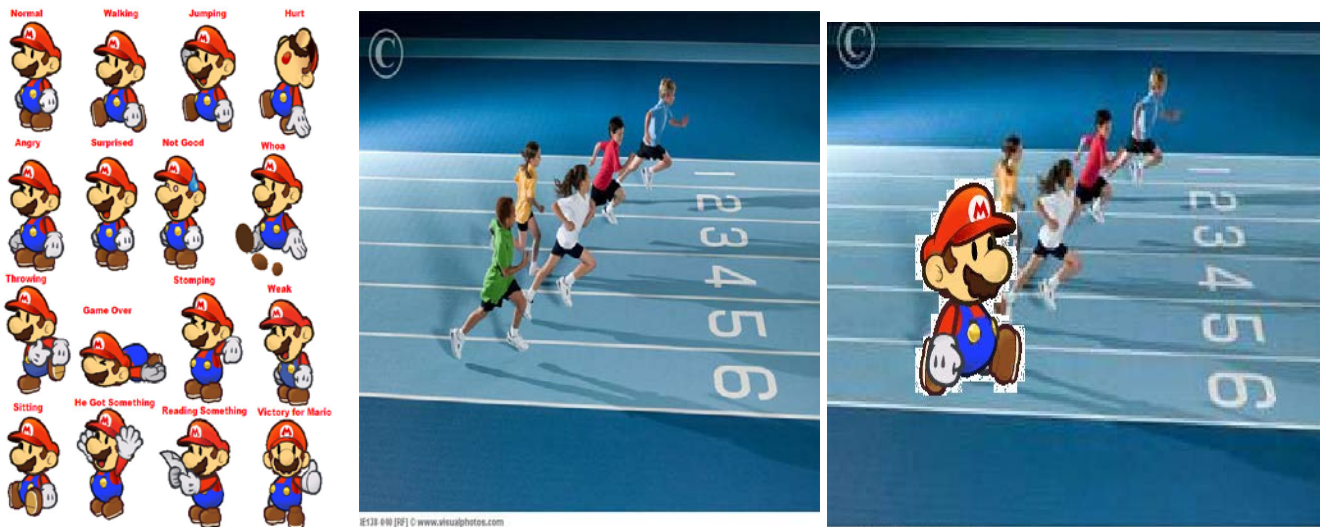


# Video Sprite Replacement

Nancy Wang

University of California: Berkeley

Nancydragon12345@gmail.com



**Figure 1: Basic Conception.**

The initial proposal for the project of this paper was to completely replace a person from a video with a cartoon character. The current focus is to use pose recognition to match the sprites instead. A set of sprites of a cartoon character (left) would be compared the individual in question in a key frame from the video (middle). After finding the best match, the cartoon pose would then be substituted into the individual's place. This set of examples sprites is of the video game character Mario from the Paper Mario series under Nintendo Co, Ltd.

## ABSTRACT

*Animation has gained much attention overtime since the creation of animated films during early 20<sup>th</sup> century. Up to present day, a popular thing to do, through various media, is to combine live-action and animation together in videos. This practice has been done by professionals and amateurs alike from film industries for movies to public internet videos.*

*However, the process can be time consuming and costly to draw or model out the animation separately and then put it into video. This paper explores methods to simplify the process by using pose recognition between two different sets of sprite images--a set of key sprites of a live-action person from a video with a set of key sprites of a cartoon character.*

**Keywords:** animation, affine, cartoon, control joints, key

frame, motion tracking, skeletons, retargeting, rotoscoping, key sprites, toon sprites, video.

## INTRODUCTION

Animation is created by combining visual and motion style together. Characters created for this type of medium can be portrayed from realistic to highly exaggerated facial expressions and movements. The production of these is usually handled by teams of professionals in the film and other media industry.

## Background

The problem with animation in general is that it can be really time-consuming or costly to do. In the early days of traditional animation, animators would have to draw out everything by hand. Animators would draw out each frame individually, with each sequential one slightly differing from the one before to simulate movement. Considering animations are run at 24 frames per second on average, requiring 12 to 24 drawings per second, an one-hour animated movie would need up to 86400 sketches. [4]

An early method of recording human motion was rotoscoping. For older films such as Disney Studio's *Snow White*, animators would trace animation over film footage of live actors playing out the scenes in order to copy the movement for realistic motion of their animated characters. However, motion capturing through this process was still done generally painstakingly by hand. [7]

Presently, motion capture has been a standard tool by many animated film industries such as Pixar and Dreamworks to apply motion to 3d models. This is done by having live action actors wear special suits from which computer can record their movements. However, such equipment, software, and personnel use tend to be very expensive.

### Motivation

It is interesting to try to match the cartoon sprite one selects to that of a real life person on video such without hand-drawing everything. This can be difficult to do since the poses might not match well or the motion can be very complex. Another problem is that most well-developed motion capturing software and equipment are too expensive for most of the populace. While it is readily used for huge industries under entertainment companies such Pixar and Disney, this is generally not easily available by small production groups or individuals.

What we suggest is to try to apply some existing vision based techniques to attempt to replace live-action people from existing videos with our own selected cartoon character sprites.

### RELATED WORK

Some previous work have developed some methods to address the problem. Concerning cartoon capture and retargeting of animated characters, Bregler [1] focused on isolating the motion style of an existing cartoon animation and apply the same style to a new output domain. He splits the cartoon motion into affine and key-shape deformations. This involves estimating motion vectors that approximate the contours of the input key shapes, which is then used to apply to the target key-shapes.

In their study of Sign Language Pose Recognition, Pennock and Gringold [6] applies a robust version of the Lucas-Kanade [3] algorithm. Through this, they constrain the dimensionality of the affine transform from 6D to 4D, keeping only rotation and translation properties, and recognizes hand poses from one direction only. They had come across errors in which some poses were misclassified when read. Bregler mentions in his paper that algorithms such as Horn-Schunk's [2] and Lucas-Kanade that calculate affine warp flow in 1D is not precise for non-linear deformations.

Junjun and Zhang's [5] work into sketch-based skeleton-driven animation 2D is based on using creating a skeleton and moving its joints to create deformations of figures. In first, involves detecting the silhouette of the figure and applying triangulation and thinning algorithms to obtain a skeleton. Deformation of the target is created through manually moving the skeleton joints to obtain the new figure.

Considering these referred works, we would like to focus on matching the shapes of one real-life person with the key-shapes of a cartoon character by comparing the optical-flow error between them.

### METHODS

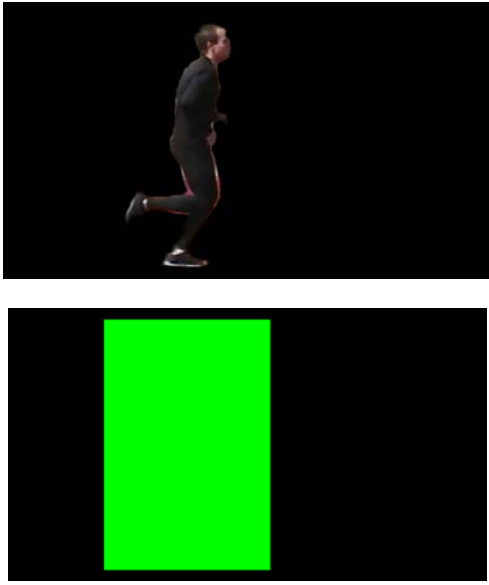
This paper involves character pose recognition by finding the pose with the least affine optical flow error between characters. Some preprocessing of images and videos are required before applying them into our system.

### Inputs

For the first step, we want to take a video to track the motion of an individual recorded in the video. For the purposes of concentrating on the individual's movement only, we use Adobe After Effects to video mat the background. This is done by using the program's rotobrush tool on the person of interest in the first frame, which will automatically be applied to each subsequent frame, with some user editing to fix any rotoscope lines that may go out of place. The rotoscope masks obtained from this are used later in the Matlab code for pose matching in the key frames.

Using the same video after rotoscoping has been done, we use Adobe After Effects to apply motion tracking on the individual to track his movement across the screen. The tracked motion is then applied to a marker that encompasses the individual. Because the motion trackers are not very precise on tracking joints, we set the motion tracker on the whole figure. Since videos are difficult to handle in Matlab, the frames of the two videos, acquired from rotoscoping and motion tracking the original, are converted into images files. The key frames in both videos are preselected to be put to use in our program.





**FIGURE 2:** During preprocessing, the original video (top) is imported in Adobe After Effects. The rotobrush feature helps rotoscope the individual (middle) in every frame. Motion tracking of individual is applied to Green Box marker (bottom).

Our Matlab program is given three sets of data: a set of key frames of the video that contains the rotoscoped masks of the person of interest, a set of key frames of the video that contains the rotoscoped person motion tracked with a green marker, and a set of cartoon sprites to be used to substitute in to the retargeted images (as seen in example of frame 43 in Figure 2). Each group of images is read as an array of images. The number of cartoons sprites does not necessarily have to be the same amount as the number of key frames. The main idea of this paper is to extract key shapes of a real-life person and match to the toon sprites accordingly.

Our program then iterates through the array of marked key frames in order to build an image array of key-shape sprites. After detecting the green-box marker, it crops the key sprite from the corresponding rotoscoped image according to the position of the marker. This array of key sprites will be used to compare against the poses of the toon sprites.

### Comparing Key Shapes

After the key-shape sprites are obtained, the program starts comparing each one to each toon sprite pose. This is done by calculating the error in between each pair of the key sprite and the toon pose. For calculations, we will mainly be using Bregler's affine parameters and equations in finding the optical flow error between the masks of the shape of each pair of sprites. The affine parameters at time frame  $t$  are defined as

$$\theta(t) = [a_1, a_2, a_3, a_4, d_x, d_y]$$

**FIGURE 3:** affine parameters from Bregler [1]

with  $a_1, a_2, a_3,$  and  $a_4$  describing rotation, x/y scale, and shear, and  $d_x$  and  $d_y$  for the x/y translation.

We acquire image gradient of the difference between key-sprite and toon-sprite.

$\Delta I = [I_x, I_y]$  which is the image gradient of  $I_t$  in x and y direction.

The affine parameters theta ( $\theta_{aff}$ ) are then obtained thusly.

$$\begin{aligned} err_{aff} &\approx \sum_{s_i \in S} (I_t(s_i) + \Delta I(s_i) \cdot (A \cdot s_i + D) - I_0(s_i))^2 \\ &= \sum_{s_i \in S} (H_i \cdot \theta_{aff} + z_i)^2 \\ \theta_{aff} &= [a_1, a_2, a_3, a_4, d_x, d_y]^T \\ H_i &= [I_x(i) \cdot x_i, I_x(i) \cdot y_i, I_y(i) \cdot x_i, I_y(i) \cdot y_i, I_x(i), I_y(i)] \\ z_i &= I_t(i) - I_0(i) \end{aligned}$$

**FIGURE 4:** equation 11 [1]

From Bregler's reference, the standard least-squares solutions of this linearized term is defined as

$$\theta_{aff} = (H^T \cdot H)^{-1} \cdot H^T \cdot Z$$

$$H = \begin{bmatrix} H_1 \\ \dots \\ H_N \end{bmatrix} \quad \text{and} \quad Z = \begin{bmatrix} z_1 \\ \dots \\ z_N \end{bmatrix}$$

**FIGURE 5:** equation 12 and 13 [1]

After affine parameters theta are obtained, we calculate the affine flow warp of the key-sprite  $S$  using the newly acquired theta.

$$warp(\theta, S) = \begin{bmatrix} a_1 & a_2 & d_x \\ a_3 & a_4 & d_y \end{bmatrix} \cdot S$$

**FIGURE 6:** equation 1 [1]

Once we get the warp( $\theta, S$ ), we calculate the error between the images with the following error equation

$$err_{image} = ||I_t(warp(\theta, S)) - I_0(S)||^2$$

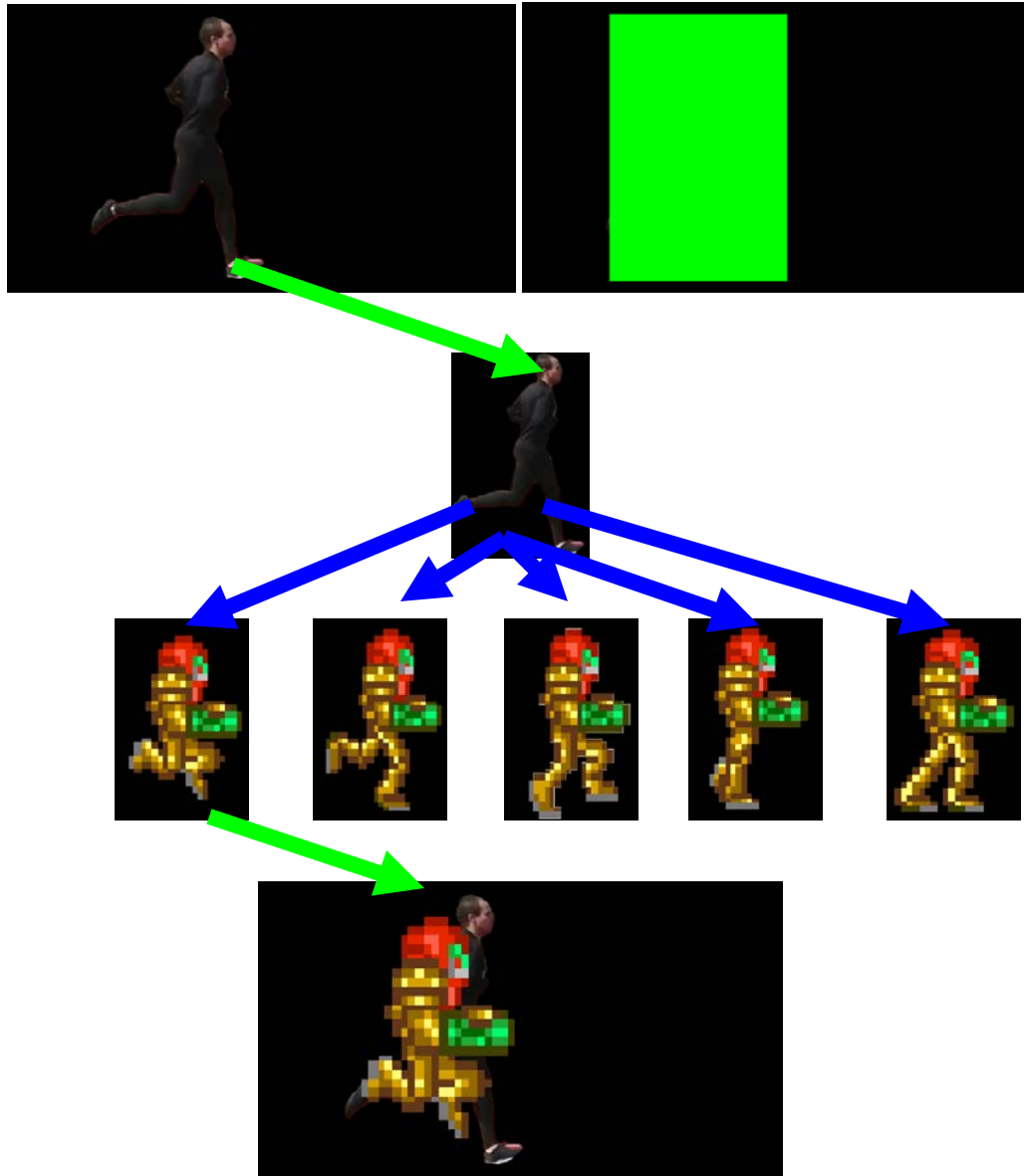
**FIGURE 7:** equation 8 [1]

with  $I_0$  as the toon image. Finally, The error calculated at each pixel is summed up.

With this error calculations done for each toon sprite for the current key sprite, the toon pose with the least amount of error compared with the key sprite is then pasted onto the position of the real-life person in the original key frame.

For the purpose of quick comparison, we set up the output of the system to overlays the toon sprite over the original figure so the user can quickly see how close their poses match visually.

Otherwise, the system can be easily modified to paste the toon sprite in a new blank image with a black background of equal size to the original image. The new images obtained may be applied with video matting to add the original background or any other backgrounds in post-processing.



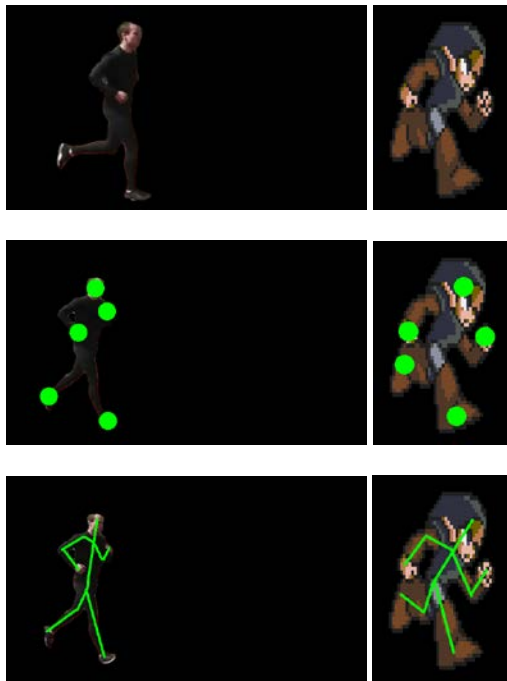
**FIGURE 8:** Quick visual overview of calculation process using Key Frame 9 (frame 40 of original video) as an example. The key sprite of the individual from the original picture (top left) is cropped out based on the position of the Green Box Marker (top right). Each cropped image (second row) is compared to a set of toon sprite images (third row, sprites of Samus

Aran from the Metroid games series). The pose with the least amount of error, in this case the first one, is then chosen to place at the position or the original individual.

### Comparing with Control Points and Skeletons

Ideally, we would have like to be able to apply motion tracking to the individual joints of the original real-life person. However, when using Adobe After Effects, the motion trackers would constantly linger or skew off track, and it would be immensely difficult and time-consuming to retarget each of the motion tracked points for every frame.

In an attempt to simplify the pose-recognition process, we want to try using control points and skeletons on each sprite instead of comparing the key-shapes in their entirety. To do so, we proceed in manually creating two copied versions of the sets of the original images and sets of toon sprites with these markers. The first version of the copies have the end joints (the head, hands, and feet) of each sprite clearly marked with green circles. The second version of the copies are marked with lines along the limbs, main body, and the head to represent the skeleton of each figure.



**FIGURE 9:** Examples using original frame 5 (left column) and one of the Dark Link sprites from the Legend of Zelda game series (right column)

**Top row:** original version of images

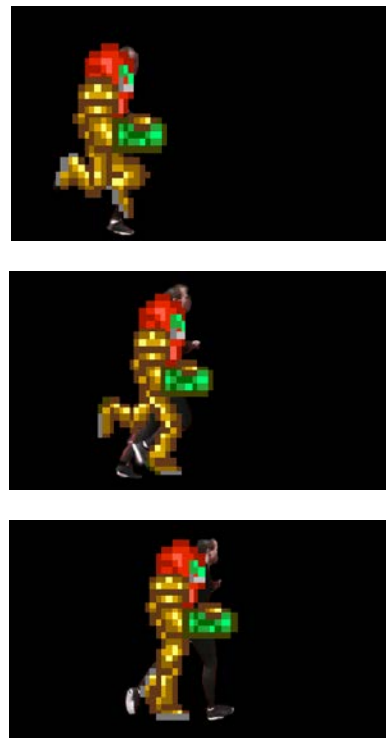
**Middle row:** Sprites marked with control joints

**Bottom row:** Sprites marked with lines for skeletons

The system we defined in Matlab has also been modified to incorporate these new inputs for processing. With the parameters set in the code to read the images marked by either control points or skeletons, the system would ideally read and detect the green markers in the from the modified images, and create masks based on those. This would be used to calculate the errors between the markers in the images. However, the system has come across some problems with this, which will be explained further under the Results Section

### RESULTS

For our initial results using pose recognition between the key shapes of the live person and cartoon sprite, we get relatively good results. With some minor mismatches for a few of the key frames, the poses of the toon sprites matched pretty well with that of the real-life individual in the video. The toon sprite is laid over the original person to easily compare their poses in each frame. Complete results acquired through this project can be seen in figure 13.



**FIGURE 10:** Outputed results of key frames 5 (top), 10 (middle), 15 (bottom), overlaid with Samus Aran sprites. Obtained through pose-recognition of key-shapes.

Since we only provided five toon poses for the Samus Aran sprites, it stands to reason that the some of the poses

would look somewhat off. Also, there is also to consider that all the Samus Aran toon sprites provided only shows her right arm, equipped with her primary weapon. So one cannot really compare the upper limbs of the jogger and that of Samus's. Aside for that, cartoon characters do not necessarily be perfectly human-shaped; as long it matches well for the most part it should be fine.

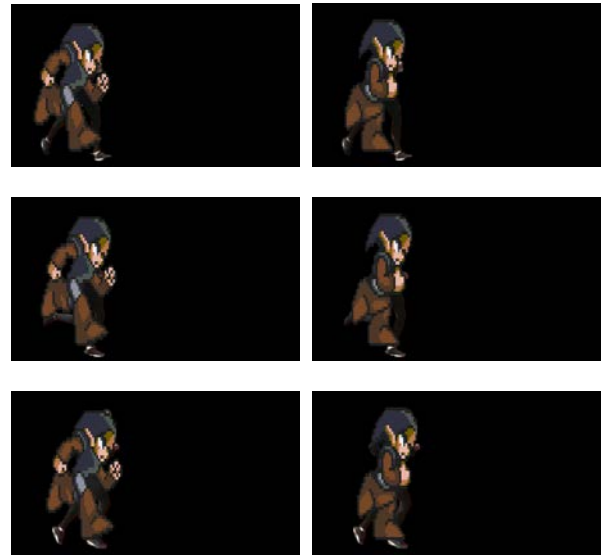
As for the Dark Link sprites, these results turn out better considering there was more sprites provided (Dark Link had nine toon poses as opposed to Samus's five). As with the Samus's output results, complete results acquired can be seen in figure 13



**FIGURE 11:**  
Outputted results of key frames 4 (top), 8 (middle), 11 (bottom), overlaid with Dark Link sprites. Obtained through pose-recognition of key-shapes.

Given a more variety of poses than the Samus Aran sprites, we can see the poses of the Dark Link sprites match even better. However, some of the matching does look different for some key frames. For instance, one may notice in key frame 8 of figure 11 that this cartoon character's legs does not quite extend all the way since his limbs are generally shorter and thicker than the original individual's. Aside from that, the comparison of the poses between these two characters worked out nicely.

However, the same cannot be said about the other inputs marked with control points and skeletons lines. With key-shape comparison, the system takes about an hour to running through all the error calculations to output all fifteen key frames. We had hoped to improve on the pose matching of our system by only comparing either the control joints or the skeletons of each figure. However, in either case, the outputs images end up overlaying the same selected toon poses on every image.



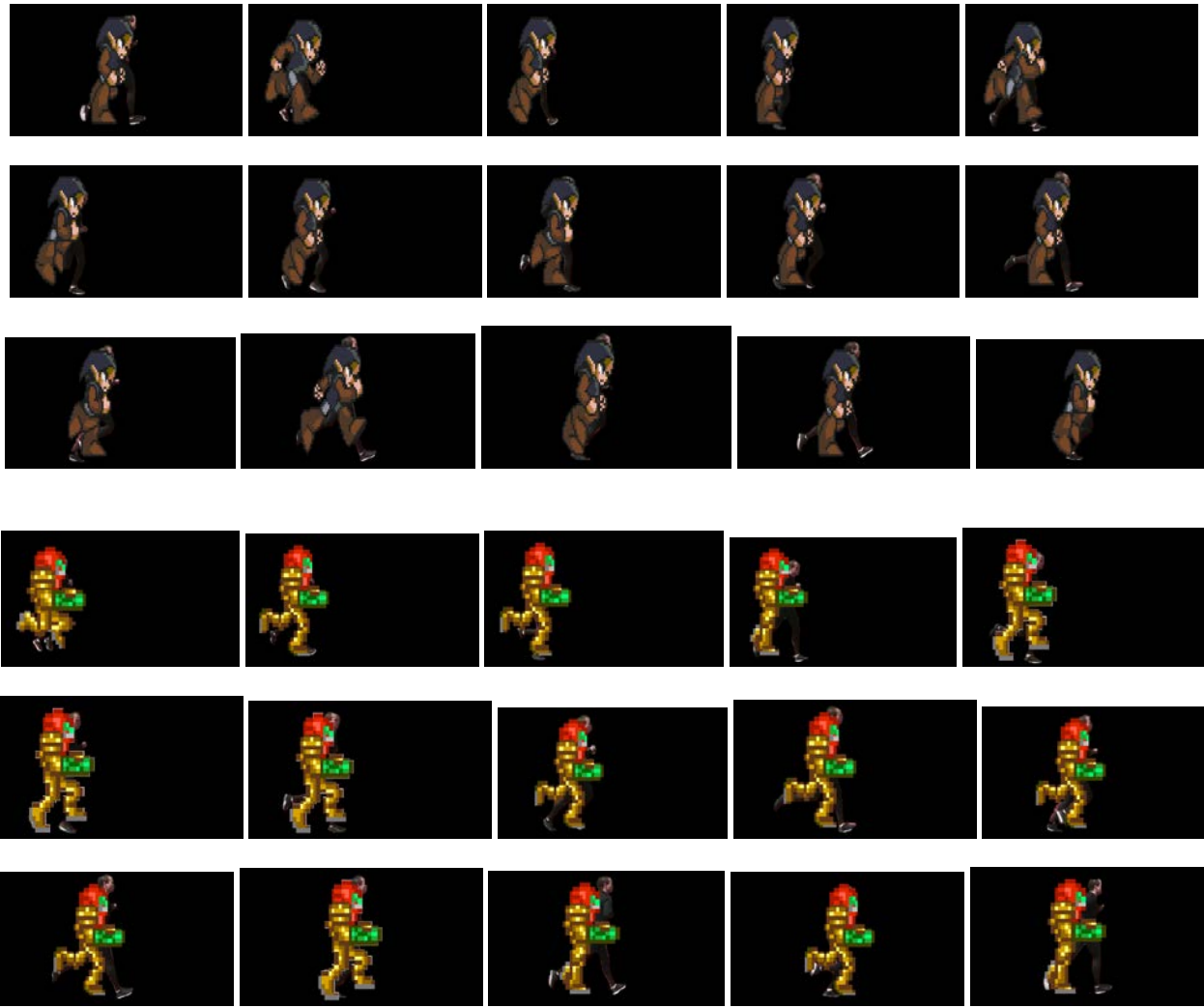
**FIGURE 12:**  
Outputted results of key frames 4 (top), 5 (middle), 6 (bottom), overlaid with Dark Link sprites. Left Column: Obtained from comparing control joints. Right Column: Obtained from comparing skeletons.

In both cases, this may be due to some of the limbs, as well as the markers, overlapping each other, so those poses may generate large errors, and would most likely be left out. Also, since the joint and skeleton markers were applied in manually through preprocessing, the actual joints and skeletons of the figures might not be well accurately presented.

## DISCUSSION

Comments and advice were received on our project after presenting the visualization of the project to our peers and others. One commenter noted that this system can also be applied by replacing cartoon characters with real-people as well in live-action animation; it would be interesting to use this to switch between any two characters from either the animated or real-life domain.

Another also mentioned that comparing characters with control points or skeletons would help better solve the problem. Currently, the system incorrectly handles these



**FIGURE 13:**

Output Results using Dark Toon Link sprites and Samus Aran sprites.

First Row: Dark Toon Link sprites for key frames 1 to 5; Second Row: Dark Toon Link sprites for key frames 6 to 10;

Third Row: Dark Toon Link sprites for key frame 11 to 15; Fourth Row: Samus Aran sprites for key frames 1 to 5;

Fifth Row: Samus Aran sprites for key frames 6 to 10; Last Row: Samus Aran sprites for key frame 11 to 15.

Results for skeleton and control joint marked images are left out since only one toon pose is applied in both cases.

inputs so further work must be looked into. Aside from pose-matching, the system could be further improved on by implemented key-shape interpolation between key frames to help create animation between the key poses.

More examples with this system may be shown later upon. With this system, we wanted to keep improving the system with consistent inputs.

### FUTURE WORK

In the future, we would want to consider better implementation of control joints and skeletons to track the movements of an individual's limbs more accurately. Further progress into this may be seen in readings into of

Sketch-based skeleton-driven projects and other related works mentioned earlier in this paper.

This system can also be extended to facial recognition by applying control points to track facial muscle contraction and relaxation.

### ACKNOWLEDGEMENTS

We would also like to thank all of those that have commented on the system and provided advice on it. We would also like to thank everyone whose images, algorithms, and ideas we have referenced and built upon for this projects. Finally, we thank Maneesh Agrawala and Floraine Berthouzoz for their teachings and advices in

during the whole course of CS294-69: Computational Photography and Image Manipulation.

## References

1. BREGLER, C., LOEB, L., CHUANG, E., DESHPANDE, H. 2002. *Turning to the Masters: Motion Capturing Cartoons*. In *Proceedings of SIGGRAPH 2001*, ACM SIGGRAPH.
2. HORN, B, AND SCHUNK, G. 1981. Determining Optical Flow. *Artificial Intelligence 17*.
3. LUCAS, B. D., KANADE, T. 1981. An Iterative Image Registration Technique with an Application to Stereo Vision. *Proc. 7th Int. Joint Conf. on Art. Intelligence*.
4. NYSTROM, J.E. 1972-1997. *How an Animated Cartoon is Made*. <http://www.sci.fi/~animato>
5. PAN, JUNJUN., AND ZHANG, JIAN J. *Sketch-Based Skeleton-Driven 2D Animation and Motion Capture*. 2011. National Centre for Computer Animation, Media School, Bournemouth University, UK.
6. PENNOCK, C., AND GINGOLD, Y. *Sign Language (Pose) Recognition. 2004*. <http://cs.nyu.edu/~ccp252/classes/vision/final/index.html>
7. STURMAN, D. J. 1999, March 13. *A Brief History of Motion Capture for Computer Character Animation*. Siggraph.org