Final Project Report

Computer Animation Techniques

Space time sketching of character animation

Submission Date: 2017.01.12

Group Members:

Name	Student ID
Name 1 – Raziur Rahman Totha	Student ID- 116030990096
Name 2 – Jiannan Ye	Student ID- 5132409029

Table of Contents

Group Members:

Name	Student ID	Tasks Assigned **
Name 1 – Raziur Rahman Totha	Student ID-116030990096	• Understanding the given Research Paper • Report Documentation • Presentation
Name 2 - Jiannan Ye	Student ID-5132409029	• Coding of the proposed idea • Implementation • Presentation

1. Introduction

We present a new animation editing tool “StrokeTool” for Computer Graphics. The goal of this tool is to mitigate the issues of quality animations that deals with high level artistic principles using keyframe animation.

The aim of this project was to develop a method to compute a dynamic line of action (DLOA) from a single stroke space time curve(STC) and to let the animators have both spatial and temporal control simultaneously making co-ordination of shape over time much easier which will allow the animators to draft a full coordinated motion using motion abstraction and to map the sketches indicating movement to the actual animation of the character.

This report also presents our technical approach which is based on the paper “Space time sketching of character animation” by Martin Guay, Remi Ronfard, Micheal Gleicher and Marie-Paule-Cani which was published in ACM SIGGRAPH 2015.

It will enable all the users to quickly obtain results including beginner to skilled artists to match their desires and expectation encouraging them to spend more time polishing their animation. It will speed up the production and allow instantaneous results and visualization as the animation is refining it by composing it of different layers of motion. Now using our tools 2D sketch artists can actually create a full 3D motion.

1.1. Related Work

In previous works, partial solutions are proposed to control different aspects of the animation, i.e. the shape (poses) and motion (timing and trajectories), separately.

Keyframing, being the more popular one which was Inspired from the hand-drawn animation tradition where computer animation is often decomposed into sequences of poses in time. Specifying each pose individually and controlling the timing through a timeline remains to this day the established pipeline.

Instead of sequencing poses, researchers have used the idea of layering partial motions. The user selects body parts or degrees of freedom and performs while the motion is recorded. Although performing can be very intuitive, coordinating different layers of performance can be hard. As a result, the animator cannot directly control the pose (shape) at a specific time.

Several works propose specifying as input high level motion abstractions like doodles and trajectories, along with preexisting motion-often humanoid-to “fill” the missing information a small number of point trajectories are sketched to drive characters by using strong priors based on previously captured motions.

Nonlinear deformers such as sinae and waves motions can be applied to a character’s shape to produce nice-looking animation effects in several authoring softwares such as Blender and Maya. But such effects typically give very little control to the user. More sophisticated methods for physically-based simulation have been proposed to synthesize realistic character animation but they suffer from the same problem. One work attempted to combine physical simulation with gesture-based control but cannot prevent nonlinear chaotic behaviors.

Several authors have proposed to create 3D animation by sketching 2D paths embedded in a 3D world. These are typically drawn onto the floor to control the global rigid 2D configuration of the character over time.

1.2. Proposed Work in Assigned Research Paper

The main concept in the proposed method of the assigned research paper is to enable space-time sketching and allow the animators to draft a full coordinated motion using a single stroke called the space-time curve (STC). From this curve and the speed at which it was drawn, we automatically initialize a dynamic line of action (DLOA) that drives the motion of a 3D character through projective constraints. The dynamic models for the line's motion are entirely geometric, require no pre-existing data, and allow full artistic control., which itself drives a specific body line in the character model (e.g. spine, tail, wings, etc - or combination of these). We enable refinement of the resulting coarse motion through over-sketching the DLOA at specific times or adding secondary lines or controls. Both the space-time curve and the DLOA are planar, defining projective constraints for the 3D animated character. This allows simpler user input and enables us to decouple expressive motion specification, done from a target camera viewpoint, from a specific 3D character model. The same DLOA can be re-applied to other body parts or to different characters, enabling the user to draft animations even before the character is fully modeled.

1.3. Limitations of Proposed Work

The biggest drawback would be that it is limited to only projective control. As sketching to freely create and refine 3D animations is a difficult problem. In this paper, sketching has been used to specify 3D animations from a given viewpoint. The notion of a moving line of action has been used for abstracting a part of the character's body allowing the same moving line to be applied to different body parts or characters.

The proposed new space-time abstraction for sketching motion (the STC) enables an animator to initialize a full coordinated motion with a single stroke-providing immediate display and allowing for rapid refinement. This method brings several limitations.

In the paper the main focus was on the motion of a single line and illustrated our concept with simple character morphologies (lamp, cactus and dragon). It would require more work to drive the motion of a multi-legged figure (humanoid, or dog) from a single moving line making the whole process quite difficult to implement as we are trying to simultaneously control both spatial and temporal properties of the model. Three specific ways of creating a DLOA from a space time curve was mentioned: path-following and two richer mechanisms for extracting coordinated motion and trajectory from intuitive strokes. The resulting motion can then be refined, which increases the range of possible results. While which behavior to adopt between the bouncing, rolling and path-following based on the geometric features found in the strokes, can be detected, automatically differentiating with wave curves and other unit keyframes is not possible.

Another focus was on planar movement plus out-of-plane secondary motions and twist, since many motions meant to be seen by viewers are in fact planar. While the proposed DLOA models can be directly extended to 3D curves, the actual editing of 3D curves remains a challenge.

1.3.1 Critical Review:

The paper introduced the new concept of space-time sketching for freeform animation of 3D characters but coordinated animation—possibly including squash and stretch—can be drafted with a single stroke. Other strokes and controls can then be added, enabling the user to progressively sculpt and refine motion. Which will make it easier for beginners to start and create animations. The animation is designed by sketching in 2D from viewpoints of interest: it defines a dynamic line of action used as a projective constraint that robustly drives the 3D character. The resulting independence between animation control and the 3D character model enables the user to simultaneously edit the 3D model during the design stage which will save a lot of time for even the skilled artists. But due to it's approach to control both spatial and temporal properties at the same time, the core idea remains difficult to implement limiting the model to only projective control. And it definitely proves that using keyframes alone is not the best way to do animation more productively. Further research on improving the motion of the path of the object and enabling it to freely follow and deform in some arbitrary manner and the design of bipedal and quadrupedal motion—driven by a single moving line abstraction may be a great addition to the whole animation world.

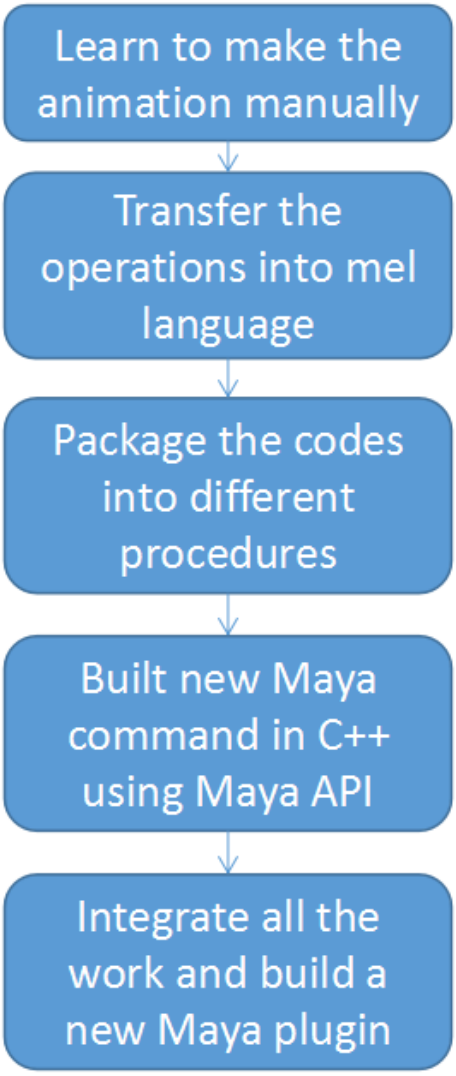
2. Proposed Research Work

This tool is built for artists including who is not experienced in key-frame animations to highly experienced CG animators. This tool only requires a minimum amount of knowledge of Maya. The user just needs to know how to draw a stroke and specify a model that will move according to that stroke. Generally, it can be very useful for concept artists and 3D animators to speed-up the animation pipeline from a 2D Sketch of a model to a full 3D animation. It can also be used by designers to visualize the range of movement a character can perform.

We just tried to implement the idea of the paper as much as we could. The approach to the method mentioned in the paper is already quite good enough. As the paper did not provide any API, tool or plug-in to realize the concept explicitly, so we built the Maya plug-in so that the users can be benefitted from it.

Maya already provides the ability to make animations using it's built-in commands and plug-ins. All we did is to combine some built-in functions with some of the new functions we made. And we made some essential methods that were constructed very well which can be used later on by any other developers to implement the idea easily.

3. Methodology

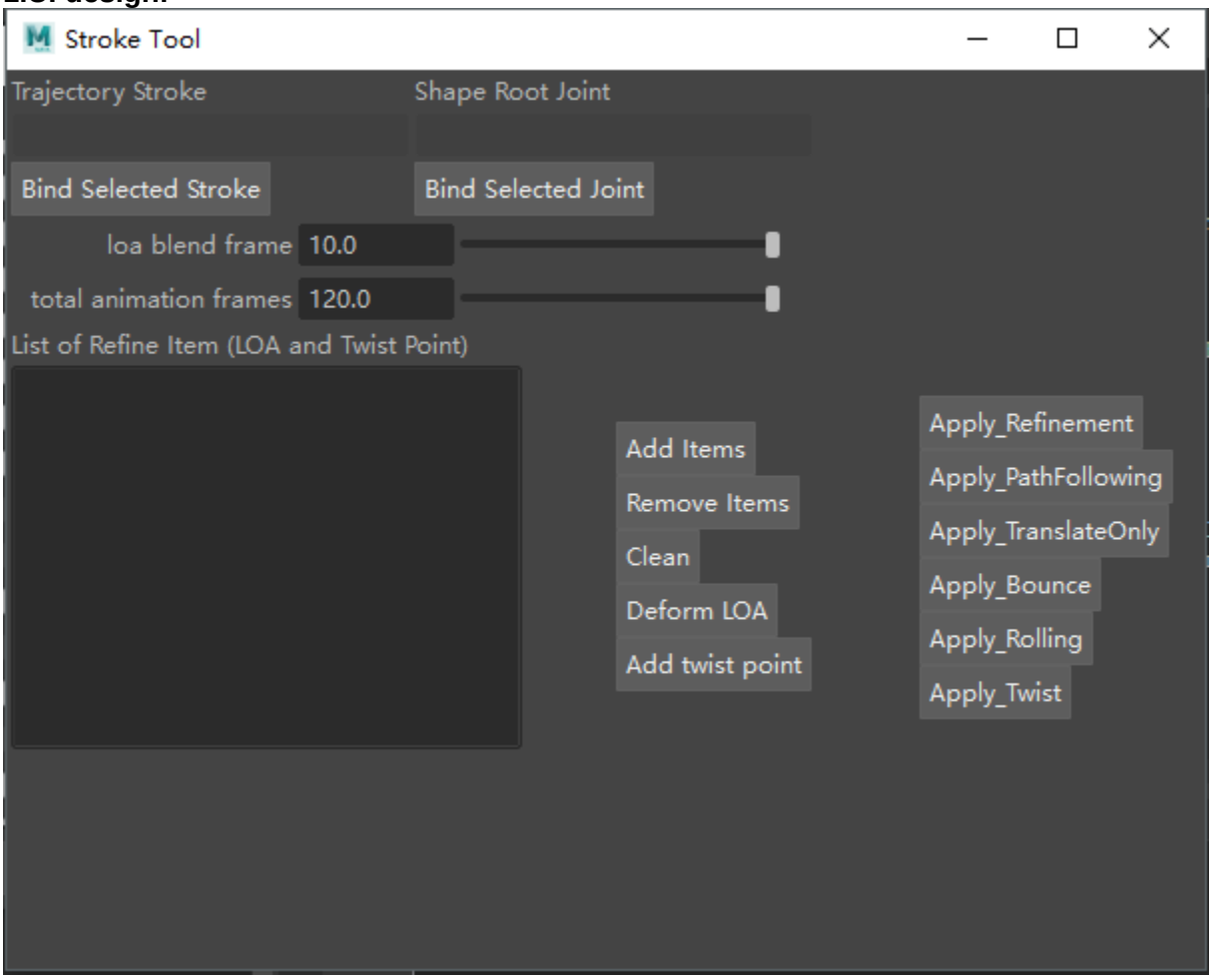


1.Maya is a very powerful software which can build very complicated animation. It can achieve every thing means itself is difficult.
2.At first, we should learn to make the animation manually using Maya.
3.The operations done in Maya are sequential and can be describe as mel (Maya embedded language) code.
4.Then the series of codes should be packaged into procedures which can be reused whenever it is needed.
5.Some commands with certain functionalities can't be found in Maya, but we can build a new Maya command in C++ using Maya API.
6.We can intergrate all the work into a Maya plugin and register the new command when initialize the plugin. And load the mel file and declare the global procs and variables.
7.We also provide a UI with buttons to implement the user interaction.

4. Implementation

Most of the plugin are written in Maya embedded language (MEL) with Maya's built-in commands and newly created commands and computational nodes coded by C++ using Maya API.
A series of MEL commands can be packaged into a proc (procedure). So the mel file Window.mel is built up with many different procs.

1.UI design:



Use rowLayout and columnLayout to arrange the Layout. text, textField, button floatSlideGrp, and textScrollList are the elements used in the windows. When the buttons are clicked, a command would be run.
The first three rows help to select items from Maya and show the names or IDs of the selected ones in the textField.
Each button are bound to a proc that print the names of items in the textField.

```
global proc BindSelectedTrajectory()
global proc BindSelectedJoint()
```

Then next two rows use two floatSliderGrp to specify two parameters loa blend frame which means how long is blend time the model use to deform form a orignal shape to a LOA matching shape and total animation frames.

floatSliderGrp -label "loa blend frame" -field true -value 10 -min 1 -max 10 -fieldMinValue 1 -fieldMaxValue 100 sld_blend;
floatSliderGrp -label "total animation frames" -field true -value 120 -min 1 -max 120 -fieldMinValue 1 -fieldMaxValue 500 sld_total;

In the next row, there are three columns, the first two columns contain a textScrollList and several buttons to add the select items in Maya to the textScrollList and show its name or ID in the textScrollList. And remove a LOA or clean all the LOA in the textScrollList.
And clicking the “Deform LOA” Button will deform the joints selected to the LOA.
In the third column, several buttons are shown here to generate the animation according to different situations.
The procs bound to the buttons are listed here and the important ones will be described clearly in the following parts.

Button “Deform LOA” - global proc DeformToLOA();
Button “Deform And Blend” - global proc DeformToLOA();
Button “Apply” - global proc ApplyAnimation();
Button “Apply_PathFollowing” - global proc ApplyAnimation_FlowMotionPath();
Button “Apply_Bounce” - global proc ApplyAnimation_Bounce();
Button “Apply_Rolling” - global proc ApplyAnimation_Rolling();
Button “Apply_PathFollowingWithTwist” - global proc ApplyAnimation_Twist();

2. PathFollowing Animation and Translate only:
pseudocode:
select the trajectory stroke and the Joints of a 3D model;
specify the rootJoint and leafJoint at the very end;
create a ikHandle of the Joints using a mel command ikHandler -sol ikSplineSolver -ns 30 which means ...
set the attribute “dTwistControlEnable” of the newly built ikHandler to 1;
create a pathAnimation of the Joints along the trajectory with flags -follow (false for PFWithoutDeform and true for PathFollowing);

pathAnimation -fractionMode true -follow true -followAxis x -upAxis y -worldUpType "vector" -worldUpVector 0 1 0 -inverseUp false -inverseFront false -bank false -startTimeU `playbackOptions -query -minTime` -endTimeU `playbackOptions -query -maxTime`;

For PathFollowing Animation: one more command should be run. run flow command to create Lattice and Base for the model and divide to model into pieces to accomplish the deformation when the model is moving along the trajectory curve.

flow -divisions 30 2 2 -objectCentered 1 -localCompute 0 -localDivisions 2 2 2 $ikSplineName;

3. Rolling:

Detect loops in the trajectory curve by self Intersection detection.
pseudocode:

global proc float[] GetSelfIntersection(string $c){}

get the spans of a single curve;
detach the curve into spans;
detect if there are intersection between every two spans;
get the u value of the intersection position; (a nurbscurve can be described as f(u)) (there would be two unique u values if there is one loop along the curve)

```
global proc float[] GetRollingParam(string $curveName){}
```
there is a redundancy in the result of GetSelfIntersection(), so we need to take the unique values out;

```
global proc ApplyAniamtion_Rolling(){}
```
select the trajectory stroke and the Joints of a 3D model;
specify the rootJoint and leafJoint at the very end;
call the proc ApplyAnimation_FlowMotionPath() and get the return as a ikHandle;
set the attribute “ikFkManipulation” of the ikHandle to 1;
get the rolling parameter in the curve;
for every loop in the curve do
    get the frames when the model is entering a loop and leaving a loop as blend_in_frame and blend_out_frame; keyframe the two frames set the “ikBlend” of the ikHandle to 1 showing it still following the path; in the time from blend_in_frame+1 to blend_out_frame-1 (including the center_frame = (blend_in_frame + blend_out_frame)/2), set the attribute “ikBlend” of the ikHandle to 0 in order to make the model stop the deformation according to the path;

```
setAttr ($ikhandle + ".ikBlend", 0);
setKeyframe { ($ikhandle + ".ikb") };
```

    at the centerframe, set the rotateZ of the rootJoint as the average of the rotateZ of the model at the blend_in_frame and blend_out_frame to make the rotation animation;
done;

4.Refinement of PathFollowing with Over-sketching keyframes:
Over-sketching keyframes are the frames at which the model is matching to the additional LOAs drawn on the trajectory curve and refine the movement.
Just like the bouncing, animation refinement is based on the PathFollowing Animation and with LOA refinement in the middle of the path.
pseudocode:
select the trajectory stroke and the Joints of a 3D model;
specify the rootJoint and leafJoint at the very end;
call the proc ApplyAnimation_FlowMotionPath() and get the return as a ikHandle;
set the attribute “ikFkManipulation” of the ikHandle to 1;
get all the LOAs listed in the textScrollList;
get the blendFrames and totalFrames from the two floatSliderGrp;
for (every LOA in the list) do
    get the intersection node of the LOA and trajectory curve.
    get postion of the node in the trajectory curve (fraction p : 0-1);
    compute the three keyframes p * totalFrame - blendFrames (blend_in_frame) , p * totalFrame + blendFrames (blend_out_frame) and p * totalFrame (p * totalFrame as the center frame of the refinement) and set keyframes for the ikHandle.ikb;
    in the time from blend_in_frame+1 to blend_out_frame-1, set the attribute “ikBlend” of the ikHandle to 0 in order to make the model stop the deformation according to the path;

```
setAttr ($ikhandle + ".ikBlend", 0);
setKeyframe { ($ikhandle + ".ikb") };
```

    at the centerframe call the proc DeformToLOA_Pure() to achieve the local-length preserving deformation and make transition animation to and from the centerframe;

```
DeformToLOA_Pure();
for($j in $childrenJoints)
{
        setKeyframe {($j)};
} done
```

5. Bouncing (Refinement of translation)

Bouncing is based on the Translation Animation and with LOA refinement in the middle of the path.

pseudocode:

select the trajectory stroke and the Joints of a 3D model;
specify the rootJoint and leafJoint at the very end;
call the proc ApplyAnimation_TranslateOnly() and get the return as a ikHandle;
set the attribute “ikFkManipulation” of the ikHandle to 1;
get all the LOAs listed in the textScrollList;
get the blendFrames and totalFrames from the two floatSliderGrp;
for (every LOA in the list) do
    get the intersection node of the LOA and trajectory curve.
    get postion of the node in the trajectory curve (fraction p : 0-1);
    compute the three keyframes p * totalFrame - blendFrames (blend_in_frame) , p * totalFrame + blendFrames (blend_out_frame) and p * totalFrame (p * totalFrame as the center frame of the refinement) and set keyframes for the ikHandle.ikb;
    in the time from blend_in_frame+1 to blend_out_frame-1, set the attribute “ikBlend” of the ikHandle to 0 in order to make the model stop the deformation according to the path;

```
setAttr ($ikhandle + ".ikBlend", 0);
setKeyframe { ($ikhandle + ".ikb") };
```

    at the centerframe call the proc DeformToLOA_Pure() to achieve the local-length preserving deformation and make transition animation to and from the centerframe;

```
DeformToLOA_Pure();
for($j in $childrenJoints)
{
        setKeyframe {($j)};
}
```

done

6.DeformToLOA

The proc DeformToLOA is a central procedure in the project and it achieves the deformation of a joints skeleton to one that matches the specified LOA curve.

```
global proc DeformToLOA_Pure()
{
    string $root = `textField -query -text tf_rootJoint`;
    string $childrenJoints[] = `listRelatives -ad $root`;
    string $end = $childrenJoints[0];
    CurveIKCmd -root $root -end $end;
    //registered command plugin
    //CurveIKCmd is going to deal with the deformation
}
```

CuvreIKCmd is not a built-in command of Maya, it is a command plugin we built using the Maya C++ API.

The basic structure:

```
class CurveIKCmd : public MPxCommand
{
private:
public:
        CurveIKCmd();
        virtual ~CurveIKCmd();
        static void* creator() { return new CurveIKCmd(); }
        MStatus doIt(const MArgList& args);
        static bool builtLocalSkeleton;
};
```

The most important part is the function doIt:

pseudocode:

```
MStatus CurveIKCmd::doIt(const MArgList& args)
{
        1.select the curve;
        2.select the rootJoint and endJoint;
        3.build the local skeleton along the joints;
        4.get the length of local skeleton and the new curve and get the strech factor;
        5.get the new uValue of each joints on the new curve;
        6.get the length of every bones on the old skeleton and new curve and get the strech vector (only consider the length not the thickness)
        7.get the rotation axis by  compute vector between the new direction of the bone and the old direction of the bone ^  (1,0,0) and the rotation angle by using the Maya command angleBetween;
        8.make the rotation and scaling of every bone;
        9.set the position of the rootJoint to the new curve's start point;
}
```

5. Results

1.DeformToLOA:

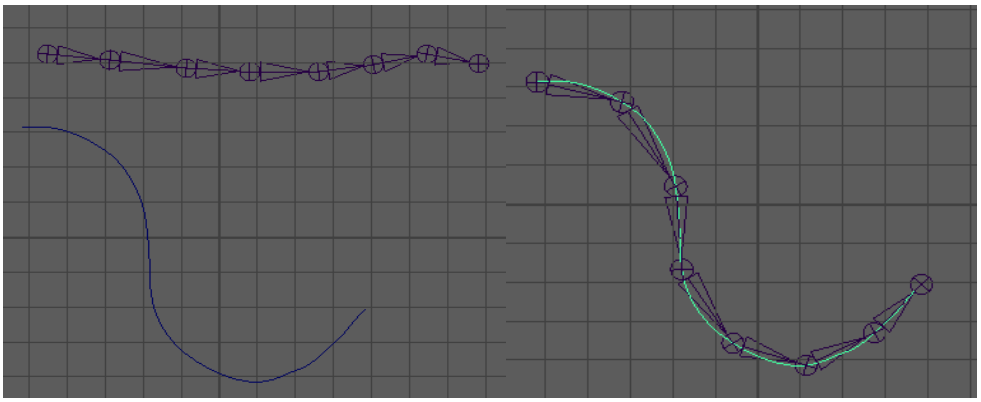


Figure 5.1

First, we test the effect of the deformation by the Command CurveIKCmd.
We can see that every joints are perfectly positioned on the curve and the ratios of lengths between two different bones perverse.

2.PathFollowing

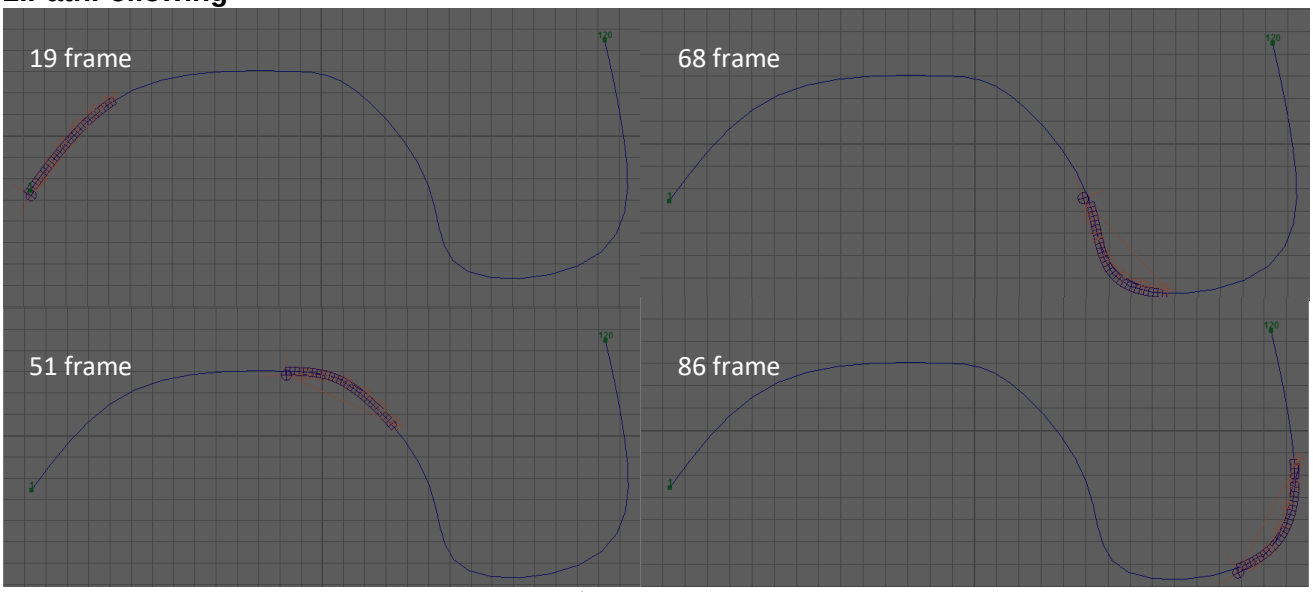


Figure 5.2

In the screen shot, we can see that the skeleton is moving along the trajectory curve and the deforming its shape according to the local shape.

3.Translation

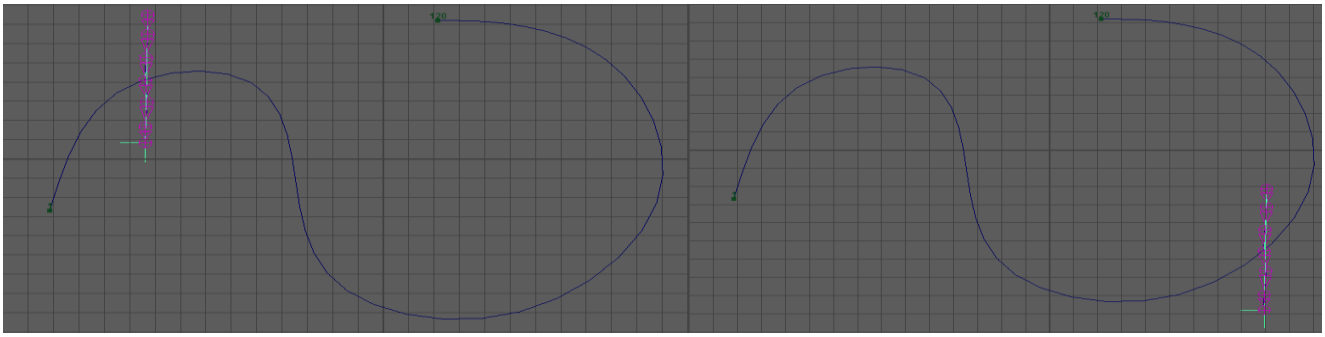


Figure 5.3

Translation just translate the center point of the skeleton along the curve without deformation.

4.Bouncing (Refinement of Translation)

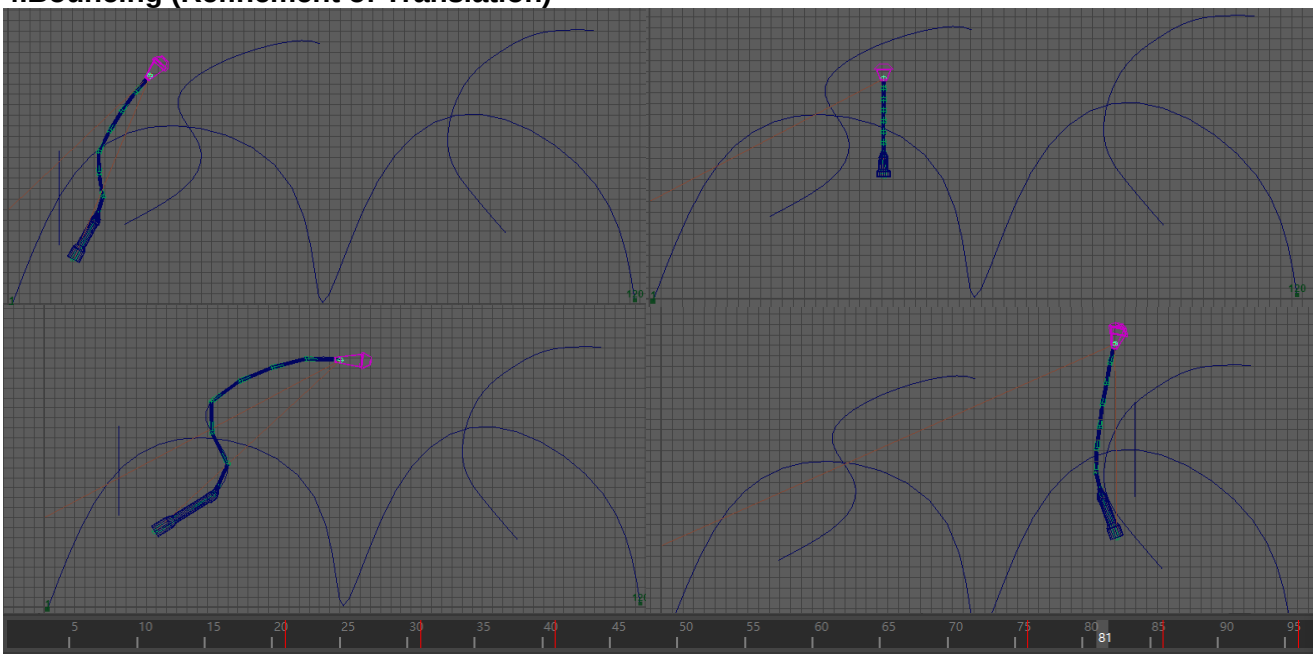


Figure 5.4

Bouncing is aiming at simulate the jumping or bouncing like a human. We need to use LOA along the trajectory curve and insert keyframes in the middle to adjust the poses when the model is moving in the air.
The result shows that we can achieve the animation that the model can deform to the LOA and recover in certain specified blend frames.

5.Refinement of PathFollowing

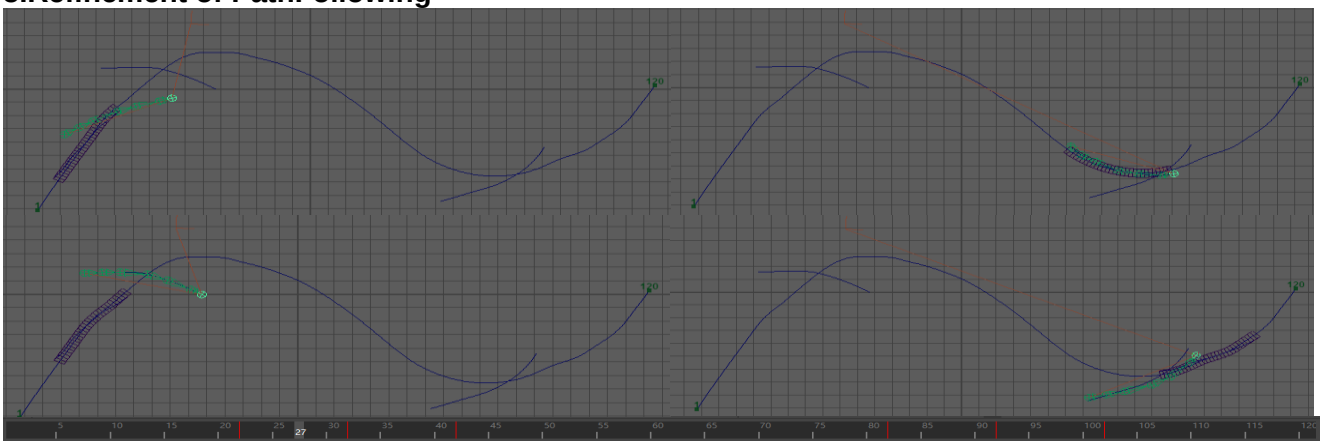


Figure 5.5

During the path following animation, we want to refine the poses of the model using additional LOA.
The result shows that we can successfully animate the model can deform to the LOA and recover in certain specified blend frames.

6.Rolling

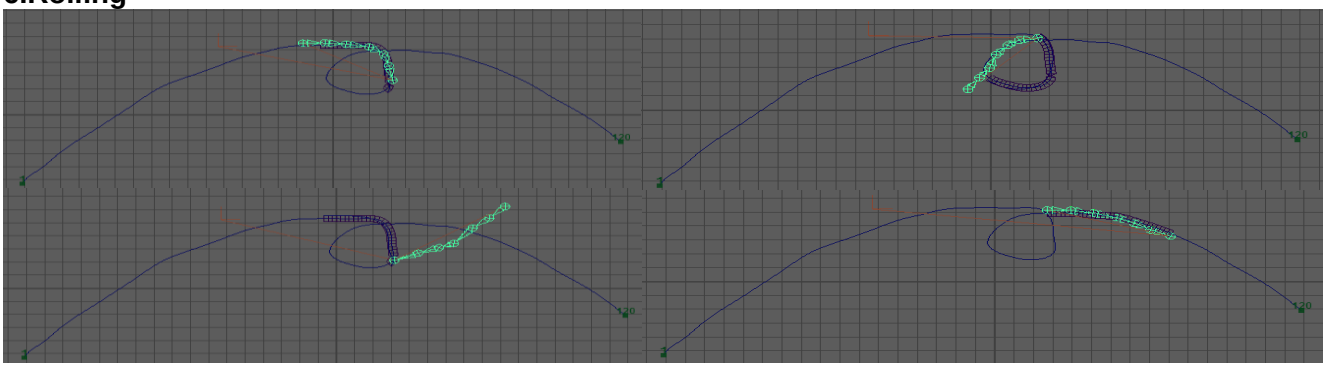


Figure 5.6

One difficult thing about rolling is that if there is a loop in the trajectory curve, the model should rotate its tail to prevent selfcollision.
The result shows our method can avoid selfcollision and give a natural rolling animation of the model.

6. Conclusion & Future directions

Conclusion

- We build a Maya plugin called StrokeTool. Our tool will free animators from nasty and repetitive key framing processes. With our tool, high quality animation with simple character morphologies can be generated given a trajectory and shape matching strokes. Additional shape matching curve can be added to refine the original automatically generated animations.
- The results shows that we almost achieve the basic idea of the paper. And package it as a Maya plugin that anyone can easily get and deploy in their environment and use it in Maya. And the animations built by our tool have the similar quality of the animations proposed by the paper and the results are consistent with the original paper.
- There are many limitaions in our work. The plugin has some bugs and may doesn't function well after several times. And the perspectives have some restrictions. And the deformation effects are highly related to the quality of the 3D model. In addition, more work should be done to improve the quality in the future directions.
- In conclusion, we make a Maya plugin to help build animation with space-time sketching.

Future direction

- In the proposed work, we didn't achieve good effect of rolling and bouncing and didn't finish the functionality of adding twists to the animation.
- In addition, we should do more work to achieve animation along different axes. And we can make animation such as a dragon flies along with a path and waves its wings at the same time.

7. References

- 1.Martin Guay, R´emi Ronfard, Michael Gleicher, Marie-Paule Cani. Space-time sketching of character animation. ACM Transactions on Graphics, Association for Computing Machinery, 2015, 34 (4), pp.1. <10.1145/2766893>. <hal-01153763>
- 2.https://www.youtube.com/watch?v=9d3LOCbFNy4
- 3.https://www.researchgate.net/publication/277664449_Spacetime_sketching_of_character_animation_Space-time_sketching_of_character_animation/figures?lo=1
- 4.http://help/autodesk.com/view/MAYAUL/2017/CHS//
- 5.http://kesen.realtimerendering.com/sig2015-changelog.html
- 6.http://www.liyiwei.org/papers/workflow-siga15/preprint.pdf
- 7.http://www.cc.gatech.edu/~karenliu/Performance.pdf