2D CARTOON RIGS FROM UNCORRESPONDED VECTOR GRAPHICS

by

Nusha Mehmanesh

A Thesis
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Master of Science
Computer Science

Committee:

_____	Dr. Yotam Gingold, Thesis Director
_____	Dr. Craig Yu, Committee Member
_____	Dr. Jyh-Ming Lien, Committee Member
_____	Dr. Sanjeev Setia, Chairman, Department of Computer Science
_____	Dr. Kenneth S. Ball, Associate Dean for Research and Graduate Studies

Date: _____ Spring Semester 2019
George Mason University
Fairfax, VA

2D Cartoon Rigs From Uncorresponded Vector Graphics

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at George Mason University

By

Nusha Mehmanesh
Bachelor of Science
George Mason University, 2017

Director: Dr. Yotam Gingold, Professor
Department of Computer Science

Spring Semester 2019
George Mason University
Fairfax, VA

Acknowledgments

I would like to express my sincere gratitude to my advisor Dr. Yotam Gingold for his continuous guidance and patience. I could not have imagined having a better advisor and mentor for my Masters study. I would also like to thank Dr. Adrien Bousseau for imparting his immense knowledge and expertise on this research.

Table of Contents

List of Tables

List of Figures

Abstract

2D CARTOON RIGS FROM UNCORRESPONDED VECTOR GRAPHICS

Nusha Mehmanesh

George Mason University, 2019

Thesis Director: Dr. Yotam Gingold

This thesis introduces a novel algorithm that finds the correspondence between different poses of character(s) in two consecutive vectorized 2D key frames.This algorithm also derives a set of handles that allow the illustrator to animate the character to any desired pose. This is done in the hopes of reducing the time consuming manual work done by the animators in the process of in-betweening between key frames and to allow the animator to produce other frames by using the handles. The automated iterative approach described in this thesis finds the set of best possible transformations between two subsequent key frames by minimizing an energy function and performing a multi-label optimization. The resulting transformations are then used to calculate the blending weights that minimizes the displacement error. The final transformations can then either be interpolated to produce intermediate frames at a user specified rate or modified to animate the character.

Chapter 1: Introduction

The goal of 2D cartoon animation is to illustrate the movement of a character or object by utilizing drawings known as frames. The drawings are then displayed consecutively at a fixed rate of usually 24 frames per second to show the character's motion. While the goal of every 2D cartoon production is to produce the mentioned drawings, the process in which these drawings are obtained can vary. In a Traditional 2D animation pipeline every frame is hand-drawn which requires more manual label but gives the illustrator more artistic freedom. On the other hand in a non-traditional process, in addition of having the choice of hand-drawing each frame, the illustrator is able to produce skeletons for the characters of the film through a process called rigging. This skeleton can then be used to animate the character to any pose by moving individual parts of the structure.

In a traditional pipeline, the animator first draws the subject in consecutive different poses known as key frames. The key frames show the subject in poses that are a specific time period apart and therefore by themselves are not sufficient to produce smooth motion when displayed. Because of this the animator(s) draw a sequence of images in which the character transitions between two consecutive key frames (see figure 1.1). This process is known as in-betweening and can be very time consuming and labor intensive.

The first goal of this thesis is to provide an automated method of producing in-between frames in hopes of aiding the illustrator in this time consuming process. In this thesis, it is assumed that the hand-drawn images are pre-processed to produce the clean and vectorized key frames used as input. Since in vector graphics the drawings of the characters are made of well-defined parametric curves, assuming no body part obscures another, a point on the curve in one pose of the character will have a corresponding point on other poses of the same character. Therefore, this problem can be thought of as a point-to-point correspondence problem and once solved the key frames can be interpolated to produce in-between images.

Figure 1.1: Examples of hard-drawn key and in-between frames

Source: http://www.austincc.edu

Another goal of this thesis is to not only allow the animator to interpolate between the two inputted frames but to also aid the illustrator through a non-traditional pipeline by automating the rigging process. In this thesis we find a low-dimensional set of handles to allow interpolation withing the space of all given poses of the character(s). To this aim, the curves are modeled as undergoing a weighted blend of rigid transformations which allows an illustrator to modify the transformations to animate the character.

To achieve the previously stated goals, the method described in this thesis first finds the point-to-point correspondences between the given key frames then uses the computed transformations to perform an inverse skinning decomposition. These two steps together result in a set of transformations and a set of handles which can be used to both interpolate between the inputted frames and to animate the character in general.

In the following chapters we first discuss some background information and related work in this field to identify how our approach varies from other methods. Then we describe our method and detail its steps for further understanding. The results of our Algorithms are then illustrated in chapter 5 and topics for future research are discussed in chapter 6.

Chapter 2: Background

In traditional 2D animation production, the hand-drawn key frames represent the key actions taken by the characters in the story arc. Each key frame, like any hand-drawn image, is the result of many rough sketching attempts. Once the sketching is finished, the drawing is then used as a reference by an illustrator to reproduce a cleaner image by aggregating and/or replacing strokes. These clean images are then considered key frames. In our method we not only limit the input images to clean drawings but we also require the input frames to be vectorized i.e. each line or stroke in the image will be represented by one or more well-defined parametric curve. More accurately, we only consider input images that consist of a set of cubic bezier curves (see figure 2.1).

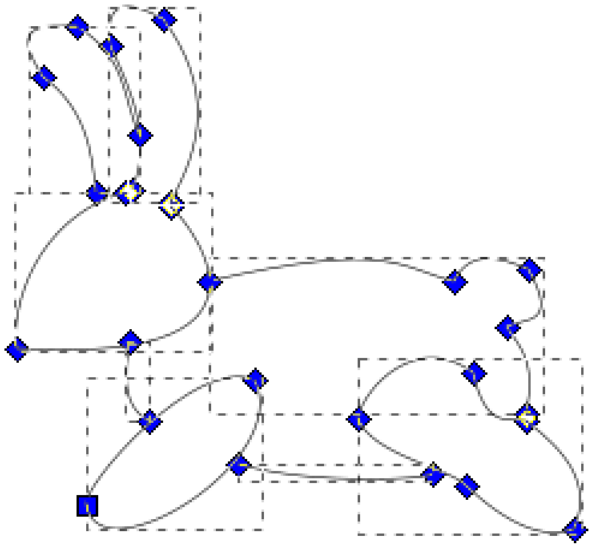
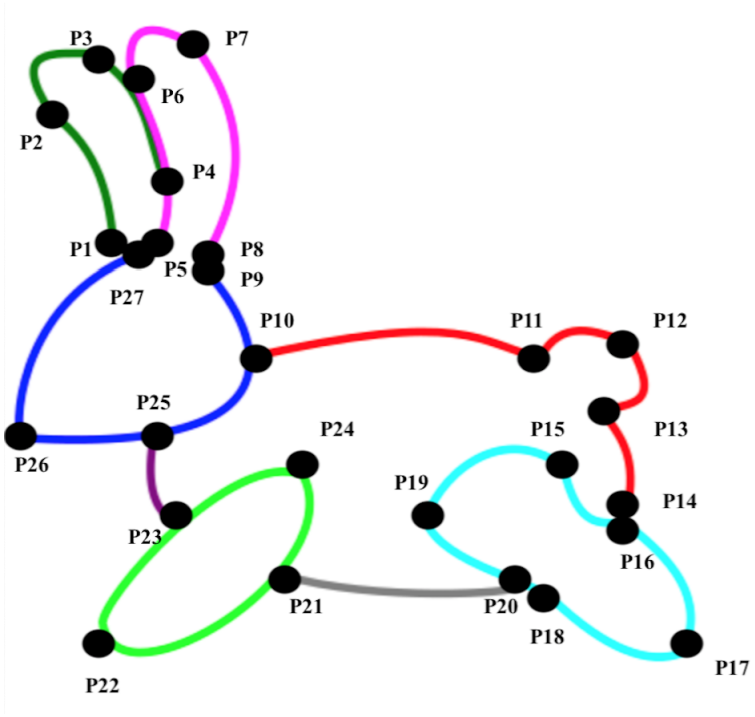


Figure 2.1: Example of a vectorized image consisting of only cubic bezier splines. In this image blue points represent the endpoints of cubic bezier curves and each box contains a chain of bezier curves/a single spline.

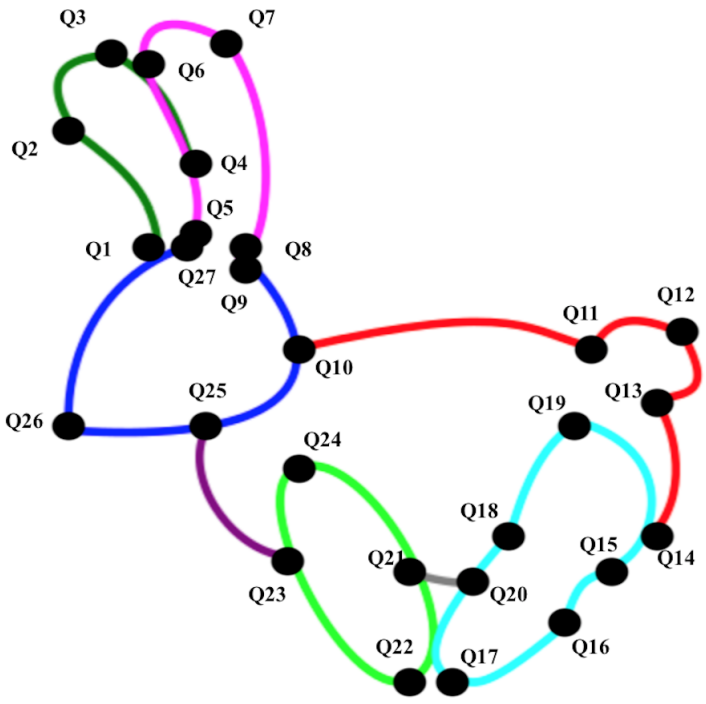
As mentioned in the previous chapter, the goal of the first step of this method is to find the point to point correspondence for two inputted frames. While it is difficult to mathematically represent a good correspondence, we can determine if a correspondence is desirable visually. To illustrate this we look at the two poses of a rabbit in figure 2.2. In this example it is easy to see that the desired point to point correspondence would result In Points Pi in figure 5.3a getting mapped to the Points Qj in figure 5.3b in increased numeric order.

Moreover we can observe that the group of bezier curves illustrating a particular body part like the leg tend to roughly undergo the same transformation. Knowing this, the challenge is to group the curves that are likely to move together in one cluster and calculate a transformation that represents that cluster's movement from one pose to another. The details of how we compute these clusters and transformations will be discussed in Chapter 4 in which we detail our approach.

Once these clusters and the transformations are identified, knowing the point to point correspondence and having the transformations, an inverse skinning decomposition can be performed to give us a set of transformation weights that minimize the error between the transformed points in one frame and their corresponding points in the other frame. Having the blending weights and the clustering information, the illustrator can then modify the transformations to animate the character. Overall the desired output for this step of the algorithm is a set of weights that best illustrates the deformation of the character when modifying the skeletal structure of the character(s).



(a) Reference pose



(b) Deformed pose

Figure 2.2: Example of a desired grouping of curves for two consecutive key frames

Chapter 3: Related Work

To be able to automate the process of in-betweening and rigging, we first need to find the correspondence between the points in two consecutive key frames. Once the correspondence is found we can then either interpolate between those drawings to produce the in-between frames or derive a set of handles for manipulation. However finding the correspondences is a non-trivial task. While we won't be able to discuss all potential problems and their many proposed solutions, for a simple overview please refer to [1]. In this thesis we focus on addressing one of the main concerns which is that hand-drawn characters tend to undergo free-form deformations and so finding the correspondence can be quite difficult. Other problems such as obscured parts are not addressed, hence the inputted drawings are restricted to images without occlusion.

In the past two decades, various methods for computer assisted in-betweening and automated rigging have been proposed. The most popular class of approaches for finding the correspondence between drawings are the so called stoke-based methods. In such methods each frame is considered to be a collection of strokes or splines. Therefore the main objective is to find the correspondence between the set of curves in two consecutive frames. These methods often eliminate undesirable correspondences based on a set of predefined rules and then proceed to identify the best possible correspondence based on an energy function. Since the elimination rules are based on a set of visual and usually style specific assumptions, their performance can vary based on the input. Restrictive style specific elimination rules can limit the effectiveness of some methods to a particular set of inputs[2, 3].

Stroke-based Methods that are currently used during production of 2D cartoons such as the algorithm in BetweenIT, require clean input with neatly connected curves [4]. This is because this algorithm uses connectivity information and so drawings containing gaps

between curves or intersecting lines can result in undesirable results that will require manual correction. To reduce the work done in case of undesirable results other algorithms such as CACAI which rely on shape information incorporate user feedback into the steps of the algorithm [5]. Our approach is fully automated and goes beyond just finding the correspondence.

Another state of the art stroke-based approach is the FTP-SC Algorithm which is proposed for a small set of high confidence correspondences that can be Incorporated into our approach. Again the FTP-SC algorithm focuses on the correspondence problem while our method also automates rigging which allows the animator to produce other poses of the character in addition to in-betweens [6].

In contrast to stroke-based approaches that aim to find correspondence, layer-based and skeleton based methods often times require either the correspondence to be known or they require other information to be specified by the user [7, 8]. The information needed might include but is not limited to color, texture, shape boundaries and etc. The method described in Tooncap [9] is considered layer-based and requires the user to annotated the reference frame. our approach finds the correspondence and the handles by only using the curves of the input.

In addition to the methods discussed above, the second stage of our algorithm for character rigging is similar to the SSDR algorithm by Le and Deng [10]. However the SSDR algorithm operates on 3D input and requires the correspondence to be known before hand while our approach also finds the correspondence in the first step of our algorithm.

Chapter 4: Our Approach

4.1 Overview

In this section we first provide an over view of the Algorithm and then proceed to describe the steps in more detail. This algorithm finds the correspondence by considering all possible rigid transformations for each pair of points from one key frame to the other. It then eliminates undesired transformations based on an energy minimization function and recalculates the remaining labels in an iterative process. After a constant number of predetermined iterations the resulting label assignment is used to compute the optimum linear blending weights to transform the character from one pose to another while minimizing the reconstruction error. Finally the resulting transformations and weight are used to interpolate between key frames or to animate the character to any pose.

4.2 Uniform Sampling

The inputed key frames are vectorized hand-drawn characters in which strokes are represented as cubic bezier curves. To be able to calculate a dense correspondence and to eliminate unwanted local ambiguities when searching for possible correspondences uniform discrete sampling is first performed on all curves in the key frames at a user specified sampling rate (see figure 4.1). Using the value of the curve parameter and the control points of the Bezier curve to which the sample belongs to, the tangents at the sampled points are computed and normalized for all keyframes.

Algorithm 1: Overview

Data: $Frame1_{curves} = \{c_1, c_2, ..., c_m\}$, $Frame2_{curves} = \{c_1, c_2, ..., c_m\}$
Result: $Labels = \{l_{p_1}, l_{p_2}, ..., l_{p_n}\}$, $Weights = \{w_{p_1}, w_{p_2}, ..., w_{p_n}\}$

$F1_{points}, Fr1_{\hat{tgs}} = UniformSampling(F1_{curves}, sampleRate)$
$F2_{points}, F2_{\hat{tgs}} = UniformSampling(F2_{curves}, sampleRate)$
$MkScalingUniform(F1_{points}, F2_{points})$
$Rotations, Translations = InitializeTransformations(F1_{points}, F2_{points})$
$itr \longleftarrow 0$
while $itr < ITRCONST$ **do**
    $MultiLabelOPT_{uni}(Frame1_{points} \mapsto Frame2_{points})$
    $MultiLabelOPT_{uni}(Frame2_{points} \mapsto Frame1_{points})$
    $MultiLabelOPT_{bi}(Frame2_{points}, Frame1_{points})$
    $DeriveClusters()$
    $RecomputeTransformations()$
end
$ComputeBlendingWeights()$
$InterpolateTransformations()$
$OutputAnimation()$
return $labels, Weigts$

4.3 Uniform Scaling

The key frames are provide to us as svg files and The scaling of the characters can affect both the energy terms and the weights calculated. Because of this, it is important to unify the scaling in all input images so that the units in which the calculations are made are predictable. This is done by iterating through the coordinates of all sampled points and finding the minimum and maximum in both x and y. The minimum and maximum are then used to scale all the point coordinates to a predefined size.

4.4 Initializing Transformations

Having the coordinates of the points in two consecutive key frames and assuming that each point in one frame has a corresponding point in the other, there are n possible corresponding points for each point. While the number of options are limited this information alone is not

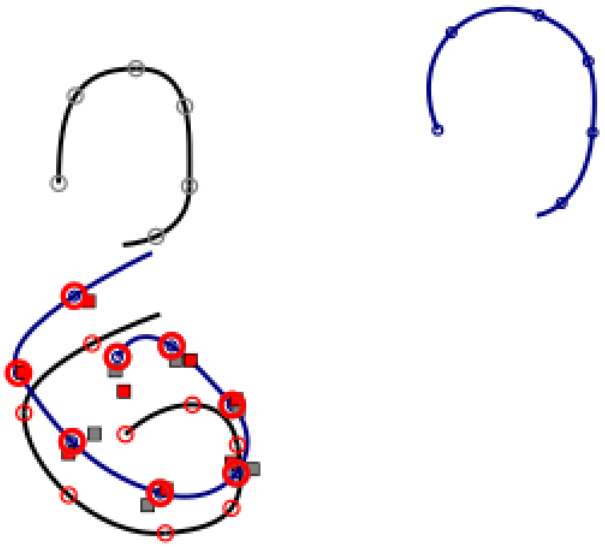


Figure 4.1: Undersampled key frames produce incorrect correspondence since the gray round dots get mapped to the gray squares.

sufficient to find a unique transformation that a point can undergo to reach its corresponding point. Because of this the unit tangents is added to the coordinates of their respective point to produce a second point. Now the two points in one frame and their corresponding two points in the other frame can be used to compute a unique best-fitting rigid transformation (see Algorithm 1). The unique transformation between the now two cluster of points is computed using singular-value decomposition (see algorithm 2).

4.5 Unidirectional multi-label optimization

Now having n^2 transformations or labels a label assignment is computed in both forward (from $Frame_1$ to $Frame_2$) and reverse direction (from $Frame_2$ to $Frame_1$). This is done by minimizing the energy function below:

$$E(labeling)_{uni} = DataCosts_{uni}(labeling) + SmoothCosts_{uni}(labeling) + LabelCosts_{uni}(labeling) \tag{4.1}$$

Algorithm 2: Initializing transformations

Data: $Frame1_{points} = \{p_1, p_2, ..., p_n\}$, $Frame2_{points} = \{p_1, p_2, ..., p_n\}$,
$Frame1_{t\hat{g}s} = \{\vec{v_1}, \vec{v_2}, ..., \vec{v_n}\}$, $Frame2_{t\hat{g}s} = \{\vec{v_1}, \vec{v_2}, ..., \vec{v_n}\}$

Result: $Rotations = \{R_1, R_2, ..., R_{n^2}\}$, $Transtions = \{T_1, T_2, ..., T_{n^2}\}$

for $p_i \in Frame1_{points}$ *and* $v_i \in Frame1_{t\hat{g}s}$ **do**
  for $q_i \in Frame2_{points}$ *and* $u_i \in Frame2_{t\hat{g}s}$ **do**
    $A \longleftarrow \begin{bmatrix} p_i \\ p_i + v_i \end{bmatrix}$ $B \longleftarrow \begin{bmatrix} q_i \\ q_i + u_i \end{bmatrix}$
    $R, T \longleftarrow ComputeTransformations(A, B)$
    $Rotations \longleftarrow Rotations + \{R\}$
    $Translations \longleftarrow Translations + \{T\}$
    return $Rotations, Translations$
  end
end

Algorithm 3: Computing unique transformation between two clusters

Data: $Cluster_1 = \{p_1, p_2, ..., p_n\}$, $Cluster_2 = \{p_1, p_2, ..., p_n\}$

Result: Rotation, Translation

for $p_i \in Cluster_1$ *and* $q_i \in Cluster_2$ **do**
  $\bar{p} \longleftarrow \bar{p} + p_i$
  $\bar{q} \longleftarrow \bar{q} + q_i$
end
$\bar{p} \longleftarrow \bar{p}/n$
$\bar{q} \longleftarrow \bar{q}/n$
$X \longleftarrow \begin{bmatrix} (p_{x1} - \bar{p}) & (p_{x2} - \bar{p}) & ... \\ (p_{y1} - \bar{p}) & (p_{y2} - \bar{p}) & ... \end{bmatrix}$
$Y \longleftarrow \begin{bmatrix} (q_{x1} - \bar{q}) & (q_{x2} - \bar{q}) & ... \\ (q_{y1} - \bar{q}) & (q_{y2} - \bar{q}) & ... \end{bmatrix}$
$ComputeSVD(X * Y^t)$
$Rotation \longleftarrow V * Determinant(V * U^t) * U^t$
$Translation \longleftarrow \bar{q} - (Rotation * \bar{p})$
return $Rotation, Translation$

The reason for computing a label assignment in each direction separately is that it allows us to derive an initial correspondence which is then used to refine the smoothness term in the energy function.

Here we have a finite set of points in the first frame and a finite set of transformations or labels L. A labeling l is assignments of labels in L to points in first frame. The individual points are referred to with small letters p and q, label of point p is denoted by l_p, and the set of all label-point assignments is denoted by l, that is $l = \{l_p | p \in Frame\}$. The goal is to find a label assignment that minimizes this energy function. In the following subsections each term in the energy minimization function is explained in detail.

4.5.1 Data Terms

The first term in the energy function is called the data term, and it is the sum of the costs of assigning label l_{p_i} to point p_i for all points. In other words the data term is the cost of a particular label assignment. To set the penalty for each point label pair, the point p_i in the first frame is transformed by label l_{p_i}. the resulting point is then used to calculate the minimum distance between this point and all the points in the second frame. This minimum value is then set as the data cost of the point label pair. However since this distance is a floating point number, it is converted to an integer and then used to set the cost.

$$DataCosts_{uni}(labeling) = \sum_{i=1}^{n} D(p_i, l_{p_i}) \tag{4.2}$$

$$D_{uni}(p_i, l_{p_i}) = min(|(R_{l_{p_i}} \cdot p_i + T_{l_{p_i}}) - q_i|) \tag{4.3}$$

4.5.2 Smoothness Terms

The smoothness cost specifies a penalty for choosing labels l_{p_i} and l_{p_j} for neighboring points p_i and p_j. In this term points p_i and p_j are considered neighbors if they belong to the same curve or curves that share an endpoint. This is because such points are more likely going

to undergo similar transformation. Therefore assigning a penalty for neighbors that didn't choose the same label can aid proper clustering and can direct the algorithm to choose identical labels for connected curves. The smoothness term which is the second term in the energy function is the sum of all the penalties paid for not picking the same label for neighboring points. The neighborhood relations specified on the set of points need to be symmetric, that is if p_i is a neighbor of p_j then p_j should be a neighbor of p_i. Here it is also assumed that for each pair of neighbors that need to be penalized, the penalty is only added once to the energy term not twice.

$$SmoothCosts_{uni}(labeling) = \sum_{i=1}^{n} V(p_i, p_j, l_{p_i}, l_{p_j}) \tag{4.4}$$

$$V_{uni}(p_i, p_j, l_{p_i}, l_{p_j}) = \begin{cases} 0 & \text{if } l_{p_i} = l_{p_j} \\ & \forall p_i, p_j \in Neighbors \\ 10 & \text{otherwise} \end{cases} \tag{4.5}$$

4.6 Bidirectional multi-label optimization

Having computed a unidirectional correspondence in the forward direction — from the reference frame to the deformed frame — and, separately, in the reverse direction — from the deformed frame to the reference frame —, this computed information can be incorporated into a bidirectional energy that seeks consistency in the labeling (see equation 4.6).

$$\begin{aligned} E(labeling)_{bi} = & DataCosts_{bi}(labeling) + SmoothCosts_{bi}(labeling) + \\ & LabelCosts_{bi}(labeling) \end{aligned} \tag{4.6}$$

Knowing that transformation matrices associated with each label in the reverse direction can be derived by computing the inverse of the transformations in the forward direction,

the set of unique labels chosen by either unidirectional assignment can be utilized to redefine the terms of the bidirectional energy function. In the data term of the bidirectional energy function, both directions are considered by using the inverse transformations when setting the costs of the deformed frame's points. Moreover the bidirectional Smoothness term is augmented with an additional term by utilizing the unidirectional label assignment information computed in the previous steps.

Having the label assignment in both forward and reverse direction, first for each each point in the reference frame the selected label transformation is applied and the nearest point in the deformed frame is identified to be the corresponding point (see algorithm 4). The same is done for points in the deformed frame by applying the inverse transformations. Now having the corresponding points in both directions an adjacency matrix A is derived in which $A(p_i, q_j) = 1$ for point p_i and its correspondence q_j. This matrix can then be utilized to set the additional cost in the bidirectional Smoothness term. In the following subsections each term in the energy minimization function is explained in detail.

Algorithm 4: Computing Adjacency Matrix

Data: $Frame1_{points} = \{p_1, p_2, ..., p_n\}$, $Frame2_{points} = \{q_1, q_2, ..., q_n\}$, $UniLabels_{F1} = \{l_{p_1}, l_{p_2}, ..., l_{p_n}\}$, $UniLabels_{F2} = \{l_{q_1}, l_{q_2}, ..., l_{q_n}\}$
Result: Adjacency Matrix A

$A \longleftarrow ZeroMatrix(n^2, n^2)$
for $p_i \in Frame_1$ **do**
  $p\prime_j \longleftarrow R_{l_{p_i}} * p_i + T_{l_{p_i}}$
  $q_j \longleftarrow Min(|p\prime_i - q_j|)$
  $A(i, j + n) \longleftarrow 1$
  $A(j + n, i) \longleftarrow 1$
end
for $q_j \in Frame_2$ **do**
  $q\prime_i \longleftarrow R_{l_{q_j}} * q_j + T_{l_{q_j}}$
  $p_i \longleftarrow Min(|q\prime_j - p_i|)$
  $A(i, j + n) \longleftarrow 1$
  $A(j + n, i) \longleftarrow 1$
end
return A

4.6.1 Data Terms

While the list of all possible transformations was computed from transforming points from one frame to the next, the list of all possible transformations that transform points in the reverse direction can be easily computed. Now to set the data term for the bidirectional multi-label optimization, the forward transformations are applied to points on the reference frame and the reverse transformations are applied to the deformed frame to find the closest corresponding point. In other words, the bidirectional data term can be derived by combining the data terms used in the two unidirectional optimizations.

$$DataCosts_{bi}(labeling) = \sum_{i=1}^{n} D(p_i, l_{p_i}) + \sum_{i=n}^{n^2} D(q_{n-i}, l_{q_{n-i}}) \quad \forall p_i \in Frame_1, \forall q_j \in Frame_2 \tag{4.7}$$

4.6.2 Smoothness Terms

As discussed earlier, the smoothness cost specifies a penalty for choosing labels l_{p_i} and l_{p_j} for neighboring points p_i and p_j. In the unidirectional multi-label optimization only points that belong to the same curve or neighboring curves are considered to be neighbors since they are more likely to undergo the same transformation. However, it can also be said that if the unidirectional multi-label optimization leads to the mapping of a point in one frame to a point in the other frame, the chosen pair of points have a higher probability to be correct correspondence for each other. While this information can be used to refine the smoothness term, it is less reliable than the the neighbor relations defined in the unidirectional smoothness term. Because of this although the corresponding pairs will be considered neighbors, to reflect this neighbor relation is less reliable, the assigned penalty for such neighbors in the bidirectional smoothness term is reduced . The value of the penalties are user specified constants and in our case the penalty for reliable neighbors is set to 100 while the penalty

for unreliable neighbors is set to 10.

$$SmoothCosts_{bi}(labeling) = \sum_{i=1}^{n^2} V(p_i, p_j, l_{p_i}, l_{p_j})$$
$$\forall 1 \leqslant i \leqslant n, p_i \in Frame_1, \forall n \leqslant i \leqslant n^2, p_i \in Frame_2 \tag{4.8}$$

$$V_{bi}(p_i, p_j, l_{p_i}, l_{p_j}) = \begin{cases} 0 & \text{if } l_{p_i} = l_{p_j}, \forall p_i, p_j \in ReliableNeighbors \\ 100 & \text{if } l_{p_i} \neq l_{p_j}, \forall p_i, p_j \in ReliableNeighbors \\ 10 & \text{if } l_{p_i} \neq l_{p_j}, \forall p_i, p_j \in UnreliableNeighbors \end{cases} \tag{4.9}$$

4.7 Recomputing Transformations

Having computed the bidirectional label assignment, the clusters in both forward and reverse direction are derived. Since there is a possibility that multiple points in one frame are mapped to the same points in the other, the label assignment might be more accurate in one direction versus the other. Because of this, the accuracy of the label assignment in each direction is measured by counting the number of points covered in the corresponding pose. Having two values (one for each direction), the coverage of the label assignments are compared and the direction with the larger coverage is chosen for recomputing the transformations.

As discussed earlier, when trying to compute a unique transformation for a pair of points, due to the lack of sufficient information, two additional points were derived by adding unit tangents. This resulted in two clusters which were then used to compute a unique transformation. However after the multi-label optimization is performed, each cluster will have more points and hence it is important to recompute the transformations based on the current clusters to have more accurate labels. To recompute the transformations 3 is performed on each pair of corresponding clusters.

4.8 Termination Criteria

At the end of each Iteration of the algorithm, to ensure that good labels are not eliminated too soon without any chance of reconsideration n random labels are selected from the n^2 initial possible choices and inserted into the list of labels to be considered in the next iteration. It is worth noting that the number of iterations in this algorithm is a constant determined by the user and in our case it was picked to be 5.

4.9 Blending Weights and Transformation Interpolation

After the termination of the main loop of the algorithm, the final label assignment is identified. Now having the point to point correspondence and a set of transformations we will solve for the blending weights of each pair of corresponding points [10]. The optimal set of weight that minimize the reconstruction error from the reference to the deformed pose can be calculated using a quadratic programming solver 4.10. Having the weights, the transformations can be manipulated to produce different poses of the character. Moreover, having the transformations and the blended weights the user can interpolate between the frames by interpolating the transformations 5.

$$\begin{aligned} min_{reconstruction} E &= \sum_{i=1}^{n} \|q_i - \sum_{j=1}^{|B|} w_{ij}(R_j p_i + T_j)\|^2 \\ w_{ij} &\geqslant 0, \forall i, j \quad \text{and} \quad \sum_{j=1}^{|B|} w_{ij} = 1, \forall i \end{aligned} \tag{4.10}$$

Algorithm 5: Interpolate Transformations

Data: $Frame1_{points} = \{p_1, p_2, ..., p_n\}$,
$Labels = \{l_1, l_2, ..., l_n\}$,$Weights = \{W_1, W_2, ..., W_n\}$ $numberOfFrames$

Result: In-between Frames

$t \longleftarrow 0$
for $j = 1$ **to** *numberOfFrames* **do**
  $Rs_t \longleftarrow \{\}$ $Ts_t \longleftarrow \{\}$
  for $l_i \in Labels$ **do**
    $deg \longleftarrow RotationAngle(R_{l_i})$
    $R_t \longleftarrow RotationFromAngle(deg * t)$
    $Rs_t \longleftarrow Rs_t + \{R_t\}$
    $Ts_t \longleftarrow Ts_t + \{T_{l_i} * t\}$
    $t \longleftarrow j * 1/numberOfFrames$
  end
  $Frame_t \longleftarrow \{\}$
  for $p_i \in Frame1_{points}$ **do**
    $R_{p_it} \longleftarrow w_{i1} * R_1 + w_{i2} * R_2 + ... + w_{im} * R_m$
    $T_{p_it} \longleftarrow w_{i1} * T_1 + w_{i2} * T_2 + ... + w_{im} * T_m$
    $Frame_t \longleftarrow Frame_t + \{R_{p_it} \ * p_i + T_{p_it}\}$
  end
  $DrawFrame(Frame_t)$
end

Chapter 5: Results and Applications

To test the algorithm the method described in this thesis was implemented in C++. The implementation accepts the reference and deformed frames as svg files and uses svg++ — a third party svg parser library — to parse the two svg file contents (see figure 5.1 a and b). The parsed cubic bezier curves are then uniformly sampled at the user specified rate. The points are then used to perform the algorithm. To perform the multi-label optimization another third party library called GCO was used [11–13]. Once the weights and the final transformations are computed the program outputs the following svg files:

1. An svg file containing the point clusters of the reference frame and the corresponding clusters in the deformed frame (forward clusters). Each pair of corresponding clusters use a unique color (see figure 5.1c).

2. A series of svg files that contain the interpolated frames at the user specified rate (see figure 5.1 d to i).

To evaluate the algorithm a series of images were used as test inputs and the results were manually examined. The results of some of these images are shown in figure 5.1 to 5.9. Overall, this algorithm performed fairly well in cases in which unique or distinguishable local features were present but performed inconsistently in cases that contained non-distinguishable features. For example, in figure 5.1 and 5.2 the s-shaped drawing at the bottom of the reference frame was correctly mapped to the s-shape drawing in the deformed frame. This is because the entire shape consists of connecting curves which are identified as neighbors and hence are encouraged by the smoothness term to be clustered together.

In contrast while the input frames in figure 5.1 and 5.2 are the same, varying the sample rate resulted in different correspondences in the presence of global ambiguity. The reason

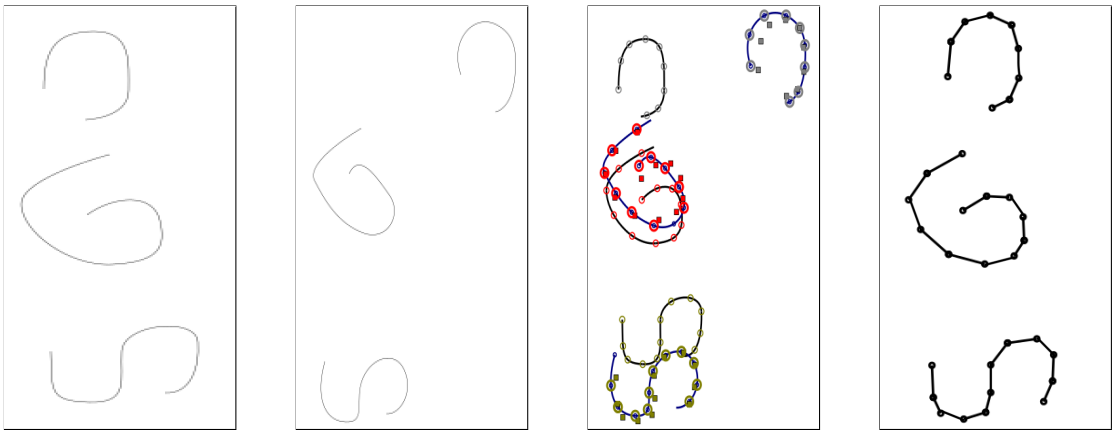
Input Images	Number of Curves	Sampling Rate (points per curve)	Runtime (minutes per iteration)
Simple	11	8	0.654
Rabbit	22	8	10.36
Fish	23	8	13.42
Man	26	8	23.68
Seahorse	29	5	12.7
Penguin	29	5	11.9
Jellyfish	41	4	21.3

Table 5.1: Runtime of the examples in minutes per iteration

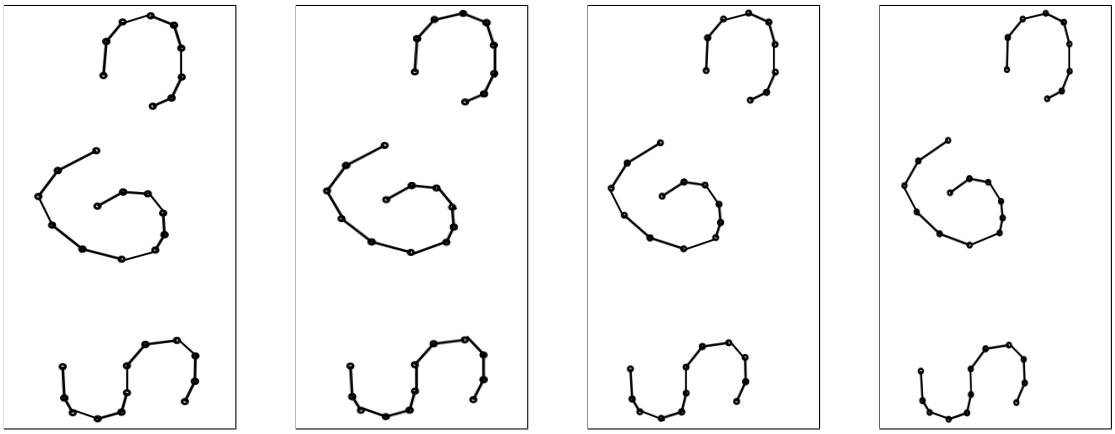
for this inconsistency is that for the c-shaped drawing on the top of the reference frame both the c-shape and part of the curve in the middle shape are both good corresponding choices locally. Moreover since overlapping correspondences are not disallowed and there is no logic for global information, this behaviour is as expected but not desirable.

As discussed in the previous example the outcome of this algorithm can vary depending on the curve sample rate. However in some cases unlike the previous example increasing the sample size can result in better correspondences. This is because some times slight differences in shape become more apparent when the sample rate is higher and higher sample rates can eliminate aliasing.

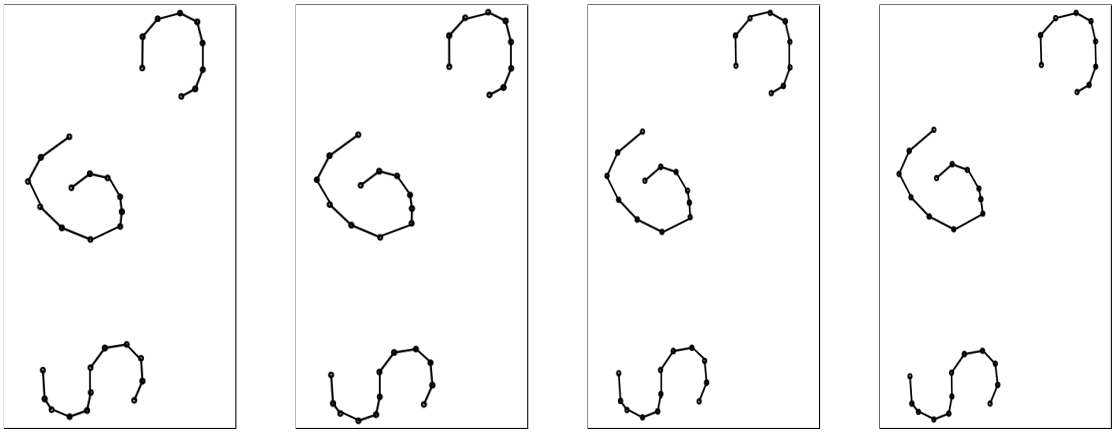
An interesting observation in results such as the rabbit in figure 5.4 is that in some cases the quality of the bidirectional label assignment varies from one direction to another in parts of the image. In these cases even though the label assignment in the reverse direction finds a better correspondence for the belly of the rabbit, this correspondence is not chosen. This is because determining the quality of the assignment solely based on the amount of coverage does assess the quality well.



(a) reference (b) deformed (c) correspondence (d) frame 2

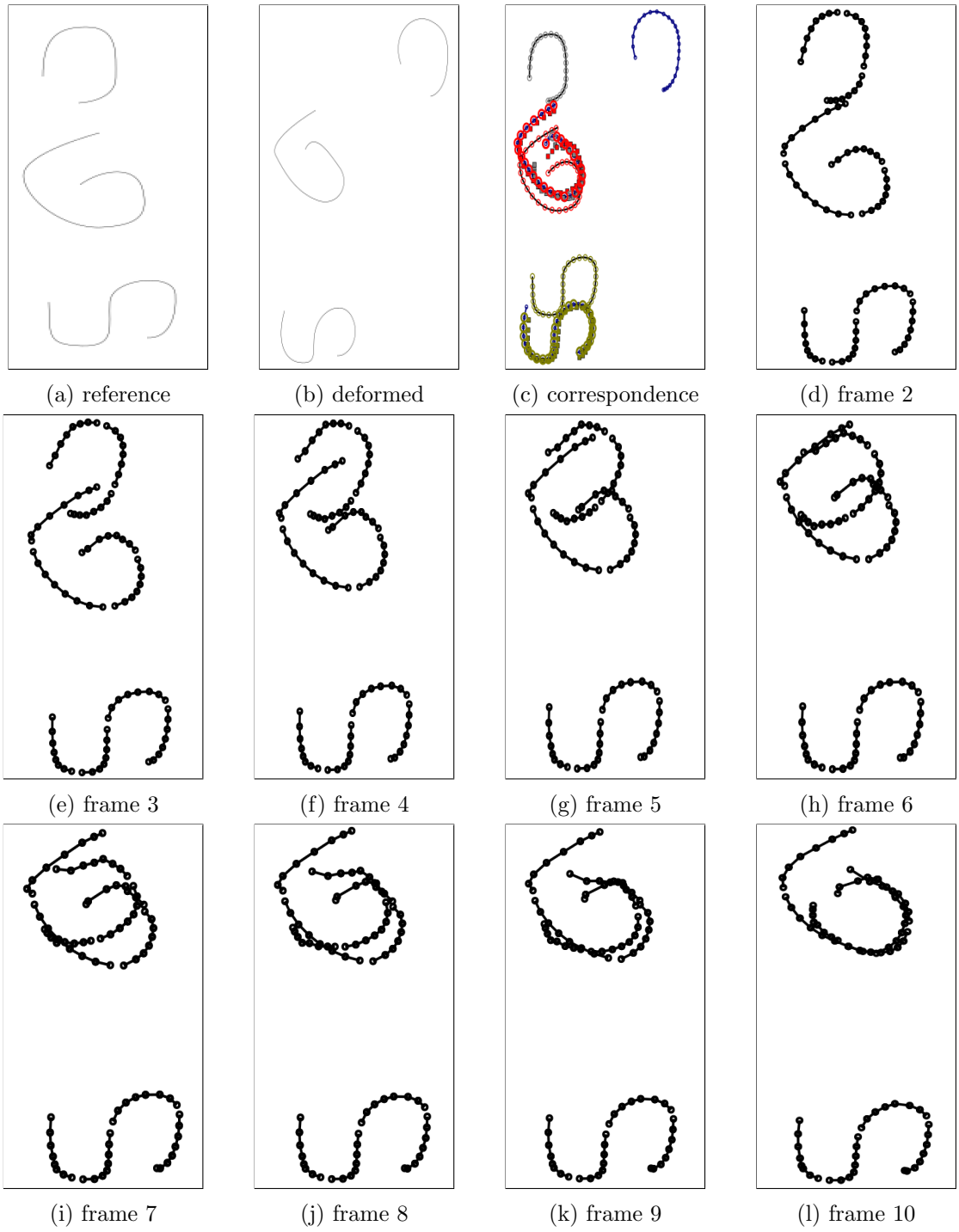


(e) frame 3 (f) frame 4 (g) frame 5 (h) frame 6



(i) frame 7 (j) frame 8 (k) frame 9 (l) frame 10

Figure 5.1: Simple shape sampled at 5 points per curve

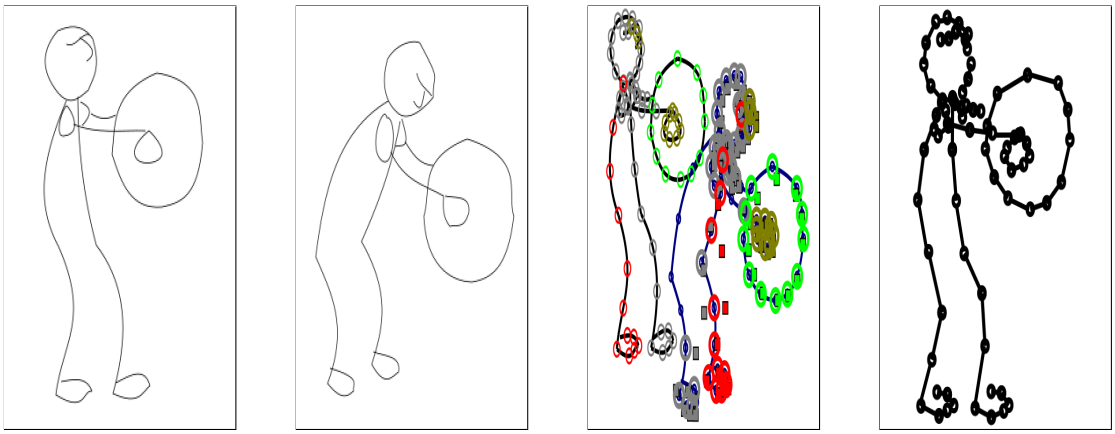


(a) reference (b) deformed (c) correspondence (d) frame 2

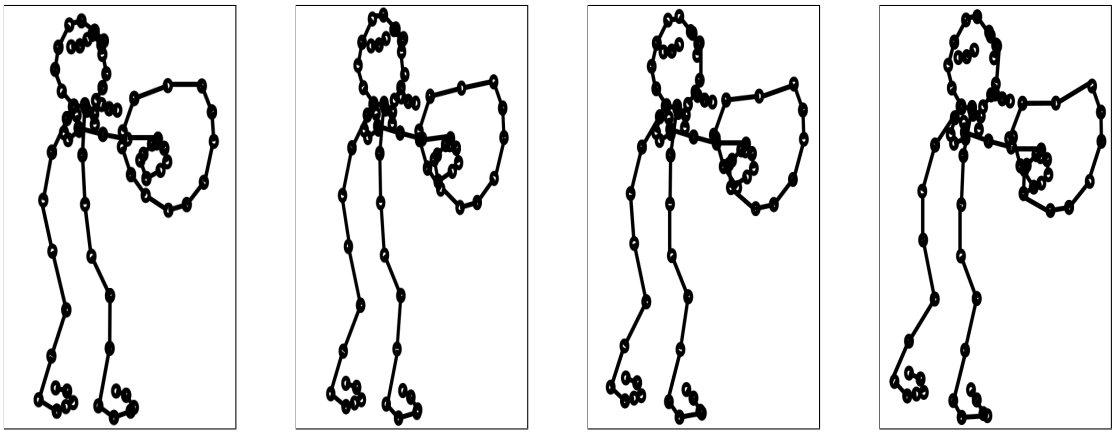
(e) frame 3 (f) frame 4 (g) frame 5 (h) frame 6

(i) frame 7 (j) frame 8 (k) frame 9 (l) frame 10

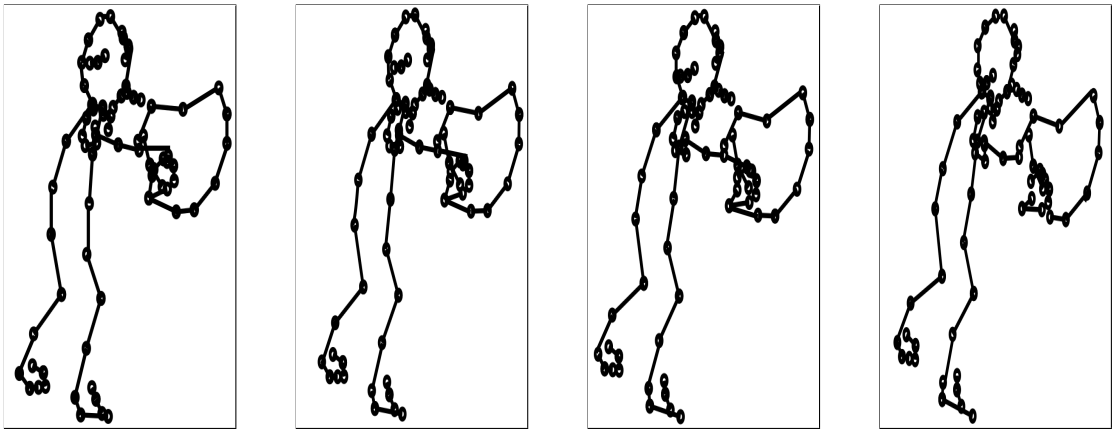
Figure 5.2: Simple shape sampled at 8 points per curve



(a) reference (b) deformed (c) correspondence (d) frame 2

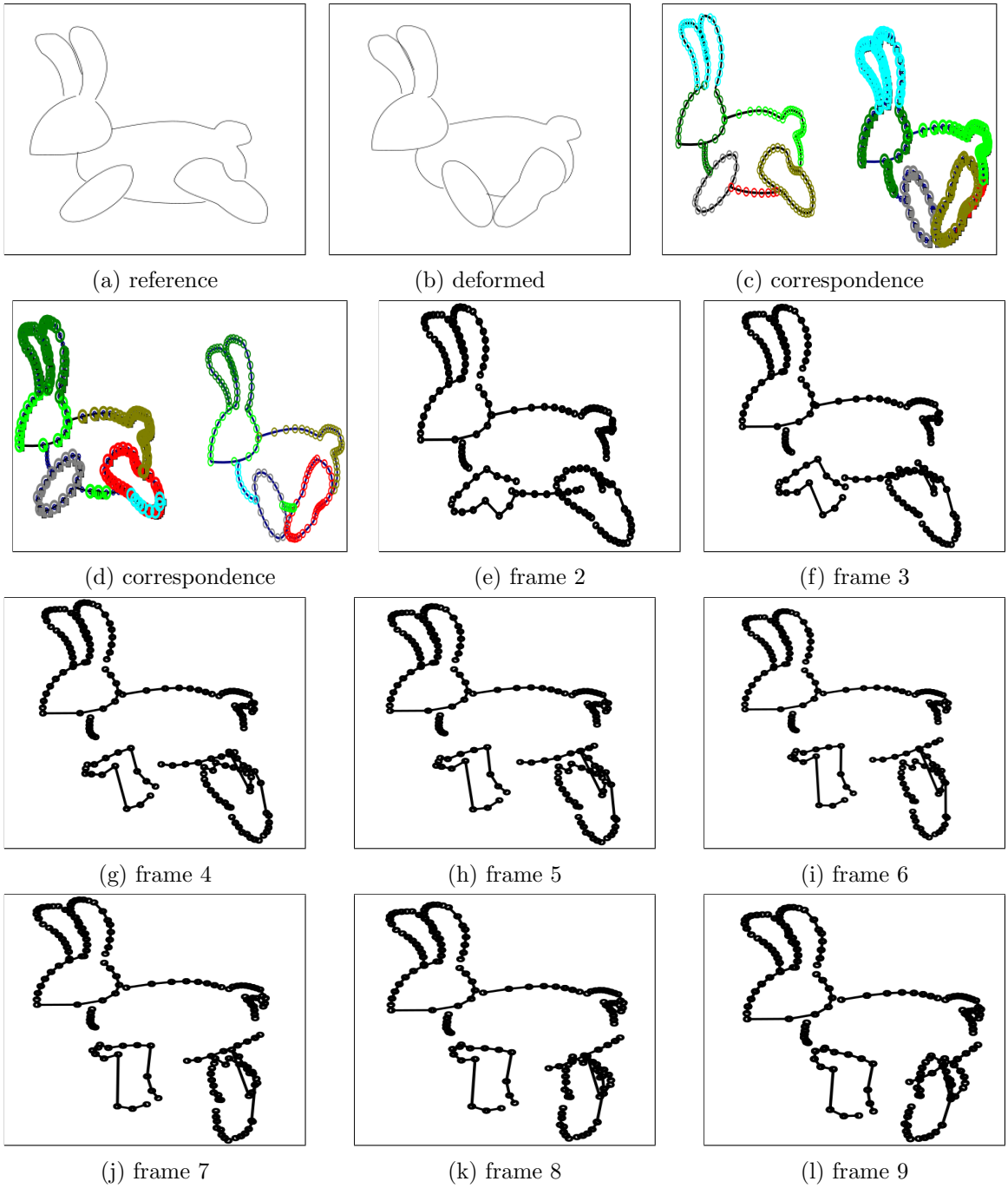


(e) frame 3 (f) frame 4 (g) frame 5 (h) frame 6



(i) frame 7 (j) frame 8 (k) frame 9 (l) frame 10

Figure 5.3: Man sampled at 5 points per curve



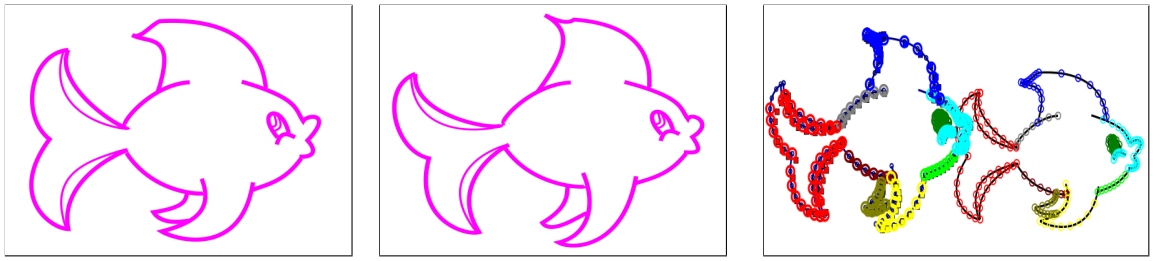
(a) reference (b) deformed (c) correspondence

(d) correspondence (e) frame 2 (f) frame 3

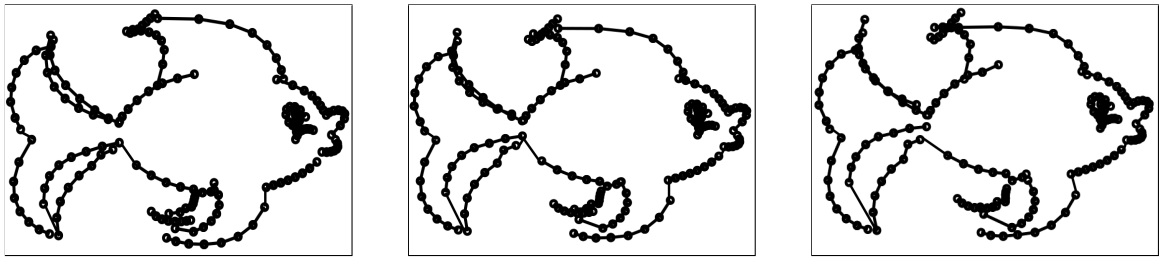
(g) frame 4 (h) frame 5 (i) frame 6

(j) frame 7 (k) frame 8 (l) frame 9

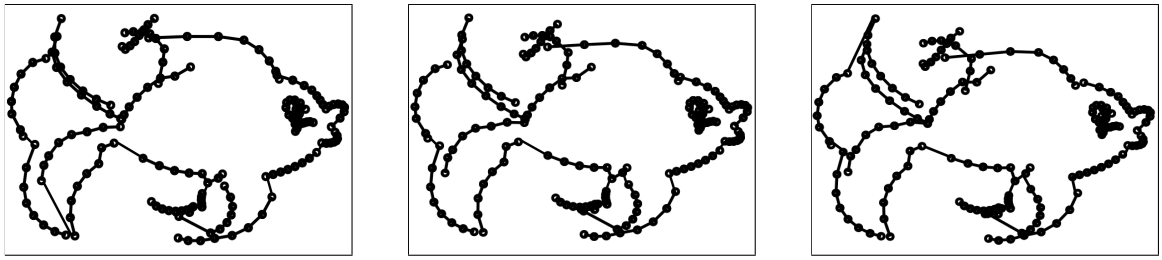
Figure 5.4: Rabbit sampled at 5 points per curve



(a) reference (b) deformed (c) correspondence



(d) frame 2 (e) frame 3 (f) frame 4



(g) frame 5 (h) frame 6 (i) frame 7

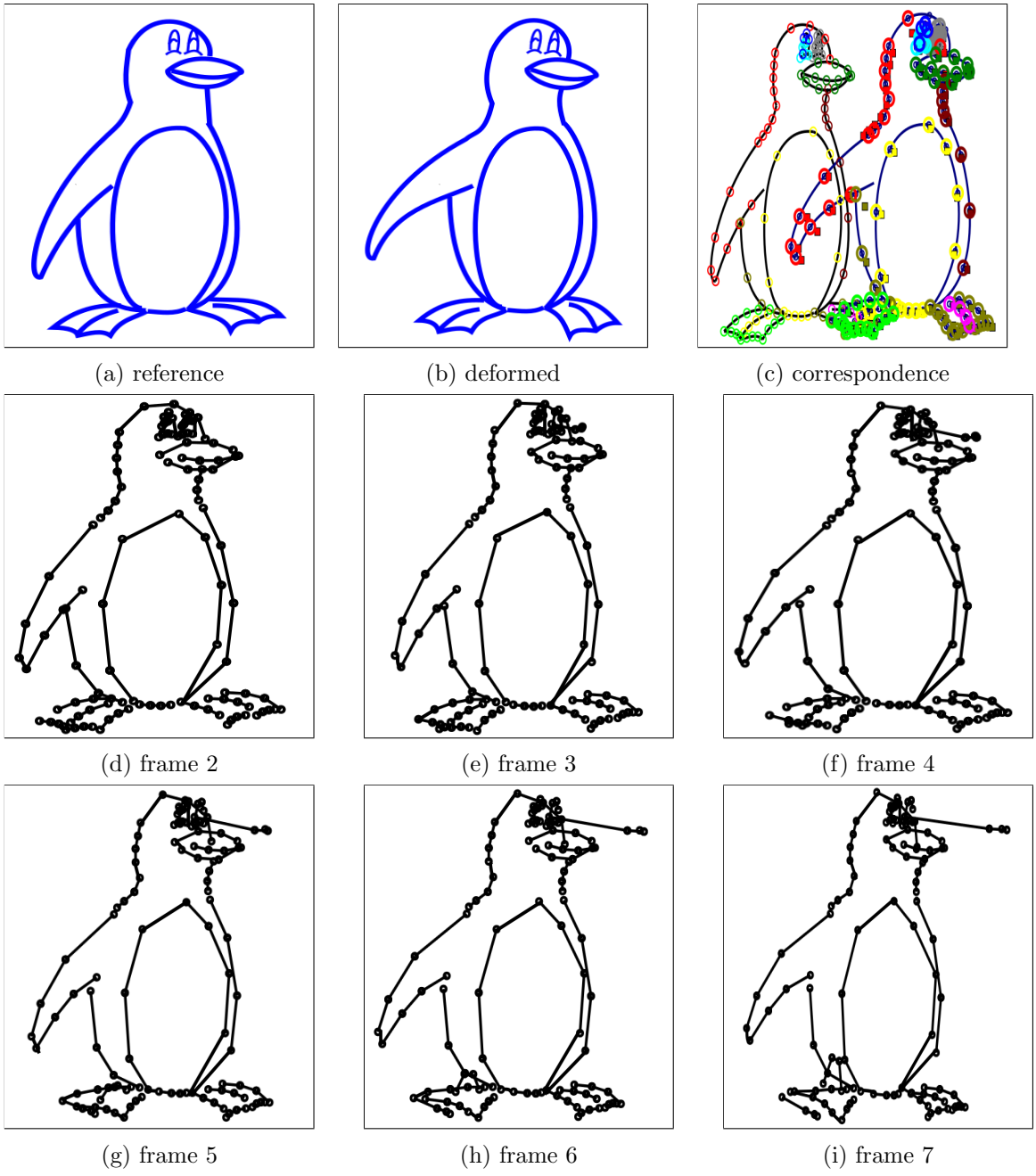
Figure 5.5: Fish sampled at 8 points per curve

(a) reference (b) deformed (c) correspondence

(d) frame 2 (e) frame 3 (f) frame 4

(g) frame 5 (h) frame 6 (i) frame 7

Figure 5.6: Seahorse sampled at 8 points per curve

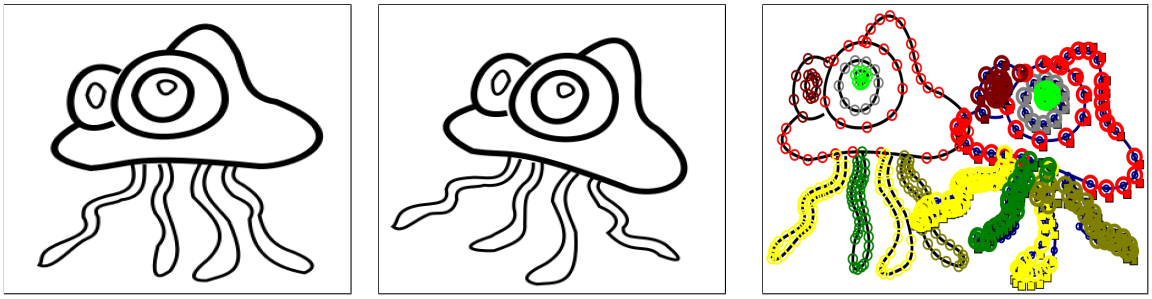


(a) reference (b) deformed (c) correspondence

(d) frame 2 (e) frame 3 (f) frame 4

(g) frame 5 (h) frame 6 (i) frame 7

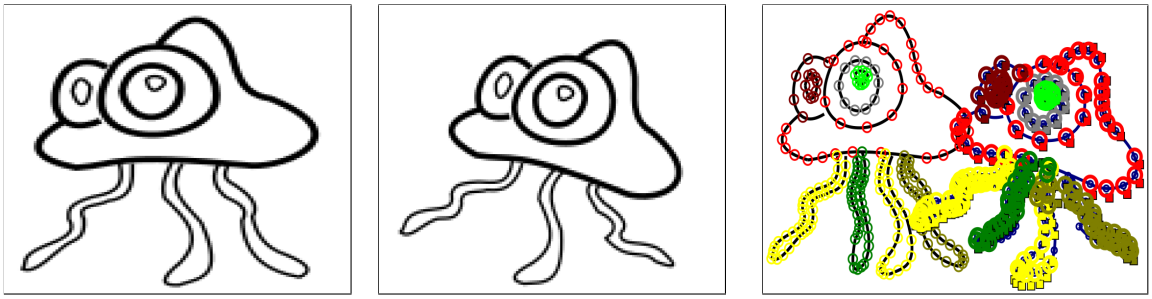
Figure 5.7: Penguin sampled at 8 points per curve



(a) reference (b) deformed (c) correspondence

(d) frame 1 (e) frame 10

Figure 5.8: four legged Jellyfish sampled at 8 points per curve



(a) reference (b) deformed (c) correspondence

(d) frame 1 (e) frame 10

Figure 5.9: Three legged Jellyfish sampled at 4 points per curve

Chapter 6: Conclusions and Future Work

This algorithm accepts two vectorized 2d key frames enables both frame interpolation and production of new frames though a set of handles. This approach is fully automated and requires no additional information from the user. Moreover the number of maximum iterations of this algorithm, sampling rate of the curves and the penalty paid by undesirable assignments can be changed by the user if desired.

Overall this algorithm performed well for images with distinct local features but the algorithm requires further improvement to be able to handle ambiguous cases such as the ones discussed in the results chapter. A possible way of resolving the current shortcomings related to this is to add logic that adjusts the incorrect choices based on global parameters. In the future we are planning to modify the algorithm to include such logic.

In addition, we also need to prevent the algorithm from assigning overlapping correspondences. One way of doing this is to remove one of the reference frame clusters that maps to the overlapping region along with the over lapping region and calculating a label assignment that minimizes the energy function. We can then remove the other reference cluster and put the previous one back in and repeat the energy calculation. Having two energy terms, we can then choose to assign the overlapping region to the cluster which produced a higher energy term. There are currently other modifications in progress with respect to the smoothness term. The algorithm as is encourages the bidirectional label assignment to be similar to the unidirectional label selections by assigning a small smoothness penalty to corresponding points that choose different labels. However in cases in which the forward label assignment and the reverse label assignment lead to different correspondences there will be a penalty independent of the label assignment. This is because choosing one results a penalty for the non-selected choice. This will not aid the label assignments and so we would like to limit the penalty to pairs that are found to correspond to each other in both

directions.

Overall, we would like to further improve the algorithm by adding redefining the bidirectional smoothness term and introducing a logic that adds global context. We would also like to test the algorithm on more complex and diverse set of inputs to further improve the algorithm to handle more realistic examples. However the algorithm as is, performs as expected and is able to produce acceptable results.

Bibliography

[1] E. Catmull, "The problems of computer-assisted animation," *SIGGRAPH Comput. Graph.*, vol. 12, no. 3, pp. 348–353, Aug. 1978. [Online]. Available: http://doi.acm.org/10.1145/965139.807414

[2] A. Kort, "Computer aided inbetweening," in *Proceedings of the 2Nd International Symposium on Non-photorealistic Animation and Rendering*, ser. NPAR '02. New York, NY, USA: ACM, 2002, pp. 125–132. [Online]. Available: http://doi.acm.org/10.1145/508530.508552

[3] B. Whited, G. Noris, M. Simmons, R. Sumner, M. Gross, and J. Rossignac, "Betweenit: An interactive tool for tight inbetweening," *Comput. Graph. Forum*, vol. 29, pp. 605–614, 05 2010.

[4] ——, "Betweenit: An interactive tool for tight inbetweening," *Comput. Graph. Forum*, vol. 29, pp. 605–614, 05 2010.

[5] W. Yang, "Context-aware computer aided inbetweening," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 2, pp. 1049–1062, Feb 2018.

[6] W. Yang, S. H. Soon, Q. Chen, H.-Z. Liew, and D. Sýkora, "Ftp-sc: Fuzzy topology preserving stroke correspondence," *Comput. Graph. Forum*, vol. 37, pp. 125–135, 2018.

[7] N. Burtnyk and M. Wein, "Interactive skeleton techniques for enhancing motion dynamics in key frame animation," *Commun. ACM*, vol. 19, no. 10, pp. 564–569, Oct. 1976. [Online]. Available: http://doi.acm.org/10.1145/360349.360357

[8] F. Di Fiore, P. Schaeken, K. Elens, and F. Van Reeth, "Automatic in-betweening in computer assisted animation by exploiting 2.5d modelling techniques," in *Proceedings Computer Animation 2001. Fourteenth Conference on Computer Animation (Cat. No.01TH8596)*, Nov 2001, pp. 192–200.

[9] X. Fan, A. H. Bermano, V. G. Kim, J. Popović, and S. Rusinkiewicz, "Tooncap: A layered deformable model for capturing poses from cartoon characters," in *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, ser. Expressive '18. New York, NY, USA: ACM, 2018, pp. 16:1–16:12. [Online]. Available: http://doi.acm.org/10.1145/3229147.3229149

[10] B. H. Le and Z. Deng, "Robust and accurate skeletal rigging from mesh sequences," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 84:1–84:10, Jul. 2014. [Online]. Available: http://doi.acm.org/10.1145/2601097.2601161

[11] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 23, no. 11, pp. 1222–1239, Nov. 2001. [Online]. Available: https://doi.org/10.1109/34.969114

[12] V. Kolmogorov and R. Zabih, "What energy functions can be minimizedvia graph cuts?" *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 147–159, Jan. 2004. [Online]. Available: https://doi.org/10.1109/TPAMI.2004.1262177

[13] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, Sep. 2004.

Curriculum Vitae

Nusha Mehmanesh began her collegiate education by enrolling at Northern Virginia Community College. She the transferred to George Mason University to study Computer Science. She graduated *Magna Cum Laude* with a Bachelor of Science in Computer Science from George Mason University in 2017. During that time, she also completed the requirements to be accepted into George Mason Universitys Computer Science Masters program.