Master of Science in Informatics at Grenoble
Master Informatique
Specialization Graphics, Vision, and Robotics

Sketch-Based Posing and Interaction of Multiple Characters: Animating Dancing Couples

Sarah Anne Kushner

Wednesday June 21, 2017

Research project performed at Inria Grenoble – Rhône-Alpes

Under the supervision of:

Prof. Marie-Paule Cani	Dr. Rémi Ronfard
École Polytechnique	Inria Grenoble
Paris-Saclay	Rhône-Alpes

Defended before a jury composed of:
Prof. James Crowley
Prof. Joëlle Thollot
Prof. Laurence Nigay
Assistant Prof. Dominique Vaufreydaz

June 2017

Abstract

3D human characters are difficult to pose and animate on their own, let alone while interacting with each other. Even experienced animators require a lot of time setting poses over time to create precise animations. The line of action, an artistic tool used mainly by cartoonists to indicate expressive poses, translates well as a concept to 3D and due to previous research is, available in a convenient and intuitive sketch-based interface. With the line of action, a user can sketch a line in the shape that they want a selected body part of a character to take.

This technique has been proven to work well with a single character. However, extending the line of action to work for multiple characters is not a simple task. Many issues come with animating multiple characters who interact with each other. Avoiding collisions, maintaining contacts, and twisting the characters become rather difficult.

We aim to extend the line of action to handle cases involving multiple characters, with a dancing couple as a case study. Dance is a good example of interactions; there are many cases of characters touching each other and moving separately. The characters come together and move apart over the course of a dance sequence. To allow the line of action to span over multiple characters in an intuitive way, we propose an algorithm to merge the skeletons of the characters to be able to pose the characters as one.

Résumé

Les personnages humains en 3D sont difficiles à positionner et à animer par eux-mêmes, sans parler de l'interaction des les uns avec les autres. Même les animateurs expérimentés ont besoin de beaucoup de temps pour établir des poses au fil du temps afin de créer des animations précises. La ligne d'action est un outil artistique utilisé principalement par les caricaturistes pour indiquer des poses expressives. Il se traduit bien par un concept à 3D et, grâce à des recherches antérieures, est disponible dans une interface basée sur l'esquisse pratique et intuitive. Avec la ligne d'action, un utilisateur peut esquisser une ligne décrivant la forme souhaité que la partie partie du corps sélectionné d'un personnage prenne.

Cette technique s'est avérée bien fonctionner avec un seul personnage. Cependant, étendre la ligne d'action pour travailler avec plusieurs personnages n'est pas une tâche simple. Beaucoup de problèmes apparaissent dans l'animation de plusieurs personnages interagissant les uns avec les autres. Eviter les collisions, maintenir les contacts et tordre les personnages devient plutôt difficile.

Nous visons à étendre la ligne d'action pour gérer les cas impliquant plusieurs personnages, avec un couple dansant comme étude de cas. La danse est un bon exemple d'interactions ; Il y a plusieurs cas où les personnages se touchent et se déplacent séparément. Les personnages se rassemblent et s'écartent au cours d'une séquence de danse. Pour permettre à la ligne d'action de s'étendre sur plusieurs personnages de manière intuitive, nous proposons un algorithme pour fusionner les squelettes des personnages afin de pouvoir positionner les personnages comme s'ils y en avait q'un seul.

Acknowledgement

I would like to officially say thank you to those without whom this thesis would not have been possible. It brings me joy to acknowledge the following people.

Thank you to David Breen, Marcello Balduccini and Daryl Falco from Drexel University for being not only excellent and inspiring mentors and teachers, but also for helping and supporting me on my journey to study in France.

James Crowley deserves credit for making my move to France viable, and putting up with my numerous emails. I would also like to thank him for teaching me in the first semester of M2 along with all the other MoSIG M2 professors. Thank you to the Fondation partenariale Grenoble INP for funding my first semester.

I would like to express my sincere gratitude to both my supervisors Marie-Paule Cani and Rémi Ronfard for their invaluable assistance, guidance, and patience through the process of a Master thesis.

To the IMAGINE engineers Maguelonne Beaud de Brive and Julien Daval: thank you for answering my development questions and teaching me the mysterious ways of Rumba and Blender.

I have so much appreciation for Tom. Thank you for letting me think out loud to you on the phone; comforting me when I freak out to you on the phone; sending me pictures of our baby bunny, Cinnamon Louise Bun; and providing me with constant encouragement all the way from the United States.

It wouldn't be fair not to mention and thank my dear dancer friends Ruth and Vivian for answering my sometimes obvious and weirdly specific questions about your experiences in dance and choreography.

Shout out to all the C1 bus drivers for getting me to and from Inria every day. Merci, au revoir.

Lastly, thank you to the jury for reading this report and listening to my presentation.

Contents

— 1 —
Introduction

1.1 Background

3D animation can be a painstakingly tedious activity. To create a desired animation, animators go through the long process of keyframing. Keyframes are set positions that define the start and end points of a movement in an animation, which are essentially are sequences of poses in time. Typically, animators assign keyframes to certain frames and poses over the animation, and in-between motions are then generated by a computer. To get an accurate animation, artists usually must assign many keyframes, then spend time adjusting and editing them to be more precise. The fact that industry professionals take so much time and effort to do this shows that for an amateur or untrained artist, creating *good* 3D animation is close to impossible.

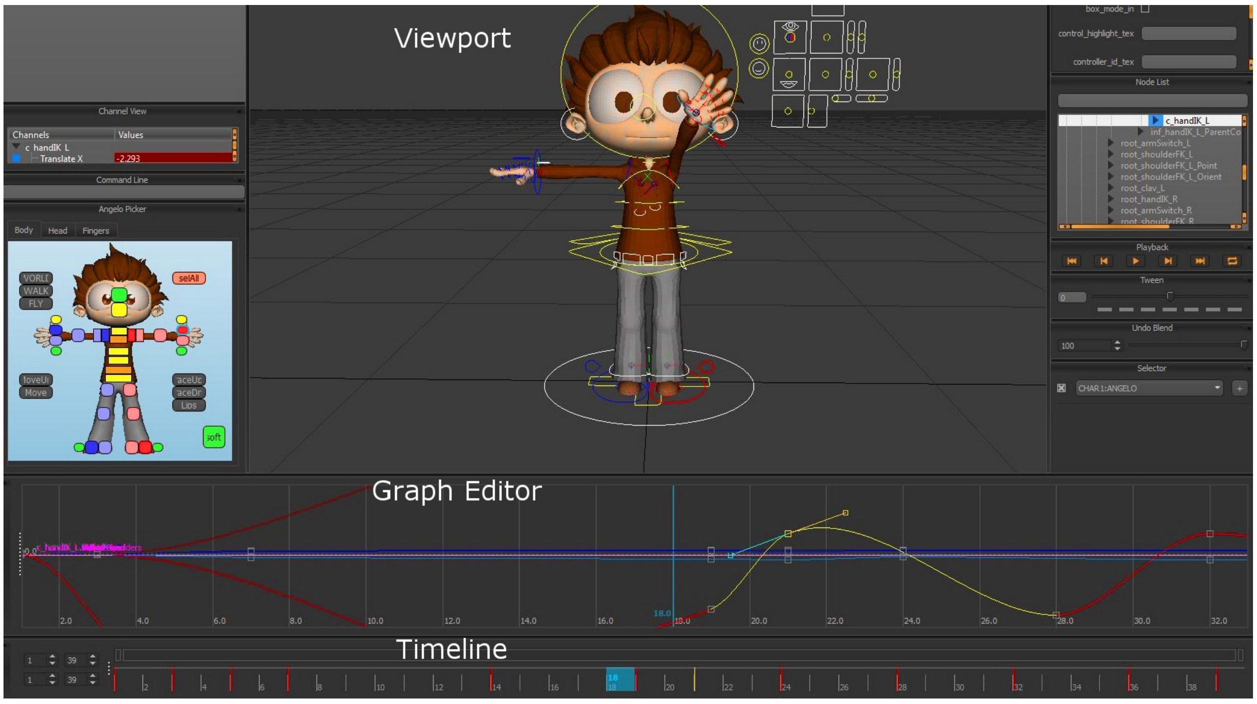
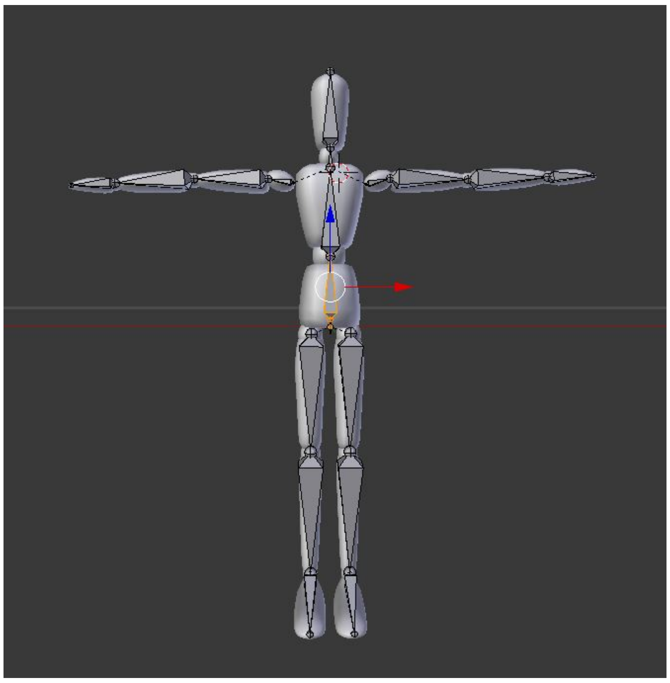


Figure 1.1 – Example of keyframing in Rumba. Red lines on the timeline indicate keyframes for an animation of the character waving.

In Figure 1.1, the character has keyframes attached to it on the timeline, which represent its various positions and rotations in time. In this case, the curves and keyframes shown are only attached to the character's left wrist for 39 frames, just over a second of animation. Imagine how many more keyframes it would take to animate his whole body.

1.2 Problem Statement

Among the most complicated characters to animate in 3D animation are humanoid characters. To ease this task, animators create a skeleton for their character called a **rig**, that consists of joints connected by rigid links (bones) to give a structure to the character. Humanoid rigs can range in complexity from somewhat simple to extremely complicated depending on the amount of detail desired by the user. The structure is a hierarchy of joints that can also be seen as a tree with a root, which in the humanoid case, is usually the pelvis. The leaf nodes of this tree, which are located at the maximal parts of the body, are called **end effectors**. Leaf nodes come at the end of a **kinematic chain**, which can be followed back up to the root. **Controls** for joints decide how to deform the geometric mesh of the character.


(a) Example of a humanoid skeleton.

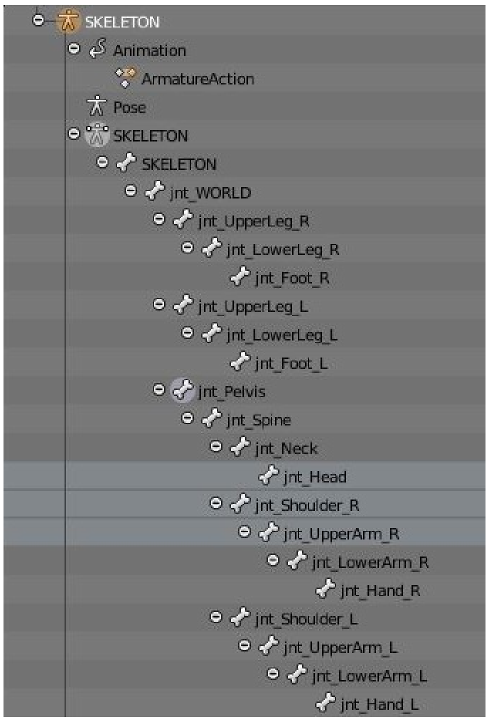

```
SKELETON
  Animation
    ArmatureAction
  Pose
  SKELETON
    SKELETON
      jnt_WORLD
        jnt_UpperLeg_R
          jnt_LowerLeg_R
            jnt_Foot_R
        jnt_UpperLeg_L
          jnt_LowerLeg_L
            jnt_Foot_L
        jnt_Pelvis
          jnt_Spine
            jnt_Neck
              jnt_Head
            jnt_Shoulder_R
              jnt_UpperArm_R
                jnt_LowerArm_R
                  jnt_Hand_R
            jnt_Shoulder_L
              jnt_UpperArm_L
                jnt_LowerArm_L
                  jnt_Hand_L
```
(b) The hierarchy of joints corresponding to the skeleton.

Figure 1.2 – Humanoid skeleton shown in Blender ([4]).

1.2.1 Kinematics

Forward and inverse kinematics are two general animation methods used mainly in situations in which articulated characters (jointed characters) need to move according to some constraints. In order to animate this structure successfully, controls are added that allow for forward and inverse kinematics. These controls help the animator move the character into poses that will then act as keyframes.

Forward kinematics (FK) is a method of calculating the position and orientation of the end effector (i.e. a hand or foot) given the positions and angles of the joints higher up in the chain all the way to the root.

Inverse kinematics (IK) is the method opposite of forward kinematics. That is, the goal is to calculate the angles and positions of joints in the chain, given the angle and position of the end effector. This goal is much harder to reach, seeing that more information needs to be calculated than is given. This problem is underconstrained, which means there can be more than one correct configuration that satisfies the constraints or there can even be no viable configurations. Many IK algorithms exist to calculate joint angles and positions.

Inverse kinematics are clearly more desirable for an animator since it is easier and faster to pose a character and have the joint angles automatically computed than it is to manipulate the character's joints directly.

1.2.2 The Line of Action

In practice, both forward and inverse kinematic controls are used in combination. Although this is the standard, even manipulating controls can be time-consuming. Sketch-based interfaces have started to become a plausible option for both animators and those with less experience. Many papers have been authored dealing with the sketch-based posing of a single humanoid character, discussed later in chapter 2. However, the animation of multiple characters comes with its own unique set of challenges as well. The problem is discovered when humanoid characters interact, namely when they are in close proximity to each other or when they touch each other.

The line of action is the concept of imagining a line that extends through the character's main action. It is commonly used by cartoonists in gesture drawings and the early stages of story-boarding to accentuate the motion and shape of the character. These lines are often dramatic in shape but smooth and simple in quality, usually containing only one or two extrema. The line of action goes through the majority of a character's body or through a part of the body and has a clear direction. See Figure 1.3.

The line of action is useful for expressive posing in a 2D medium. This concept transfers over to 3D sketch-based posing very intuitively. Shown in chapter 3, posing with the line of action works incredibly well for a single character. It provides a more straightforward and quick way of animating a single humanoid character. Extending the line of action to more than one character is nontrivial. Left on its own, many undesirable effects like collisions make the tool frustrating to use, rather than convenient.

1.3 Proposed Approach

The use case for this research in multi-character animation is a dancing couple. There are many combinations of poses a human can be in, let alone two humans *and* the two humans interacting. Specifically, a clip from the film "The Band Wagon" ([16]) was used to observe the common motions and poses in a dancing couple.

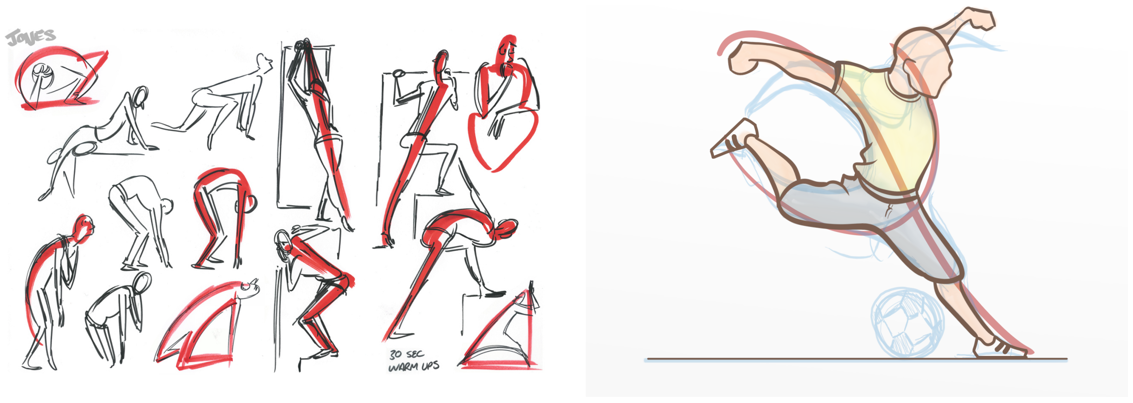


Figure 1.3 – Line of action examples from [27] and [26].

Using the previous work ([7], [8], and [9]), already in the process of being developed by the IMAGINE team, I extend the functionality of the line of action to be applied to multiple characters. Their software acted as the baseline with which to compare the new features. Baseline poses and animations were made by taking important keyframes from the film clip and recreating them. The amount of time spent running the program, the number of clicks, and number of lines drawn were recorded.

Since characters' skeletons are trees, a novel approach to solving this problem was combining these kinematic trees into one, in a new data structure with special attributes for posing and animating. An interesting way to evaluate this notion of animating a combined character was to, again, reproduce the same poses and animations as in the baseline, but this time using the new structure instead.

1.4 Contents of this report

In the following chapter (chapter 2), I cover the state-of-the-art for sketch-based posing and dance in animation. I will talk about the existing sketch-based systems used for posing articulated characters, covering the benefits and limitations of each. I discuss a brief history of dance notation – how choreographers and dancers use sketching on paper to brainstorm and communicate their ideas of motion, formation, and pose of dance.

In chapter 3, I discuss in depth the limitations of the current line of action concept used in practice. The biggest problem I will cover is the contacts between two characters.

Chapter 4 covers exactly how a solution was reached and what it entails. The approach to extending the line of action took a sizeable portion of my work, so I will take time to explain the systematic observation of our dance use case. Our proposed solution is described in detail.

Chapter 5 is where I go over the methods of validating our solution. Discussion of the lessons learned during this project and the concluding thoughts on the process are explored in chapter 6 and chapter 7, respectively.

— 2 —
State-of-the-Art

2.1 Animation of Single Characters

2.1.1 Sketch-Based Animation

Guay et al. originally proposed the line of action as a posing tool. In “The Line of Action: an Intuitive Interface for Expressive Character Posing” ([7]), there were several contributions: adapting the line of action (LOA) in the form of a new notation to posing 3D characters, solving the LOA problem as an optimization problem, and validating this technique by having users recreate poses from images.

From this paper came a prototype, in which a user can draw a line in the shape they want a kinematic chain to take. There is a limited amount of maximal, connected, linear kinematic chains, each of which is called a **body line**. Maximal means that the body line must begin and end at an extremity. A connected body line has edges (bones) between every joint (node) in the chain. Linear means the body line must be a straight chain through the tree with no branching, so each node is degree 1 or 2. Though these constraints limit the number of possible chains which can be selected, each character added to the problem adds more body lines. If characters are interacting or touching, this number increases dramatically to the point where a new technique needs to be introduced.

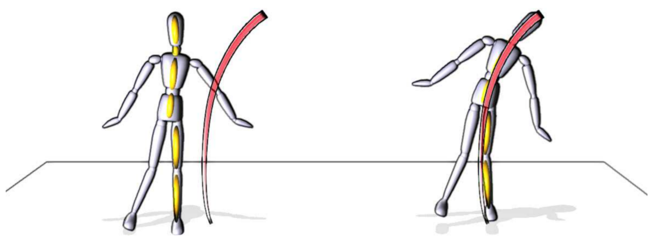


Figure 2.1 – An example of a maximal, connected, and linear selected body line from the head to the left foot.

Engineers in the IMAGINE team at Inria have implemented the solution of the LOA problem within software called Rumba. In this software, the user draws a selection line to select one of the 10 kinematic chains. Then the user draws a second target posing line to move the body line to a new shape and position. Their system works extremely well for a single humanoid character, and even multiple humanoid characters which are strictly separate from each other.

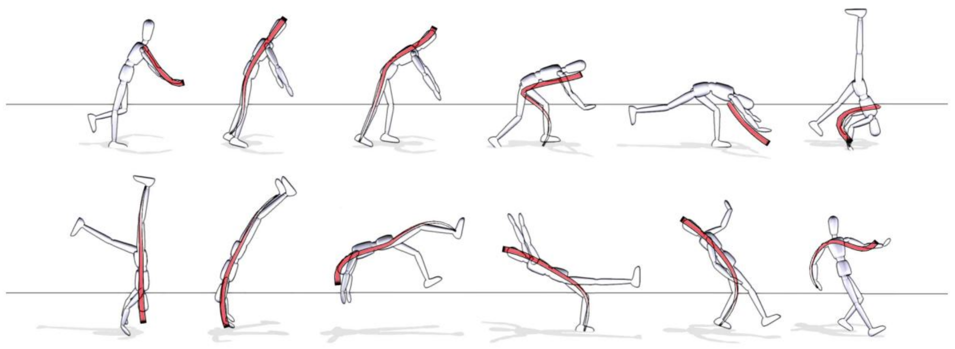


Figure 2.2 – One character's keyframes using the line of action technique from [7].

In [9], Guay et al. continued their work on the line of action to create a technique for animation called space-time sketching, in which a user can draw a line in the path they want a model to take and it will be animated accordingly. As the character follows the path, its model bends and changes shape in a physically realistic way. Their system currently supports creating different movements with the path such as bouncing, rolling, and twisting.

The biggest improvement from their first paper to [9] in regard to this work is the linear time algorithm for matching a 3D skeleton to a drawn LOA. This more numerically stable approach to matching the character to the line makes for faster results than the formerly used minimization problem.

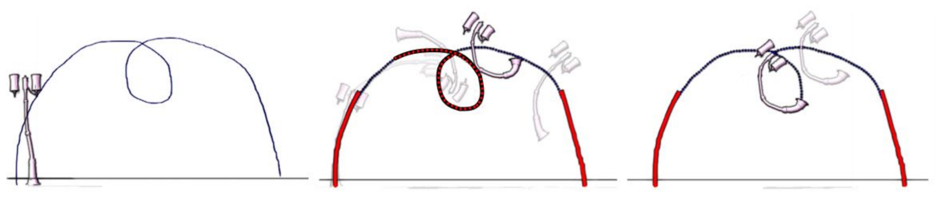


Figure 2.3 – A lamp post model rolling. [9].

Similarly, [14] and [21] respectively describe École Polytechnique Fédérale de Lausanne's and Pixar's use of sketching to pose characters. In [14], slight improvements were made in fitting the body line to the LOA, and their algorithm is still linear. They also provide automatic body line selection based on the joint the user selects plus the direction of their sketching. There is no mention in either paper about handling the interactions of characters.

"SketchiMo: Sketch-based Motion Editing for Articulated Characters" ([3]) is yet another paper which takes advantage of the convenience and ease of a sketch-based interface. This time, Choi et al. take an already saved animation and visualize the motion through time as different curves. The purpose is to allow the user to edit an existing animation, not create one. Additionally, their demo and paper focus on the animation of a single character.

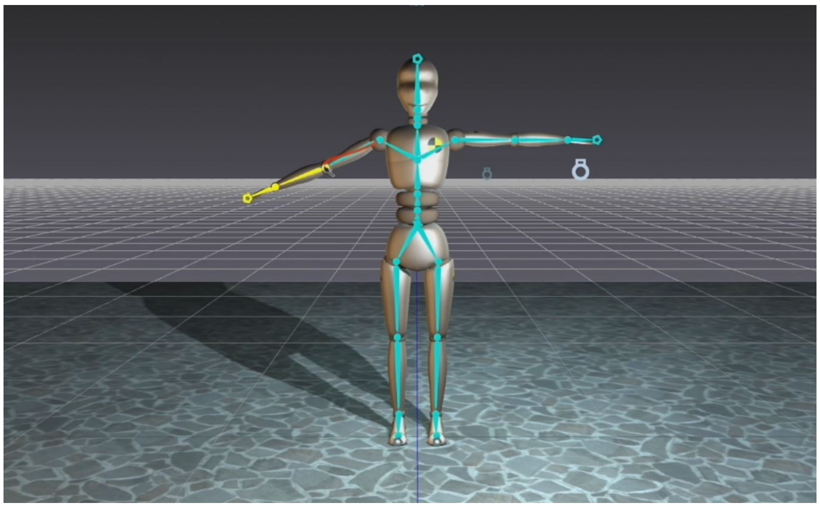


Figure 2.4 – [14]'s sketch-based posing system. As the user draws starting from the shoulder, the rest of the arm is highlighted as the automatically selected body line.

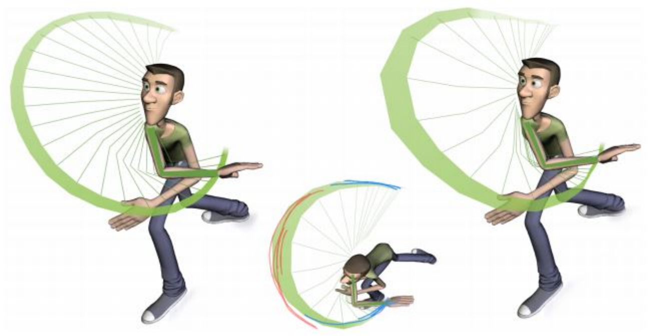


Figure 2.5 – Animation re-timing in [3].

2.1.2 Articulated Characters

Chris Hecker's "My Adventures with Inverse Kinematics" ([10]) compares and contrasts different inverse kinematics algorithms to animate a character climbing a rock wall. Articulated character animation is challenging due to the combination of constraints that limits the viability of a pose. In his work with the cyclic coordinate descent (CCD) algorithm, whose calculations must start at a fixed root, Hecker decides to dynamically change the root of the human's kinematic tree based on what makes the most sense. The idea of root changing can be particularly helpful in the case of combining kinematic trees, though this talk only deals with one character.

Articulated characters are complicated to animate. To this end, generating animation is sometimes easier than doing it by setting a character's pose over and over. Many researchers tackle animation problems using motion capture data instead of manually keyframing. The following articles explore generating animation and reusing animation.

In "Motion Graphs" ([13]) and "Style-Based Inverse Kinematics" ([6]), the authors give two different generative methods used to build animations from motion capture sequences. [13] shows how to construct a graph encoding the different orderings of clips that could possibly make up a realistic animation, while [6] makes use of a generative machine learning model to easily apply a likely and reasonable style to an existing clip.

"Using an intermediate skeleton and inverse kinematics for motion retargeting" ([17]) is an important paper since it touches upon the idea of changing a skeleton, albeit for retargeting motion rather than creating. This paper describes the usage of an intermediate skeleton when retargeting animation from one character hierarchy to another.

2.2 Animation of Multiple Characters

Interactions of multiple characters has never been explored in sketch-based animation, but interactions in general have proved to be another difficult research problem. The following papers show the difficulties of animating by hand, since they all heavily rely on motion capture to provide the animations.

[20] focuses on fighting as an example of close character interactions. This paper also uses motion capture data of single characters to create action-level motion graphs to generate realistic and competitive interactions.

[11] and [23] aim to describe the dynamic spatial relations between characters and also between characters and objects. They both propose an interaction surface or mesh which is multi-structured. [11] uses this surface to adapt motion capture data to accomodate new constraints and different character sizes. [23] uses the interaction bisector surface, seen in Figure 2.6, to retrieve similar motions from a database given a 3D animated clip.

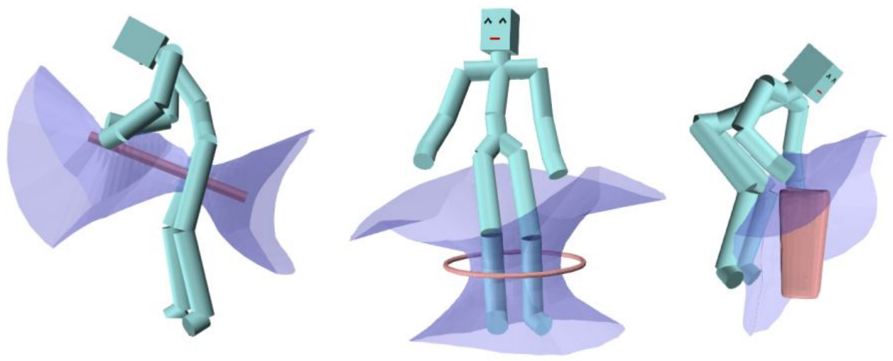


Figure 2.6 – The interaction bisector surface.

2.3 Animation of Dancers

2.3.1 Dance Notation

Valerie Sutton is a choreographer responsible for inventing Sutton Dance Writing, introduced in [22]. Sutton notation remains the closest inspiration for the line of action. A human dancer is represented as a stick figure drawing, putting emphasis on the expressive shape of poses that are prevalent in dance.

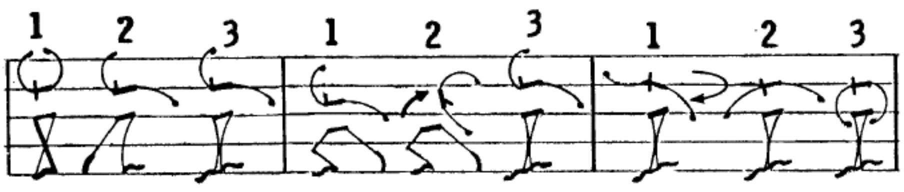


Figure 2.7 – An example of Valerie Sutton's Dancewriting.

Benesh Movement Notation (introduced in [2]) is analogous to Sutton notation, but for motions instead of poses. This tells a dancer how to transition from one poses to another, exactly like interpolation between keyframes. This notation is complex and not intuitive; it requires a certain amount of study before one can learn to write a dance in this form.

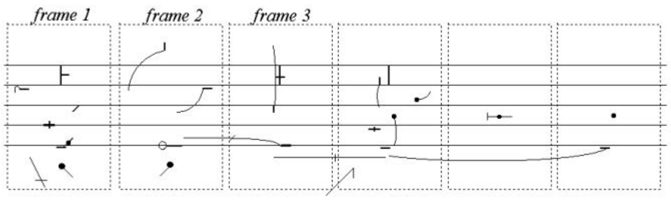


Figure 2.8 – An example of Benesh Movement Notation.

Labanotation is another way of recording human movement. Rudolph Laban invented it with the purpose of annotating dances. Rather than use figures though, Labanotation uses more abstract notions to describe properties of a movement such as its direction, body part, duration, and quality. The Dance Notation Bureau has documented dances using this more precise and technical method, though it is arguably not ideal for artists to quickly write a dance.

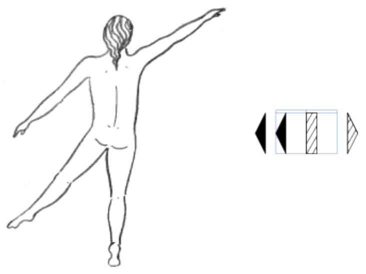


Figure 2.9 – An example of Labanotation from [25].

2.3.2 Dance in Animation

There is a strong interest in portraying dance in animation, which can be seen in many examples of films over the years:

It has progressed from live action to 2D animation to 3D animation. It is safe to say that dance will always be an important part of the human experience and will therefore continue to be represented in art, including film and animation.

(a) A capture from *Beauty and the Beast.*

(b) Buzz LightYear and Jessie dance in *Toy Story*.

(c) A scene from *Tangled.*

Figure 2.10 – Images from [24].

So it seems almost obvious that researchers must also be interested in dance in animation. In Gleicher's 1998 paper, [5], he describes a method of retargetting motion from a motion capture database to 3D articulated characters of different sizes, of which an application is dancing.

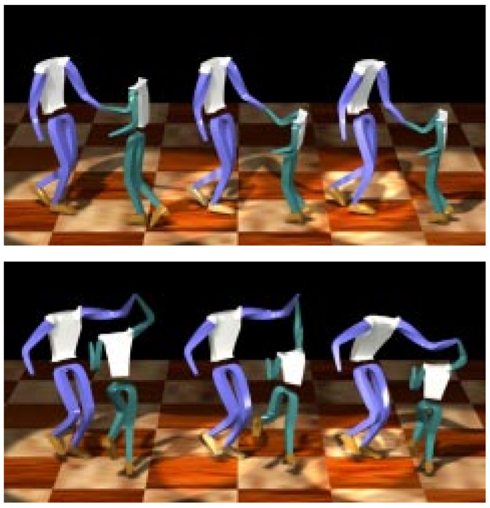


Figure 2.11 – Motion retargetting applied to dance from [5].

Based on the 2005 paper [1], which involves a system for composing and editing dances using Labanotation, Matthews et al. experimented with procedurally generating dance motion in the paper [15] using foot motions and basic patterns. [18] and [19] synthesize music with an animation generated from a motion graph built from motion capture data. It attempts to capture the emotional aspects of human dance, which is a shared goal of the line of action.

— 3 —

Challenges for Posing with Line of Action

The line of action, both on paper and used for 3D animation, is an effective technique for the expressive posing of a character. A few major problems arise when interactions are involved.

3.1 Maintaining Contact

3.1.1 Ground Contact

Maintaining contact is already an issue with one character, exacerbated when more than one character is involved. Contacts for one character are constraints between the character's body parts and itself or other objects in the scene, like the floor, for instance. It would not normally be desirable to have a character's foot go through the floor; usually keeping it level with the floor is a necessary constraint.

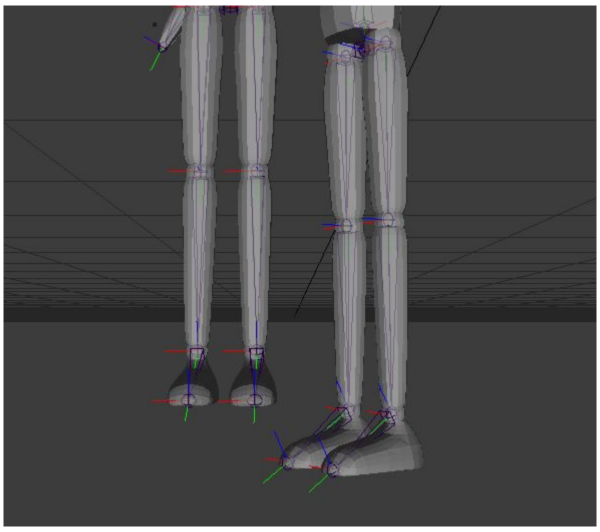


Figure 3.1 – It's hard to tell where the floor is if every LOA moves a character closer to the target line.

3.1.2 Contact Between Characters

Now when other characters are introduced to the scene, there are even more options for potential contacts between body parts of one character and body parts of another. In our case, we only have two characters, but this research could be further generalized to a larger number of characters.

Say we want two dancers to hold each other while spinning. In one frame, it may be simple to get the characters into this specific pose without colliding with each other and touching at just the right locations. When the next keyframe is drawn, the characters may move apart from each other. In a complex dancing scene, there may be many contacts at different times changing from pose to pose, or staying constant from pose to pose. Both situations are challenging for the line of action to handle.

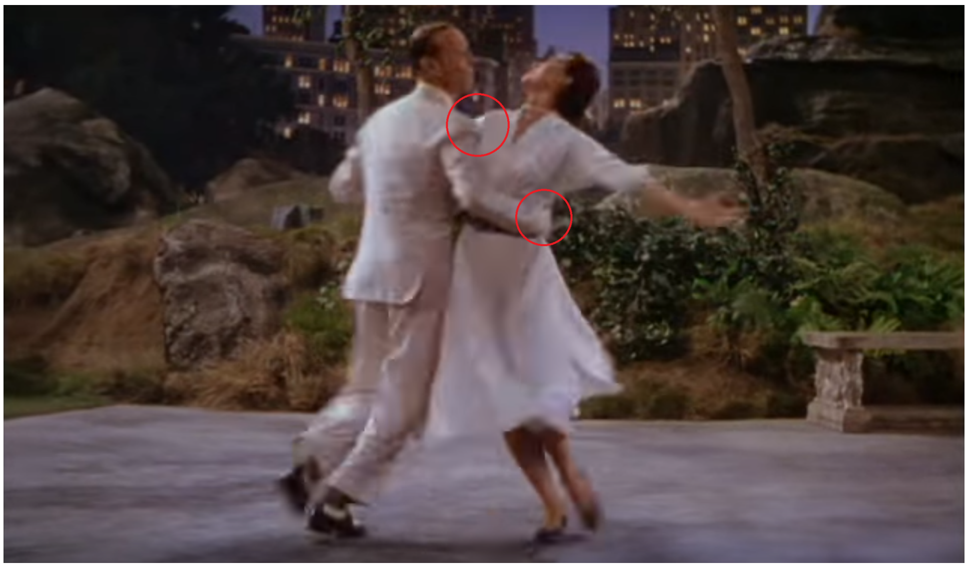


Figure 3.2 – There is clear contact between their shoulders and his hand on her hip. There are also contacts occluded in this view.

3.2 Twisting

Twisting of a 3D character is an inherently 3D problem, while LOAs are 2D lines. When a user draws a selection or posing line, the viewing plane constraint makes sure it is drawn on an invisible plane perpendicular to the camera normal.

The twisting in [9] covers twisting within a character's body, for instance the shoulders turning while the hips stay constant. It also handles twisting around an axis for simpler characters at marked places on the space-time curve. This is not exactly the twisting seen in many dances. Twisting in dance is usually around an axis planted at the position where the character's foot makes contact with the floor.

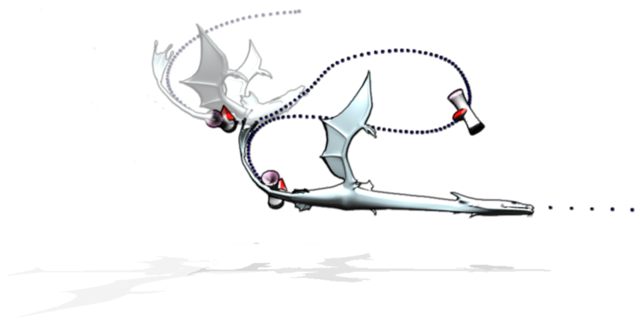


Figure 3.3 – The dragon in [9] spins around the space-time curve at specified coordinates.

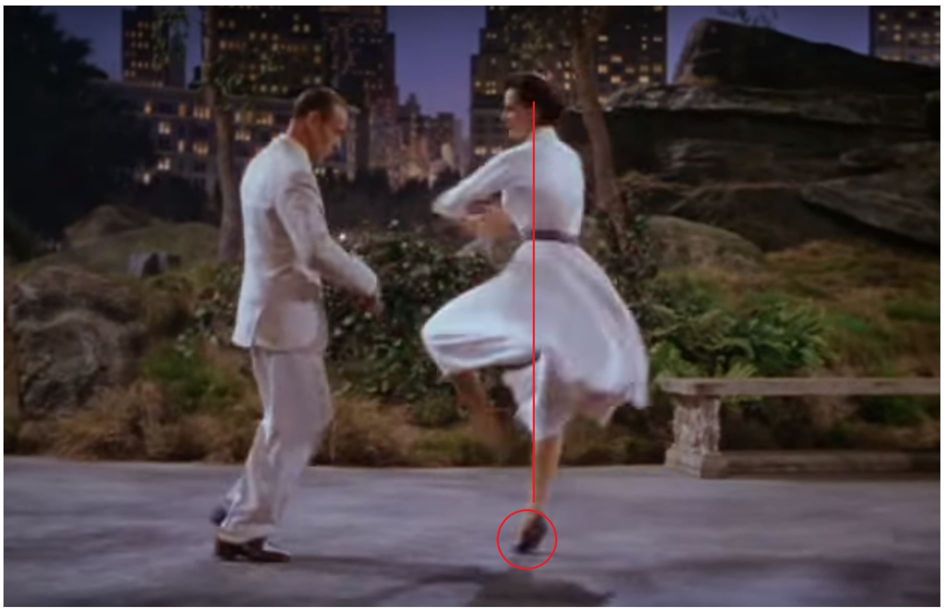


Figure 3.4 – The woman spins around her own axis maintaining contact with the ground.

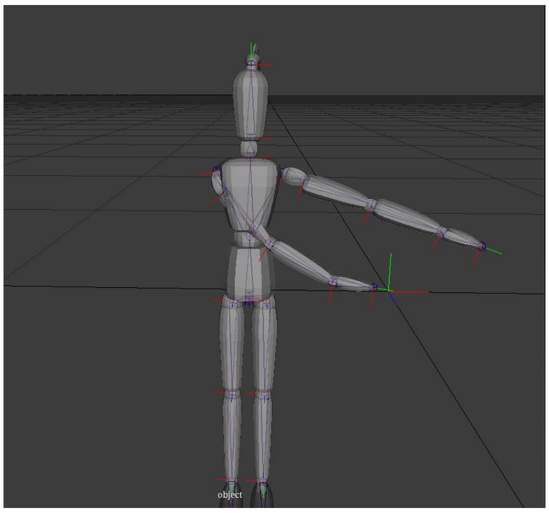
3.3 Collisions

3.3.1 Self-Collisions

Self-collisions are when a part of the character's body collides with another part of its own body. The line of action technique does not do any constraint checking or collision detection, so unrealistic poses are possible.

3.3.2 Collisions Between Characters

As the name implies, collisions between characters involve characters' body parts go through each other. Properly handling contact may solve some cases of collision.


(a) A self-intersection where a character's arm goes through her body.

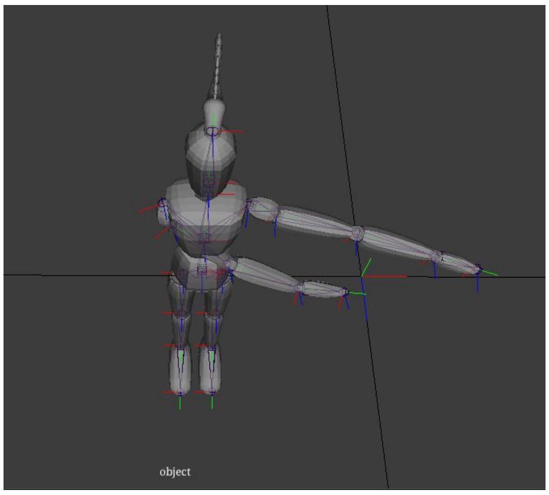
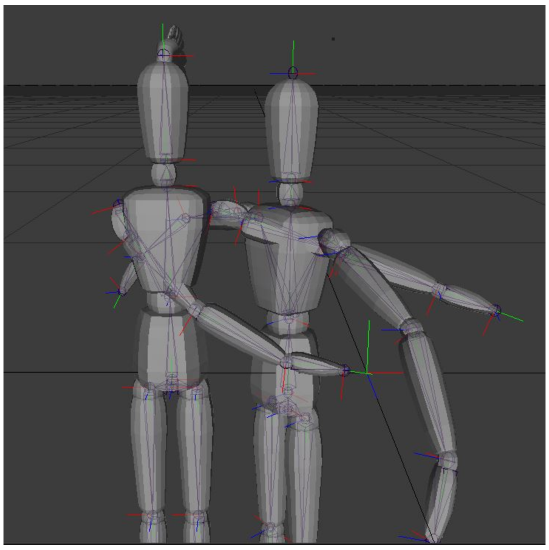

(b) Another view of the self-intersection.

Figure 3.5 – An example of a self-collision.


(a) Two characters collide.

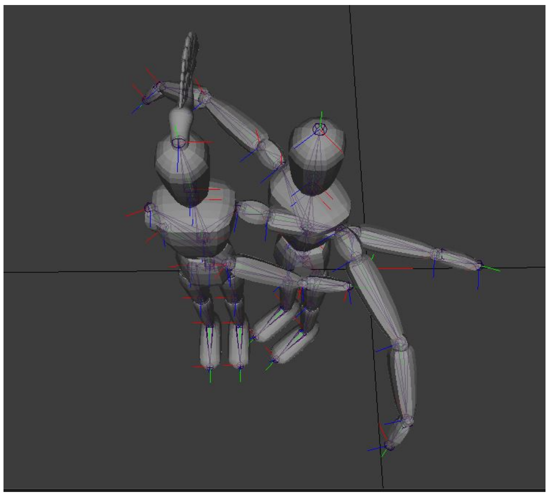

(b) In this intersection, one character's arm goes through another's body.

Figure 3.6 – An example of a collision.

— 4 —

Extending the Line of Action to Multiple Characters

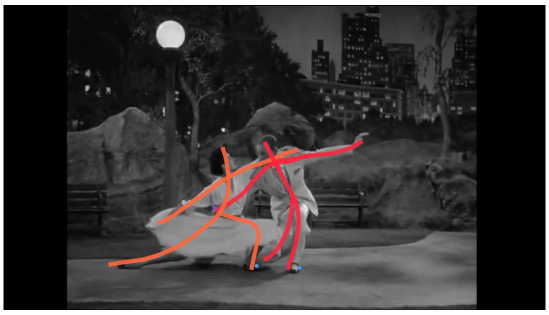
4.1 Analysis of a Typical Video Sequence

The previous line of action method in practice is still more tedious than it could be when working with more than one character. In order to work towards the goal of improving the line of action, it is necessary to observe the common combinations of limbs in a typical dance scene.

There is a dance scene in the film “The Band Wagon,” where Cyd Charisse and Fred Astaire start out by walking together side by side, exchanging twirls until it morphs completely into a swing style dance. This scene is our use case for inventing a notation which extends seamlessly to more than one character.

Almost inverting the problem at hand, we start from a pose to find the appropriate lines of action by tracing over the shapes of the dancers’ bodies using the Grease Pencil Tool in Blender. It was carried out with the purpose of clarifying the properties and limitations of the typical line of action. After seeing at which point the actions started to repeat, a shorter sub-clip was chosen. Annotation was performed in five different ways:

1. tracking and marking each characters’ contact with the ground
2. tracking and marking each characters’ contact with each other
3. tracing lines of action for each character separately every few frames, using the previous line of action notation – the baseline method.
4. tracing lines of action for the characters and treating them as one using as few strokes as possible every few frames.
5. once the main common combinations were established from the above two methods, tracing lines of action only for the selected extreme combination poses.


(a) The typical lines of action trace out the whole skeleton like Sutton notation, and in the worst case is 3 lines per character. This may not seem like such an overwhelming amount, but keeping consistency through a whole clip is cumbersome.

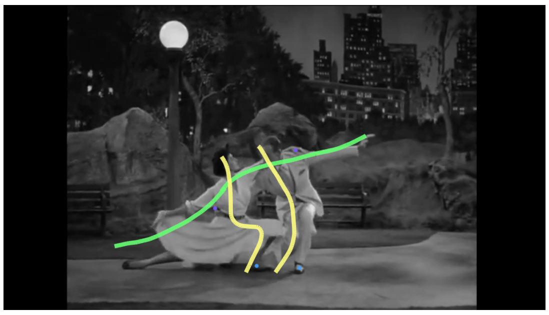

(b) Treating the dancers as one character allows the user to draw fewer lines to indicate the same pose. LOAs indicate position and orientation, so the secondary line in green controls both the arm and leg on the left.

Figure 4.1 – Comparing the old line of action and the new.

Through this process, certain patterns and common poses emerged, enabling the development of an improved notation for illustrating the poses of characters during their interactions with each other. Patterns included symmetry, parallelism, and repetition. Of course, using these patterns and assumptions to our advantage is what drove the development of a solution. It seemed that during many instances in the clip, it would be more efficient to draw lines over the two dancers as if they were one character. This morphed "combined" character should be easier to pose and animate.

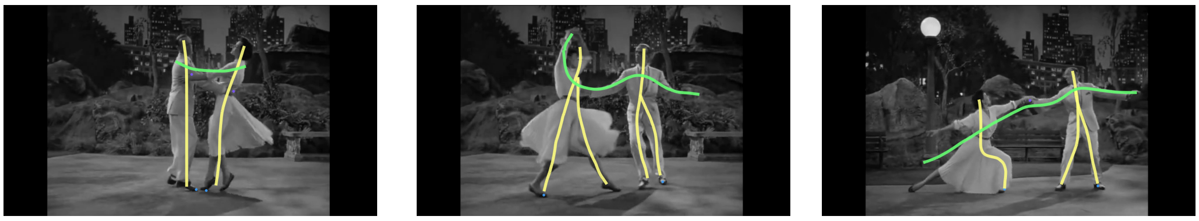


Figure 4.2 – A selection of tracings over the dance clip from "The Band Wagon."

4.1.1 Notation

Taking inspiration from Sutton Notation, we propose a new notation for representing the poses of two characters, both together and separately.

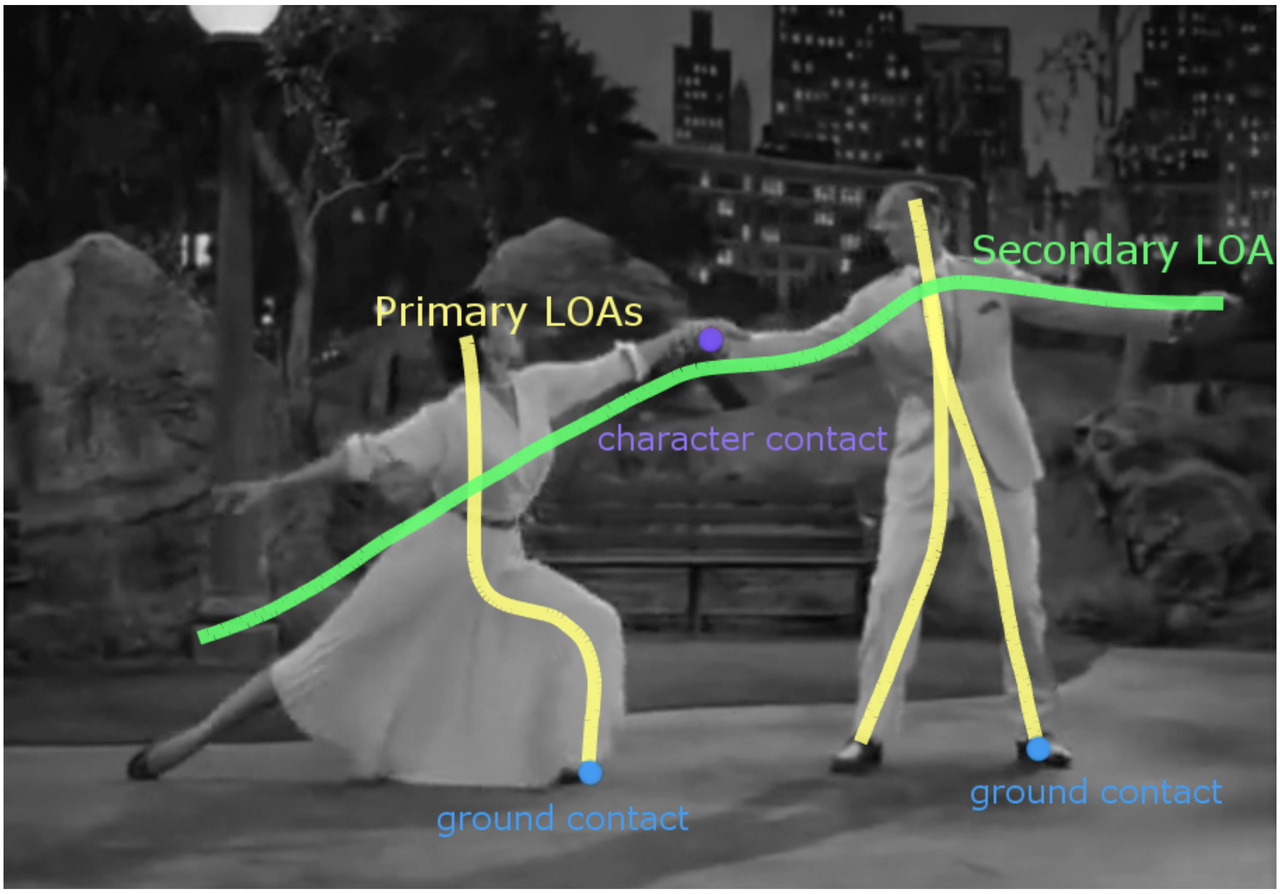


Figure 4.3 – Labeled notation.

To determine which poses for these dancers were common, I annotated the video, keeping track of contact points between the two characters (purple), the contact points with the characters and the ground (blue), the main LOAs (yellow), and the secondary LOAs (green). Once the video was annotated, I categorized the keyframes into how many LOAs it took to create that pose. I chose the most extreme examples of each combination that I could find.

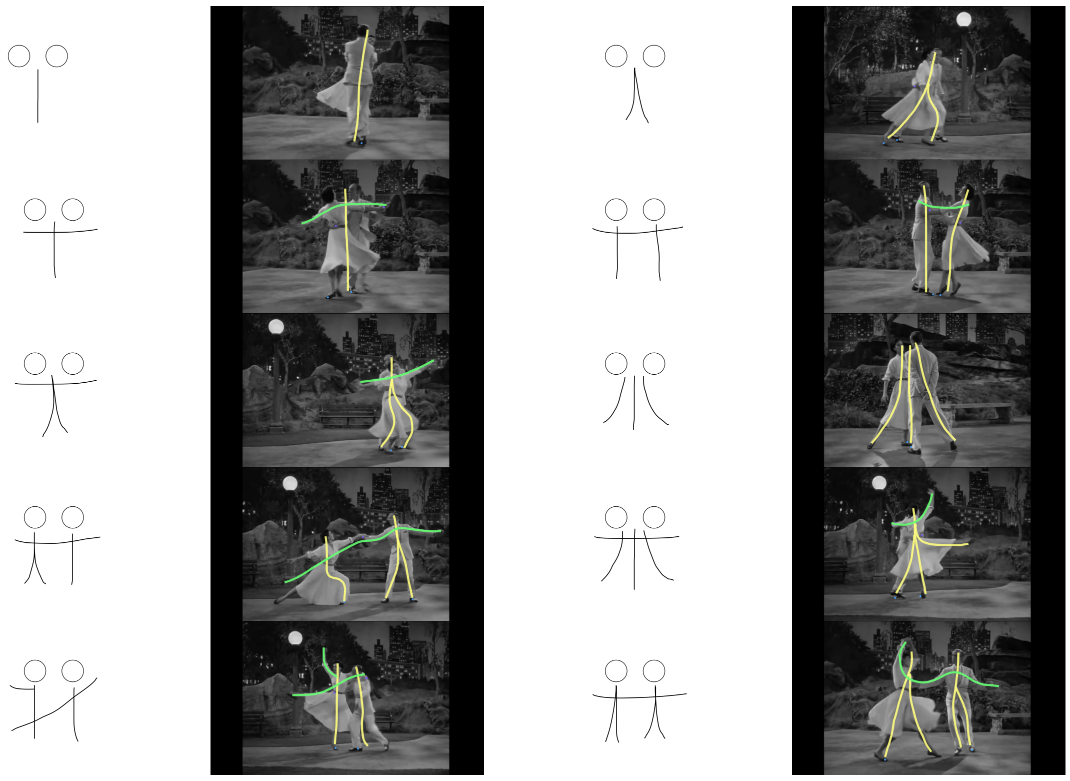


Table 4.1 – Common LOA combinations and their corresponding examples in the video.

4.2 Merging Kinematic Trees

To be able to apply this new notation, the kinematic trees of the two dancing characters need to be dynamically merged and/or separated at various frames of the animation. Rather than changing the whole LOA concept and matching algorithm, we reduce the more complex problem of using LOAs on multiple characters to the original LOA problem and utilize the same iterative method to pose. At each keyframe, the kinematic trees of the two characters are to be merged in a consistent way that preserves the tree structure. Then this new structures is controlled by the new proposed lines of action.

From [9], since there is a viewing plane constraint on which the user draws the posing line, only one angle per joint has to be calculated to get from its current position to the target position. Starting from one end of the selected body line, local joint rotations are found using this equation:

$$\theta_i(t) = \angle(Px_{i+1}(t) - Px_i(t), z_{i+1}(t) - z_i(t)) \tag{4.1}$$

where Px is the joint directed projected onto the viewing plane at time t and z is the drawn joint direction determined by the LOA. The current joint i is the parent of joint $i+1$. Then the angles θ_i are applied to each joint to move it to the new pose. This method is fast and performs well. So instead of changing it, we merge kinematic trees of characters into one combined tree.

Internally, both the tree and graph structures of characters' skeletons are stored. There is also another type of node called a "separator" node, which is purely virtual and acts as a sort of local root to allow for more body line options.

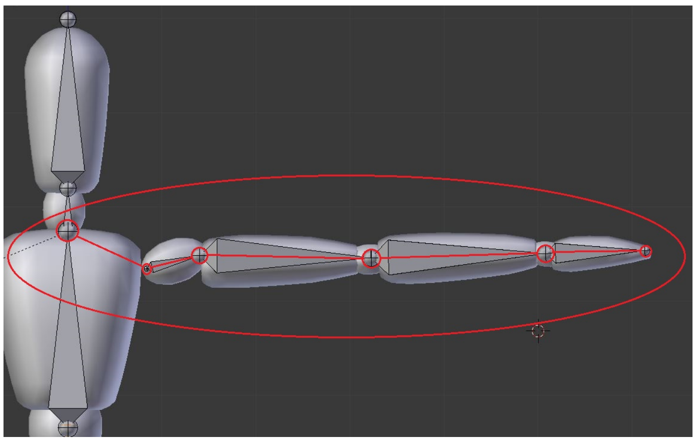
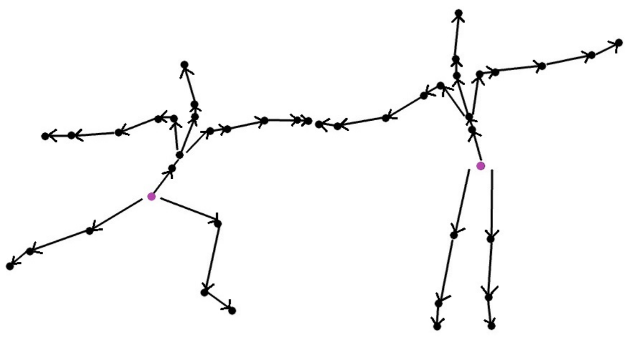
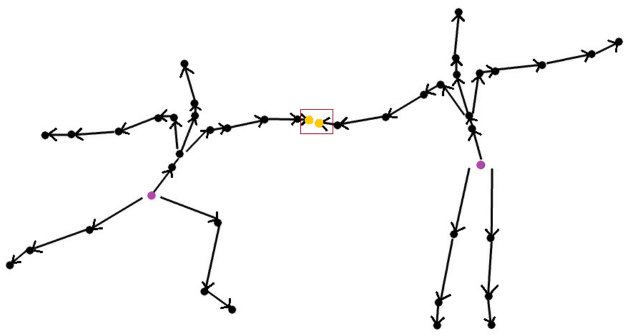
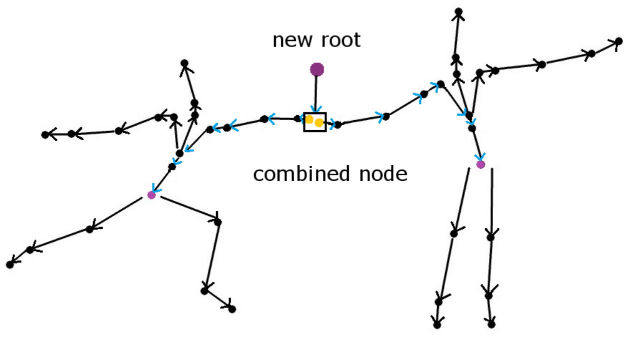


Figure 4.4 – In between the spine and the shoulder joints is a separator node, so the user can select only the arm as a body line.

4.2.1 Example


(a) The two characters' kinematic trees.


(b) A pair of joints are selected to be combined and merge the two hierarchies. These particular nodes do not share a parent. So, a virtual root needs to be created.



(c) The new root is made the parent of the combined node. Then the edges to each original root are reversed to keep the data structure as a tree.

Figure 4.5 – An example of the algorithm.

4.2.2 Algorithm

```
Data: list of character hierarchies h, list of corresponding joints pairs
Result: a single merged hierarchy
if pairs is empty then
    create a root node mega_root;
    foreach hierarchy in h do
        make root of hierarchy a child of mega_root;
        return mega_root;
    end
end
foreach correspondence in pairs do
    foreach joint in correspondence (2) do
        go up hierarchy to find closest root or separator;
        if roots are same then
            connect joints into a mega_node;
            union the two nodes children;
        end
        else
            create a root node mega_root;
            make each joint a child of mega_root;
            reverse edges from each joint to their respective original nodes;
        end
    end
end
enumerate cycles in using Johnson's algorithm ([12]);
make each cycle into a new mega_node;
return mega_root;
```

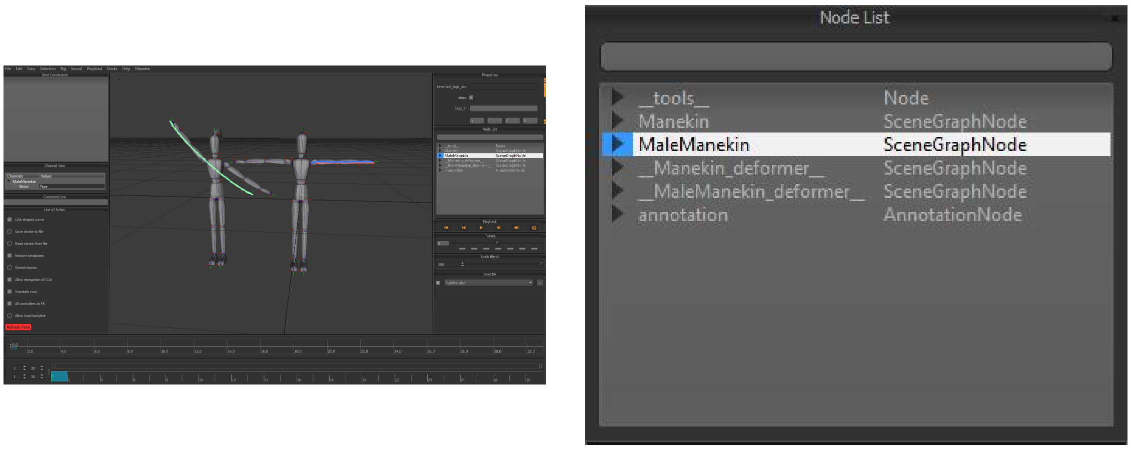
Algorithm 1: The *merge_hierarchies* function.

A combined node is made by averaging frames (translation and rotation) of other nodes, effectively becoming a rigid body. A combined node made from a cycle averages all the frames of the nodes in the cycle with higher weight on the combined nodes within the cycle. A combined node made by connecting just two nodes averages the two frames and adds a virtual root. By combining nodes this way, we convert the two characters' trees into one, which is the goal.

A kinematic tree must be just that, a tree. It cannot have cycles and each node can only have one parent, so Johnson's algorithm is executed to find all simple cycles in the graph. From this all the nodes in the cycle are merged into its own combined node.

4.3 Interface

To select a body line for one character, the user clicks on which model they want either in the viewport or in the node list on the right hand side panel and activates annotation mode. Then they can press and hold shift and draw over the character, tracing the limb they want to select. A path is found between the two closest controllers to the endpoints of the LOA. If a path is found, the blue line shows what was finally selected. The red line is what the user drew and the green line shows a posing line.

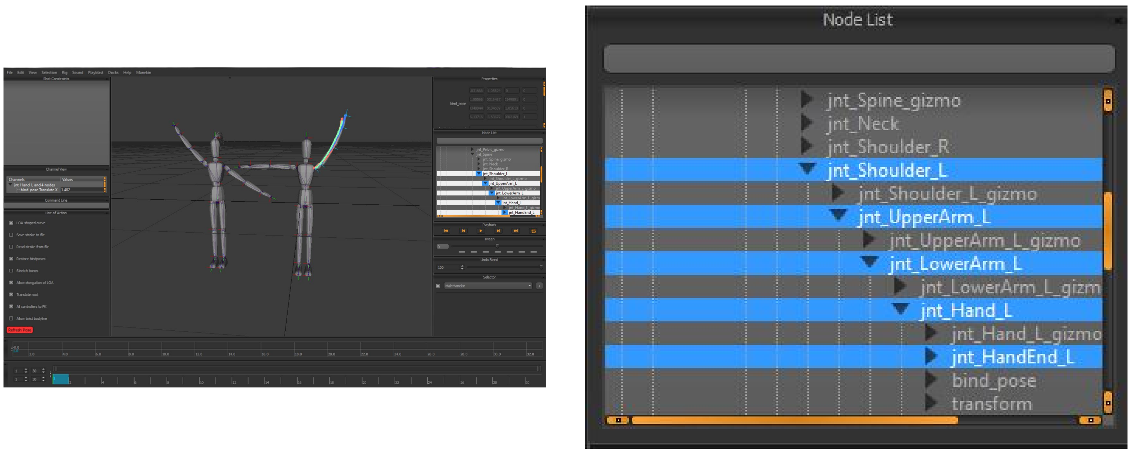


Node List

__tools__	Node
Manekin	SceneGraphNode
MaleManekin	SceneGraphNode
__Manekin_deformer__	SceneGraphNode
__MaleManekin_deformer__	SceneGraphNode
annotation	AnnotationNode

Figure 4.6 – Selecting a body line.

To pose a character after a proper body line is selected, the user holds the control key while drawing a line with the mouse and Equation 4.1 is used to match the body line to the LOA after the user releases the mouse. The green line shows the LOA. Note that now in the node list, all joints contained in the body line are selected, which allows the user to set keyframes using the LOA.



Node List

jnt_Spine_gizmo
jnt_Neck
jnt_Shoulder_R
jnt_Shoulder_L
jnt_Shoulder_L_gizmo
jnt_UpperArm_L
jnt_UpperArm_L_gizmo
jnt_LowerArm_L
jnt_LowerArm_L_gizm
jnt_Hand_L
jnt_Hand_L_gizmo
jnt_HandEnd_L
bind_pose
transform

Figure 4.7 – Posing a body line.

While in annotation mode, the user can then enter a sub-mode called joint selection mode. They can then click on pairs of joints and if valid, they will be highlighted in yellow. The user can deselect a joint before it goes into a pair, but when two joints are selected they are automatically put into a pair.

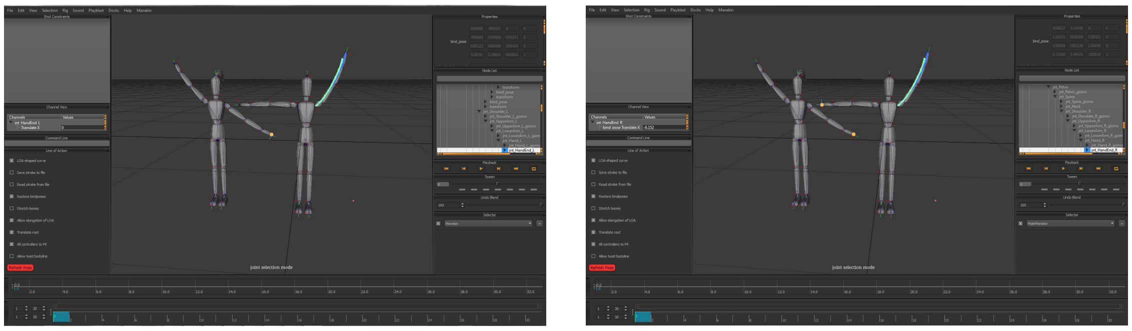


Figure 4.8 – Selecting a pair of joints to connect.

— 5 —

Experimental Validation of Solution

5.1 Carefully Selecting Sample Keyframes

Based on the chart Table 4.1 in chapter 4, the general LOA combinations, concrete keyframes from the video were chosen as examples. A few keyframes in particular stood out as ones with reasonable transitions between them. It makes sense that these poses would be close in time. Three of the keyframes together make for a very short but interesting and complex clip, which is used as the baseline with which to compare the proposed method.

5.2 Establishing a Baseline for Comparison

5.2.1 User Study

```
------------------------------
         USER STUDY:
------------------------------
Study performed on: Tue Jun 13 2017 at 21:40
Total time running: 0:04:42
Number of user clicks in the viewport: 18
Number of selection lines drawn click+shift: 2
Number of posing lines drawn click+ctrl: 3

------------------------------
end of user study
------------------------------
```

The metrics recorded during the posing of characters in Rumba are: how long the program has been running since its initial launch, how many clicks the user made in the viewport for any reason, the number of posing lines drawn, and the number of selection lines drawn.

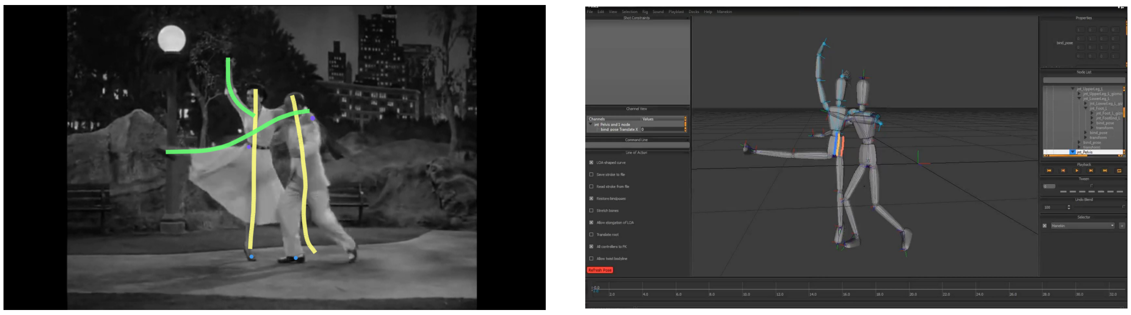


Figure 5.1 – The first baseline pose.

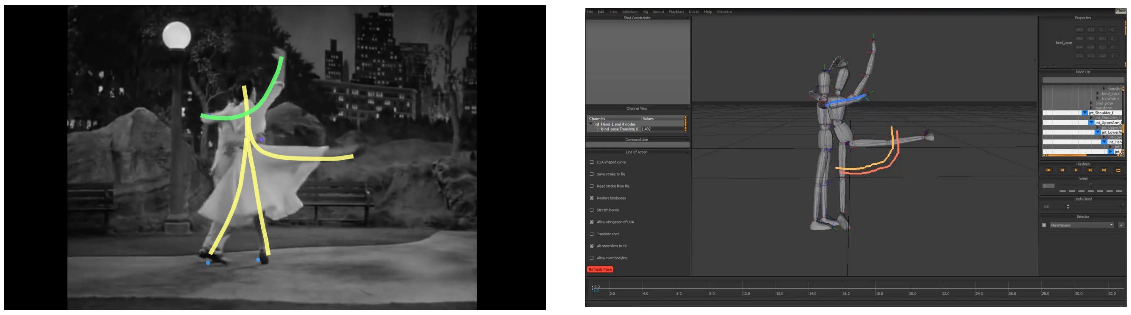


Figure 5.2 – The second baseline pose.

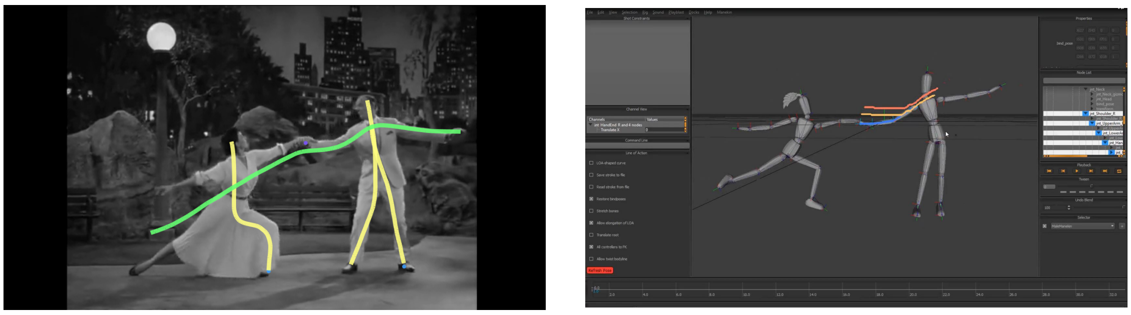


Figure 5.3 – The third baseline pose.

Baseline User Study Results for Three Key Poses				
	Total time	Number of clicks in viewport	Number of selection lines	Number of posing lines
Pose 1	11 minutes 24 seconds	145	22	45
Pose 2	8 minutes 49 seconds	118	22	27
Pose 3	11 minutes 59 seconds	152	30	39

Table 5.1 – User studies for the baseline poses using the standard line of action method.

5.2.2 Experiment

The updated user study metrics include all the metrics used in the baseline, plus the number of joints selected and the number of merges made. The same three poses made with the proposed technique are compared in accuracy to the reference photo (visually), and in efficiency and effort based on the metrics.

```
------------------------------
         USER STUDY:
------------------------------
Study performed on: Tue Jun 13 2017 at 21:40
Total time running: 0:04:42
Number of user clicks in the viewport: 18
Number of selection lines drawn click+shift: 2
Number of posing lines drawn click+ctrl: 3
Number of joints selected: 2
Number of merges made: 1

------------------------------
end of user study
------------------------------
```

— 6 —
Limitations and Discussion

6.0.1 Future Work

Due to time contraints on the project, not all the cases in Table 4.1 were tested. Only the three selected baseline poses shown in Figure 5.1, Figure 5.2, and Figure 5.3 were ever considered in practice. The algorithm described in the report to merge hierarchies assumes that the characters being merged are already in the desired position for connection. This means that if the user wanted the characters to be holding hands – a contact to be maintained – the hands should be already touching when the user selects and merges joints. Ideally, the user should be able to specify whether or not they want a merge to create a contact. Currently, if they do want a contact, the hands in this example should move to be touching. If not, the combined node's averaged frame will be somewhere in between the two hands, whether they are close in position or not. This could possibly be solved with a better joint selection method.

6.0.2 Joint Selection

The joint selection interface idea was originally to have the user encircle the joints they want to be merged. This comes with problems and advantages, the benefit being that it may be more intuitive to the user. However, a method for detecting all joints in the 2D circle would be challenging. Along with that difficulty, there would be some ambiguity as to which joints the user actually wants in their circle. Some joints will be behind others in the 3D space and who is to say whether those unseen joints should or should not be included in the selection?

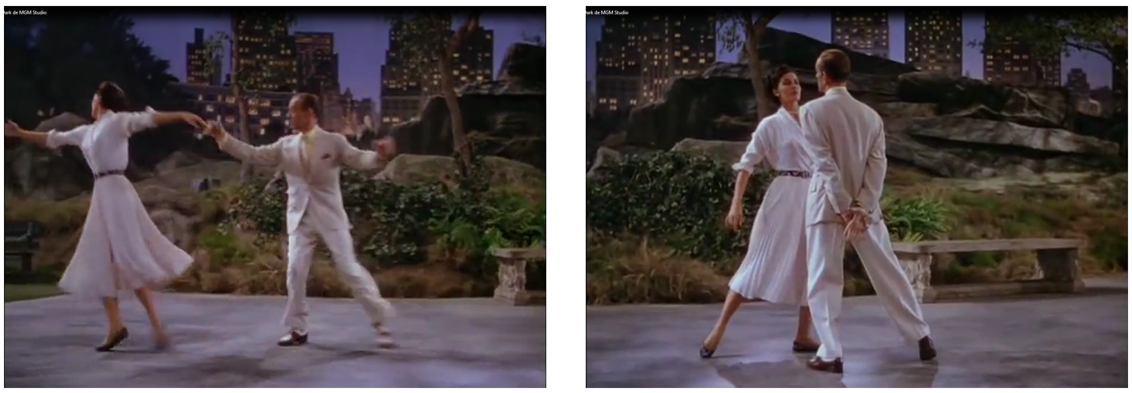
To that effect, currently only pairs of joints can be selected and merged. It could be interesting to see what changes, if any, would be needed for the merging algorithm if the user were allowed to select an arbitrary number of joints. There is more potential for creating cycles that way, which would make very large combined nodes. On the one hand, less computation would be required in the body line selection and posing because there are fewer joints to choose from in the merged hierarchy. But on the other hand, the resulting rigid body made of nodes could gradually increase to be bigger than the user intended and makes for a less precise pose.

6.0.3 User Constraints

Users are only allowed to attach and detach hierarchies at keyframes to avoid interpolation issues. If a contact made by the merged hierarchy is present in one keyframe and absent in the next, the splitting of the hierarchy would have to be done during the interpolation at some time in between the two keyframes. So for now the user can only make sets of keyframes where the actual detaching is done exactly at a keyframe.

6.0.4 Taking Advantage of Patterns

Symmetry and parallelism are both prevalent in the use case video. It would be nice to use this to make posing even faster. For example, while posing one character, the system could pose the other in the same or opposite way. This would involve mirroring the drawn LOA to match to the second characters, limb. This raises questions about which kinematic chain on the second character to select.



(a) Parallelism. (b) Symmetry.

Figure 6.1 – Patterns in the poses.

6.0.5 Twisting

In future work, the twisting problem mentioned in chapter 3 would be very nice to explore, since the amount of spins the dancing characters do is quite high. The characters spin separately and together, so the technique would need to work on both separate and combined kinematic trees.

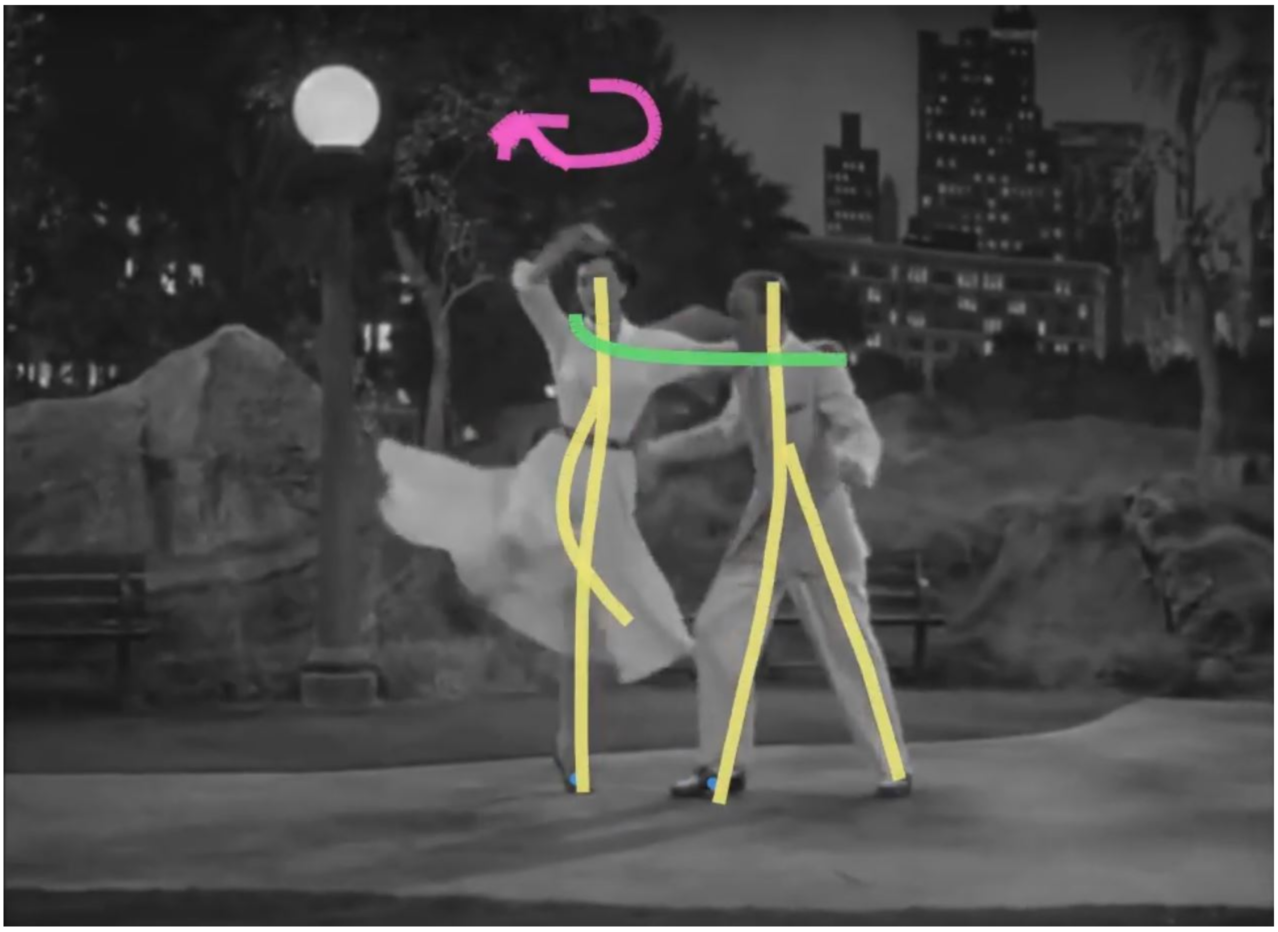


Figure 6.2 – One idea for a twisting notation.

— 7 —
Conclusion

The goal of the thesis was to extend recent research at IMAGINE, restricted to the animation of a single character, to the challenging case of several interacting characters. While the capability to animate a single character by drawing a curve is present, there is no established sketch-based way to animate multiple characters either interacting or moving separately. The sketch-based animation of dancing couples captures the challenge of animating two characters both separately and together.

First, I observed and analyzed the current methods of creating character animations with and without interactions, from sketch-based animation to generating animation through motion graphs made from existing data. I looked at the way artists, cartoonists, and choreographers draw lines to indicate motion, specifically that of dancers.

I have identified what I think those tools lack for the task of posing dancing characters. Based on my findings, I created an algorithm for being able to describe and reproduce the coordinated movements of two dancers during a dance step. I evaluated the current line of action tool with metrics that could later be compared to the combined kinematic tree approach.

Dance is simply an introductory example for the design and evaluation of the proposed techniques, but the methods have the possibility to be applicable more generally.

— A —
Appendix

A.1 User Study Files

Pose 1

```
------------------------------
        USER STUDY:
------------------------------
Total time running: 0:11:24
Number of user clicks in the viewport: 145
Number of selection lines drawn click+shift: 22
Number of posing lines drawn click+ctrl: 45

------------------------------
end of user study
------------------------------
```

Pose 2

```
------------------------------
        USER STUDY:
------------------------------
Total time running: 0:08:49
Number of user clicks in the viewport: 118
Number of selection lines drawn click+shift: 22
Number of posing lines drawn click+ctrl: 27

------------------------------
end of user study
------------------------------
```

Pose 3

```
------------------------------
        USER STUDY:
------------------------------
Total time running: 0:11:59
Number of user clicks in the viewport: 152
Number of selection lines drawn click+shift: 30
Number of posing lines drawn click+ctrl: 39

------------------------------
end of user study
------------------------------
```

A.2 Links to Accompanying Videos

- The Band Wagon Dancing in the Dark original scene - `https://www.youtube.com/watch?v=duLFwcsc6Nc`
- The 10 main keyframes are annotated, along with inter-character contacts annotated by purple dots and ground contact annotated by blue dots - `https://www.youtube.com/watch?v=Xz3sYEbq_eo`
- Densely annotated sub-clip using the *previous* LOA method - `https://www.youtube.com/watch?v=02IyEHd3IfI`
- Densely annotated sub-clip using the *proposed* LOA method - `https://www.youtube.com/watch?v=q7cbzwNr7vY`
- A side by side comparison of the two methods, with twists indicated in pink - `https://www.youtube.com/watch?v=TejQGyKDqwc`
- Sped up 8x screen cast of the making of Pose 1 in Rumba - `https://www.youtube.com/watch?v=MCniYCZt9bg`
- Sped up 8x screen cast of the making of Pose 2 in Rumba - `https://www.youtube.com/watch?v=058-576J4lM`
- Sped up 8x screen cast of the making of Pose 3 in Rumba - `https://www.youtube.com/watch?v=EDtk6OqCbDE`
- The baseline animation clip for future work - `https://www.youtube.com/watch?v=vdu0ETFJ95g`

Bibliography

[1] Tom Calvert, W Wilke, Rhonda Ryman, and Ilene Fox. Applications of computers to dance. *IEEE computer Graphics and Applications*, 25(2):6–12, 2005.

[2] Marguerite Causley. *An Introduction to Benesh: Movement Notation*. Ayer Company Pub, 1980.

[3] Byungkuk Choi, JP Lewis, Yeongho Seol, Seokpyo Hong, Haegwang Eom, Sunjin Jung, Junyong Noh, et al. Sketchimo: sketch-based motion editing for articulated characters. *ACM Transactions on Graphics (TOG)*, 35(4):146, 2016.

[4] Blender Foundation. Blender. `https://www.blender.org/`, 2006–2017.

[5] Michael Gleicher. Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 33–42. ACM, 1998.

[6] Keith Grochow, Steven L Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. In *ACM transactions on graphics (TOG)*, volume 23, pages 522–531. ACM, 2004.

[7] Martin Guay, Marie-Paule Cani, and Rémi Ronfard. The line of action: an intuitive interface for expressive character posing. *ACM Transactions on Graphics (TOG)*, 32(6):205, 2013.

[8] Martin Guay, Rémi Ronfard, Michael Gleicher, and Marie-Paule Cani. Adding dynamics to sketch-based character animations. In *Proceedings of the workshop on Sketch-Based Interfaces and Modeling*, pages 27–34. Eurographics Association, 2015.

[9] Martin Guay, Rémi Ronfard, Michael Gleicher, and Marie-Paule Cani. Space-time sketching of character animation. *ACM Transactions on Graphics (TOG)*, 34(4):118, 2015.

[10] Chris Hecker. My adventures with inverse kinematics. In *Game Developer's Conference Proceedings, Retrieved June*, volume 22, page 2008, 2002.

[11] Edmond SL Ho, He Wang, and Taku Komura. A multi-resolution approach for adapting close character interaction. In *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*, pages 97–106. ACM, 2014.

[12] Donald B Johnson. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing*, 4(1):77–84, 1975.

[13] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *ACM transactions on graphics (TOG)*, volume 21, pages 473–482. ACM, 2002.

[14] Mentar Mahmudi, Pawan Harish, Benoît Le Callennec, and Ronan Boulic. Artist-oriented 3d character posing from 2d strokes. *Computers & Graphics*, 57:81–91, 2016.

[15] Elizabeth A Matthews and Geoffrey B Matthews. Procedural generation of cuban dance motion. In *Computer Games (CGAMES), 2011 16th International Conference on*, pages 293–297. IEEE, 2011.

[16] Vincente Minnelli. *The Band Wagon*. Metro-Goldwyn-Mayer (MGM), 1953.

[17] Jean-Sébastien Monzani, Paolo Baerlocher, Ronan Boulic, and Daniel Thalmann. Using an intermediate skeleton and inverse kinematics for motion retargeting. In *Computer Graphics Forum*, volume 19, pages 11–19. Wiley Online Library, 2000.

[18] Takaaki Shiratori, Atsushi Nakazawa, and Katsushi Ikeuchi. Dancing-to-music character animation. In *Computer Graphics Forum*, volume 25, pages 449–458. Wiley Online Library, 2006.

[19] Takaaki Shiratori, Atsushi Nakazawa, and Katsushi Ikeuchi. Synthesizing dance performance using musical and motion features. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 3654–3659. IEEE, 2006.

[20] Hubert PH Shum, Taku Komura, and Shuntaro Yamazaki. Simulating competitive interactions using singly captured motions. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, pages 65–72. ACM, 2007.

[21] Ryan Stelzleni, Bret Parker, Tom Hahn, Sarah Shen, Dan McGarry, and Chen Shen. Sketch to pose in pixar's presto animation system. In *ACM SIGGRAPH 2015 Talks*, page 26. ACM, 2015.

[22] Valerie Sutton. *Sutton Movement Shorthand: Writing Tool for the Dance*. Movement Shorthand Society Press, 1979.

[23] Xi Zhao, Myung Geol Choi, and Taku Komura. Character-object interaction retrieval using the interaction bisector surface. In *Computer Graphics Forum*, volume 36, pages 119–129. Wiley Online Library, 2017.

Internet Sources

[24] Liz Cerezo. 15 unforgettable disney dance scenes!, 2013. Online.

[25] Merle erl Idris Ghyssaert. about labanotation: determination of position. Online.

[26] Matt Jones. Line of action, 2011. Online.

[27] Juan M. Line of action!, 2012. Online.