SLIDE 2 – INTRODUCTION

Motivate with 3 statements.

One: Style is important for character animation.

Two: **Creating motion in one style is a pain.**
Keyframing is painful for animators.
Motion capture is painful for motion actors.
Techniques reduce pain, but still pain.

Third statement corollary to second.
Creating motion in more styles is more pain.
If you create a motion in one style,
but want that same motion in another style,
you often have to repeat lot of work.

SLIDE 3 – GOAL

Our work aims to reduce amount of pain involved by addressing task of **style translation –** which we define as **process of taking a motion in one style, and changing it to another style.**

Our method allows users to define the desired style translations using examples.

For instance, you might provide short examples of normal walks, sneaky walks. Using these, method learns the difference between normal and sneaky. Doing so allows it to translate new normal walks to sneaky style.

SLIDE 4 – MOTION GENERATION

More context by mentioning related work.
First, look at some motion generation techniques.
Roughly categorize **by how they create motions.**

- Some concatenate examples
- Some interpolate examples
- Some draw from generative statistical models.
- Some optimize physical energy functions.

Algorithmically, quite different.
High level, **[C]** all model the space of “good” motions,
Where “good” = “realistic” “natural”

How do they do this?

SLIDE 5 – MOTION GENERATION

First two **combine discrete examples of good motions.**

Second two **define continuous goodness functions over space of all motions.**

Techniques used by following general template.

LEADIN – Select a good motion …

SLIDE 6 – MOTION GENERATION

Select … that meets demands given by animator.
For instance – might **select** concatenation of examples, and **demand** it follows a path.

Also could do style translation.
[C] Define good motion as realistic, in desired style.
[C] Demand selected motion is similar to input.
Conceptually good – practically, has limitations.

[C] Since space of realistic motions is complex, mogen techniques need to **[C]** approximate.
Since **result is selected from approximation**, subtle details of input get **lost during translation process.**

[C] Similarity demand hard to satisfy.
Involves expensive optimizations either when **generating** motion or **comparing** it to the input.

SLIDE 7 – MOTION TRANSFORMATION

Get around limitations by viewing style translation as motion transformation. Model differences between good motions – instead of space of good motions.

Reasonable for style translation.
Suppose trying to model exaggeration.
[C] Could model **complex process** that generates.
[C] Or model exaggeration directly.

Contrived example, but point is modeling differences often easier than modeling everything.
Result: motion transformation techniques are usually **fast and capable of preserving detail.**

SLIDE 8,9,10 – MOTION TRANSFORMATION

Some examples …

Motion warping: **Models differences as** smooth scales/offsets of motion curves. **Changes** motions to satisfy kinematic constraints.

[C/9] Frequency filtering: **Models differences as** adjustments of frequency bands. **Changes** motions to have different styles – style translation!
Unfortunately, frequency limited; no timing.

Plus, requires users to define styles manually. Often **easier and more intuitive** to define by example.

[C/10] One approach that does this is emotional transforms: **Models differences as** adjustments of timing and amplitude. Also limited, no frequencies.

Our goal is to combine the best qualities …
We try to make style translation fast and accurate by viewing it as motrans. **We try to make it easy** by letting translations be defined by examples. **Use a model** that captures freq, timings, scalings, offsets …

LEADIN: So, let's see what we do …

SLIDE 11 – METHOD

First step – get examples that **implicitly define the desired style translation. Each example pair of motions –** one in input style, other is **what you want it's translation to look like.**

Since motions differ in style, **usually have significant timing differences.** This fact complicates the task of modeling the translation.

SLIDE 12, 13, 14 – METHOD

To resolve, **find timing differences by figuring out which frames of input match to which frames of output [C].** In this picture, first frame … We call this correspondence step. Leave details until later **[C]**.

Using correspondence, we warp **[C/13]** output motion to **strip away** timing differences. But don't want to discard … **[C/14]** So store timings within the frames.

This gives a framewise mapping between motions that doesn't lose original timing information.

SLIDE 15 – METHOD

Mappings used to estimate parameters of style translation model **[C]**. I'll discuss what, how to estimate later **[C]**. **For now, let's just see how to use the model once you have it.**

SLIDE 16 – METHOD

Start with new frame of motion in input style.
[C] Run through the model, which translates style and produces proper timing.
[C] Apply timing parameter.
To handle entire motions, repeat for each frame.

SLIDE 17, 18 – METHOD (SUMMARY)

Summarize method, but on one slide.

[C] Given example motions,
[C] a correspondence algorithm
[C] finds matching frames.
[C] The motions are warped
[C] to give framewise mappings
[C] which are used to estimate
[C] a style translation model.
This is used to process **[C]** new frames,
One at a time.

Questions at bottom correspond to …
[C/18] These two parts of the method.

So, let's get to the first one.
READ THE SLIDE.

LEADIN: The standard way …

SLIDE 19, 20, 21, 22 – CORRESPONDENCE

Standard way is DTW; optimizes this objective.

[C/20] **In this expression,**
U and Y are motions represented by vectors.
Contain discrete samples of motion curves.

[C/21] **W** is the **time-warp function.**
Nonuniformly stretches U to same length as Y**;**
makes it possible to compute their distance.

In words, find **temporal adjustments** of U to bring it as close as possible to Y. Commonly used, but problem for our purposes.

Motions in different styles have different poses.
In other words, temporal and spatial differences.

[C/22] Objective accounts for **spatial differences** using **temporal adjustments** … leads to bad results.

SLIDE 23, 24, 25 – CORRESPONDENCE

We add ability to make **spatial adjustments.**
Type is borrowed from motion warping, which **models differences between motions as smooth scales and offsets. That's what this objective is doing …**

[C/24] Spatial adjustments performed on U using **scale vector A** and **offset vector B …** [C/25]
Smoothness approximated using finite differences.

In words … find **temporal and spatial adjustments** of U … **to bring it as close as possible** to Y … while keeping spatial adjustments as smooth as possible.

SLIDE 26, 27 – CORRESPONDENCE

To optimize, use coordinate descent.
Alternate between optimizing **W using DP**,
And optimizing **A and B by solving linear system.**
We call this algorithm
[C/27] **ITERATIVE MOTION WARPING.**

And that is how we compute correspondence.

LEADIN: Let's move on to the next question …

SLIDE 28 – METHOD

On to the next question. ***READ THE SLIDE.***
Let’s look more closely at the input to this step.

SLIDE 29 – MODELING

Remember from visual description … correspondence gives framewise mapping …
Let’s **[C]** carry over notation and name these things.
Putting it this way, goal of modeling is to find relationship between UTs and YTs.

SLIDE 30 – MODELING

Simple way, linear regression.
Handles simple translations, like exaggeration, but not ones that depend on velocity, acceleration, frequency.

Furthermore, **regression methods in general** ignore fact that **the UT-YT mappings are samples from time series.** In other words, could **reorder mappings** and still get same model.

We can resolve these issues with a few simple modifications.

SLIDE 31 – MODELING

First, add state variable XT, depends on UT **and itself** during previous time-step.
This self-dependency allows it to capture things like velocity, acceleration, frequency, and so on.
Also **reflects temporal nature of the data.**

SLIDE 32 – MODELING

By letting YT also depend on state, we get a
Linear Time-Invariant System

Even though model is more powerful than linear regression, not much slower to use. Adds a few more matrix operations.

But estimation is much more difficult.
Have to solve A,B,C,D … **but only given** Us and Ys.
Don’t know Xs … don’t even know dimension.

Thankfully, there are several existing techniques that’ll perform the estimation task.

LEADIN: The one that we chose to use …

SLIDE 33 – MODELING

The one we chose is called Numerical Subspace … **Provides optimal estimates of all parameters, without needing precise tweaking or getting trapped in local minima.**

SLIDE 34 – MODELING

We use algorithm to estimate individual models for each joint … i.e., **each joint is translated independently.**

A contrast to motion generation techniques, which need to identify dependencies between different joints.

When you're **generating** walking, **need to know that arms swing roughly out-of-phase.**

But when you're **translating** walking, input **already going to have this property** – and it can get passed **directly to the output.**

In other words: **Desired dependencies can be drawn from the input, instead of the model.**

Many motion transformation techniques rely on this **for their speed and quality.** For us, benefit is lot fewer parameters → lot fewer examples to define translation.

SLIDE 35 – METHOD

That completes description of core of method.
Almost there; not quite done.
Since method **based purely on data**, has **standard limitations of these sorts of techniques.**

SLIDE 36 – POSTPROCESSING

We take care of two of these issues in a postprocessing phase.

[C] First – doesn't enforce physical or kinematic constraints. Most visible effect – sliding feet.
To fix, use **simple classifier to learn footplant annotations.** Used to **clean up sliding feet using existing techniques.**

[C] Second – model is only complete as data it's given.
If you provide examples of walking, **unlikely to handle inputs like backflips.** Could add examples, but **often impractical or impossible of every strange motion that you might encounter.**

In these cases, we apply a simple heuristic.
It **[C] detects inputs not likely to be translated correctly,** and **[C] just passes them through instead.** Motivation: **no translation preferable to bad translation.** More details in paper.

LEADIN: Let's move on to some results …

SLIDE 37 – RESULTS

Move on to results.

SLIDE 38 – LOCOMOTION

First evaluation applies method to 5 different styles of locomotion: *READ STYLES.*

For each style, captured **short walk cycles** in
5 different turning angles and **3 different speeds.**
Total, about a minute for each style.
Clips about to show **straight-angle, medium-speed.**

SLIDE 39 – EXAMPLES (LOCOMOTION)

RECITE STYLE NAMES AS THEY APPEAR. HOLD.

Used these examples to estimate style translation models from NORMAL to each of the other styles.

SLIDE 40 – TRANSLATION (LOCOMOTION)

We applied each of the style translation models to this **new NORMAL-style walk.** Unlike examples, contains random, continuous changes in speed and turning angle.

HOLD FOR BLOWOUT*.** Here's the original motion translated to supermodel … ***REPEAT.

SLIDE 41 – NUMBERS (LOCOMOTION)

In all these tests – characters have **35 DOFs, sampled at 120fps.**

Building style translation models slow – several hours.
Using much faster – roughly **realtime framerates.**

Most of computation in **postprocessing sliding feet – The core of our method – the LTI system – could translate at about 10000 frames per second.**

SLIDE 42 – TRANSLATION (LOCOMOTION)

Here are some more exciting translations.
***ADLIB*.**

SLIDE 43, 44, 45 – FIGHTING

Next evaluation translates some fighting moves.
We captured motions in two styles: weak and aggressive. These included punches and kicks, aimed at three different heights.

[C] On the top are some weak examples, and on the bottom, some aggressive ones.
[C] Here's a new weak fighting motion, obviously not very threatening … and this is that motion translated to the aggressive style.

LEADIN: CONCLUSION

SLIDE 40 – CONCLUDE

Conclude where I started -- going backwards.

3: [C] Creating motion in more styles is more pain. That's what we tried to address. By **providing a fast, easy method to translate the style of motions**, we hope that our method can make this statement less true.

2: [C] Creating motion in one style is a pain. We didn't address this – wasn't our intention.

Since our method does motion transformation, still needs to get input motion from somewhere: Keyframing, mocap, or … motion generation techniques.

Those techniques are great at addressing this statement, and we're not trying to replace them.

Instead, hope our work can **complement their capabilities in many of the applications where they're already useful.**

We also hope our method can be useful independently, when all you want to do is change a motion that you already have.

Finally, just for completeness …

[C] Style is important for character animation.