^2 - X_n^2) \\ &+ (\cos^2(\phi) - \sin^2(\phi))(Y_nX_n).\end{aligned}$$

We now normalize by letting $Y_n = 0$, which is equivalent to assigning

$$\begin{aligned}\cos(\phi) &= \frac{X_n}{\sqrt{X_n^2 + Y_n^2}} \\ \sin(\phi) &= \frac{-Y_n}{\sqrt{X_n^2 + Y_n^2}}.\end{aligned} \tag{3.2}$$

Substituting these values into our expression for $\overline{X^{01}(n)}$, we obtain

$$I_1^T(n) = \frac{1}{2}\sum_{i=1}^{n}(Y_{i-1}X_i - Y_iX_{i-1}).$$

Thus, we can derive an invariant, numerical approximation to $I_1(t)$ by discretizing our integral variable $X^{(01)}$ with the trapezoidal rule before applying the invariantization process.

We now discretize $Y^{(11)}$ and follow the same process as above to construct an invariant, numerical approximation to $I_2(t)$ using the trapezoidal rule.

We discretize $Y^{(11)}(t) = \int_0^t X(s)Y(s)dY(s)$ using the trapezoidal rule to obtain $Y_T^{(11)}(n)$, $1 \leq n \leq N$,

$$Y_T^{(11)}(n) = \frac{1}{2}\sum_{i=1}^{n}(X_iY_i + X_{i-1}Y_{i-1})(Y_i - Y_{i-1}).$$

Then, prolonging the action of Eq. 3.1

$$\begin{aligned}\overline{Y_T^{(11)}(n)} &= \frac{1}{2}\sum_{i=1}^{n}(\overline{X}_i\overline{Y}_i + \overline{X}_{i-1}\overline{Y}_{i-1})(\overline{Y}_i - \overline{Y}_{i-1}) \\ &= \frac{1}{2}\sum_{i=1}^{n}((\cos(\phi)X_i - \sin(\phi)Y_i)(\sin(\phi)X_i + \cos(\phi)Y_i)\end{aligned}$$

$$+ (\cos(\phi)X_{i-1} - \sin(\phi)Y_{i-1})(\sin(\phi)X_{i-1} + \cos(\phi)Y_{i-1}))$$
$$(\sin(\phi)(X_i - X_{i-1}) + \cos(\phi)(Y_i - Y_{i-1})).$$

Using Eq. 3.2 we get

$$I_T^{(11)}(n) = \frac{1}{2(X_n^2 + Y_n^2)^{3/2}} \sum_{i=1}^{n} \quad [(X_n X_i + Y_n Y_i)(-Y_n X_i + X_n Y_i)$$
$$+(X_n X_{i-1} + Y_n Y_{i-1})(-Y_n X_{i-1} + X_n Y_{i-1})]$$
$$[-Y_n(X_i - X_{i-1}) + X_n(Y_i - Y_{i-1})]$$

We can simplify this formula by canceling some terms as we expand the summation. Recalling that $(X_0, Y_0) = (0, 0)$, we find the following simplifications.

$$\sum_{i=1}^{n} (X_i Y_i^2 - X_{i-1} Y_{i-1}^2) X_n^3 = (X_0 Y_0^2 - X_n Y_n^2) X_n^3 = X_n^4 Y_n^2$$
$$\sum_{i=1}^{n} (Y_i^3 + 2X_{i-1}^2 Y_{i-1} - 2X_i^2 Y_i - Y_{i-1}^3) X_n^2 Y_n = (Y_n^3 - 2X_n^2 Y_n) X_n^2 Y_n = X_n^2 Y_n^2 - 2X_n^4 Y_n^2,$$
$$\sum_{i=1}^{n} (X_i^3 - X_{i-1}^3 + 2X_{i-1} Y_{i-1}^2 - 2X_i Y_i^2) X_n Y_n^2 = (X_n^3 - 2X_n Y_n^2) X_n Y_n^2 = X_n^4 Y_n^2 - 2X_n^2 Y_n^4, \text{ and}$$
$$\sum_{i=1}^{n} (X_i^2 Y_i - X_{i-1}^2 Y_{i-1}) Y_n^3 = (X_n^2 Y_n) Y_n^3 = X_n^2 Y_n^4.$$

Making these substitutions and simplifying,

$$I_2^T(n) = \frac{1}{2(X_n^2 + Y_n^2)^{3/2}} \sum_{i=1}^{n} \quad (X_n Y_{i-1} - Y_n X_{i-1})(-X_n Y_i + Y_n X_i)$$
$$(X_n X_i + Y_n Y_i - Y_n Y_{i-1} - X_{i-1} X_n)$$

$$= \frac{1}{2(X_n^2+Y_n^2)^{3/2}} \sum_{i=1}^{n} \begin{vmatrix} X_n & Y_n \\ X_{i-1} & Y_{i-1} \end{vmatrix} \begin{vmatrix} X_n & Y_n \\ X_i & Y_i \end{vmatrix}$$

$$< X_n, Y_n > \cdot < X_i - X_{i-1}, Y_i - Y_{i-1} > .$$

We notice that $R(n) = \sqrt{X_n^2 + Y_n^2}$ is invariant under Euclidean motions and if we omit the factor in front of the sum, we still have a Euclidean invariant.

It should be noted that this formula has a geometric interpretation which gives us some insight into the second invariant. We see that the two determinants give the area of the parallelograms defined by the vectors $< X_n, Y_n >$ and $< X_{i-1}, Y_{i-1} >$, and $< X_n, Y_n >$ and $< X_i, Y_i >$ respectively. Remembering that we have a factor of one half in front of our sum, we can consider one of these determinants to be the area of the triangle defined by the respective vectors. We can consider the scalar product of $< X_n, Y_n >$ and $< X_i - X_{i-1}, Y_i - Y_{i-1} >$ either as a scalar projection or as a measurement of the magnitude of the two vectors multiplied by the cosine of the angle between them. If we consider the second definition, then we may factor the magnitude of $< X_n, Y_n >$ out of our sum. This will reduce our coefficient of our sum to $\frac{1}{2\sqrt{X_n^2+Y_n^2}}$ and leave us with a factor of $| < X_i - X_{i-1}, Y_i - Y_{i-1} > | \cos(\theta)$ where θ is the angle between the $< X_n, Y_n >$ and $< X_i - X_{i-1}, Y_i - Y_{i-1} >$.

Lastly, we apply the invariantization process to the integral variable $X^{(11)}(t) = \int_0^t X(s)Y(s)\,dX(s)$. We approximate using a trapezoidal rule with n nodes, $1 \leq n \leq N$.

$$X_T^{(11)}(n) = \frac{1}{2}\sum_{i=1}^{n}(X_iY_i + X_{i-1}Y_{i-1})(X_i - X_{i-1}).$$

Prolonging the action of Eq. 3.1, we get

$$\begin{aligned}\overline{X}_T^{(11)}(n) &= \frac{1}{2}\sum_{i=1}^{n}(\overline{X}_i\overline{Y}_i + \overline{X}_{i-1}\overline{Y}_{i-1})(\overline{X}_i - \overline{X}_{i-1}) \\ &= \frac{1}{2}\sum_{i=1}^{n}(\cos(\phi)\sin(\phi)(X_i^2 + X_{i-1}^2) + \cos^2(\phi)(X_iY_i + X_{i-1}Y_{i-1}) \\ &\qquad - \sin^2(\phi)(X_iY_i + X_{i-1}Y_{i-1}) - \cos(\phi)\sin(\phi)(Y_i^2 + Y_{i-1}^2)) \\ &\qquad (\cos(\phi)(X_i - X_{i-1}) - \sin(\phi)(Y_i - Y_{i-1})).\end{aligned}$$

Substituting Eq. 3.2 in to $\overline{X}_T^{(11)}(n)$, we get

$$\begin{aligned}I_3^T(n) = \frac{1}{2(X_n^2 + Y_n^2)^{3/2}}\sum_{i=1}^{n}&(X_nY_n(X_i^2 + X_{i-1}^2 + Y_i^2 + Y_{i-1}^2) + (X_n^2 - Y_n^2)(X_iY_i + X_{i-1}Y_{i-1})) \\ &(X_n(X_i - X_{i-1}) + Y_n(Y_i - Y_{i-1})).\end{aligned}$$

We have four discrete integral invariants

$$\begin{aligned}R(n) &= \sqrt{X_n^2 + Y_n^2} \\ I_1^T(n) &= \frac{1}{2}\sum_{i=1}^{n}(X_{i-1}Y_i - X_iY_{i-1}) \\ I_2^T(n) &= \frac{1}{2(X_n^2 + Y_n^2)^{3/2}}\sum_{i=1}^{n}(X_nY_{i-1} - Y_nX_{i-1})(-X_nY_i + Y_nX_i) \\ &\qquad (X_nX_i + Y_nY_i - Y_nY_{i-1} - X_{i-1}X_n) \\ I_3^T(n) &= \frac{1}{2(X_n^2 + Y_n^2)^{3/2}}\sum_{i=1}^{n}(X_nY_n(X_i^2 + X_{i-1}^2 + Y_i^2 + Y_{i-1}^2) + (X_n^2 - Y_n^2)(X_iY_i + X_{i-1}Y_{i-1})) \\ &\qquad (X_n(X_i - X_{i-1}) + Y_n(Y_i - Y_{i-1})).\end{aligned} \tag{3.3}$$

When the invariantization process with respect to the special Euclidean group was applied to the integral variables $Y(t)$ the result was a constant invariant. Since constant invariants will

not be of use to us in curve matching applications, we disregard these invariants.

Now let us derive invariants using Simpson's rule for the discretization rather than the trapezoidal rule. We are only able to apply Simpson's rule in situations where the number of nodes is even, so we will assume that this is the case.

First, we consider

$$\begin{aligned} X^{(01)} &= \int_0^t Y(s)\, dX(s) \\ &= \int_0^t Y(s)X'(s)\, ds. \end{aligned}$$

To discretize this expression we will require a discrete approximate to the first derivative of $X(s)$. We choose to approximate with Taylor's expansions about three points. We chose this approximation for two reasons. Our quadrature technique is second order and using Taylor's expansions about three points will be of the same order. The second order Taylor approximation also has the benefit of only using the same subinterval of nodes as Simpson's rule. This allows the approximation of the integrand on each subinterval to be reliant only upon itself. This will help prevent an error in one subinterval from pervading throughout the entire approximation.Using these approximations with Simpson's Rule allows us to write

$$\begin{aligned} X^{(01)}(n) = \frac{1}{3} \sum_{i=1:2:n-2} & Y_i \left(\frac{-3}{2} X_i + 2X_{i+1} - \frac{1}{2} X_{i+2} \right) + 4Y_{i+1} \left(\frac{-1}{2} X_i + \frac{1}{2} X_{i+2} \right) \\ & + Y_{i+2} \left(\frac{1}{2} X_i - 2X_{i+1} + \frac{3}{2} X_{i+2} \right) \end{aligned}$$

where $1 \leq n \leq N$ and $i = 1 : 2 : n-2$ means i is increased by two at each iteration until $i = n-2$. Now we rotate, substitute in the associated transformations, and simplify.

$$\overline{X^{(01)}(n)} = \sum_{i=1:2:n-2} \left(\frac{-1}{2} Y_i^2 + \frac{1}{2} X_i^2 + \frac{1}{2} Y_{i+2}^2 - \frac{1}{2} X_{i+2}^2 \right) \cos(\phi) \sin(\phi)$$

$$
\begin{aligned}
&+ (Y_{i+2}X_{i+2} - Y_iX_i)\cos^2(\phi) \\
&+ \frac{2}{3}\left(Y_iX_{i+1} - Y_{i+2}X_{i+1} - Y_{i+1}X_i + Y_{i+1}X_{i+2}\right) \\
&+ \frac{1}{2}\left(Y_iX_i - Y_{i+2}X_{i+2}\right) + \frac{1}{6}\left(Y_{i+2}X_i - Y_iX_{i+2}\right).
\end{aligned}
$$

By invariantizing, we obtain

$$
\begin{aligned}
I_1^S(n) = \sum_{i=1:2:n-2} &\left(\frac{-1}{2}Y_i^2 + \frac{1}{2}X_i^2 + \frac{1}{2}Y_{i+2}^2 - \frac{1}{2}X_{i+2}^2\right)\left(\frac{-Y_nX_n}{Y_n^2 + X_n^2}\right) \\
&+ (Y_{i+2}X_{i+2} - Y_iX_i)\left(\frac{Y_n^2}{Y_n^2 + X_n^2}\right) \\
&+ \frac{2}{3}\left(Y_iX_{i+1} - Y_{i+2}X_{i+1} - Y_{i+1}X_i + Y_{i+1}X_{i+2}\right) \\
&+ \frac{1}{2}\left(Y_iX_i - Y_{i+2}X_{i+2}\right) + \frac{1}{6}\left(Y_{i+2}X_i - Y_iX_{i+2}\right).
\end{aligned}
$$

Similarly, we may derive invariant, numerical approximations to $I_2(t)$ and $I_3(t)$ using Simpson's rule.

We have derived discrete integral invariants, using the trapezoidal rule and Simpson's rule, by applying to our integral variables before we begin the invariantization process as detailed in 3.1.1.

3.1.2 Discrete Invariant Signatures

In Chapter 2, we discussed the benefits of using invariants to construct signatures to be used for curve recognition and matching. Discrete invariant signatures are obtained in a similar way be simply plotting one discrete invariant against another.

We first demonstrate how discrete signatures appear graphically.

Example 3.1.1. Let us return to the spiral given in 2.2.4, $(x(\theta), y(\theta)) = (\theta\cos(\theta), \theta\sin(\theta)), \theta \in$

$[0, 4\pi]$. We will now consider this spiral given discretely by 200 points along the curve. This discrete representation is shown in Figure 3.1.

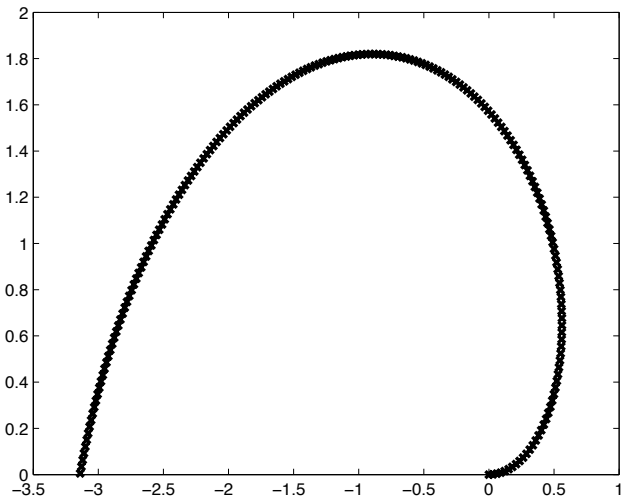


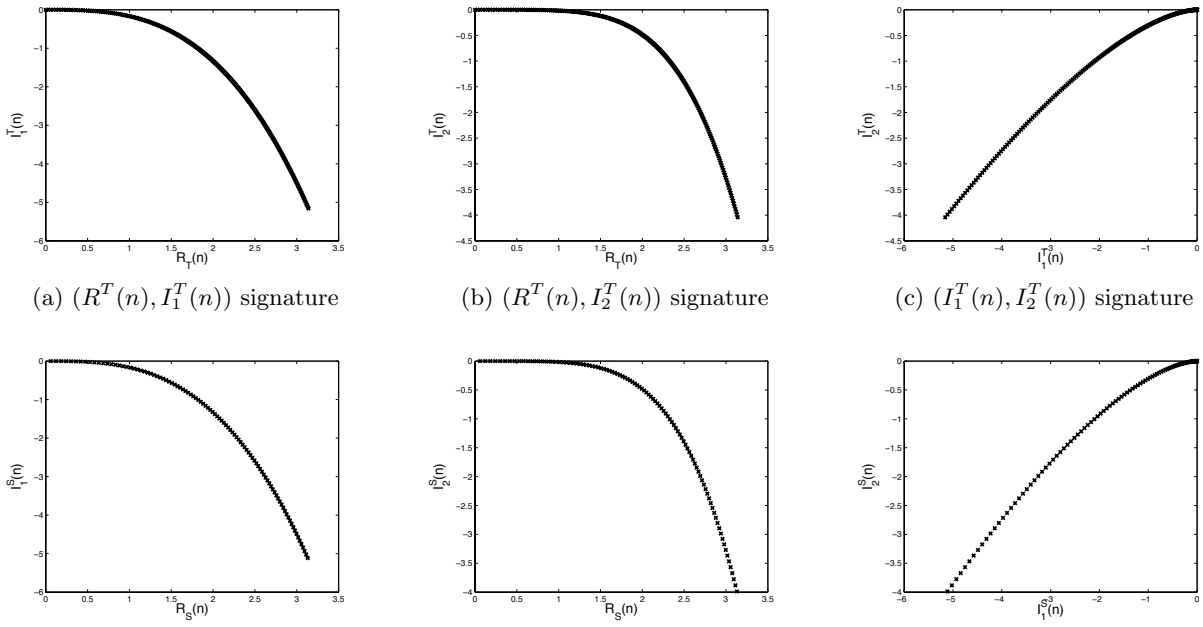
Figure 3.1: Plot of 200 points along the curve $(\theta\cos(\theta), \theta\sin(\theta)), \theta \in [0, \pi]$.

Shown in Figure 3.2 are the discrete invariant signatures of this curve.

We note that our discrete invariants do not require any specific assumption on the step sizes, neither regarding uniformity or density of nodes which is an significant asset in practice.

For a given smooth curve, the discrete invariants will converge to the associated continuous invariants as the number of sampled values N goes to infinity. We make this statement more precise in the next section. It should be noted that the usefulness of our new discrete invariants is not linked to their approximating properties. These latter properties are, however, useful when analyzing the dependence of sampling on the discrete invariants, see 3.1.3.

We have derived discrete invariants approximating $R(t), I_1(t)$, $I_2(t)$, and $I_3(t)$ using both the trapezoidal rule and Simpson's rule. It is possible to derive discrete invariants with more additional quadrature rules. Hereafter, we choose to work with the simplest versions of dis-



(a) $(R^T(n), I_1^T(n))$ signature (b) $(R^T(n), I_2^T(n))$ signature (c) $(I_1^T(n), I_2^T(n))$ signature

(d) $(R^S(n), I_1^S(n))$ signature (e) $(R^S(n), I_2^S(n))$ signature (f) $(I_1^S(n), I_2^S(n))$ signature

Figure 3.2: Discrete invariant signatures of 200 points along $(x(\theta), y(\theta)) = (\theta\cos(\theta), \theta\sin(\theta))$ for $\theta \in [0, \pi]$.

crete invariants, i.e., those derived from the trapezoidal rule: $I_1^T(n), I_2^T(n)$, and $I_3^T(n)$. We will henceforth denote the invariants simply as $\tilde{R}(n), \tilde{I}_1(n), \tilde{I}_2(n)$, and $\tilde{I}_3(n)$. They are algebraically simpler than invariants derived from other quadratures and work equally well when used for curve analysis. The theoretical results derived below apply with only obvious changes to the analysis of discrete integral invariants based on other quadratures.

3.1.3 Consistency of the Discrete Invariant

We want to consider how choosing different samplings of the same curve may effect the $(\tilde{R}(n), \tilde{I}_1(n))$ signature. To analyze this difference, we first examine the error incurred when we choose to use a sampling of points along a curve to approximate $R(t)$ and $I_1(t)$ rather than use its parametric formula.

Consider a curve $\Gamma_E = \{f(t) = (x(t), y(t)) : t \in [0, T]\}$ and a sampling of this curve,

$\Gamma_D = \{(x_i, y_i) : x_i = x(t_i), y_i = y(t_i), t_i \in [0,T], i = 0, \ldots, N\}$, so the points in Γ_D are on Γ_E. We consider the curve signatures created by pairing R and I_1 and define the distance between the associated curve signatures, S_E and S_D, to be

$$\begin{aligned} \|S_E - S_D\| &= \max_i \|S_E(t_i) - S_D(i)\| \\ &= \max_i \sqrt{(R(t_i) - \tilde{R}(i))^2 + (I_1(t_i) - \tilde{I}_1(i))^2}. \end{aligned} \tag{3.4}$$

However, since $(x_i, y_i) = (x(t_i), y(t_i))$, the nodes on the sampling of the curve are also on the parametric curve. Thus, $R(t_i) = \tilde{R}(i)$, so this expression simplifies to

$$\begin{aligned} \|S_E - S_D\| &= \max_i \sqrt{(I_1(t_i) - \tilde{I}_1(i))^2} \\ &= \max_i |I_1(t_i) - \tilde{I}_1(i)| \\ &= \max_i \left| \int_0^{t_i} x(s)\, dy(s) - \frac{1}{2} x(t_i) y(t_i) - \frac{1}{2} \sum_{j=1}^{i} (x_{j-1} y_j - x_j y_{j-1}) \right|. \end{aligned}$$

But now since $x_i = x(t_i)$, $y_i = y(t_i)$ and $(x_0, y_0) = (0,0)$, we have

$$\begin{aligned} \frac{1}{2} \sum_{j=1}^{i} (x_{j-1} y_j - x_j y_{j-1}) + \frac{1}{2} x_i\, y_i &= \frac{1}{2} (x_0 y_1 - x_0 y_0 + x_1 y_1 - x_1 y_0) \\ &\quad + \frac{1}{2} (x_1 y_2 - x_1 y_1 + x_2 y_2 - x_2 y_1) \\ &\quad + \ldots \\ &\quad + \frac{1}{2} (x_{i-1} y_i - x_{i-1} y_{i-1} + x_i y_i - x_i y_{i-1}) \\ &= \frac{1}{2} \sum_{j=1}^{i} (x_j + x_{j-1})(y_j - y_{j-1}), \end{aligned}$$

and thus

$$\|S_E - S_D\| = \max_i \left| \int_0^{t_i} x(s)\, dy(s) - \frac{1}{2} \sum_{j=1}^{i} (x_j + x_{j-1})(y_j - y_{j-1}) \right|. \tag{3.5}$$

In other words, the discrepancy between S_E and S_D corresponds exactly to the error of integration for the composite trapezoidal rule. The general result recalled at the beginning of this chapter assumes uniform meshes. If the mesh is not uniform, we observe

$$\int_{y_{j-1}}^{y_j} x\,dy = \frac{x_j + x_{j-1}}{2}(y_j - y_{j-1}) - \frac{1}{2}\int_{y_{j-1}}^{y_j} x''(y)(y_j - y)(y - y_{j-1})\,dy.$$

Therefore, we have for x of class $\mathcal{C}^2$

$$\begin{aligned}\int_0^{t_i} x(s)\,dy(s) - \frac{1}{2}\sum_{j=1}^{i}(x_j + x_{j-1})(y_j - y_{j-1}) &= \sum_{j=1}^{i}\left\{\int_{t_{j-1}}^{t_j} x(s)\,dy(s) - \frac{1}{2}(x_j + x_{j-1})(y_j - y_{j-1})\right\}\\ &= -\frac{1}{2}\sum_{j=1}^{i}\int_{y_{j-1}}^{y_j} x''(y)(y_j - y)(y - y_{j-1})\,dy\\ &\leq \frac{1}{12}|x''|_\infty \sum_{j=1}^{i}|y_j - y_{j-1}|^3,\end{aligned}$$

with $|x''|_\infty = \sup_{t\in(0,T)}|\frac{\ddot{x}\dot{y}-\dot{x}\ddot{y}}{\dot{y}}|$ and where the dots denote derivatives with respect to the parameter t. The above representation is invalid if $\dot{y}$ vanishes. However, a similar formula can be derived by expressing I_1 as an integral with respect to x, instead of y, through integration by parts. For a regular smooth curve, the corresponding estimate will be locally valid since $\dot{x}$ and $\dot{y}$ can't simultaneously vanish. Therefore, working with both x and y based representations, we have

$$\|S_E - S_D\| \leq \frac{1}{12}\min\left\{|x''|_\infty \sum_{j=1}^{N}|y_j - y_{j-1}|^3, |y''|_\infty \sum_{j=1}^{N}|x_j - x_{j-1}|^3\right\}. \tag{3.6}$$

The influence of two different samplings can be estimated in a similar way. To simplify the analysis, we assume that both samplings have the same number of nodes, namely N. If S_D is the signature obtained with the above sampling nodes $t_1, \ldots, t_N$ and $\bar{S}_D$ is the signature

obtained from evaluations at $\bar{t}_1, \ldots, \bar{t}_N$, we want to estimate

$$\|S_D - S_{\bar{D}}\|^2 \leq \max_i A^2 + \max_i B^2$$

where

$$\begin{aligned}
A &= \tilde{R}(t_i) - \tilde{R}(\bar{t}_i) \\
&= \sqrt{(x_i - \bar{x}_i)^2 + (y_i - \bar{y}_i)^2} \\
B &= \frac{1}{2}\sum_{j=1}^{i}(x_{j-1}y_j - x_j y_{j-1}) - \frac{1}{2}\sum_{j=1}^{i}(\bar{x}_{j-1}\bar{y}_j - \bar{x}_j\bar{y}_{j-1}) \\
&= \frac{1}{2}\sum_{j=1}^{i}(x_j + x_{j-1})(y_j - y_{j-1}) - x_i y_i - \sum_{j=1}^{i}(\bar{x}_j + \bar{x}_{j-1})(\bar{y}_j - \bar{y}_{j-1}) + \bar{x}_i\bar{y}_i.
\end{aligned}$$

It is easily shown that the bound on A is dependent on the difference in the samples, so we focus on the B term. Proceeding as above, we get

$$\begin{aligned}
B = \frac{1}{2}\max_i \Bigg| &\int_{\bar{y}_i}^{y_i} x\,dy - x_i y_i + \bar{x}_i\bar{y}_i \\
&+ \sum_{j=1}^{i} x''(\xi_j)\int_{y_{j-1}}^{y_j}(y_j - y)(y - y_{j-1})dy - \sum_{j=1}^{i} x''(\zeta_j)\int_{\bar{y}_{j-1}}^{\bar{y}_j}(\bar{y}_j - y)(y - \bar{y}_{j-1})dy\Bigg|,
\end{aligned}$$

where ξ_j is between y_{j-1} and y_j and ζ_j is between $\bar{y}_{j-1}$ and $\bar{y}_j$. This yields

$$\begin{aligned}
B \leq\ & |x|_\infty \max_i |y_i - \bar{y}_i| + \frac{1}{2}|y|_\infty \max_i |x_i - \bar{x}_i| \\
&+ \frac{1}{12}\sum_{j=1}^{N}\left|x''(\xi_j)(y_j - y_{j-1})^3 - x''(\zeta_j)(\bar{y}_j - \bar{y}_{j-1})^3\right|,
\end{aligned}$$

and where $|x|_\infty$ and $|y|_\infty$ are the largest values of $|x(t)|$ and $|y(t)|$ for $t \in [0, T]$. This leads to

$$\|S_D - S_{\bar{D}}\| = \mathcal{O}(\max_i |x_i - \bar{x}_i| + \max_i |y_i - \bar{y}_i|),$$

which indicates that the discrepancy of the signatures due to sampling is simply proportional to the discrepancy in the sampling nodes.

3.2 Using Signatures for Matching Curves with a Fixed Initial Point

Assume we are given two sets of points each representing a curve. The connectivity of the curve is implied by the order in which the coordinates are given. We will denote the first discrete curve as $\gamma = (x_i, y_i), i = 1, \dots N$ and the second discrete curve as $\overline{\gamma} = (\overline{x}_j, \overline{y}_j), j = 1, \dots, M$.

Compute Discrete Signatures

The first step of our curve matching algorithm is to compute invariants for each discrete curve. The choice of invariants is determined by which group action we consider. Here we focus on the special euclidean group and consider the signatures based on R, I_1 and I_2. As previously mentioned, invariants are dependent upon choice of initial point and thus, to compute signatures, an initial point on the discrete curve must be identified. For open curves, there are two obvious choices for initial point. We will first choose (x_1, y_1) on γ as our initial point. The resulting invariants we will denote as $[\tilde{R}(i), \tilde{I}_1(i), \tilde{I}_2(i)], i = 1, \dots, N$. Letting $(\overline{x}_1, \overline{y}_1)$ serve as our initial point on $\overline{\gamma}$, we will compute the invariants $[\tilde{\overline{R}}(j), \tilde{\overline{I}}_1(j), \tilde{\overline{I}}_2(j)], j = 1, \dots, M$.

We cannot assume that the two discrete curves listed with the same orientation. Failing to account for this will prevent the algorithm from identifying matching curves. The following example illustrates the dependence of discrete signatures of curves upon initial point.

Example 3.2.1. Consider the curve, $\hat{\gamma} = (x, 3x\cos(x)), x \in [0, 10]$, given discretely by 200 points along the curve as pictured in Figure 3.3.

We compute the (R, I_1), (R, I_2) and (I_1, I_2) signatures first with the initial point $(0, 0)$ then again with $(10, 30\cos(10))$ as initial point. These signatures are shown in Figure 3.4. Despite being signatures of the same discrete curve, we see that the signatures do not correspond when

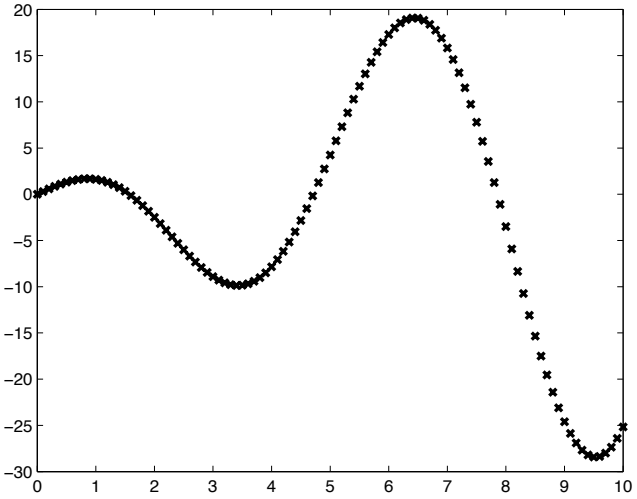


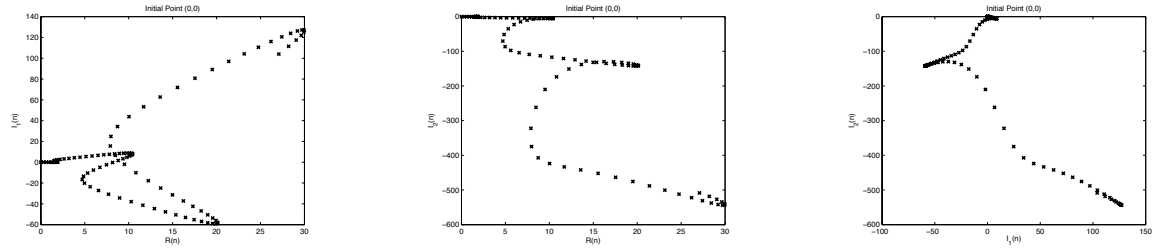
Figure 3.3: Discretely given $\hat{\gamma} = (x, 3x\cos(x)), x \in [0, 10]$.

we do not use the same initial point for the computation of invariants.

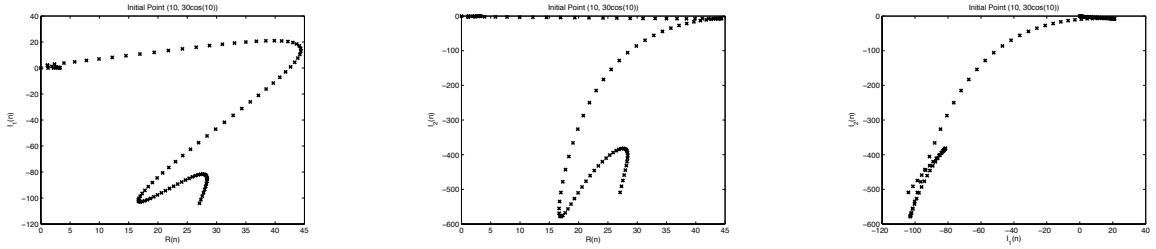
We need to compute the invariants associated with both orientations for one of the open discrete curves involved in our matching problem. We arbitrarily choose the first curve, but we note that choosing the second curve would not change the matching algorithm or the results. Thus, we reverse the direction of the first curve and compute the invariants $[\tilde{R}^R(n), \tilde{I}_1^R(n), \tilde{I}_2^R(n)]$.

The applications we consider below (character recognition and automated puzzle solving) only call for considering the special Euclidean group. As a result, for this description, we choose to use the $(\tilde{R}(n), \tilde{I}_1(n))$ signature and remark that the comparison algorithm does not change when a different choice of discrete invariants is used.

We now have to compare curve signatures. After we adopt an appropriate notion of distance, the idea is, for open curves, to start at one of the ends of the curves, and then simultaneously follow each signature curve as long as the distance remains below a certain threshold. The "length" of the match is then recorded. Closed curves have no natural starting points; this presents additional challenges that we address below.



(a) Initial point $(0, 0)$.



(b) Initial point $(10, 30\cos(10))$.

Figure 3.4: Signature curves associated with differing choice of endpoint used for initial point of the open curve, $(x, 3x\cos(x)), x \in [0, 10]$.

We start by reviewing classical notions of distance for sets and curves.

Distance

We now have to decide what distance measure we would like to use. Two classical distances between curves are the Hausdorff distance and the Fréchet distance.

Definition 3.2.2. Let P and Q be two sets of points in $\mathbb{R}^2$. The directed Hausdorff distance from P to Q is

$$h(P, Q) = \max_{p \in P} \min q \in Q \|p - q\|,$$

where $\|\cdot\|$ is the usual Euclidean norm.

The **Hausdorff distance** between P and Q is

$$H(P, Q) = \max\{h(P, Q), h(Q, P)\}.$$

Definition 3.2.3. The **Fréchet distance** between two curves is defined as

$$F(P,Q) = \inf_{\alpha,\beta} \max_{t\in[0,1]} \|A(\alpha(t)) - B(\beta(t))\|,$$

where $P, Q : [0,1] \to \mathbb{R}^2$ are parametrizations of the two curves and $\alpha, \beta : [0,1] \to [0,1]$ range over all continuous and monotone increasing functions.

The **discrete Fréchet distance**, developed by Eiter and Mannila [32], uses the vertices of polygonal curve approximations to estimate the Fréchet distance.

We are concerned with how the distance between two directed curves changes as we proceed along them; our measure must thus take the direction of the curves into account. Neither the Hausdorff nor the Fréchet distance satisfies this requirement. The Hausdorff distance allows for backwards travel along a curve. The Fréchet distance does not allow backwards travel, but it does allow for varying speeds of travel along the curves. It is also possible to remain stationary on one curve while progressing along the other.

The following example illustrates why both the Hausdorff and Fréchet distances fail to satisfy our requirements and may lead to false positives when identifying members of curve classes.

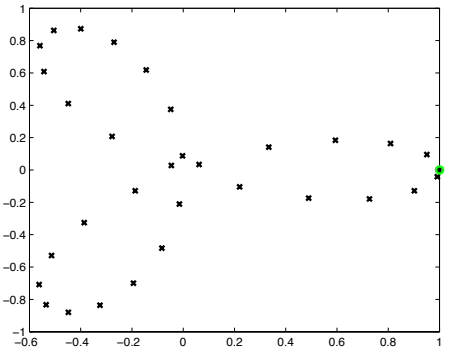
Example 3.2.4. We take 32 points generated by a uniform t along a three leafed rose over two different domains.

$$(t, \cos(3t)), t \in [0, \pi] \tag{3.7}$$

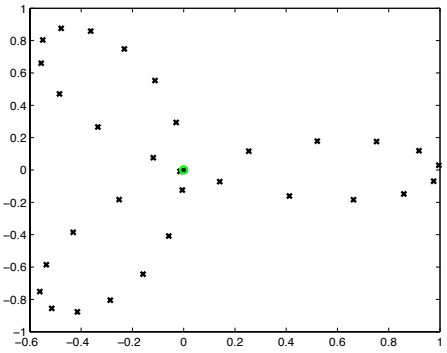
$$(t, \cos(3t)), t \in \left[\frac{\pi}{2}, \frac{3\pi}{2}\right] \tag{3.8}$$

One set of points, Eq. 3.7, has initial point $(0,1)$. We refer to this as curve one. The second set, Eq. 3.8, has initial point $\left(\frac{\pi}{2}, 0\right)$. We refer to this as curve two. These sets, with initial points highlighted in green, and their $(R(n), I_1(n))$ signatures are shown in Figure 3.5.

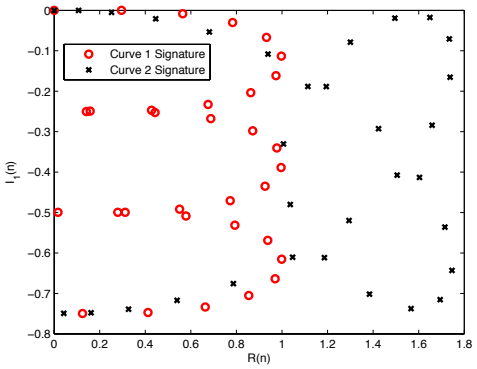
From their initial points at $(0,0)$, the signature curves quickly move away from each other.



(a) Discrete curve given in Eq. 3.7.



(b) Discrete curve given in Eq. 3.8.



(c) Plots of $(\tilde{R}(n), \tilde{I}_1(n))$ signatures of curve one and curve two.

Figure 3.5: Plots of Eq. 3.7 and Eq. 3.8 with initial points highlighted in green and plots of the respective $(\tilde{R}(n), \tilde{I}_1(n))$ signatures.

While the signatures move apart if we consider progressing along them node by node at the same rate, the signature of curve one remains close to some nodes of the signature of curve two. The Hausdorff and Fréchet distances consider the shortest distance to any node on the other signature and would lead here to a "false positive". For our purposes, we consider these two signatures to be quite different and would like for the distance between the two to reflect this, thus neither the Hausdorff nor the Fréchet distance will be a good distance measure for our applications.

Secondly, we would like to be able to consider how the distance between the curves changes as we progress along them. For this reason, we prefer a measure that results in a vector of distances at various nodes along the curves rather than one result for the entire length. The easiest way to generate this vector is to measure the Euclidean distance between the nodes on each curve. Taking the distance between nodes would ensure that our distances were directed. This criterion, however, may be misleading and lead to false negatives, as illustrated in the next example.

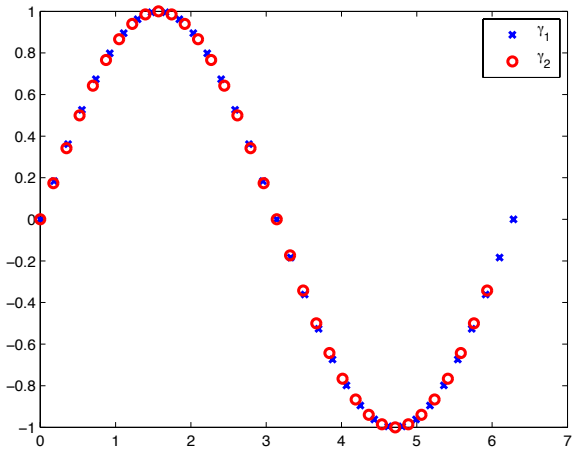


Figure 3.6: Plots of two samplings of $\sin(x)$.

Example 3.2.5. Let us consider two different samplings of $f(x) = \sin(x)$. Our first curve is $\gamma_1 = (x_i, \sin(x_i))$, $x_i = (i-1)\frac{\pi}{17}$ where $i = 1, \ldots, 35$. Our second curve is $\gamma_2 = (\bar{x}_j, \sin(\bar{x}_j))$, $\bar{x}_j = (j-1)\frac{\pi}{18}$ where $j = 1, \ldots, 35$. The plots of γ_1 and γ_2 are pictured in Figure 3.6.

Since γ_1 and γ_2 are two samplings of the same curve, we would like for our algorithm to detect this. The signatures of the two samplings are plotted in Figure 3.7. Both samplings of the curve have 35 nodes, so we may measure the distance between the signatures by taking the distance between like numbered nodes. If we measure the node-to-node distance between

each pairing of the signatures, we obtain the results shown in Table 3.1. Therefore, a notion of distance based purely on nodal values misses the fact that there is a common underlying curve.

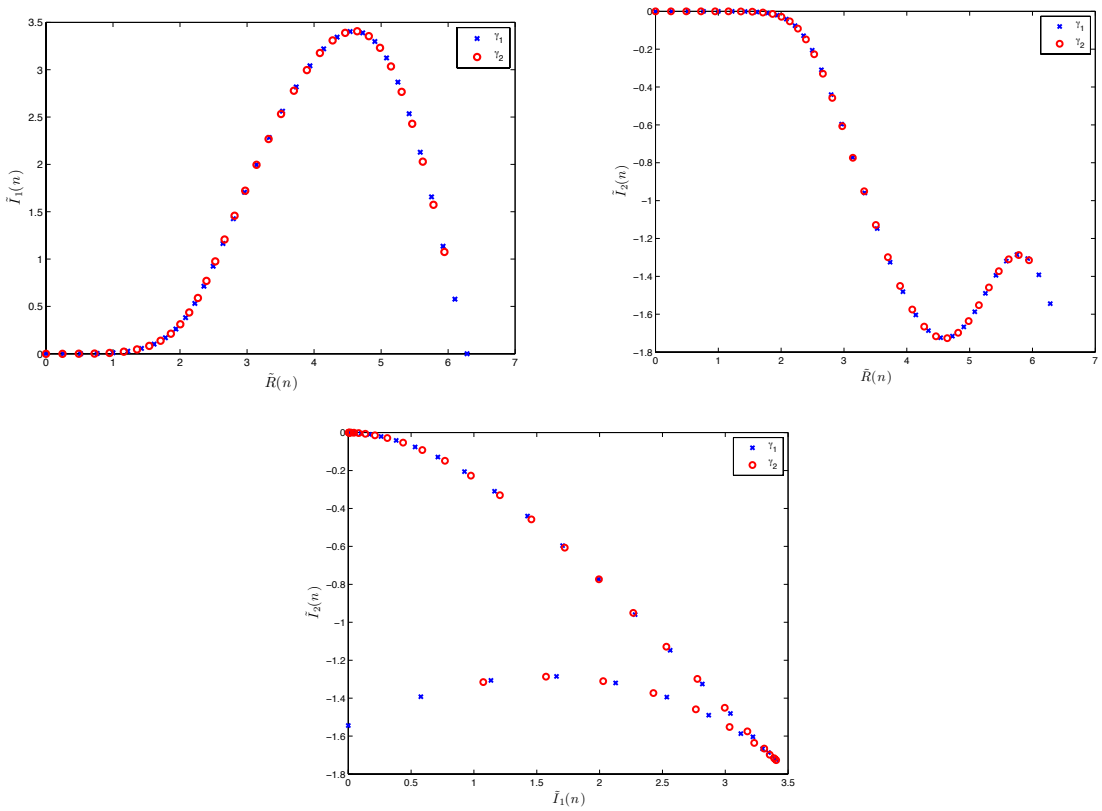


Figure 3.7: Signature curves associated with different samplings of $(x, \sin(x))$.

Table 3.1: Nodal distances of signatures associated with different samplings of $\sin(x)$.

Signature	$(\tilde{R}(n), \tilde{I}_1(n))$	$(\tilde{R}(n), \tilde{I}_2(n))$	$(\tilde{I}_1(n), \tilde{I}_2(n))$
Nodal Distances	0	0	0
	0.0144	0.0144	0
	0.02812	0.02812	0.0001177
	0.04054	0.04053	0.0006951
	0.05121	0.05116	0.002265
	0.05996	0.05971	0.005497
	0.06707	0.06614	0.01113
	0.07342	0.07068	0.01991
	0.08062	0.07389	0.03251
	0.09078	0.07666	0.04947
	0.1058	0.08032	0.07119
	0.1267	0.0867	0.09788
	0.1534	0.09788	0.1295
	0.1848	0.1156	0.1655
	0.2194	0.1402	0.2049
	0.2552	0.1704	0.2455
	0.2898	0.2038	0.2846
	0.3206	0.2367	0.3184
	0.345	0.2657	0.3428
	0.3603	0.2874	0.3536
	0.3645	0.2995	0.3471
	0.3564	0.3011	0.3205
	0.3361	0.2933	0.272
	0.3069	0.28	0.2013
	0.277	0.2673	0.1097
	0.2627	0.2627	0.001021
	0.2846	0.2701	0.1233
	0.3501	0.2866	0.2553
	0.4492	0.3037	0.3904
	0.5682	0.313	0.5241
	0.696	0.3105	0.6532
	0.8236	0.3014	0.7763
	0.9428	0.3027	0.8929
	1.046	0.3367	1.002
	1.127	0.4092	1.099

Though all three pairings of signatures result from samplings of the same curve and the signatures have the same shape, we see that this likeness is not reflected in the nodal distance measures. This occurs as a result of the fact that we are only measuring distance between nodes and not taking into account the fact that two signatures may be exactly the same when we use linear interpolants to connect the nodes regardless of the nodal distances. This node-to-node measurement method will prefer two signatures with nodes that are close over two signatures where one interpolated signature lays directly on top of the other but the nodes are not the same.

We would like for our measure to result in small values when two interpolated signatures remain close as we progress along them. For this reason, we base our distance measure on the **distance from the nodes of one signature to the piecewise linear interpolants of the other signature and vice versa**, see Figure 3.8.

Any notion of closeness for discrete curves implicitly relies on the properties of the underlying meshes. For a discrete curve represented by N nodes, we define the average size of the mesh as

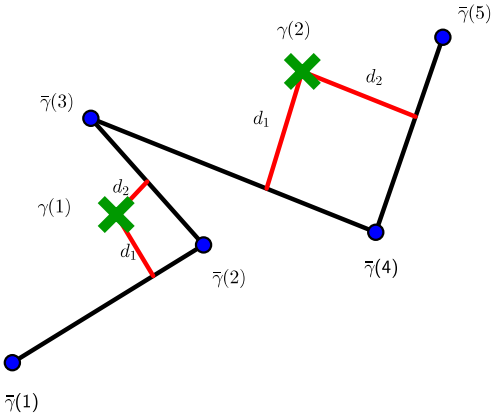
$$h = \frac{1}{N}\sum_{i=1}^{N} h_i, \text{ where } h_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}, i = 1, \ldots, N.$$

When comparing curves, we assume that

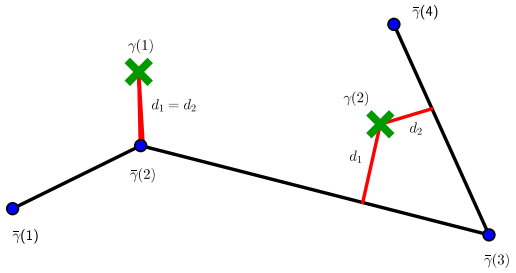
1. the average mesh sizes are "similar",
2. the standard deviation of the mesh sizes (i.e., $\sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(hi - h)^2}$) are "small".

We do not make these requirements more precise as (i) their fulfillment on the original curves does not imply they are valid on the signatures and (ii) in practice, we use linear interpolation of the signatures and (nearly) uniform redistribution of the nodes [1]. We have found that the algorithmic simplifications gained by working with nearly uniform meshes far outweigh the price of re-interpolation of the signatures.

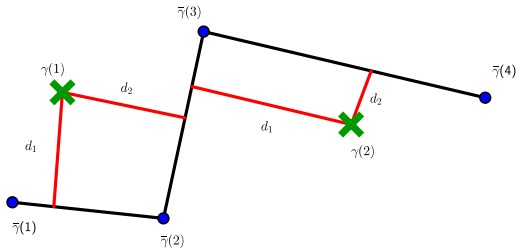
[1] The notion of quasi-uniformity of meshes in numerical analysis is an asymptotic property. Here, the (original) mesh size is fixed.



(a) Situation where minimum distance is to second interpolant.



(b) Situation where minimum distance is to a node.



(c) Situation where minimum distance is to the first interpolant.

Figure 3.8: Illustration of node to linear interpolant measuring scheme.

We now define our measure. Consider two signatures given by nodes, $(\tilde{R}^{\gamma_1}(i), \tilde{I}_1^{\gamma_1}(i)), i = 1, \ldots, N_1$ and $(\tilde{R}^{\gamma_2}(j), \tilde{I}_1^{\gamma_2}(j)), j = 1, \ldots, N_2$. Suppose we measure from nodes on γ_1 to piecewise linear interpolants of γ_2. The result is a vector of length N_1 such that the i^{th} entry is the minimum distance from the i^{th} node of S_1 to a linear interpolant of S_2. As previously stated, we do not expect the nodes on the two signatures to perfectly "line up" in spite of the above mentioned redistribution process. Similarly, we do not expect the i^{th} node of one signature to perfectly align with the i^{th} piecewise linear interpolant on the other signature. We do, however, want to find the minimum distance to interpolants in the same region. As a result, we will take the minimum distance to *two* consecutive linear interpolants.

We also note that the two signatures may not have the same number of nodes. If signature one has N_1 nodes and signature two N_2 nodes, then we will let $N = \min\{N_1, N_2\}$. We will then find the distance associated with each node from the first node to the N^{th} node on one signature to the interpolants on the other signature. These distances will be stored for each computation and computed for the nodes on each signature, so our results when comparing two discrete curves with fixed intitial points will be two distance vectors of length N.

The process of determining the distance from nodes on γ to $\overline{\gamma}$ is outlined.

procedure CurveDist

Input:

node(i,γ), node(i,$\bar{\gamma}$), $i = 1, \ldots, N$

Output:

d(i), $i = 1, ..., N$, distance vector

$k = \bar{k} = 0$

for $i = 1 : N$ **do**

construct $\overline{\text{int1}}$ = interp($i + \bar{k}$, $\bar{\gamma}$) and $\overline{\text{int2}}$= interp($i + \bar{k} + 1$, $\bar{\gamma}$)

compute d_1 = dist(node(i, γ), $\overline{\text{int1}}$) and d_2 = dist(node(i, γ), $\overline{\text{int2}}$)

set $D(i) = \min(d_1, d_2)$

if $d_2 = D(i)$ **then**

$\bar{k} = \bar{k} + 1$

end if

construct int1 = interp($i + k$, γ) and int2 = interp($i + k + 1$, γ)

compute d_1 = dist(node(i, $\bar{\gamma}$), int1) and d_2 = dist(node(i, $\bar{\gamma}$), int2)

set $\bar{D}(i) = \min(d_1, d_2)$

if $d_2 = \bar{D}(i)$ **then**

$k = k + 1$

end if

set $d(i) = \min\{D(i), \bar{D}(i)\}$

end for

end procedure

In the above procedure, interp(i, γ) refers to the linear segment between nodes $i-1$ and i of γ, i.e.,

$$\text{interp}(i,\gamma) = \left\{ (1-\lambda) \begin{bmatrix} x_{i-1} \\ y_{i-1} \end{bmatrix} + \lambda \begin{bmatrix} x_i \\ y_i \end{bmatrix}, \lambda \in [0,1] \right\}.$$

Further, dist refers to the Euclidean distance. We have for instance

$$\text{dist}(\text{node}(i,\bar{\gamma}), \text{interp}(\text{i},\gamma)) = \min_{P \in \text{ interp}(\text{i},\gamma)} \|\text{node}(i,\bar{\gamma}) - P\|.$$

In the special case where the minimum distance between node and linear interpolant is that of the distance between node and the left node of the interpolant, we do not advance forward in the interpolants for the next match, i.e. do not advance k (or $\bar{k}$). Note that it is necessary to compute the symmetric distance, i.e., to consider both nodes on γ and an interpolated $\bar{\gamma}$ as well as nodes on $\bar{\gamma}$ and an interpolated γ. The next example illustrates this.

.

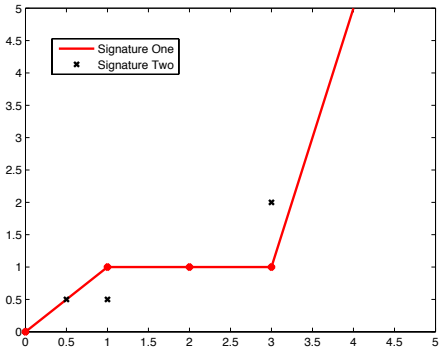
Example 3.2.6. Consider the coarse signatures in Figure 3.9. For the purposes of this example, the original curves are not important, nor is it important which invariants are used in the signature.

The resulting distances are listed in Table 3.2 in one case and in Table 3.3 in the other.

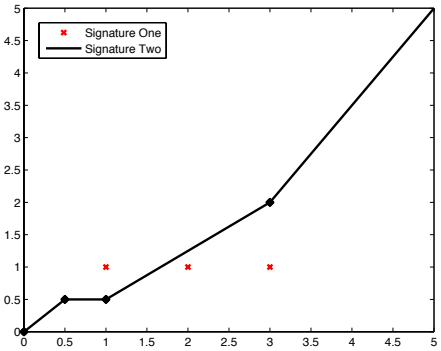
Table 3.2: Distances from nodes on signature one to linear interpolants of signature two.

Associated Node	1	2	3	4	5
Distance	0	0.3215	0.4849	1.0750	1.9904

The need to symmetrize is obvious.



(a) Nodes of signature one plotted with linear interpolants of signature two.



(b) Nodes of signature two plotted with linear interpolants of signature one.

Figure 3.9: Signatures of two curves plotted.

Table 3.3: Distances from nodes on signature two to linear interpolants of signature one.

Associated Node	1	2	3	4	5
Distance	0	0	0.3215	1.2005	1.5273

Choosing the Best Match for a Given Situation

The vector d constructed in the procedure CurveDist has as entries consecutive discrete distances between given curves. Assessments about the closeness of two corresponding curves are based on the analysis of this vector. Our choice of norm to quantify that vector is closely related to the notion of closeness one would use for specific applications. To illustrate this graphically, which of the blue or red curve is closer to the black one in Figure 3.10?

Once more we must decide what is most important in our matching. In the case of character recognition, we would like to consider the distances between the two signatures overall rather than the length of a match before a certain distance tolerance is reached. We may want to consider the overall maximum distance or perhaps the average distance between the two curves. If we choose to consider the overall maximum distance, then we may reject a match such as signature 3 when compared to signature 1 in Figure 3.10 where one signature exactly follows

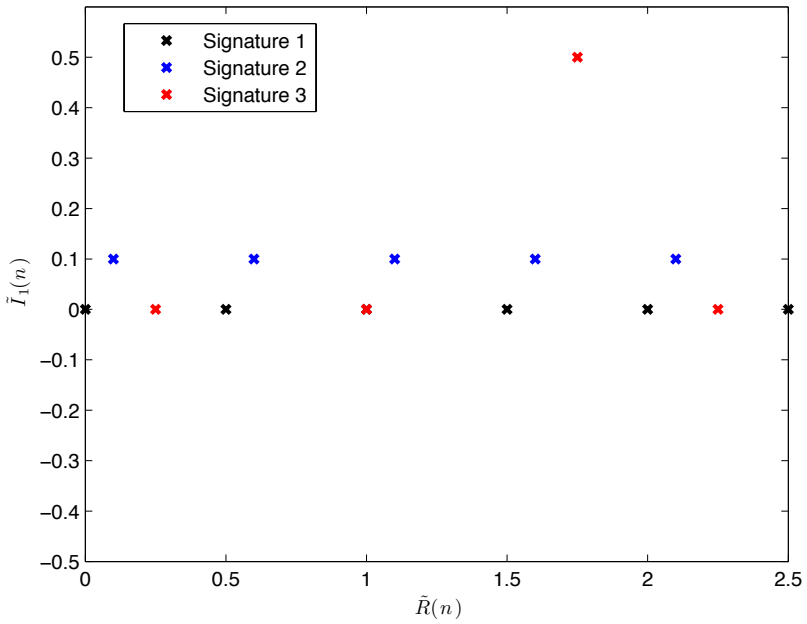


Figure 3.10: Sample signatures illustrating benefits and pitfalls of methods to choose a best match.

the signature to be matched except for one large spike. To avoid this pitfall, we might consider the average distance over the whole length of the signatures. However, using this measure, a signature that maintained a constant distance from the signature to be matched would be considered as good a match as a signature that matched exactly except for one node such as the signatures in Figure 3.10. We must know what is of highest importance to us in our match. For instance, if we value most that the signature is approximated well for the entire length we would have different requirements for choosing a match than in a case where a close match the whole time excepting one node is acceptable. This matter must be decided on a case by case basis depending on the situation.

For character recognition, we define the best match as being the signature that most closely approximates the given signature for its entire length. To decide upon a best match out of several possible curves, we first decide for each signature which orientation is the best match

to a given curve. Once we have decided the best orientation for each possible match, we then use a binary decision tree to determine the optimal match. Decision trees will be discussed in detail later.

After a best match is decided upon, the final transformation (if desired) for the curve is found by minimizing the node-to-node distance function over the region of the match. To create the tightest match possible, we use nodes resulting from a fine mesh sampling, if possible, for our matching algorithm rather than nodes from a coarse sampling that we may use to generate the list of possible matches. A detailed matching algorithm and example of character matching are provided in Chapter 4.

3.3 Using Signatures for Matching Curves without a Fixed Initial Point

It is important to note here that we will, without loss of generality, begin with all of our closed curves labeled in a clockwise orientation from the chosen starting point. While the goal of some matching algorithms is to match two identical closed curves, our chosen application, puzzle assembly, aims to find interlocking curves. In order to be able to match two closed curves in this application, we will have to reverse the direction of one of the curves. Without this reversal, the collection of area to compute the $\tilde{I}_1$ invariant would not occur in the same order.

Compute Discrete Signatures with Each Node on Piece as a Starting Point

Due to the dependence of the signature on its starting point as illustrated in 3.2.1, the problem of matching two closed curves is more difficult than that of two open curves. We can no longer avoid the issue of initial point by simply reorienting one curve and using the invariants associated with both orientations. For a closed curve with N nodes, we need to compute the invariants, $(\tilde{R}, \tilde{I}_1)$, associated with each of the N nodes as the initial point. We introduce the notation $\tilde{I}_1(a, b)$ to denote the first invariant computed with (x_a, y_a) as the initial point to the point

(x_b, y_b). We will discuss how this need affects the complexity of our algorithm later.

To begin the process of finding the best match on two closed curves, we input two matrices listing the edge coordinates of two closed curves. We assume data is given in a meaningful order which represents the clockwise directed curve. The nodes are ordered in such a fashion, i.e. $(x_i, y_i), i = 1, \ldots, N$. We should note that $(x_1, y_1) = (x_N, y_N)$. Once we have computed the invariants associated with this choice of initial points, we must calculate the invariants associated with every other node as the initial point. There are two methods to find these new invariants.

Option One: Rotation of the Data

We can generate the invariants associated using each node as a starting point by iteratively rotating through the matrix of nodes. We compute the discrete invariants associated with each ordering. Once we have computed the invariants for a given initial point, we reorder the nodes with the new starting point being the second point from our previous ordering. We then progress clockwise from the new initial point around the piece and arrange the points in this fashion. Thus, we have removed the original starting point and amended it to the end of the previous vector. To keep creating new closed curves, we append the new initial point to the end of the new vector. This new vector becomes our new closed curve data and we compute the discrete invariants of this newly ordered curve.

We repeat this method until we have rotated through all possible starting points on the closed curve. The invariants are stored in matrices such that the row signifies where that starting point was located in our original arrangement of the data and the column identifies how many nodes we have advanced from the starting node for that row. For instance, the entry $\tilde{I}_1(i, j)$ is the first discrete invariant computed from starting point (x_i, y_i) to node numbered (x_{i+j}, y_{i+j}) if $i + j < N$ or node $(x_{i+j-N+1}, y_{i+j-N+1})$ if $i + j \geq N$ in our original labeling of the data. See Figure 3.11 for a visual representation of this measure.

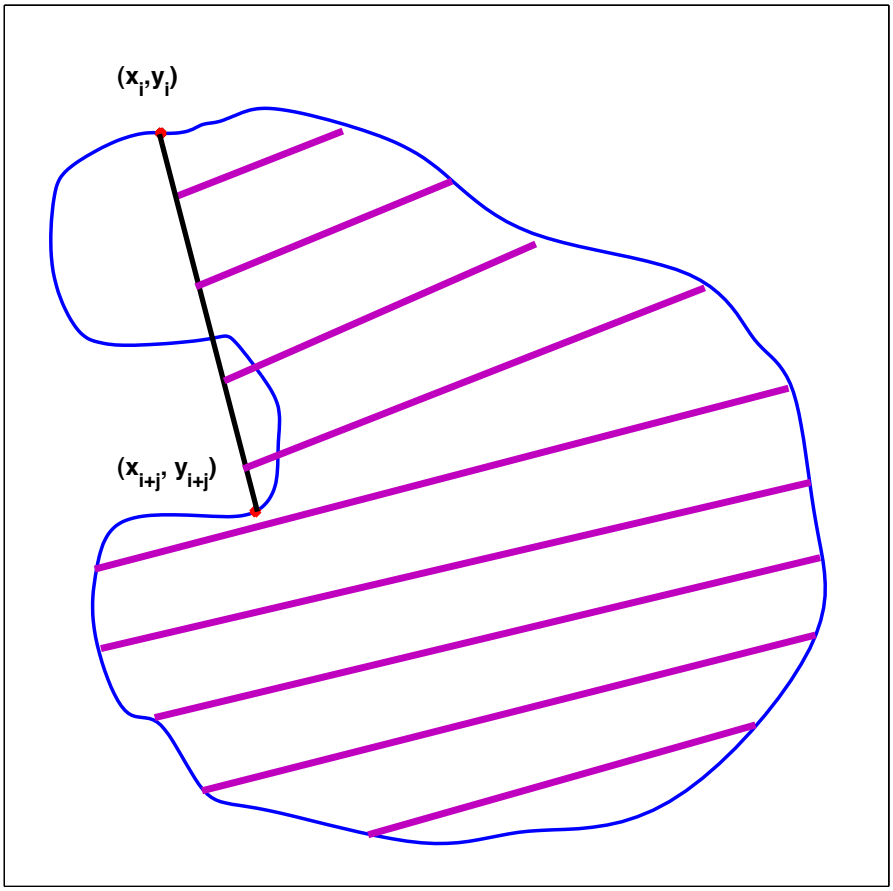


Figure 3.11: Visual illustration of first integral invariant for a closed curve.

Option Two: Determinant Shift

We can also generate the curve invariants associated with using each node as a starting point by using the fomula described below. We call the original starting point on the curve (x_1, y_1). Our new initial point will be (x_a, y_a) where $1 < a < N$ and N is the length of our matrix.

We let $\tilde{R}(a, a+i)$ denote the $\tilde{R}$ invariant with initial point (x_a, y_a) and end point (x_{a+i}, y_{a+i}).

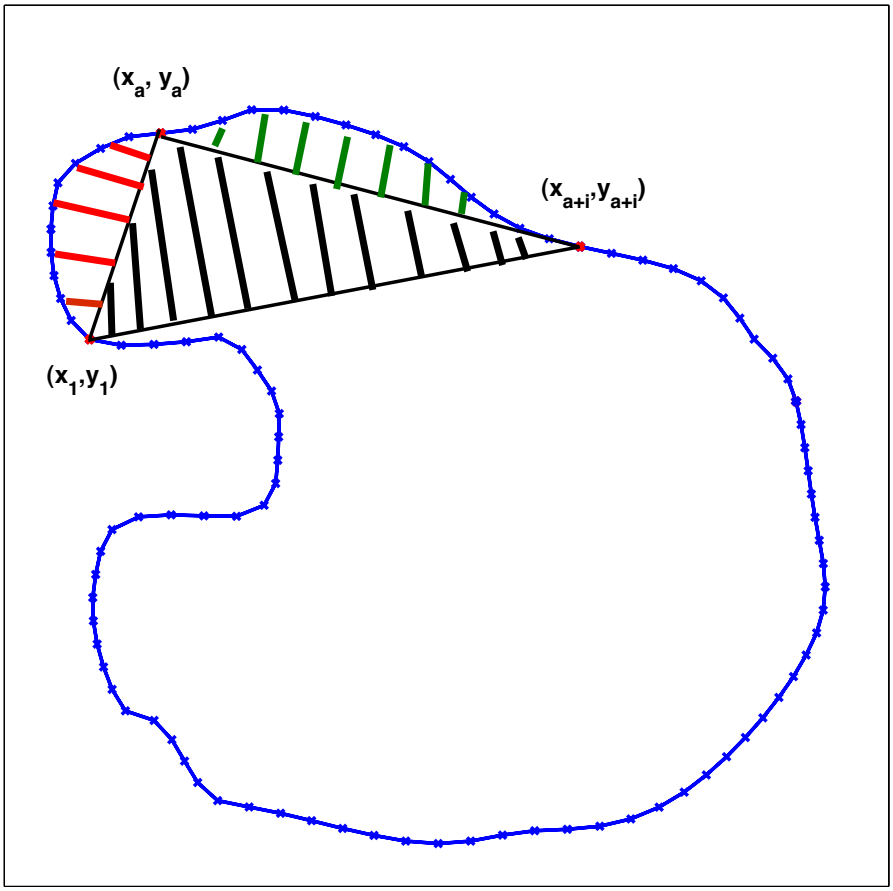


Figure 3.12: Visual depiction of determinant shift of signatures when $a + i < N$.

To find the $\tilde{R}$ invariant with our new initial point, we recall that this invariant is defined as the distance of the current point from the initial point. Thus we can compute the new invariant, $\tilde{R}(a, a+i)$ for any $1 < i < N$ using the formula

$$\tilde{R}(a, a+i) = \sqrt{(x_{a+i} - x_a)^2 + (y_{a+i} - y_a)^2}. \tag{3.9}$$

As a result of using closed curves, we should note that $(x_{a+i}, y_{a+i}) = (x_{a+i-N+1}, y_{a+i-N+1})$ when $a + i \geq N$.

The formula for computing $\tilde{I}_1(N)$ with changing initial points is easily derived visually by remembering the geometric definition of the first discrete invariant, $\tilde{I}_1(i)$, as the signed area between the curve and the secant line between the initial point and the point (x_i, y_i). There are two cases which we must consider as we derive this formula.

In the first case, we have traveled i nodes forward from our new initial point and $a+i < N$. Therefore, in this case our new end point to which we are computing is between our initial point and the original initial point in our ordering. This case is visually represented in Figure 3.12. We would like to find a formula using what we have previously found to compute $\tilde{I}_1(a, a+i)$. We have already computed the values for $\tilde{I}_1(1, a)$, the area between the curve and the secant line connecting (x_1, y_1) and (x_a, y_a), which is represented by the red shaded area in the figure, and $\tilde{I}_1(1, a+i)$, the area between the curve and the secant line connecting (x_1, y_1) and (x_{a+i}, y_{a+i}), which is represented by the entire shaded area in the figure. From Figure 3.12, we can see that $\tilde{I}_1(a, a+i)$ is equivalent to $\tilde{I}_1(1, a+i)$ less $\tilde{I}_1(1, a)$ and the area of the triangle formed by $(x_1, y_1), (x_a, y_a)$, and (x_{a+i}, y_{a+i}). Thus, we arrive at the following formula,

$$\tilde{I}_1(a, a+i) = \tilde{I}_1(1, a+i) - \tilde{I}_1(1, a) - \frac{1}{2}\begin{vmatrix} x_1 & x_a & x_{a+i} \\ y_1 & y_a & y_{a+i} \\ 1 & 1 & 1 \end{vmatrix}$$

for the case when $a + i < N$.

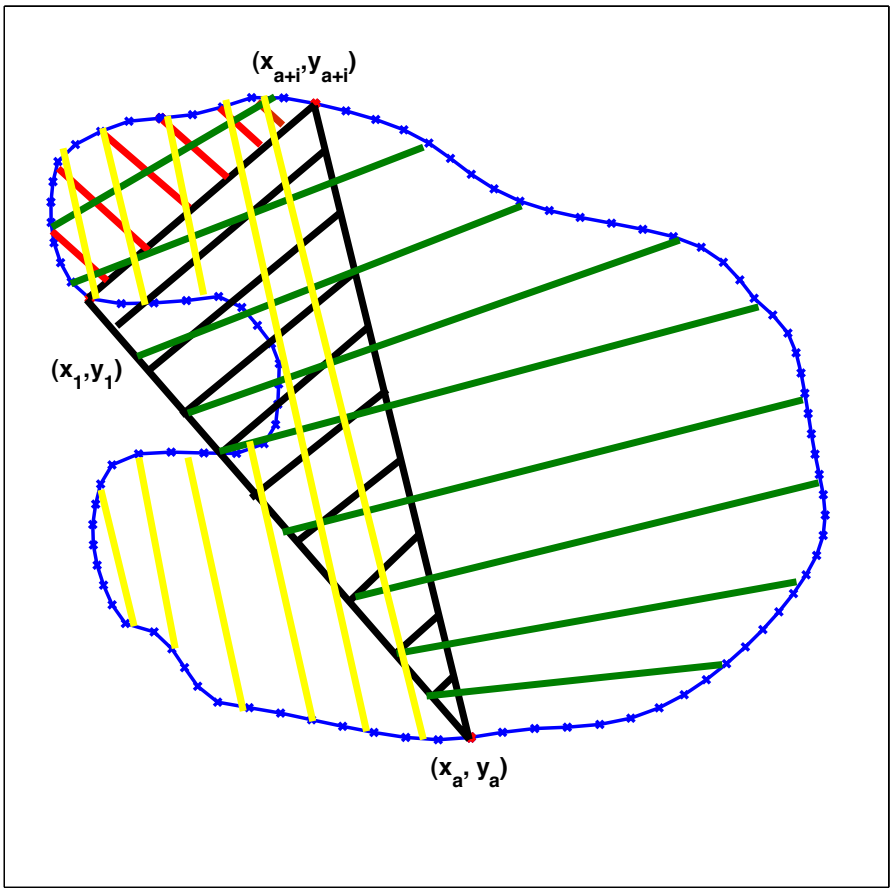


Figure 3.13: Visual depiction of determinant shift of signatures when $a + i \geq N$.

When $a+i \geq N$ this formula will no longer find the correct value for $\tilde{I}_1(a, a+i)$. Figure 3.13 provides a better geometric understanding of what occurs in this case. First, we note that for a sufficiently small sampling size, $\tilde{I}_1(1, N-1)$ will approximate the area trapped within the closed curve. We use this approximation in place of the area of the closed curve in our formula. From Figure 3.13 we see that the area between the curve and the secant line connecting (x_{a+i}, y_{a+i}) to (x_a, y_a), represented by the yellow shaded area, can be found by taking the area enclosed by the curve and subtracting the area between the curve and the secant line connecting (x_1, y_1) and (x_a, y_a), represented by the green shaded area, and then adding back in the area of the triangle formed by $(x_1, y_1), (x_{a+i}, y_{a+i})$, and (x_a, y_a), represented by the black shaded area, and the area between the curve and the secant line from (x_1, y_1) to (x_{a+i}, y_{a+i}), represented by the red shaded area. Recall that $(x_{a+i}, y_{a+i}) = (x_{a+i-N+1}, y_{a+i-N+1})$ when $a+i \geq N$. In this case, we have

$$\tilde{I}_1(a, a+i) = \tilde{I}_1(1, N-1) - \tilde{I}_1(1, a) + \tilde{I}_1(1, a+i-N+1) + \frac{1}{2}\begin{vmatrix} x_1 & x_{a+i} & x_a \\ y_1 & y_{a+i} & y_a \\ 1 & 1 & 1 \end{vmatrix}$$

when $a+i \geq N$.

Thus, if we choose to use this method to compute all of the invariants for every possible initial node on a closed curve, we use the formula,

$$\tilde{I}_1(a, a+i) = \begin{cases} \tilde{I}_1(1, a+i) - \tilde{I}_1(1, a) - \frac{1}{2}\begin{vmatrix} x_1 & x_a & x_{a+i} \\ y_1 & y_a & y_{a+i} \\ 1 & 1 & 1 \end{vmatrix}, & a+i < N \\ \\ \tilde{I}_1(1, N-1) - \tilde{I}_1(1, a) + \tilde{I}_1(1, a+i-N+1) + \frac{1}{2}\begin{vmatrix} x_1 & x_{a+i} & x_a \\ y_1 & y_{a+i} & y_a \\ 1 & 1 & 1 \end{vmatrix}, & a+i \geq N \end{cases} \tag{3.10}$$

Both of our options of iteratively computing the invariants associated with each initial point are of order N. Our choice of which method to employ depends upon any storage restrictions we may have. If storage is at a minimum, we prefer to use the determinant shift method of computing the invariants. In our algorithm used for puzzle assembly in Chapter 4, we are not concerned with the amount of storage we may use, so we will rotate the data.

We will compute the signatures of both curves using each node on the curve as the initial point. Suppose we have two closed curves of lengths N and M respectively. In this case our results would be three matrices, one for each discrete invariant, for each piece of sizes $N \times N$ and $M \times M$ respectively. For any of the six matrices, the i^{th} row will be the associated invariant of the piece with (x_i, y_i) as the starting point.

Choosing the Best Match for a Given Situation

For the comparison of two closed curves, we use the same metric as in the description of the matching algorithm for two open curves with one difference. The method that we use to determine the best match of two curves without fixed initial points is different from that used to match two curves with fixed initial points as a result of what results we desire. In the case

of comparison between two curves without fixed initial points, we are generally interested in finding the parts of the curves which match most closely.

For two closed curves for which we do not expect an exact match for the entire length of the curve, e.g. puzzle pieces, we would like to locate the parts of each curve where the best match can be made. In this situation we must balance our want for the closest possible match (distance wise) and our desire for the longest match (node wise). We want to find the parts of the original curves associated with the smallest distance between the signatures for the longest length of nodes. We choose our best match based on how many nodes we may progress on the signatures before the distance between the two exceeds some given maximum tolerance. In this case, we may determine an appropriate maximum tolerance in the context of a given application. We also declare a significant length of nodes and discard all match candidates that do not extend at least this many nodes.

As before, we will choose a curve that we would like to match. We will then find the segment of best match for each pairing of curves and then decide which of these segments of matches is best overall. For each pairing of closed curves, we must compare signatures associated with each possible initial node on each piece. We choose an initial node on the first curve and then compute distances from this signature to the signature of the second curve. We iterate through the signature associated with each initial point on the second curve. In order to save computing time, when the distance between the signatures exceeds the maximum tolerance, we stop computing and move to the next initial node. Once we have computed these distances, we begin the algorithm again with the next initial point on the first curve.

For each pair of signatures of the two curves, we record how many nodes we can advance forward before the distance exceeds the maximum tolerance. We then find the longest match, node wise, from among every initial point pairing and record this match and any other match within a certain amount (generally a percentage of the significant number of nodes) of matching node length. For each match, we record the starting point on curve one, the starting point on curve two, and the length of nodes for which it traveled before exceeding the maximum tolerance.

If the shortest length of match within these candidate matches is shorter than the significant number of nodes, we reset the new tolerance to be eight percent of the current tolerance and repeat our process.

If the shortest length of match within all possible best matches is longer than the significant number of nodes, we set the new tolerance to be eighty percent of the current tolerance and repeat the process of gathering a list of best possible matches until we meet the following requirements.

To declare an overall best match, we require first that the match be nested, meaning that not only must the initial nodes appear as a match for the longest nodal length, but the same part of the curves must also appear for a slightly shorter match length. Secondly, we require that the best match be a persistent match. It must have appeared in at least three consecutive candidate lists as we continue to decrease our distance tolerance. As with the open curves, after a best match is decided upon, the final transformation for the curve is found by minimizing a node-to-node distance function over the part of the curves associated with the match.

A pseudocode of this algorithm will be provided, along with complexity costs and an example application, in Chapter 4.

3.4 Sensitivity Analysis

In this section, we study the following question. Assume that the (R, I_1) signatures of curves γ and $\overline{\gamma}$ are close (as made precise by Eq. 3.11 below). How close are γ and $\overline{\gamma}$ modulo translations and rotations?

To simplify the calculations, we compare the curves as written in polar coordinates (r and θ). To do so, we consider an additional assumption: γ and $\bar{\gamma}$ are continuously differentiable curves such that for any $r > 0$, γ and $\bar{\gamma}$ intersect the circle of radius r centered at the origin at most once. In that context, it is known that if two curves have the *same* (R, I_1) signature, they are equivalent up to rotation and translation [59].

We describe γ by $\theta = \phi(r)$ and $\bar{\gamma}$ by $\theta = \bar{\phi}(r)$. Through translation, we take the initial point

of each curve to be the origin. In polar coordinates, the I_1 invariant takes the form

$$I_1^{\gamma}(R) = \frac{1}{2}\int_0^R s^2 \frac{d\phi}{ds} ds.$$

We assume that two signatures are close in the following sense

$$|I_1^{\gamma} - I_1^{\overline{\gamma}}| = \left| \frac{1}{2}\int_0^r s^2 \left(\frac{d\phi}{ds} - \frac{d\bar{\phi}}{ds} \right) ds \right| \leq \epsilon, \, \forall \, r, 0 \leq r \leq R, \tag{3.11}$$

for some $\epsilon > 0$. Using integration by parts we may write

$$\left| \frac{1}{2}\int_0^r s^2(\phi'(s) - \overline{\phi}'(s))\, ds \right| = \left| \frac{1}{2} r^2(\phi(r) - \overline{\phi}(r)) - \int_0^r s(\phi(s) - \overline{\phi}(s))\, ds \right| \leq \epsilon. \tag{3.12}$$

To better understand the geometric meaning of (3.12), we consider an example.

Example 3.4.1. Let α and β be positive numbers. We consider the two spirals

$$\phi(r) = \alpha r,$$
$$\bar{\phi}(r) = \beta r.$$

These spirals are plotted in Figure 3.14 for $\alpha = 2, \beta = 3, r \in [0, \frac{3\pi}{8}]$. We define four areas according to Figure 3.14. Elementary considerations lead to the following relations

$$I_1(r) = \text{ Area 1} + \text{ Area 2}, \qquad \overline{I}_1(r) = \text{ Area 2} + \text{ Area 3}$$
$$\frac{1}{2} r^2(\phi(r) - \bar{\phi}(r)) = \text{ Area 1} + \text{ Area 4}, \quad \int_0^r s(\phi(s) - \bar{\phi}(s))\, ds = \text{ Area 3} + \text{ Area 4}.$$

Thus, if we consider our definition of closeness of signatures, $|I_1(r) - \bar{I}_1(r)| \leq \epsilon$, then we have that $|\text{Area 1} + \text{ Area 2} - (\text{ Area 2} + \text{ Area 3})| = |\text{ Area 1} - \text{ Area 3}| \leq \epsilon$. Hence, our definition of closeness implies that Area 1 and Area 3 must be of similar size. The question at hand is then whether the green and red curves approximately differ by a rotation.

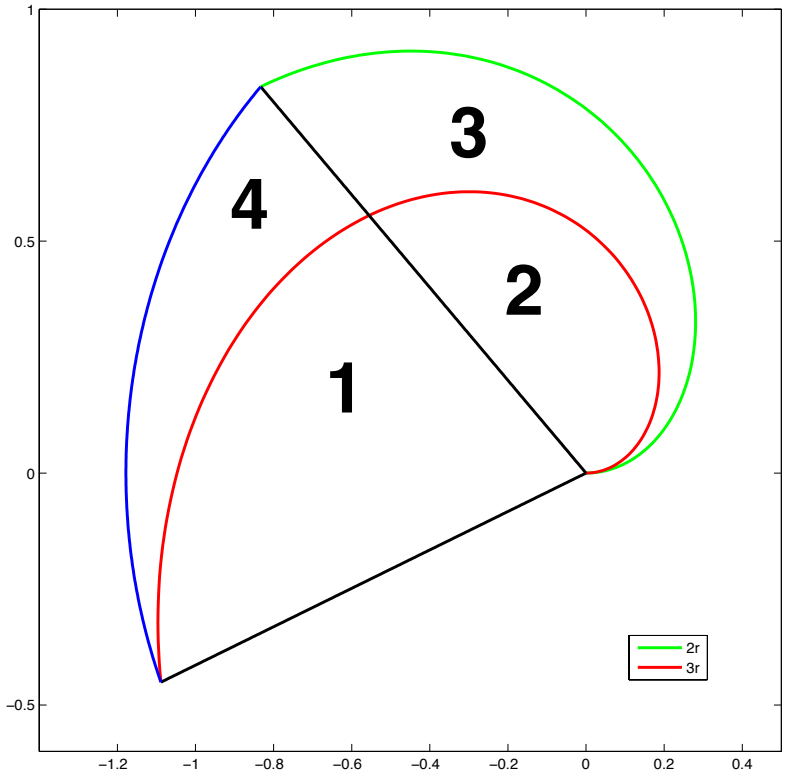


Figure 3.14: Plot of $\phi(r) = 2r$ (green), $\bar{\phi}(r) = 3r$ (red) and geometric interpretation of implications of Eq. 3.12. The blue curve is an arc of a circle centered at the origin.

We have, unfortunately, not been able to establish the desired local stability estimate of the mapping corresponding to the "inverse" of the signature. This type of results are not available even for simpler integral signatures such as the circular area signature (see however a recent partial result in that direction [20]).

Chapter 4

Applications

4.1 Character Recognition

There are two main types of character recognition - offline and online. Online recognition methods perform recognition as the character is being written. Offline recognition focuses on identifying characters after they have been completely drawn. We focus on offline recognition. A main difficulty with which we have to contend is small variations between digit classes while there are also large variations within a digit class. These variations within a class are referred to as allographs. Creating an automated method to determine allographs for a given character class is a challenging task.

We consider character recognition for single characters. This avoids the difficulty of segmentation, because we do not have to separate a word into characters. In the segmentation process with cursive words, it is difficult to identify characters and often new figures are created. In these cases, more classes of characters would be needed than those we define.

We test the recognition ability of our discrete invariant signature code by identifying the digits zero to nine as entered by various users. This problem involves matching both closed and open curves with fixed starting points, so it is a good examination of the ability of our proposed method. To accurately test the method, we have collected 240 samples of characters

from 13 different users on which to test our method for character recognition. Below we detail the algorithm used to identify each entry and also analyze its accuracy.

4.1.1 Algorithm

We face two main difficulties when identifying characters: multi-stroke characters and allographs. We first tackle the issue of multi-stroke characters. A multi-stroke character is defined as any character which cannot be drawn without lifting the pen from the paper. If we allow characters with a pen lift (or multiple lifts), the task of character recognition becomes much more complicated. Others have dealt with multi-stroke characters by simply connecting the strokes in the order in which they were drawn and considering this new character ([42]). This method, however, may cause confusion between characters that previously were not similar. For instance, a seven drawn with a crossbar may appear similar to a rotated nine once the strokes are connected. In an effort to simplify our process, we will only consider single stroke characters.

We accommodate for different handwriting and allographs by entering multiple styles of characters when there are several commonly used variants. For instance, rather than assume a user will draw a stick one, we have included both a stick one and a one with a flag in our database. To address the problem of allographs, we have prescribed a starting point and direction for each figure. Users were instructed in the standardized ways to enter each digit. We only consider figures drawn starting from the top of the figure, except in the cases detailed below. We use these standardized digits to create classes of digits.

To be able to identify characters, we divide the digits into classes. There are ten obvious classes of digits. We further subdivide two of these classes (4 and 9) for a total of *12 distinct digit classes: ones, twos, threes, open fours, closed fours, fives, sixes, sevens, eights, top to bottom nines, bottom to top nines*. For four, we have considered the drawings of the character with both an open and closed upper part. When a four is drawn with an open top, we consider only the case in which it is drawn from the top. Though this is generally drawn as a multi-stroke

digit, we have instructed users to draw this character with a single stroke. Users start at the upper left part of the four, trace down and across, then back across and create the up and down stick. This is all done in one stroke with no pen lift. When a four is drawn with a closed top, we consider it being drawn starting from the right side in one fluid stroke. The figure nine is considered as drawn from the top, loop first and then the stick, or from the bottom of the loop, up and around.

We have chosen twelve classes of digits with associated starting points and orientations that we consider to be the most common. To have a more comprehensive character recognition method, we would need to consider more ways of drawing each digit and, thus, have more classes. Also, we have declared that each figure may belong to only one class. Some identification procedures have allowed for multiple classes to apply to a single figure (e.g. [44], [42], [49]) and study the success rates of their procedure in finding the correct match within the top however many matches. For instance, a character may be classified as belonging to either the class of 4's or the class of y's. This more complicated approach is detailed in the discussion of our results.

For our procedure, characters were created using a MATLAB program that takes input points and equally distributes nodes along the splines connecting the input points. All characters were scaled to have a height of one with the original height to width ratio retained. Characters were defined with varying amounts of nodes.

When a user inputs a character, we equally distribute points along the connecting splines at a pre-defined distance. This data is used to compute the $\tilde{R}, \tilde{I}_1$, and $\tilde{I}_2$ discrete invariants for the character. We then connect the nodes along the chosen discrete signature, $(\tilde{R}, \tilde{I}_1), (\tilde{R}, \tilde{I}_2)$, or $(\tilde{I}_1, \tilde{I}_2)$, with linear interpolants and collect a declared number, say N, of equally distributed nodes along the signature. For each character, say the i-th one in our set, we create a vector $\mathcal{I}_i \in \mathbb{R}^{2N}$ which corresponds to either of the following discrete signatures

$$\mathcal{I}_i = [\tilde{R} \quad \tilde{I}_1], \quad [\tilde{R} \quad \tilde{I}_2], \text{ or } [\tilde{I}_1 \quad \tilde{I}_2].$$

For each character, we also have a response or class $\mathcal{C}_i$ that corresponds to one of the 12 classes described above.

Our classification approach uses the concept of *classification tree* from the fields of statistics and data mining, see for instance [62] and [95]. To build a tree, we first select a *training set* which is a subset of our set of characters. To do so, we choose at random an identical number of characters in each class (see below for examples).

Because our classes are categorical rather than quantitative, we introduce the concept of Gini impurity. Gini impurity measures how often a randomly chosen element from a given set would be mislabeled based on the distribution of labels in the set. For a set (leaf) containing m classes, $C_i, i = 1, \ldots, m$, the Gini impurity is

$$I_G(f) = \sum_{i=1}^{m} f_i \left(1 - f_i\right) = 1 - \sum_{i=1}^{m} f_i^2$$

where f_i is the fraction of items (digits) of class C_i in the given set. If the set contains only items belonging to one class, then $I_G(f) = 0$. If the set contains an equal amount of items belonging to each class, then $I_G(f) = \dfrac{m-1}{m}$. We will use the Gini impurity to evaluate the heterogeneity of each possible leaf in our tree.

Working with the training set exclusively, a tree $\mathcal{T}$ is then recursively constructed as follows.

1. Start with one node (ℓ) containing all characters

   - compute the Gini impurity for leaf ℓ

   $$I_G^{\ell}(f) = 1 - \sum_{i=1}^{m} f_i^2.$$

   - compute the sum of the Gini impurities for $\mathcal{T}$

   $$S = \sum_{\ell \in \text{leaves}(\mathcal{T})} I_G^{\ell}(f).$$

2. If all points in node ℓ are in the same class STOP; otherwise

    - search over all binary splits of all the components of the training set for one that minimizes S.
    - If the largest decrease in S is less than a threshold or if the resulting nodes contain less than a certain number of points STOP; otherwise
        - take the split and create two new nodes.

3. In each new node, go back to 1.

We have implemented the above algorithm using the MATLAB `classregtree` protocol from the Statistics Toolbox. A decision tree is created based on the training set. For the remaining characters, we use the tree to predict the response, i.e., the class to which each character belongs. Each step in a classification tree checks the value of one predictor. If the predictor value is above a given value, we proceed down one branch; if it is below the value, we proceed down the other branch. This process is repeated iteratively down each branch until the branch ends in only leaf nodes. These leaf nodes identify which digit class the input figure belongs to according to the values of its predictor values. There are several digits that are clearly differentiated from the others based on their signature generated vectors. The difference between other digits is more nuanced and may require several branch splits to reach a decision. Digits in the test set will be classified using the resulting tree. Decision trees for our digit classes are complicated and often there are several possible paths to being identified as the same digit.

For our experiment, we will create signatures containing 150 nodes. We combine the two chosen invariants to generate an initial vector of length 300. In an effort to make our algorithm faster and less computationally difficult, we will create a new signature generated vector that uses every tenth node from our original vector. The resulting vector of length 31 is the vector that we use to create our decision tree. The classification success rates with this shortened vector suggest that we have not lost significant accuracy with this choice.

An example of a decision tree is given in Figure 4.1. This tree was created using the $(\tilde{R}, \tilde{I}_2)$

signature with the training set comprised of 15 digits in each of the 12 digit classes. To identify a character using this tree, we begin at the top of the tree. This split specifies that if the ninth entry in our signature generated vector is less than 0.311941 then we proceed down the branch to the left. If the ninth entry is greater than or equal to 0.311941 then we use the branch to the right to continue our decision making process.

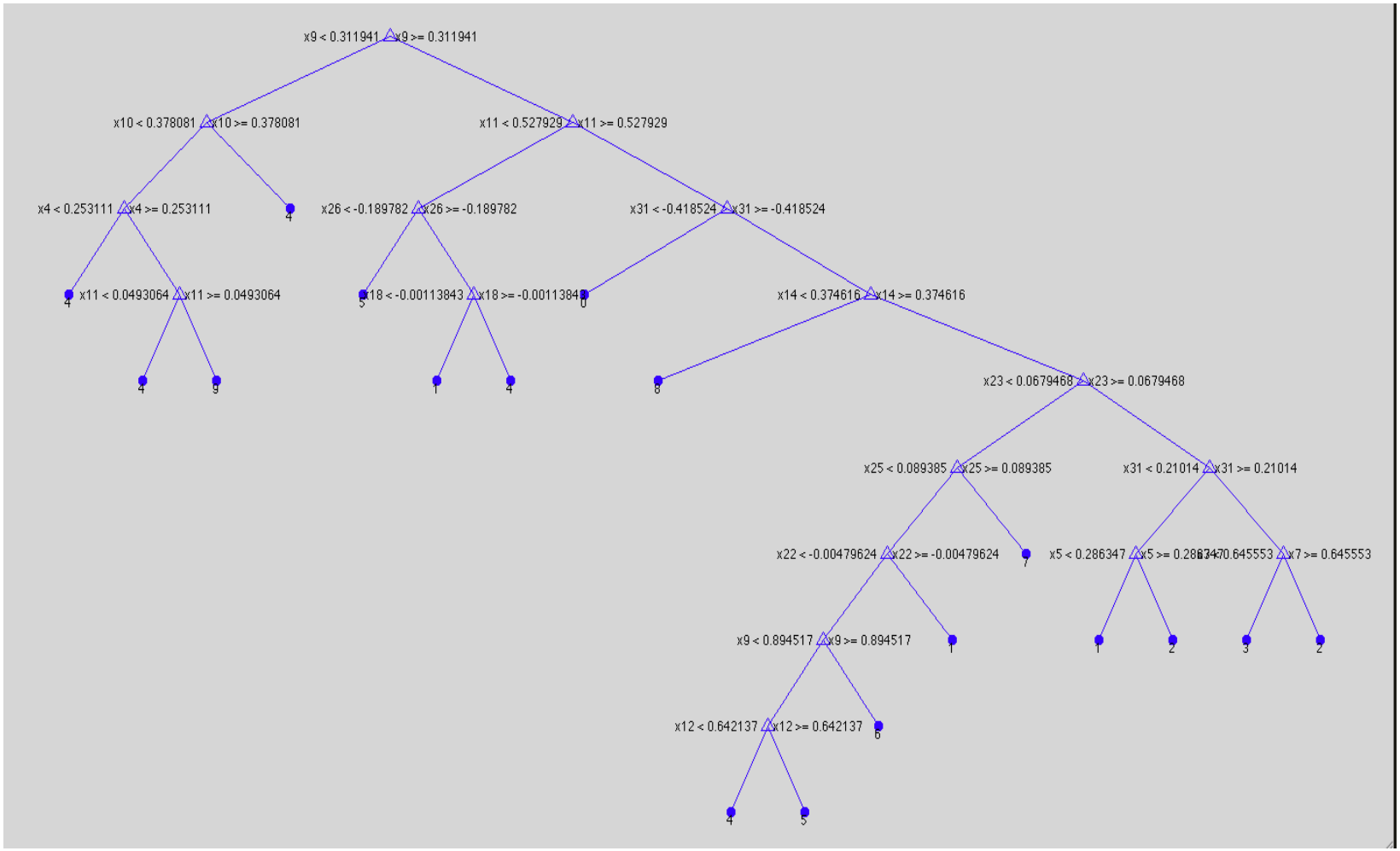


Figure 4.1: Example of a Binary Decision Tree

4.1.2 Pseudocode

This pseudocode is given for the $(\tilde{I}_1, \tilde{I}_2)$ signature, but the same process is followed for the $(\tilde{R}, \tilde{I}_1)$ and $(\tilde{R}, \tilde{I}_2)$ signatures.

Input:

- A two row matrix of ordered points representing a digit zero to nine
- A $M_1 \times M_2$ matrix of predictor values

Output:

- The top match for the input digit from 12 digit classes

Steps:

1. Using `classregtree`, create a classification tree, t, given a training set of M_1 digit signatures of length M_2 in the form $[\tilde{I}_1 \quad \tilde{I}_2]$.
2. For the user input order points, compute the associated $\tilde{I}_1$ and $\tilde{I}_2$ invariants using the formulas given in Eq. 3.3.
3. Linearly interpolate the $(\tilde{I}_1, \tilde{I}_2)$ signature and create a new $2 \times \frac{M_2}{2}$ matrix, A, of equally spaced ordered points along the signature.
4. Concatenate the rows of A to create a vector, a, of length M_2.
5. Use t to identify which of the 12 digit classes a best matches.

4.1.3 Results and Discussion

In order to evaluate the accuracy of this method, we partition our database of digits into training sets and test sets of varying sizes. We want to ensure that as the size of our training set increases our rate of correct matches increases. We have a database of 20 characters in each of the 12 digit classes. Five characters in each class are randomly chosen to create a 60 character

Table 4.1: Average Success Rates of Character Identification for Varying Sizes of Training Sets

(a) $(\tilde{R}, \tilde{I}_1)$

Size of Training Set	Success Rate
96	76.7
108	77.9
120	79.0
132	79.4
144	80.9
156	81.3
168	81.8
180	85.1

(b) $(\tilde{R}, \tilde{I}_2)$

Size of Training Set	Success Rate
96	76.7
108	77.9
120	79.0
132	79.4
144	80.9
156	81.3
168	81.8
180	85.1

(c) $(\tilde{I}_1, \tilde{I}_2)$

Size of Training Set	Success Rate
96	70.0
108	71.0
120	72.5
132	73.6
144	74.3
156	74.7
168	75.3
180	79.0

test set. Of the remaining 15 characters in each class, we first choose a given number to create a training set. This training set is used as predictor values to create a classification tree. Each of the 60 test characters is classified using this tree. For each iteration, we calculate the percentage of the test digits that were correctly identified. This entire process, including choosing the test set, is repeated 1,000 times with 96, 108, 120, 132, 144, 156, 168, and 180 characters in the training set. For each size of training set and each signature pairing, we find the average success rate over the 1,000 trials. These percentages are given in Table 4.1 and shown graphically in Figure 4.2.

As we increase the size of our training set, our rate of correctly identified digits increases. With a ratio of three training characters to each test character, we achieve a success rate of roughly 85% when using the $(\tilde{R}, \tilde{I}_1)$ or $(\tilde{R}, \tilde{I}_2)$ signatures. The higher success rate of these signatures is beneficial. We prefer to use the $(\tilde{R}, \tilde{I}_1)$ signature when possible due to its lesser

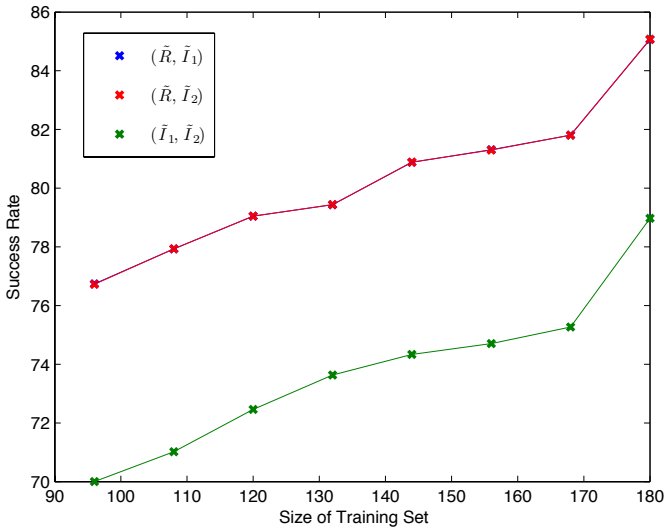


Figure 4.2: Average Success Rates of Character Identification

compuational cost. For our single step identification algorithm, this identification rate is in line with the rates we would expect from the literature.

Various methods for character recognition have been proposed: Hidden Markov Models ([13], [91], [82]), neural networks ([49], [52], [71]), genetic algorithms ([61],[40], [71]), and geometric moments ([37], [42]), among others. We focus on comparing the success rate of our method to that of other methods using similar approaches.

In [42], the $R(t)$ and $I_1(t)$ invariants are computed using Legendre-Sobolev polynomial approximations to the input curve, C. Classification is then performed by evaluating the distance from the sample to the convex hulls of the nearest neighbors. The class closest to the sample in the space of the coefficients of the truncated Legendre-Sobolev polynomial is selected as the best match. Because only one class is chosen, the success rate of this algorithm does not depend on the number of classes. It should be noted, however, that a sample may be labeled with more than one class. If the classification of the match overlaps the class of the sample in at least one symbol, the match is considered a success. For instance, if our sample is classified as a four and it is matched to a class of "4 or y" it would be considered a successful match.

Table 4.2: Success Rates of Various Character Recognition Methods

Method	**Success Rate**
Convex Hull of Nearest Neighbor [43]	78.25
Multilayer Cluster Neural Network (MCNN) with Backpropagation [71]	97.1
MCNN with Backpropagation and Genetic Algorithm [71]	97.8
Hidden Markov Models, Context-Independent and Character Tied Mixture [82]	59.9
Integral Invariants and Legendre-Sobolev Polynomials [42]	88
Coordinate Functions and Integral Invariants [42]	87.9
Coordinate Functions and Moment Invariants [42]	93.5

Golubitsky et al used 50,703 mathematical characters (not limited to digits) with 242 classes and considered multi-stroke characters. These characters were partitioned into 10 equally sized sets. Nine of these sets were used for training sets with the remaining set being the test set. Using this method, the success rate is found to be 88%. While this success rate is slightly higher than that of the method we propose, we believe the difference may be accounted for in the more sophisticated algorithm and the multiple label classes.

We evaluate how our method compares with other commonly used offline character recognition procedures. The success rates of these methods are shown in Table 4.2. When rates were given based upon the correct match being in the list of the top T classes, for a number T, we show the Top-1 rate. The success rates given are not limited to digit identification unless noted.

We see that our average success rate is comparable to the success rates of these more sophisticated classification methods.

Future experiments to improve upon the proposed method may allow for multi-stroke characters, multi label classes, and a two step classification process in an effort to improve our success rate.

4.2 Jigsaw Puzzle Assembly

The extensions of jigsaw puzzle assembly to real life problems are various. The same method used for puzzle assembly may be used for object assembly, object recognition, and scene recog-

nition, among others. As a result of its various applications and uses, the automatic solving of jigsaw puzzles has been widely studied.

We focus on apictorial jigsaw puzzle assembly with no restrictions on the shape of the puzzle or on the shape of the individual pieces. Due to the use of discrete invariants, we also need not worry about the orientation of the puzzle pieces as they are input. Many previous works (for example, [41], [110]) focused on automatic puzzle assembly have imposed certain restrictions upon the puzzle and its individual pieces. The most common restrictions are:

1. Each individual piece must have four sides and each side must contain either an "outdent" or an "indent."

2. Each piece has at most four direct neighbors, one on each side, with whom it connects via the "outdents" and "indents."

3. Once solved, the puzzle has pieces that are positioned on a grid.

4. The solved puzzle has a boundary that is a common, smooth shape, e.g. a circle or a rectangle.

We do not require any of these stipulations of our puzzle, however, we may exploit them if we have knowledge of the puzzle and its pieces before we solve it.

As noted in our descriptions of various curve matching techniques, methods are most efficient when several techniques are combined. We show that discrete invariants are a viable technique that may be effectively combined with existing techniques to accurately solve apictorial puzzles.

The problem of automatically assembling apictorial puzzles has been discussed since Freeman and Garder proposed the first computer solution of the problem in 1964, [38]. While many proposed methods exist, researchers still pursue faster, more efficient methods that are capable of solving more difficult (e.g. more pieces, irregular shaped pieces, irregular boundary) puzzles.

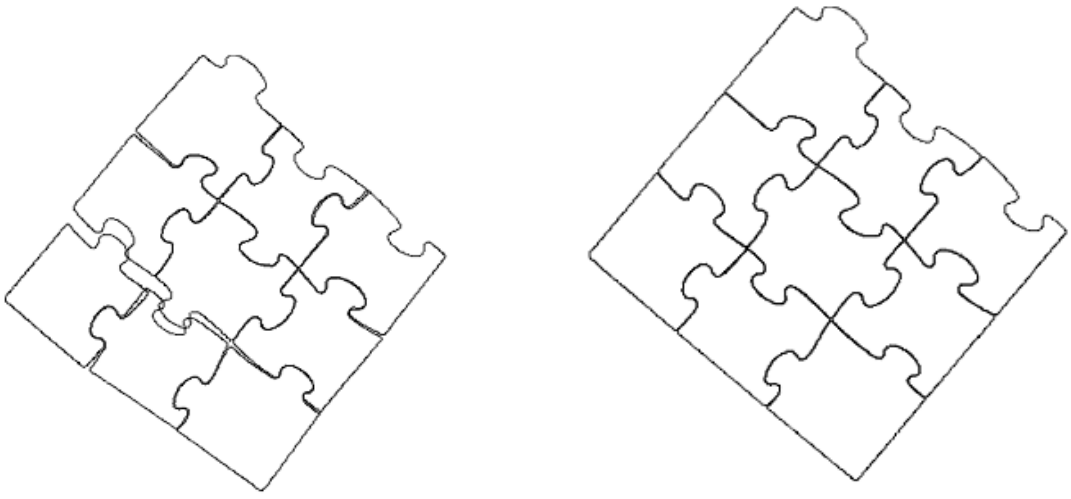
Some puzzle solvers begin by assembling the border ([41]) if it is a known and standard shape. Others begin with one piece and use a spiral type method to add pieces to this cluster ([38]).

While other methods begin with assembling clusters of pieces and then combining these clusters to solve the entire puzzle ([16]). Regardless of how the algorithm is begun, most algorithms eventually rank possible matches by some criteria to reduce the amount of work that must be done in finding the ideal mate. Once an ranked list of matches is found, some technique is used to find which of these matches provides the tightest and best fit. Often this technique is expensive and we want to avoid performing it more times than needed. Various methods to check the fit have been proposed.

In [54], Hoff and Olver proposed a method that uses a physics based approach. Each piece is considered to have an electrostatic/gravitational attractive force and the strength of attraction between two pieces is measured to quantify fit. Freeman and Garner, [38], proposed a "junction figure of merit" that measured the length of unmatched edge remaining at the junction of two pieces after a new piece had been placed. In [41], Goldberg, Malon, and Bern use a simple greedy algorithm to check and place the interior puzzle pieces. All of these techniques seek to create a digital analogue to the snap that humans feel when assembling a puzzle. This snap communicates to the assembler that a mate has been chosen successfully. It was even shown in [16], that robots are capable of feeling and registering this snap as they assemble pieces. Regardless of what it is called, this step is an integral one in any automated puzzle solver.

The difference in a puzzle assembled by using a fit finding technique only versus a fit finding technique paired with a locking tactic can quickly be seen. The resulting gaps in matches can pervade through the rest of the matches and cause problems throughout the assembly algorithm. Two solutions to a nine pieces of a puzzle are shown in Figure 4.3. Even in this puzzle with a small number of pieces the lack of a locking mechanism has a great effect.

We propose that discrete invariants can be used to rank possible matches as a first step in an automatic puzzle solver. The situation we study is that of finding the top matches for a given piece. We assume we have been given a specific piece to match. We examine the best match that would occur with each of the other pieces of the puzzle to organize the matches in order of best to worst. In our discussion, we more explicitly define the concept of best match



(a) Fit only solution

(b) Fit finding and locking technique solution

Figure 4.3: Solutions of a partial puzzle highlighting the impact of using a locking technique. Reproduced from [54].

and detail how this ordering is attained.

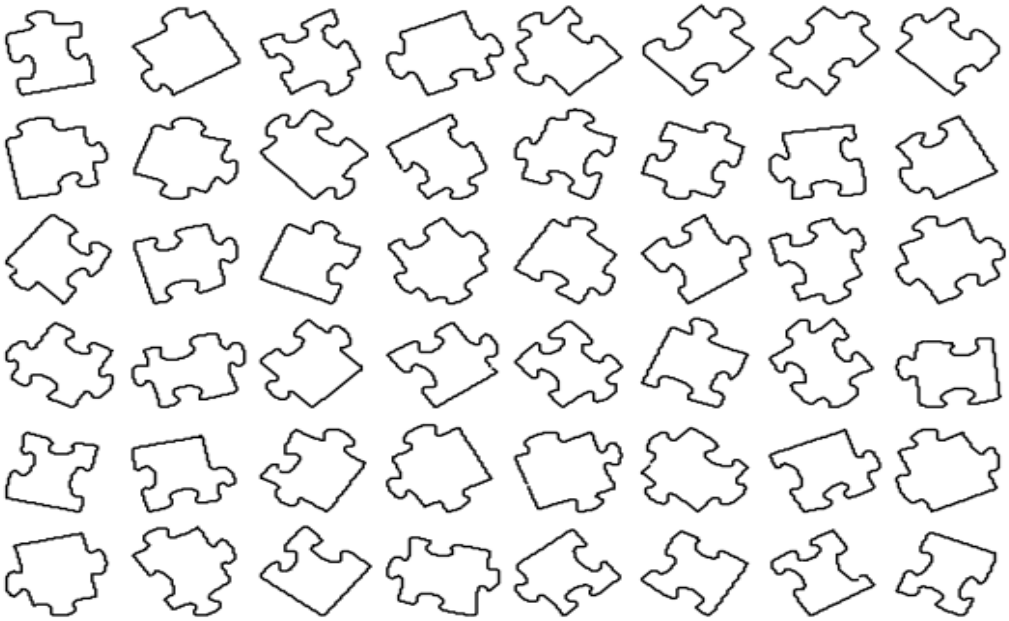
4.2.1 Preprocessing

Before we may begin our puzzle assembly, we must first obtain accurate computerized representations of each piece. Our pieces were found using a segmentation process based on the method of active contours. This process is also referred to as the method of snakes. The general idea of this method is to tighten a curve around an image until the curve meets a boundary. Some of the benefits of this method are that it allows for the segmentation of a part of the curve without regards for the content of other parts and it returns a connected boundary curve. Further details of this edge detection method are given in ([23], [24],[65]). Our digital representations of puzzle pieces were kindly shared with us by Dr. Peter Olver and Daniel Hoff. Their implementation of the active contours method was based upon code written by S. Lankton, [69].

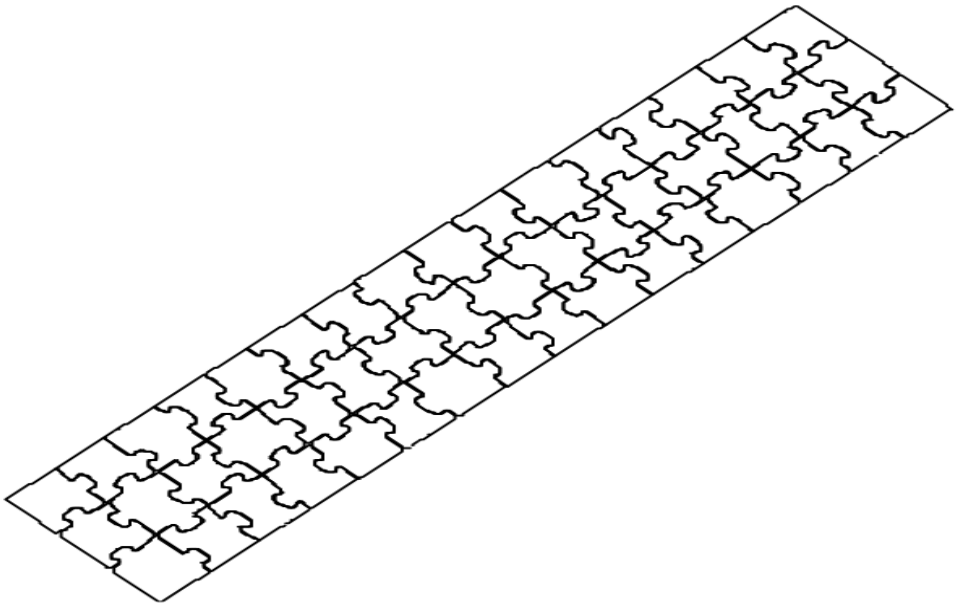
As a result of using high definition photographs of the puzzle pieces, the resolution of the

pictures must be reduced before the segmentation process may be performed. The resulting resolution meant that the segmented boundary curves found were not sufficiently smooth for the algorithms that Hoff and Olver use. Thus, they applied a preliminary smoothing operation to the piece boundaries. This process is detailed in [54].

The pieces we use are the smoothed boundary curves that resulted from these processes. We will study our algorithms as applied to the pieces of the 48 piece Rain Forest Giant Floor Puzzle, Figure 4.4. The solved puzzle is a 16 piece by 3 piece rectangle. Images of the pieces and the completed puzzle below are reproduced from [54].



(a) Pieces of the Rain Forest Giant Floor Puzzle



(b) The Solved Rain Forest Giant Floor Puzzle

Figure 4.4: The pieces of and completed Rain Forest Giant Floor Puzzle. The pieces in (a) are shown in pseudo-random order and in the orientations in which they are input to the algorithm. Reproduced from [54].

4.2.2 Algorithm

For the application in which we are interested, we are searching for the best match. Assume we have two puzzle pieces with N nodes on each. We define what exactly best match means for us. We prioritize four items:

tolerance The maximum distance allowed between the two curves. Initial tolerance is ten percent of the maximum $\tilde{I}_1$ value of both curves.

length The number of nodes from the initial node where the tolerance is exceeded. To be considered a match, the length must be longer than $.1N$.

persistence When the match remains a possible match as we decrease the tolerance. The tolerance will be decreased by twenty percent at each iteration. A match is persistent if it remains through 3 iterations.

nesting When there is another match of similar length in the same region of the piece edge. We require the second match to be within $.4N$ nodes of the original match.

We search for the best match by successively refining the tolerance. We evaluate each possible match for the four qualities listed above. The algorithm is involved and has many parameters which are defined below. The values used for this case are given.

significant number of nodes Only matches longer than this value are considered. Value is set to $.1N$.

maximum list length Lists of possible matches must be shorter than this value. Value is set to 30.

expected match length If we know the shape of the pieces, this value is the approximate expected length of a match. It is $.25N$ since we have roughly square pieces.

tolerance reduction factor The multiplier we use to reduce the tolerance. New tolerance is set to .8 of previous tolerance.

match list interval To be a match, its length must be within $.15N$ nodes of the maximum match length for the given tolerance.

persistence interval For persistence, two matches within this interval are considered equivalent. We require matches to be within two nodes.

nesting interval For nesting, two matches within this interval are considered equivalent. We require matches to be within $.4N$.

Given two puzzle pieces, the algorithm to find the best match between the two begins by computing distances. Recall that the signatures of closed curves are dependent upon the choice of initial point. Thus, we find the distance associated with each pairing of initial point on the two pieces using the method described in Chapter 3.

Next we find the length of each pairing using the initial tolerance. From the list of all lengths for this tolerance, we save the match with maximum length and all matches within its match list interval. If the list is longer than the maximum list length, we reduce the tolerance according to the reduction factor and repeat the process. If either the match list is empty or there do not exist any matches of length greater than the significant number of nodes, then we proceed to the last step of the algorithm. This match collection and tolerance reduction may be repeated at most ten times per iteration. If the match list avoids these pitfalls, we record this list, reduce the tolerance according to the reduction factor, and begin a new iteration.

Starting with the second iteration, we check for persistent matches by comparing two consecutive match lists. If a match in one list is within the match list interval of a match in the second list, it is persistent. We save all persistent matches along with how many iterations it has persisted.

Beginning with the fourth iteration, we check for nesting. We only check matches that have persisted through at least three iterations. We compare entries in the persistence list to those in the match list. If a match in the list is within the nesting interval of a persistent match, the match is nested.

These iterations end after either we have performed 30 iterations or we have found at least one match that is nested and persistent. In the case that we have performed 30 iterations, we use the last list of persistent matches (regardless of nesting or persistence) as possible matches. If there is more than one entry in our final list of possible matches, we choose the match with the smallest average node-to-node distance over the region of match.

To rank pieces as possible mates for a given piece, we perform the above procedure on the given piece paired with every other piece. After we have a best match for each piece pairing, we rank them based on average nodal distance. So that this distance is an accurate reflection of the match, we use the finest sampling of nodes available to find the distance. If a match is not of expected match length, we will add nodes symmetrically to both sides of the match region until it is the expected length. We order these distances from smallest to largest and this determines the ranking of the possible mates.

4.2.3 Pseudocode

We use the $(\tilde{R}, \tilde{I}_1)$ signature due to its lesser computation cost. We are able to use this signature because we are not concerned with possible reflection of puzzle pieces. To simplify notation, we will let $A = \tilde{I}_1$ and denote the first invariant associated with puzzle piece i as Ai.

The code below provides the best match region, nodal distances, transform used, and the new coordinates of the transformed piece for a pair of two puzzle pieces.

Input:

- $P1O$, a $2 \times m1$ matrix representing an ordered walk around one puzzle piece
- $P2O$, a $2 \times m2$ matrix representing an ordered walk around second puzzle piece
- $SigPercent$, scalar defining the percentage of total nodes needed to be a significant match
- $TolPercent$, scalar defining the percentage of A for the initial tolerance
- $n1, n2$, number of nodes to be distributed along the edge of $P1O$ and $P2O$

Output:

- $P2_T$, a $2 \times n2$ matrix representing an ordered walk around the transformed second puzzle piece
- $Trans$, a vector of length 3 containing the values of ϕ, a, and b used in the final transformation of $P2$
- $NodalDist$, a vector of length k containing the node-to-node distances over the region of match
- $BestMatchRegion$, a vector of form $[i, j, k]$ where the match region starts at $P1(:, 1)$ and $\overline{P2}(:, j)$ and continues for k nodes

Steps:

1. Create $P1$ and $P2$ by distributing $n1$ and $n2$ equidistant points respectively on the linear interpolants of $P1O$ and $P2O$.
2. $N := \min\{n1, n2\}$.
3. Define the significant number of nodes to be $S := SigPercent \times N$.
4. Reverse the order of the walk around $P2$:
   $\overline{P2}(i, j) = P2(i, n2 - j + 1)$, where $i = 1, 2$ and $j = 1, \ldots, n2$.
5. Create a $n1 \times 2 \times n1$ matrix $S1$,
   $S1 = [R1, A1] :=$ *Subalgorithm for Finding Discrete Signatures Using Determinant Shift*$(P1)$,
   where $S1(i, 1, k)$ is the $\tilde{R}$ curve invariant for $P1$ starting with $P1(:, i)$ at the k^{th} node and $S1(i, 2, k)$ is the similar value of $A1$ for $i = 1, \ldots, n1$.
6. Create a $n2 \times 2 \times n2$ matrix $S2$,
   $S2^j = [R2^j, A2^j] :=$ *Subalgorithm for Finding Discrete Signatures Using Determinant Shift*$(\overline{P2})$,
   where $S2(j, 1, k)$ is the $\tilde{R}$ curve invariant for $P2$ starting with $P2(:, j)$ at the k^{th} node and $S2(j, 2, k)$ is the similar value of $A2$ for $j = 1, \ldots, n2$.

7. $T := TolPercent \times \max\{A1, A2\}$.

8. Create a $n1 \times n2 \times N$ matrix, $Dist$,

   $Dist$= *Subalgorithm for Finding Distances between Two Curves*$(S1, S2, T, N)$

   such that $D(i, j, k)$ is the distance between $S1(i, 1:2, k)$ and $S2(j, 1:2, k)$.

9. Generate a list of possible matches, $PossibleMatchList$,

   $PossibleMatchList =$

   Subalgorithm for Choosing Possible Matches for Two Puzzle Pieces$(Dist, T, S)$.

   in the format of $[i, j, k]$ where the match region starts at $P1(i, :)$ and $\overline{P2}(j, :)$ and continues for k nodes.

10. Find the best transformation of $\overline{P2}$, the region of best match, and the nodal distances over the matching region.

    $[P2_T, Trans, BestMatchRegion, NodalDist] =$

    Subalgorithm for Transforming the Original Curves$(PossibleMatchList, P1, \overline{P2}, P1O, P2O)$

Subalgorithms for Puzzle Piece Algorithm

Subalgorithm for Finding Discrete Signatures Using Determinant Shift

Input:

- A $2 \times m$ matrix, P, of m points representing an ordered walk around a puzzle piece

Output:

- A $m \times 2 \times m$ matrix S, such that $S(i, 1, k)$ is the R curve invariant for P starting with $P(:, i)$ at the k^{th} node and $S(i, 2, k)$ is the similar value of A for $i = 1, \ldots, m$.

Steps:

1. $InitPt := P(:, 1)$

2. for i=1:m

   Use formulas in 3.1.1 to compute discrete (R^{temp}, A^{temp}) signature using this initial point.

   $S(1, 1, i) = R^{temp}$ and $S(1, 2, i) = A^{temp}$.

3. for i=2:m

   $InitPt := P(:, i)$

   for k=1:m-1

   Use the formulas in Eq. 3.9 and Eq. 3.10 to compute R^{temp} and A^{temp}.

   $S(i, 1, k) = R^{temp}$ and $S(i, 2, k) = A^{temp}$

Subalgorithm for Finding Distances between Two Curves

Input:

- $X1$, the first curve of size $2 \times N1$
- $X2$, the second curve of size $2 \times N2$
- T, a tolerance
- $N := min\{N1, N2\}$

Output:

- A vector of length N, $Dist$, where $Dist(k)$ is the distance between $X1$ and $X2$ at the k^{th} node

Steps:

Note: Let $\overline{X(s), X(s+1)}$ denote the linear interpolant between the nodes $X(s)$ and $X(s+1)$.

1. Compute the distances from the nodes of $X1$ to the linear interpolants of $X2$.

s=1

$k = 1 : N$

i. until $dist1(k) > T$:

$d_1 = dist(X1(:,k), \overline{X2(:,s), X2(:,s+1)})$

$d_2 = dist(X1(:,k), \overline{X2(:,s+1), X2(:,s+2)})$

$dist1(k) = \min\{d_1, d_2\}$

if $dist1(k) = d1,\ s = s+1$

if $dist1(k) = d2,\ s = s+2$

ii. once $dist1(k) > T$:

$dist1(k) = \infty$

2. Compute the distances from the nodes of $X2$ to the linear interpolants of $X1$.

s=1

$k = 1 : N$

i. until $dist2(k) > T$:

$d_1 = dist(X2(:,k), \overline{X1(:,s), X1(:,s+1)})$

$d_2 = dist(X2(:,k), \overline{X1(:,s+1), X1(:,s+2)})$

$dist2(k) = \min\{d_1, d_2\}$

if $dist2(k) = d1,\ s = s+1$

if $dist2(k) = d2,\ s = s+2$

ii. once $dist2(k) > T$:

$dist2(k) = \infty$

3. Combine the distances to find the final distance between the two curves.

$k = 1 : N$

$Dist(k) = \min\{dist1(k), dist2(k)\}$

Note: If $dist1(k) = dist2(k) = \infty$, then $Dist(k) = \infty$.

Subalgorithm for Finding Discrete Signatures Using Rotation of Data

Input:

- A $2 \times m$ matrix of m points representing an ordered walk around a puzzle piece

Output:

- Three $m \times m$ matrices, one representing each of $R(n)$ and $I_1(n)$ signatures with different starting points. Each row corresponds to the node in the original ordering that was used for the initial point.

Steps:

1. Call the input matrix, $P_{current}$.
2. Let the first node be the initial point.
3. Using the formulas given in 3.1.1, find the signatures for this curve with this starting node.
4. Rotate the curve $P_{current}$ to get P_{new} so that $P_{new}(:, i) = P_{current}(:, i+1)$ for $i = 1, \ldots, m-1$ and $P_{new}(:, m) = P_{current}(:, 1)$.
5. Repeat steps b-d with each P_{new} until signatures have been computed for all possible starting points, i.e. m times.

Subalgorithm for Choosing Possible Matches for Two Puzzle Pieces

Input:

- A $n1 \times n2 \times N$ matrix $Dist$
- Initial tolerance, T

- Number of nodes considered to be significant, S

Output:

- A list of possible matches, $PossibleMatchList$ in the format of $[i, j, k]$ where the match region starts at $P1(:, i)$ and $\overline{P2}(:, j)$ and continues for k nodes

Steps:

1. $T_{current} = T, T_{old} = 0$
2. While $persistence = 0$ and $nested = 0$
   (a) $i = 1 : n1$

   $D = Dist(i, :, :)$

   $j = 1 : n2$

   $d = D(j, :)$

   $k = 1 : N$

   while $d(k) \leq T$, counter $= k$

   $List(i, j) =$ counter

   Reset $k = 1$, counter $= 1$

   (b) Find $K = \max\{List\}$

   (c) For any $List(i, j) \in [K - .01S, K]$, record the associated $[i, j, List(i, j)]$ in $MatchList$.

   (d) If $K - .01S \leq S, T_{new} = 1.1\,(T_{current})$.
   If $K - .01S > S, T_{new} = .8\,(T_{current})$.

   (e) For each match in $MatchList$ check if the match has appeared in the previous two match lists. Each match that persists, store in $MatchList2$.

   (f) If $MatchList2$ is empty, $persistence = 0$ and return to step a. Otherwise, $persistence = 1$ and proceed to the next step.

(g) For each match in $MatchList2$ check for matches in $MatchList$ where i, j and k are within $.04 * N$ of the match in $MatchList2$. Store matches that meet this criteria in $PossibleMatchList$.

(h) If $PossibleMatchList$ is empty, return to step a with T_{new}. Otherwise, end code and return $PossibleMatchList$.

Subalgorithm for Transforming the Original Curves

Input:

- A list of possible matches, $PossibleMatchList$ in the format of $[i, j, k]$ where the match region starts at $P1(i, :)$ and $\overline{P2}(j, :)$ and continues for k nodes
- $P1$, a $2 \times n1$ matrix representing the ordered walk around the first puzzle piece
- $\overline{P2}$, a $2 \times n2$ matrix representing the reversed ordered walk around the second puzzle piece
- $P1O$, a $2 \times m1$ matrix representing the original walk around the first puzzle piece
- $P2O$, a $2 \times m2$ matrix representing the original walk around the second puzzle piece

Output:

- $P2_T$, a $2 \times n2$ matrix representing the transformed second puzzle piece
- $Trans$, the values of ϕ, a, and b used in the final transformation of $P2$
- $NodalDist$, a length k vector containing the node-to-node distances between the nodes of $P1$ and $P2_T$ over the matched region
- $BestMatchRegion$, a vector of form $[i, j, k]$ where the match region starts at $P1(i, :)$ and $\overline{P2}(j, :)$ and continues for k nodes

Steps:

1. $i = 1 : \text{length}(PossibleMatchList)$

   (a) $[s1, s2, k] = PossibleMatchList(i, :)$

   (b) Find the nodes on $P1O$ and $P2O$ that correspond to $[s1, s2, k]$ on $P1$ and $P2$. Denote these as $[s11, s22, k1]$

   (c) Minimize

   $$val(i) = \frac{1}{k1}\sum_{j=1}^{k1}\left\|\begin{pmatrix} P1O(1, s11+j-1) \\ P1O(2, s11+j-1)\end{pmatrix} - \begin{pmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi)\end{pmatrix}\begin{pmatrix} P2O(1, s22+j-1) \\ P2O(2, s22+j-1)\end{pmatrix} - \begin{pmatrix} a \\ b\end{pmatrix}\right\|^2$$

2. The match with the smallest minimal average nodal distance, val, is the best match, $BestMatchRegion$.

3. Minimize

   $$winner = \frac{1}{k1}\sum_{j=1}^{k1}\left\|\begin{pmatrix} P1O(1, s11+j-1) \\ P1O(2, s11+j-1)\end{pmatrix} - \begin{pmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi)\end{pmatrix}\begin{pmatrix} P2O(1, s22+j-1) \\ P2O(2, s22+j-1)\end{pmatrix} - \begin{pmatrix} a \\ b\end{pmatrix}\right\|^2$$

   to find the best values of ϕ, a, and b.

4. $P2_T = \begin{pmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi)\end{pmatrix}\begin{pmatrix} \overline{P2} \\ \overline{P2}\end{pmatrix} + \begin{pmatrix} a \\ b\end{pmatrix}$.

5. $i = 1 : k$

   $$NodalDist(i) = \|P1(s1+i-1, :) - P2_T(s2+i-1, :)\|$$

To determine which piece is the best mate for a given puzzle piece, we collect $BestMatchRegion$ and $NodalDist$ for each piece pairing and store them in $Matches$ and $NodalDisances$. Assum-

ing we have M puzzle pieces total, we use these arrays and the following pseudo-code to find the ranking of mates.

Input:

- $Matches$, a $3 \times M-1$ matrix containing the best matches and length for each pairing
- $NodalDistances$, a cell with $M-1$ entries containing the node-to-node distances of each pairing

Output:

- $RankedMatches$, a $M-1$ vector listing the ranking of pieces for mates for the given piece

Steps:

1. Compute the average nodal distance associated with each piece and store in $AvgNodal$.
2. Sort the vector $AvgNodal$ distance in ascending order and store the order of the indices of entries.
3. Using the indices of the reordered $AvgNodal$ vector, determine the ranking of pieces and store in $RankedMatches$.

The computational cost of ranking the possible matches for a puzzle piece is $\mathcal{O}(N^3)$. We may reduce the cost by altering our code to only search for the top-T matches and the cost will naturally reduce as the number of possible mates lessens in the process of assembling the puzzle.

4.2.4 Results and Discussion

To analyze the ability of the discrete invariants to rank all pieces in order of best match for a chosen puzzle piece, we will use the pieces from the Rain Forest Giant Floor Puzzle. For any given piece, there are 47 other pieces with which we examine its match. Since the assembled puzzle is three pieces high and 16 pieces across, any piece excepting the corner pieces will have

three or four possible pieces that are correct matches for it. We will not discriminate between these pieces as to which is the best match, but rather will consider the listing of any of them to be a correct match listing.

We will exploit the knowledge that the pieces are approximately square in our ranking algorithm. As a result of this knowledge, we expect the best match to span approximately a quarter of the total number of nodes on the piece. Whenever a region of best match is identified along the edge of a piece, we will standardize the length of the match so that the region of best match is centered in a span of nodes that is a quarter of the length of the piece. If we had no knowledge beforehand of the shape of the pieces or if they were not a uniform shape, the only change it would make to the algorithm is that we would standardize the lengths to be the same length as the longest match rather than to a quarter of the size of the piece. This step helps us ensure that an exact match over very few nodes will not be rated higher than a very close match over a longer number of nodes.

Table 4.3: Success Rates for a Correct Match Appearing in the Top-T Matches

	Top-1	Top-2	Top-3	Top-5	Top-10	Top-15
Success Rates	50%	58.33%	77.08%	83.33%	93.75 %	95.83 %

The success rates for a correct match being listed in the top-T matches with our method are displayed in Table 4.3. These results were found using 150 equally distributed nodes on each puzzle piece. In preliminary experiments, 150 nodes was the size for which we had the best pairwise matching success rate. We used the $(\tilde{R}, \tilde{I}_1)$ signature for all comparisons and match ranking. We see from the results that if we use the Top-10 matches out of a possible 47 matches for each piece we have a 93.75% chance of a correct match being in the list. This corresponds with only 45 pieces out of the 48 pieces having a correct match listed. This reduction is worth the risk that we may have a piece where a correct match does not appear in the list. By only checking ten possible matches with a locking algorithm rather than 48, we have greatly reduced

our computational load.

Chapter 5

Final Remarks

We have found a method to generate a new family of integral invariants. That these invariants allow for the input to be discrete is their main benefit. It is no longer necessary to either fit a continuous curve to a discrete set of points or to use a (possibly) non-invariant numerical approximation to continuous invariants. As we have shown, our family of invariants are independent of sampling and provide an efficient method of calculating invariants of curves that are given discretely.

These invariants are a viable curve matching technique. Our results in using discrete invariants for character recognition and classification and puzzle assembly shows that this method is useful in solving curve matching problems. These invariants would be especially useful when used in conjunction with one or more of the other curve matching techniques we have mentioned.

If these discrete invariants are to be seriously used in curve matching algorithms, however, there are further improvements that can be made. We have mentioned these possible improvements throughout, but will revisit the most important of them. While we were developing and coding this method, our end goal was not to write the most efficient code, but to develop a working method. The code could be optimized to shorten the run time needed for curve matching.

In the application of character recognition and classification, several improvements would

need to be made before this method could be competitive with other currently used procedures. We focused solely on the classification of digits and, even in this case, used a rather simple class breakdown of the digits. To be more efficient and accurate, new character classes need to be introduced. We would also need to consider the option of multiple classifications being assigned to characters as discussed. This system allows for a multiple layered code to narrow down the possible matches in several steps. In addition to expanding our number of classes, we would need to consider the possibility of multi-stroke characters. For simplicity, we have ruled them out in the algorithm as it stands now, but it is an unavoidable fact that many people draw digits using more than one stroke. Also, the technique would need to be expanded to consider characters other than digits, e.g. mathematical symbols, Roman letters, Greek letters.

While our invariants are useful to identify a list of top possible matches for puzzle pieces, they are not an efficient puzzle assembly tool alone. As has been shown in other puzzle assembly algorithms, [54] and [38] for instance, truly effective methods require some sort of a locking mechanism. Piece locking serves to ensure the tightest match possible between two pieces. It is the code equivalent of the click we feel when we snap two pieces together correctly. If the pieces are not tightly transformed together then we will incur increasingly large gaps between pieces as we continue assembling the puzzle. This will result in the correct location and orientation of each piece, but the puzzle will not be assembled as if we had assembled it by hand. In the case of object assembly, it is necessary to have a tight fit so that objects may be reliably assembled by machines.

Discrete integral invariants we introduce have been shown to be a useful and effective curve matching tool, especially if the improvements described above are made.

REFERENCES

[1] The darpa shredder challenge, 2011.

[2] A. Abatzoglou, A. Smith, J. WebsterLove, K. Iwancio, and I. Kogan. Invariants in computer vision. Technical report, North Carolina State University, 2007.

[3] W.S.I. Ali and F.S. Cohen. Registering coronal histological 2-d sections of a rat brain with coronal sections of a 3-d brain atlas using geometric curve invariants and b-spline representation. *IEEE Trans. Med. Imag.*, 17:957–966, 1998.

[4] H. Alt and L. J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation: A survey. Technical report, Handbook of Computational Geometry, 1996.

[5] K. Arbter, W.E. Snyder, H. Burkhardt, and G. Hirzinger. Application of affine-invariant fourier descriptors to recognition of 3-d objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:640–647, 1990.

[6] A. Ardovini, L. Cinque, and E. Sangineto. Identifying elephant photos by multi-curve matching. *Pattern Recognition*, 41:1867–1877, 2008.

[7] E. Attalla and P. Siy. Robust shape similarity retrieval based on contour segmentation, polygonal multiresolution, and elastic matching. *Pattern Recognition*, 38:2229–2241, 2005.

[8] M. Bakircioglu, U. Grenander, N. Khaneja, and M.I. Miller. Curve matching on brain surfaces using frenet distances. *Human Brain Mapping*, 6:329–333, 1998.

[9] B.A. Barsky, A.D. DeRose, and M.D. Dippe. An adaptive subdivision method with crack prevention for rendering beta-spline objects. Technical report, EECS Department, University of California, Berkeley, 1987.

[10] C. M. Bastuscheck, E. Schonberg, J. Schwartz, and M. Sharir. Object recognition by three-dimensional curve matching. *International Journal of Intelligent Systems*, 1:105–132, 1986.

[11] H. Blum. *Models for the Perception of Speech and Visual Forms*, chapter A transformation for extracting new descriptors of shape, pages 362–380. MIT Press, 1967.

[12] M. Boutin. Numerically invariant signature curves. *Int. J. Computer Vision*, 40:235–248, 2000.

[13] A. Brakensiek, J. Rottland, A. Kosmala, and G. Rigoll. Off-line handwriting recognition using various hybrid modeling techniques and character n-grams. In *IN 7TH INTERNATIONAL WORKSHOP ON FRONTIERS IN HANDWRITTEN RECOGNITION*, pages 343–352, 2000.

[14] D. Brinkman and P.J. Olver. Invariant histograms. *Amer. Math. Monthly*, 119:4–24, 2012.

[15] K. Buchin, M. Buchin, and Y. Wang. Exact algorithm for partial curve matching via the frchet distance. In *Proc. 20th ACM-SIAM Symposium on Discrete Algorithms*, 2009.

[16] B.G. Burdea and H.J. Wolfson. Solving jigsaw puzzles by a robot. *IEEE Transactions on Robotics and Automation*, 5:752–764, 1989.

[17] D.J. Burr. Elastic matching of line drawings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:708–713, 1981.

[18] E. Calabi, P.J. Olver, C. Shakiban, A. Tannenbaum, and S. Haker. Differential and numerically invariant signatures curves applied to object recognition. *Int. J. Computer Vision*, 26:107–135, 1998.

[19] E. Calabi, P.J. Olver, and A. Tannenbaum. Affine geometry, curve flows, and invariant numerical approximations. *Advances in Mathematics*, 124:154–196, 1996.

[20] J. Calder and S. Esedoglu. On the circular area signature for graphs. *SIAM Journal on Imaging Sciences*, 5:1355–1379, 2012.

[21] E. Cartan. La methode du repere mobile, la theorie des groupes continus, et les espaces generalises. *Exposes de Geometrie*, 5, 1935.

[22] E. Cartan. *Groupes finis et continus et la geometrie differentielle traitees par la methode du repere mobile*. 1937.

[23] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours, 1997.

[24] T. Chan and L. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10:266–277, 2001.

[25] Z. Chen and S.K. Sun. A zernike moment phase-based descriptor for local image representation and matching. *IEEE Transactions on Image Processing*, 19:205–219, 2010.

[26] F.S. Cohen and J. Wang. Part i: Modeling image curves using invariant 3-d object curve models - a path to 3-d recognition and shape estimation from image contours. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16:1–12, 1994.

[27] F.S. Cohen, Z. Yang, Z. Huang, and J. Nissanov. Automatic matching of homologous histological sections. *IEEE Trans. Biomedical Engineering*, 45:642–649, 1998.

[28] M. Cui, J. Femiani, J. Hu, P. Wonka, and A. Razdan. Curve matching for open 2d curves. *Pattern Recognition Letters*, 30:1–10, 2009.

[29] C. de Boor. On calculation with b-splines. *J. Approx. Theory*, 6:50–62, 1972.

[30] C. de Boor. *A Practical Guide to Splines*. Springer-Verlag, 1978.

[31] D.S. Doermann, E. Rivlin, and I. Weiss. Logo recognition using geometric invariants. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, 1993.

[32] T. Eiter and H. Mannila. Computing discrete frechet distance. Technical report, Christian Doppler Laboratory for Expert Systems, TU Vienna, 1994.

[33] O. Faugeras. *Application of Invariance in Computer Vision*, chapter Cartan's moving frame method and its application to the geometry and evolution of curves in the Euclidean, affine, and projective planes, pages 11–46. Springer-Verlag Lecture Notes in Computer Science, 1994.

[34] M. Fels and P.J. Olver. Moving coframes. ii. regularization and theoretical foundations. *Acta Appl. Math*, 55:127–208 127–208, 1999.

[35] S. Feng, I. Kogan, and H. Krim. Classification of curves in 2d and 3d via affine integral signatures. *Acta Appl. Math.*, 109:903–937, 2010.

[36] T. Fidler, M. Grasmair, and O. Scherzer. Identifiability and reconstruction of shapes from integral invariants. *Inverse Problem Imaging*, 2:341–354, 2008.

[37] J. Flusser and T. Suk. Character recognition by affine moment invariants. In *Proc. 5th International Conference on Computer Analysis of Images and Patterns*, 2007.

[38] H. Freeman and L. Garder. Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition. *IEEE Transactions on Electronic Computers*, 13:118–127, 1964.

[39] Y. Gdalyahu and D. Weinshall. Flexible syntactic matching of curves and its application to automatic hierarchical classification of silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:1312–1328, 1999.

[40] R. Ghosh and M. Ghosh. An intelligent offline handwriting recognition system using evolutionary neural learning algorithm and rule based over segmented data points, 2005.

[41] D. Goldberg, C. Malon, and M. Bern. A global approach to automatic solution of jigsaw puzzles. In *Proc. Conf. Computational Geometry*, 2002.

[42] O. Golubitsky, V. Mazalov, and S.M. Watt. Orientation-independent recognition of handwritten characters with integral invariants. In *Proc. Joint Conference of ASCM 2009 and MACIS 2009*, 2009.

[43] O. Golubitsky and S. M. Watt. Distance-based classification of handwritten symbols. Technical report.

[44] O. Golubitsky and S. M. Watt. Tie-breaking for curve multiclassifiers. Technical report.

[45] L. Van Gool, T. Moons, E. Pauwels, and A. Oosterlinck. *Geometric Invariance in Computer Vision*, chapter Semi-differential invariants for non-planar curves, pages 157–192. MIT Press, 1992.

[46] C. Gope. *View-Invariant Curve and Point-Pattern Matching with Application to Photo-Identification of Marine Mammals*. PhD thesis, University of Texas at Dallas, 2006.

[47] C. Gope and N. Kehtarnavaz. Affine invariant comparison of point-sets using convex hulls and hausdorff distances. *Pattern Recognition*, 40:309–320, 2007.

[48] C. Gope, N. Kehtarnavaz, G. Hillman, and B. Wursig. An affine invariant curve matching method for photo-identification of marine mammals. *Pattern Recognition*, 38:125–132, 2005.

[49] A. Graves and J. Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks.

[50] M. L. Green. The moving frame, differential invariants, and rigidity theorems for curves in homogeneous spaces. *Duke Math. J.*, 45(4):735–779, 1978.

[51] P. Griffiths. On cartan's method of lie groups and moving frames as applied to uniqueness and existence questions in differential geometry. *Duke Math. J.*, 41:775–814, 1974.

[52] I. Guyon, P. Albrecht, Y. Le Cun, J.S. Denker, and W. Hubbard. Design of a neural network character recognizer for a touch terminal. *Pattern Recognition*, 24:105–119, 1991.

[53] C. E. Hann and M. S. Hickman. Projective curvature and integral invariants. *Acta Applicandae Mathematicae*, 74:177–193, 2002.

[54] D. Hoff and P.J. Olver. Automatic solution of jigsaw puzzles. Preprint, 2011.

[55] D. Hoff and P.J. Olver. Extensions of invariant signatures for object recognition. Preprint, 2011.

[56] H. Hse and A.R. Newton. Sketched symbol recognition using zernike moments. In *Proceedings of the 17th International Conference on Pattern Recognition*, 2004.

[57] Z. Huang and F.S. Cohen. Affine-invariant b-spline moments for curve matching. *IEEE Transactions on Image Processing*, 5:1473–1480, 1996.

[58] K.C. Hung. The generalized uniqueness wavelet descriptor for planar closed curves. *IEEE Transactions on Image Processing*, 9:834–845, 200.

[59] K. Iwancio. *Use of Integral Signatures and Hausdorff Distance in Planar Curve Matching.* PhD thesis, North Carolina State University, 2009.

[60] C.V. Jawahar, A. Balasubramanian, M. Meshesha, and A. Namboodiri. Retrieval of online handwriting by synthesis and matchign. *Pattern Recognition*, 42:1445–1457, 2009.

[61] R. Kala, H. Vazirani, A. Shukla, and R. Tiwari. Offline handwriting recognition using genetic algorithm. *International Journal of Computer Science Issues*, 7:16–25, 2010.

[62] C. Kamath. *Scientific data mining: a practical perspective.* SIAM, 2009.

[63] H. Kauppinen, T. Seppanen, and M. Pietikainen. An experimental comparison of autoregressive and fourier-based descriptors in 2d shape classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:201–207, 1995.

[64] A. Khotanzad and Y.H. Hong. Invariant image recognition by zernike moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12:489–497, 1990.

[65] S. Kichenassamy, A. Kumar, and P.J. Olver. Conformal curvature flows: From phase transitions to active vision, 1995.

[66] Y.S. Kim and W.Y. Kim. Content-based trademark retrieval system using a visually salient feature. *Image and Vision Computing*, 16:931–939, 1998.

[67] W. Kong and B. Kimia. On solving 2d and 3d puzzles under curve matching. pages 583–590, 2001.

[68] Y. Lamdan, J.T. Schwartz, and H.J. Wolfson. Object recognition by affine invariant matching. In *Computer Society Conference on Computer Vision and Pattern Recognition*, 1988.

[69] S. Lankton. Active contours, 2007.

[70] D.J. Lee, S. Antani, X. Xu, and L. R. Long. Desing and evaluation of a curve matching-based spine x-ray image retrieval system. In *Proceedings of SPIE*, 2005.

[71] S.W. Lee and Y. J. Kim. Off-line recognition of totally unconstrained handwritten numerals using multilayer cluster neural network. *IEEE Transactions on Patter*, 18:648–652, 1996.

[72] H. Li, B.S. Manjunath, and S.K. Mitra. Registration of 3-d multimodality brain images by curve matching. In *Nuclear Science Symposium and Medical Imaging Conference*, 1993.

[73] W.Y. Lin, N. Boston, and Y.H. Hu. Summation invariant and its application to shape recognition. In *Proc. of ICASSP*, 2005.

[74] C. Liu, W. Pei, S. Niyokindi, J. Song, and L. Wang. Micro stereo matching based on wavelet transform and projective invariance. *Measurement Science and Technology*, 3:565–571, 2006.

[75] L. Lucchese, S. Leorin, and G. Cortelazzo. Estimation of two-dimensional affine transformations through polar curve matching and its application to image mosaicking and remote-sensing data registration. *IEEE Transactions on Image Processing*, 15:3008–3019, 2006.

[76] S. Manay, D. Cremers, B. Hong, A. Yezzi, and S. Soatto. Shape matching via integral invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28, 2006.

[77] S. Manay, A. Yezzi, B. Hong, and S. Soatto. Integral invariant signatures. In *Proc. of the ECCV*, 2004.

[78] J. McBride and B. Kimia. Archaeological fragment reconstruction using curve-matching. In *Conference on Computer Vision and Pattern Recognition Workshop*, 2003.

[79] G. Medioni and Y. Yasumoto. Corner detection and curve representation using cubic b-splines. *Computer Vision, Graphics, and Image Processing*, 39:267–278, 1987.

[80] B. Mehtre, M. Kankanhalli, and W. Lee. Shape measures for content based image retrieval: a comparison. *Inf. Process. Manage.*, 33:319–337, 1997.

[81] B.S. Morse. *Computation of object cores from grey-level images*. PhD thesis, University of North Carolina, Chapel Hill, 1994.

[82] P. Natarajan, S. Saleem, R. Prasad, E. MacRostie, and K. Subramanian. *Arabic and Chinese Handwriting Recognition*, chapter Multi-lingual Offline Handwriting Recognition Using Hidden Markov Models: A Script-Independent Approach, pages 231–250. Springer Berlin Heidelberg, 2008.

[83] P.J. Olver. Joint invariant signatures. *Found. Comp. Math*, 1:3–67, 2001.

[84] P.J. Olver and Mark Fels. Moving coframes i. a practical algorithm. *Acta Appl. Math*, 51:161–213, 1998.

[85] Y.H. Pang, A. Jin, and D. Ling. Palmprint authentication system using wavelet based pseudo-zernike moments features. *International Journal of the Computer, the Internet, and Management*, 13:13–26, 2005.

[86] G.A. Papakostas, Y.S. Boutalis, D.A. Karras, and B.G. Mertzios. Pattern classification by using improved wavelet compressed zernike moments. *Applied Mathematics and Computation*, 212:162–176, 2009.

[87] T. Pavlidis. Algorithms for shape analysis of contours and waveforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:301–312, 1980.

[88] T. Pavlidis. *Algorithms for Graphics and Image Processing*. Computer Science Press, 1982.

[89] E. Persoon and K.S. Fu. Shape discrimination using fourier descriptors. *IEEE Transactions on Systems, Man and Cybernetics*, 7:170–179, 1977.

[90] G. Radack and N. Badler. Jigsaw puzzle matching using a boundary-centered polar encoding. *Computer Graphics and Image Processing*, 19:1–17, 1981.

[91] G. Rigoll, A. Kosmala, and D. Willett. A new hybrid approach to large vocabulary cursive handwriting recognition. In *Proceedings of 14th International Conference on Patter Recognition*, 1998.

[92] C. Samir, A. Srivastava, and M. Daoudi. Three-dimensional face recognition using shapes of facial curves. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2006.

[93] J. Sato and R. Cipolla. Affine integral invariants for extracting symmetry axes. *Image and Vision Computing*, 15:627–635, 1997.

[94] T. Sebastian and B. Kimia. Curves vs. skeletons in object recognition. *Signal Processing*, 85:247–263, 2005.

[95] C. Shalizi. Classification and regression trees, 2009.

[96] E. Smirnova and S. M. Watt. Communicating mathematics via pen-based interfaces. In *Tenth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, 2008.

[97] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*, pages 193–242. Chapman and Hall, 1993.

[98] J. Sternby, J. Morwing, J. Andersson, and C. Friberg. On-line arabic handwriting recognition with templates. *Pattern Recognition*, 42:3278–3286, 2009.

[99] M.R. Teague. Image analysis via the general theory of moments. *J. Opt. Soc. Am.*, 70:920–930, 1980.

[100] C. Teh and R. Chin. On image analysis by the methods of moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10:496–513, 1988.

[101] F.B. ter Haar and R.C. Veltkamp. A 3d face matching framework for facial curves. In *IEEE International Conference on Shape Modeling and Applications*, 2008.

[102] Q.M. Tieng and W.W. Boles. Wavelet-based affine invariant representation: a tool for recognizing planar objects in 3d space. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:846–857, 1997.

[103] S. Uchida and H. Sakoe. A survey of elastic matching techniques for handwritten character recognition. *IEICE - Transactions on Information and Systems*, E88-D:1781–1790, 2005.

[104] G. Ucoluk and H. Toroslu. Automatic reconstruction of broken 3-d surface objects. *Computers and Graphics*, 23:573–582, 1999.

[105] M. Unel, O. Soldea, E. Ozgur, and A. Bassa. 3d object recognition using invariants of 2d projection curves. *Pattern Analysis and Applications*, 13:451–468, 2010.

[106] A. Y. Wang, A. D. Leow, H. D. Protas, A. W. Toga, and P. M. Thompson. Brain warping via landmark points and curves with a level set representation.

[107] J. Wang and F.S. Cohen. Part ii: 3-d object recognition and shape estimation from image contours using b-splines, shape invariant matching, and neural network. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 1:13–23, 1994.

[108] X.Y. Wang, Y.J. Yu, and H.Y. Yang. An effective image retrieval scheme using color, texture, and shape features. *Computer Standards and Interfaces*, 33:59–68, 2011.

[109] I. Weiss. Geometric invariants and object recognition. *International Journal of Computer Vision*, 10:207–231, 1993.

[110] H. Wolfson, E. Schonberg, A. Kalvin, and Y. Lamdan. Solving jigsaw puzzles by computer. *Annals of Operations Research*, 12:51–64, 1988.

[111] M. Xia and L. Bede. Image registration by "super-curves". *IEEE Transactions on Image Processing*, 13:720–732, 2004.

[112] D. Xu and H. Li. Geometric moment invariants. *Pattern Recognition*, 41:240–249, 2008.

[113] L. Younes. Optimal matching between shapes via elastic deformations. *Image and Vision Computing*, 17:381–389, 1999.

[114] D. Zhang and G. Lu. Review of shape representation and description techniques. *Pattern Recognition*, 37:1–19, 2004.