

YogaHero: Rating Alignment in a Yoga Pose from a Single Image

Larry Reaves
Rose-Hulman Institute of Technology

Advisor: Dr. J.P. Mellor
Rose-Hulman Institute of Technology

Abstract

We will examine a method for extracting 3D model from a 2D image of a person performing a yoga pose. This model will then be compared to the ideal model for the attempted pose. The results of this comparison will be displayed, providing feedback to the user as to how precise their alignment is. We accomplish this matching in a two step process, whose initial input is a silhouette of the person performing the yoga pose. Using an idealized version of the pose as an initial match, we first attempt to align the torso by adjusting the camera parameters and torso size. Following this, we use Powell's method to obtain our final match. The error function used in Powell's Method is a linear combination of three factors: symmetric difference between the projected model and the silhouette, an error factor for body parts that are below the floor, and an error factor for violating the constraints we have for the various angles and lengths of body segments. While we were able to obtain some good results, the overall quality is not sufficient to provide good feedback in general. Thus, we discuss some possible ways to improve on our results.

1 Introduction

We will approach the problem of retrieving a 3 dimensional model of a human from a 2D silhouette for the purpose of providing feedback to a beginning yoga student.

1.1 Yoga

Yoga is many things to many people. At its most general, it is a set philosophies and associated practices to achieve certain desirable states of mind. A person who practices yoga is called a yogi. Various forms of yoga focus on different types of practice, but they all aim to provide yogis with mental stability and physical health. Asana is a Sanskrit term which literally means to sit down. However, it is more generally used to mean holding one's body in a position with concentration. Asana is one of the eight categories of yogic practices. Hatha yoga is one form of yoga that initially focuses on introducing yoga philosophy via asanas.

1.2 Alignment in Asana

In this paper, our task is to match a 3D model of a human to a 2D silhouette obtained from an image in order to determine the pose of the person in the image. We aim to accomplish a means of aiding a yogi in their asana practice. Alignment of the body is an important aspect of practicing asanas. However, it is sometimes difficult to judge one's own alignment. Improper alignment can prevent you from fully benefiting from the pose, and some misalignments can even cause injury. When practicing with a teacher, there is a watchful eye on the student to ensure that they safely obtain benefit from the pose. However, to obtain maximal benefit from yoga it is often recommended that one practice at home. Having a system for providing feedback as to their alignment could help the beginning yoga student to safely and effectively practice asanas at home. While this is no substitute for bodily awareness, such awareness can take time to develop. By providing feedback we can make the beginning yoga student

aware of their mistakes. Once aware, they can take appropriate action to correct their posture.

1.3 System Overview

One way of providing such feedback is through a computer vision system. Such a system would have several components. In this paper we address the processing of a single image from a camera in order to create a 3D model of the yoga practitioner. We also will describe displaying the obtained model to provide user feedback. We will start with a binary image where the background is black and the pixels imaging the person are white. We believe that with knowledge of the background one of several background subtraction techniques would provide adequate input to our work, and thus we begin our computations from precomputed silhouettes. Our approach will use Powell's method to perform a minimization to find the joint positions, segment lengths, and the camera calibration. Several constraints are applied during this minimization, as discussed in section 3.3. Capturing this information will enable us to point out dangerous alignments, as well as misalignment that are not harmful, but merely not ideal. With expert guidance, this information could be used to provide the user with feedback to improve their alignment.

1.4 Related Work

Finding the pose of a human is a problem that has been tackled in many ways and for a variety of purposes. While some work has focused on recovering 2D pose from images [Micilotta et al., 2005], others have attempted recovering 3D information [Mikić et al., 2003]. While we attempt to recover 3D information, as in [Mikić et al., 2003], we use only a single image combined with more rigid constraints on the pose, whereas their approach matches pose to 3D voxel data obtained from multiple silhouettes. In both [Micilotta et al., 2005] and [Mikić et al., 2003] individual body segments are identified and pieced together. Our approach is different in this regard. We begin with a rough estimation of the initial pose and use a general minimization technique on the joint angles, segment lengths, and camera parameters in order to obtain our match.

While this technique would not work for a general pose finder, in our case we have a finite number of approximate poses. Even if we didn't know which of these initial poses we were attempting to match, a quick check of the error function for each initial configuration would probably find the correct pose to start from. One other possible strategy for achieving is using a star skeleton match this to quickly identify rough body configuration [Chen et al., 2006]. By looking at the model holistically, we can ensure that all of the segments connect properly, giving us a good approximation of the 30 joint angles that describe a human body configuration. Others have used a similar strategy of comparing possible model configurations to a single image [Lee, Cohen, 2006]. Using single images has been explored in other manners, such as constructing pose from information extracted via image processing techniques [Salti et al., 2008].

The use of a computer vision system to provide feedback about posture is not a novel idea. Efforts have been made to provide a computer user feedback about their posture for health reasons, as well as to improve productivity [Jaimes, 2005]. Like our system, it starts with a silhouette of the user and computes posture based on this. It also has a measure of “goodness” of posture, and provides warning for unhealthy postures. This is similar to the feedback we aim to provide for a yogi.

1.5 Remaining Sections

Section 2 discusses the model, from its basic structure to its constraints and limitations. In section 3, we explain how the matching between a silhouette and the 3D model is performed. Section 4 describes the feedback we provide to the user based on the 3D model we recover from the image. Some example matching results are presented in section 5, followed by discussion in section 6. Possible paths of

future work are outlined in section 7 and conclusions are drawn in section 8.

2 Model

The model is a set of articulated body segments. These segments can be modeled either as line segments or cylinders. Additionally, the display of each segment can be turned on or off individually. The angles of each joint in the model can be modified to put the model in nearly any position that is possible for a human. The origin of our model is the toe end of the right foot segment, and the right foot goes from $(0,0,0)$ to $(0,0,-fl)$, where fl is the length of the foot. From there, everything is computed based on the joint angles (see section 2.5). The camera is positioned relative to this. For most poses, a camera position that provides a good initial view of the model for most poses is $(0.45, 0.4, 2.25)$. However, for other poses, we need something drastically different in the x and y coordinates. Keeping in mind the position of the foot relative to the view we want helps in figuring out the appropriate camera position.

2.1 Segments

The model has made up of 18 segments. Except for the “pelvis” segment that represents the base of the torso, and the “head” segment, each other segment occurs twice, once on the right, and once on the left.

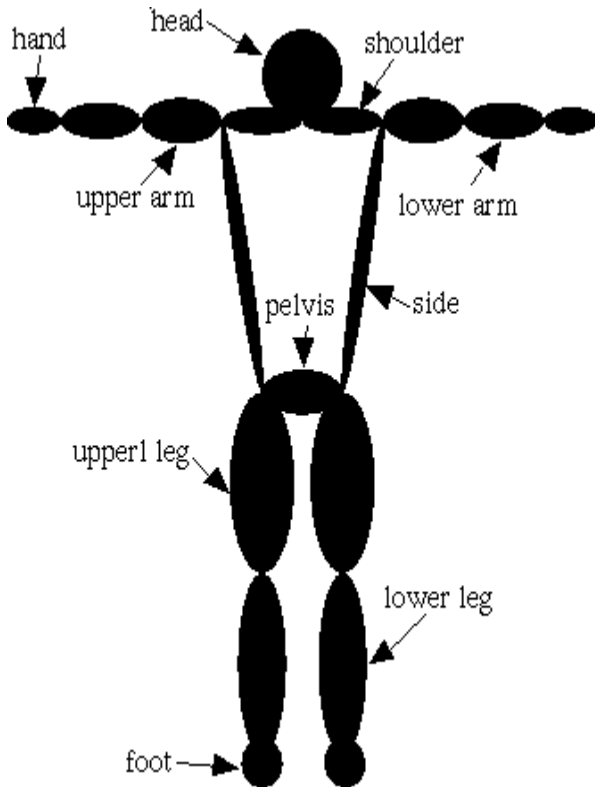


Figure 2.1: Model Segments

This gives us 10 degrees of freedom in our relative segment lengths. The relative length of each segment of the human body varies from person to person. Because of this variation, our model has variable segment lengths. The initial set of segment lengths are based on Leonardo da Vinci's Vitruvian Man. While the Vitruvian Man provides information about overall height and proportions for most of the segments in the upper body, it does not give us information about the length of the sides of the torso or about the length of the leg segments. Estimates were made for the leg segment lengths based on comparison to the known lengths of the arm segments. Combining these estimates with the total height specified by da Vinci, we were able to compute the approximate length of the the sides of the torso. While this provides us with an initial estimation of the various segment lengths, during the actual matching, these lengths are optimized. We leave the shoulder length constant, as the lengths are relative to each other. This allows us to remove one dimension from our minimization. Reducing

the dimensions contributed by the segment lengths to 9.

2.2 Constraints

Our model has two different kinds of constraints. The first of these are joint angle and segment length constraints. The human body has physical limits to how far each joint can bend. These constraints are taken into account when rating a match, and results that violate them are penalized. The constraints are fairly loose in that they allow for more range of motion that the average person can perform. However,

they are intended more as a sanity check than anything. Table 2.1 shows the angle constraints placed on each of the joints. The segment length constraints are calculated based on the starting segment lengths. The minimum is half of the starting length, and the maximum is twice the starting length. Again, these are very loose constraints, but they penalize any obviously wrong model configurations. The second type of constraint is a floor constraint. This constraint penalizes the model for having a point below the floor. Each time a segment is drawn, we check to see if its endpoint goes below $y = 0$. If it does, we increase the total floor error. See section 3.1 for details on how these constraints are used to modify the error function used in our minimization routine.

Joint	Angle	Minimum	Maximum
Neck	1	-100	100
	2	-100	100
Shoulder	1	-100	100
	2	-150	60
	Internal	-100	100
Elbow	1	-10	150
	Internal	-190	100
Wrist	1	-30	50
	2	-100	100
Hip	1	-60	150
	2	-50	190
	Internal	-100	100
Knee	1	-10	170
	Internal	-30	30
Ankle	1	-60	90
	2	-30	30

Table 2.1: Angle Constraints

2.3 Model Limitations

The model currently models the torso using five segments that are fixed in position relative to each other. While this provides a good model for many Yoga poses, it eliminates a very important class of poses, those that involve twisting of the torso. To accurately model these poses would require a much more complicated torso model and is beyond the scope of this paper. Additionally, due to the way we limit the camera parameters, both feet must be flat on the ground. However, even with our simplified model, we can match several common poses, including the majority of those poses known as “standing poses.”

2.4 Computer Representation

The model is represented as an array of 46 floating point numbers. The first 30 numbers are the angles for the various joints. The next 9 numbers represent the relative lengths of the body segments. Finally, there are 7 parameter for specifying the projection model for the camera.

2.5 Drawing the Model

The basic unit of our model is the segment. As such, we have a function for drawing segments that takes a radius and an enum value representing the segment. This function then draws the segment as a cylinder, ellipsoid, or line segment. The radius argument is ignored for the line segment. Additionally, drawing the model as line segments uses colors to indicate “correctness” as specified in Section 4-Feedback. For our tests, we used ellipsoids as they generally provided better matches than cylinders. This is most likely because our body segments are more elliptical than cylindrical. The segment is drawn from $(0,0,0)$ to $(0,0,segment)$, where *segment* is the length of the segment. After drawing the segment, `glTranslate` is called to move our origin to $(0,0,segment)$ in preparation for drawing the next segment. Following this, we check to see if our new point is below the floor (the y value is less than 0). If it is, we increase our floor error value (see section 3.3).

We begin drawing with the right foot. Thus, the toe end of the right foot segment is at $(0,0,0)$ in world coordinates. We proceed drawing the lower leg followed by the upper leg, performing appropriate rotations based on our specified joint angles. We draw the pelvis, then push the current matrix. This

allows us to come back to the left hip after we draw the left leg. We draw the left leg one segment at a time, starting with the upper leg and ending with the foot. At this point, due to our various rotations, the left foot may not be on the ground. A correction angle θ is calculated that will bring the left foot to the ground. At this point, we clear the buffer. We rotate θ degrees around the z-axis and then redraw the legs. This time, both feet will be on the ground, with $y = 0$ being the plane of the floor and the camera's up vector as $(0,1,0)$. This allows us to preserve our limited dimensionality of our camera model.

After drawing the left leg, we pop the matrix stack so we are back at the left hip. We draw the left side, using a side angle parameter calculated from the segment lengths. We again push the current matrix, as we are now at the left shoulder. We draw the left arm from upper arm to hand, before popping the matrix stack back to the left shoulder. From here, we draw the left shoulder segment and push the base of the neck. The head is drawn, and we pop back to the neck, and draw the right shoulder. Next, the right arm is drawn mirroring the left arm. Finally, the right side is drawn, again using the calculated side angle and the model is complete. All that appear on both sides are oriented so that they mirror each other. This allows us to take a pose and swap all the left-right angles and get the corresponding pose for the other side of the body. However, we will often need to adjust the camera to acquire an appropriate view.

3 Matching

For our algorithm, we assume we have as input a silhouette of the person, which can be obtained from any number of available background subtraction techniques (cite examples). From this silhouette and knowledge of which pose is being attempted, we proceed to match our 3D model to the 2D silhouette, which will enable us to identify the angles of the major joints of the body. The first step in this matching is to make a rough estimate as to the position of the camera. Following this we invoke Powell's method (Section 3.2) on a 46 dimensional formulation of our problem. 30 of these dimensions are the joint angles we are looking for. There are 9 dimensions for the lengths of the various segments. The remaining 7 dimensions are the camera parameters we need to accurately project our model.

3.1 Camera Estimation

We perform a two step process to adjust the model so that it approximately lines up with the image. In the first step, we align the center of mass of the image with the center of mass of the projected model. We then perform a similar step, with only the torso showing. The first step allows us to get fairly close, while the second step should provide a decent match for the torso. Additionally, in the second step, we perform a minimization on the side and pelvis segment lengths, and optionally the scale. This is done as a lower dimensional version of our main search method (Powell's).

We will explain how the alignment is done using the second step as an example as it is slightly more complicated. Beginning with our model, we project the torso using a predefined camera that depends on the pose we are investigating. From this projection, we calculate the center of mass of the torso (Figure 3.1). We then AND this projection with the silhouette. The center of mass of this new image is then calculated (Figure 3.2).

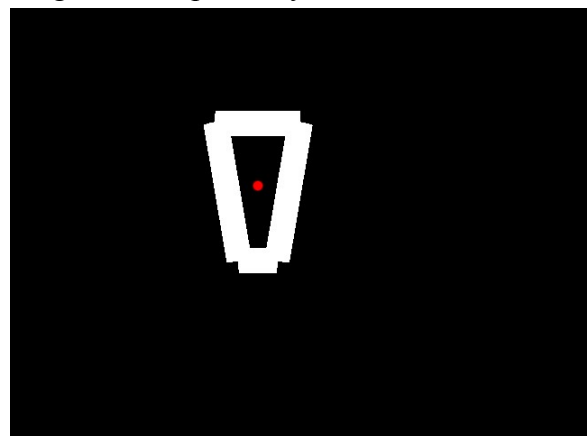


Figure 3.1: Torso Projection with Center of Mass

We then attempt to align the center of mass of the two images by adjusting the camera position. The new center of mass will be closer to the real center of mass for the torso than the original projected center of mass. We can move to projection's center of mass toward this new center of mass by moving the camera in the plane parallel to the image plane in the direction specified by the vector obtained by subtracting the projected center of mass from the new center of mass. Because of the limitations on our camera, we know a change in y in the image can be accomplished by a change in y in the camera as it's line of sight is always level. For a change in x in the image, we can calculate the appropriate direction as the

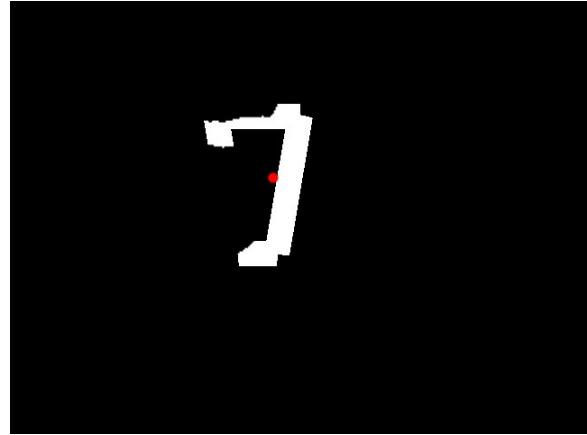


Figure 3.2: AND of Torso and Image with Center of Mass

negative inverse of the line from the camera to the point it is looking projected onto the floor. This will give us a line through the $y = 0$ plane that is parallel to the image plane. Using this, we can move the camera and the point we are looking at in the appropriate direction.

In the first step, we calculate the centers of mass from the projection of the whole model and the image, whereas in the second step, we use the projection of the torso and the AND of this projection with the image. We continue to adjust by small increments so long as the symmetric difference between the image and the projection continues to decrease. Additionally, during each iteration of the second step a 3-4 dimensional minimization is done. The lengths of the pelvis and side segments are optimized. Also, the scale may be included depending on command line arguments.

3.2 Minimization with Powell's Method

Powell's method for multidimensional minimization is a common techniques for multidimensional minimization when no derivative information is available. The basic idea of this method is to begin with a set of directions in the multidimensional space. Following this, the error function is minimized along each direction sequentially. One pass through each of the directions gives us a new direction, which is the combination of how we moved in the other directions during that step. We use this new direction to replace the direction that gave us the largest decrease from this iteration. This direction will be a major component of our new direction, so replacing it will help keep the directions from becoming linearly dependent, which would prematurely reduce the dimensionality of our problem. We repeat this process as long as our error function continues to decrease [Press, et. al. 1992].

For our problem, we use the identity matrix as our initial set of directions. This gives us one direction for each angle, segment length, and camera parameter. Our error function consists of three components. First, we have the area of the symmetric difference between the silhouette and the projected model. This gives us a good overall measure of how close the model and the silhouette match. Second, we have a floor constraint, which penalizes the model if any of the segments pass through the floor, as defined by the position of the foot segments. Finally, we have joint angle and segment length constraints, which penalize the model if the angle goes beyond what is physically possible for a given joint or the segment lengths go out of their bounds.

3.3 Error Function

The error function is computed as a linear combination of three factors: the symmetric difference

between the projection of our model and the image, an angle and segment length error, and a floor error. The symmetric difference parameter is simply is the number of white pixels in the symmetric difference of the image and the projection. It is unweighted. The other two error values have weights that can be specified at runtime. The angle and segment length error is calculated as follows:

```
asl_error(values, min, max):
    error = 0
    for i in range(0..len(values)):
        if values[i] < min[i]:
            error += (1.0 + abs(min[i] - values[i]))8
        if max[i] < values[i]:
            error += (1.0 + abs(max[i] - values[i]))8
    return error
```

The floor error is calculated in a similar manner, except there is no maximum value and the minimum value of the y coordinate is always 0. The error function calculates the three values and returns

```
symmetric_difference + asl_weight*asl_error +
floor_weight*floor_error
```

4 Feedback

Our system provides feedback to the user in the form of a color-coded stick figure representation of our model as shown in Figure 4.1. The color of each joint is interpolated based on the cumulative error from all the angles at that joint. This error is then bounded and normalized to produce a value between 0 and 1, with 0 representing no error, and 1 representing maximum error. 0 error is mapped to green, and an error of 1 is mapped to red. Intermediate values are interpolated linearly in the RGB color space. Thus, green indicates that the angles are close and red indicates a misalignment. For each joint the color is calculated as follows:

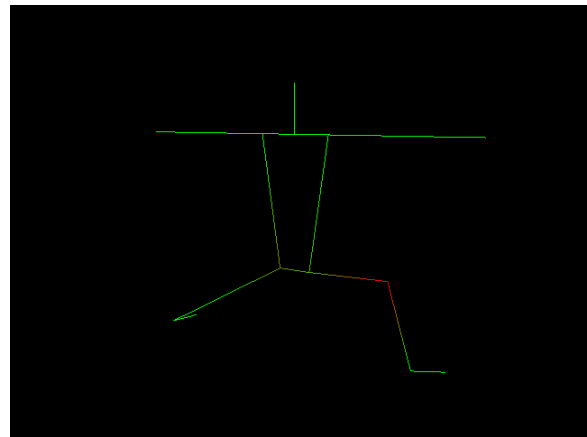


Figure 4.1: User Feedback

```
get_color(joint):
    error = 0.0
    for angle in joint.angles:
        error += abs(model[angle] - pose[angle])

    if len(joint.angles) > 0.0
        error /= len(joint.angles)
```

```
if error > 10 degrees:
    error = 10.0

error /= 10.0

# red
color[0] = error
# green
color[1] = 1.0 - error
#blue
color[2] = 0.0
return color
```

4.1 Comparing Match to Ideal

The error is based on the ideal version of the pose which is used as the starting point for matching. While even a yogi with many years of experience may not be able to fully realize this ideal, it is the goal of many forms of yoga to achieve perfect posture in each pose. As such, we feel it is useful to set the bar high. However, there are often a series of progressively more difficult variations of poses that one can practice to help then achieve the full pose. When using our method in the context of aiding a yogi in their practice, such a sequence could be used as intermediate goals leading up to the ultimate goal.

4.2 Types of Feedback

There are a couple of possibilities for feedback beyond the right/wrong determination outlined. For example the system could analyze the angles and provide warnings if the user is in a position that can lead to damage. An example of such a situation that anyone who has done yoga should have been warned about is allowing your knee to go past your ankle in most standing poses. This can lead to knee problems if not corrected. Additional, feedback could be given about left-right asymmetry. If the user is able to perform the pose better on one side, it is suggested that they perform the pose three times, starting and ending with the weak side [Coulter, 2001]. We could also determine when the imbalance has been overcome and inform the user so that they can resume working both sides evenly. Both of these types of feedback can be produced by examining the output of our matching without any complex calculations.

5 Results

Tests were run on 17 test images covering six different poses. For each image, matching was done using 49 different combinations of parameters. Various weights for the two types of constraints were explored, as well as to test if scale optimization should be included in our prematch alignment. Below we will explore several matches, both successful and unsuccessful. The six poses that we attempted to match are chair pose (Figure 5.1), forwardbend (Figure 5.2), forwardbend-wide (Figure 5.3), sideangle (Figure 5.4), triangle (Figure 5.5), and warrior 2 (Figure 5.6).

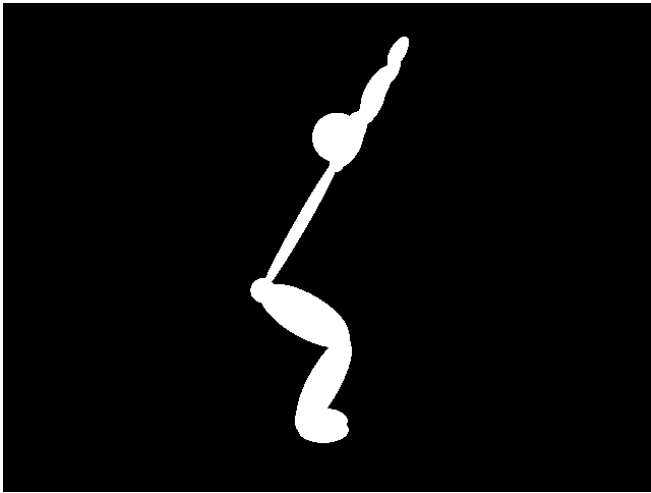


Figure 5.1: Chair Pose

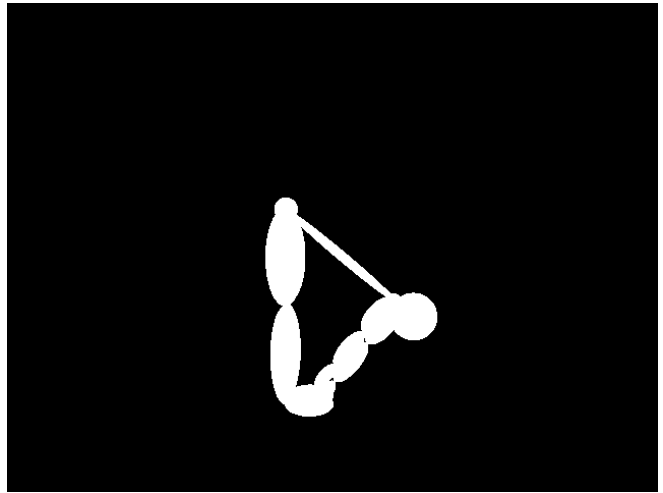


Figure 5.2: Forward Bend

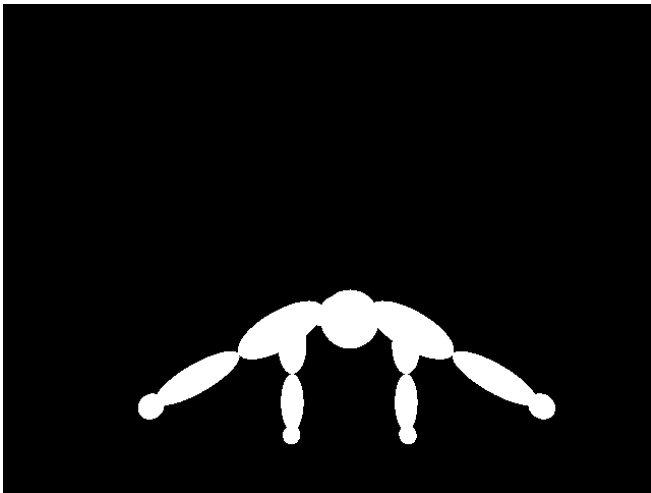


Figure 5.3: Forward Bend (Wide)

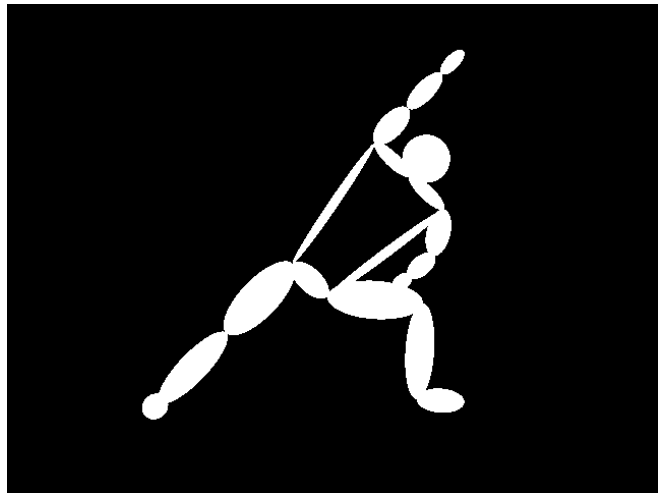


Figure 5.4: Side Angle Pose

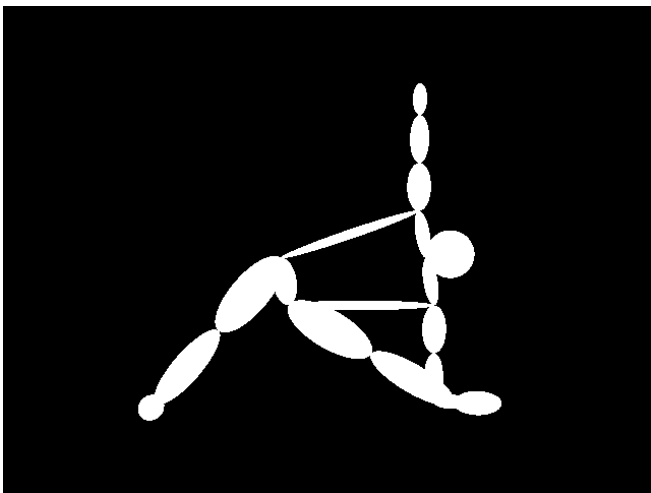


Figure 5.5: Triangle Pose

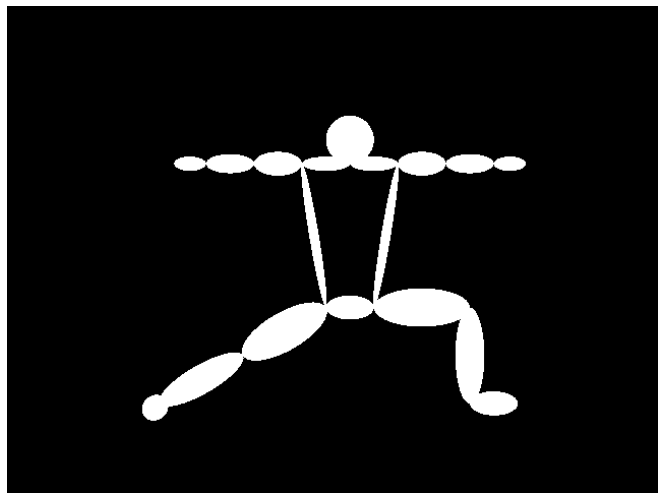

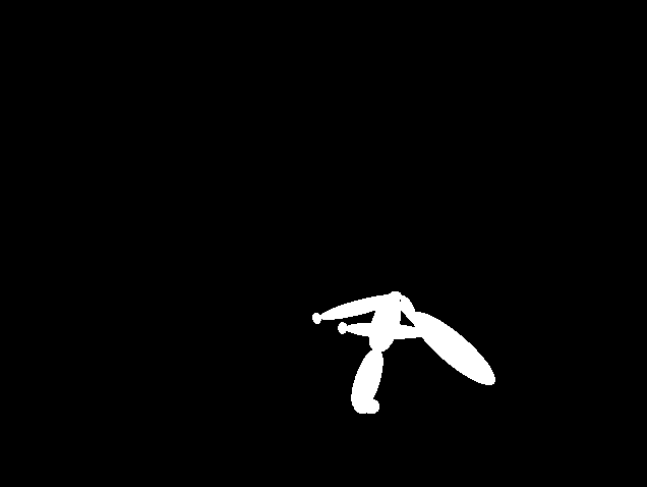
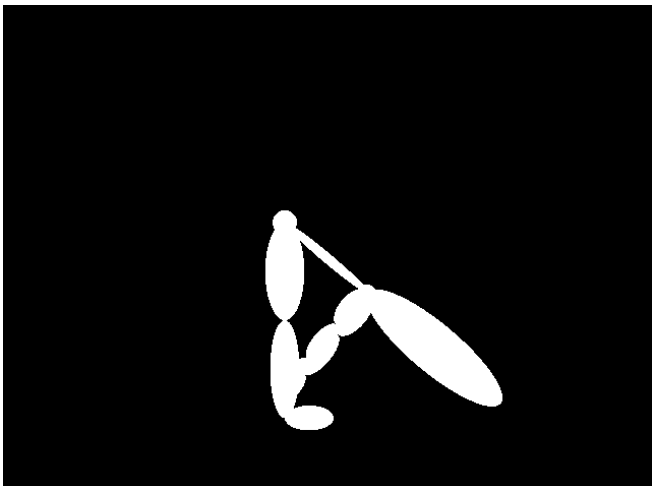
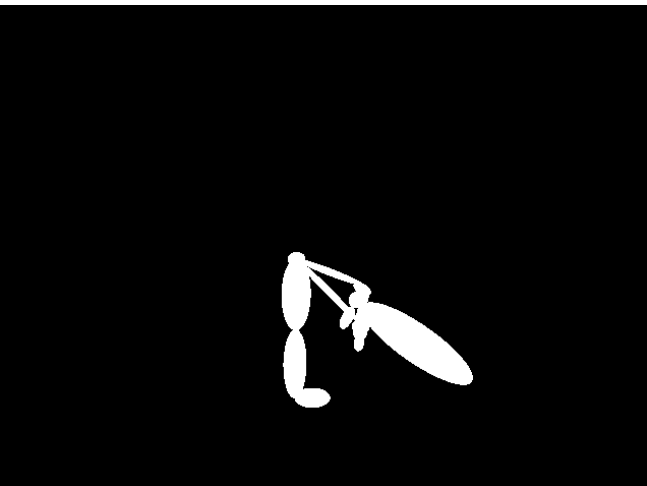
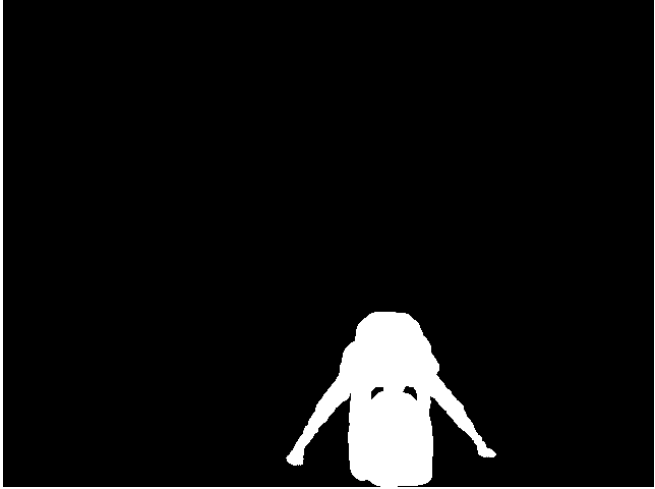
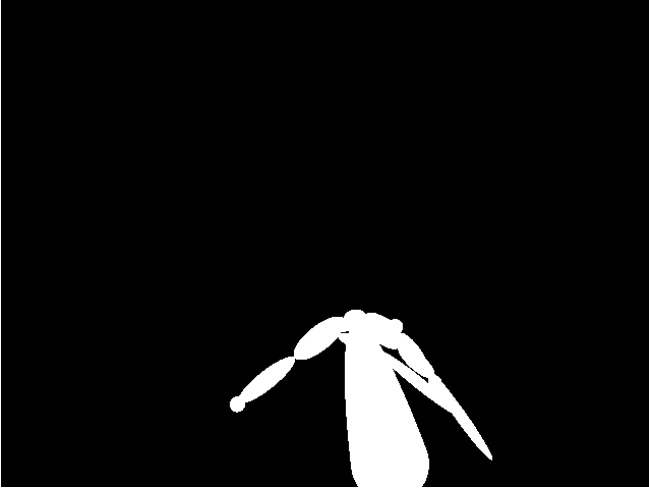
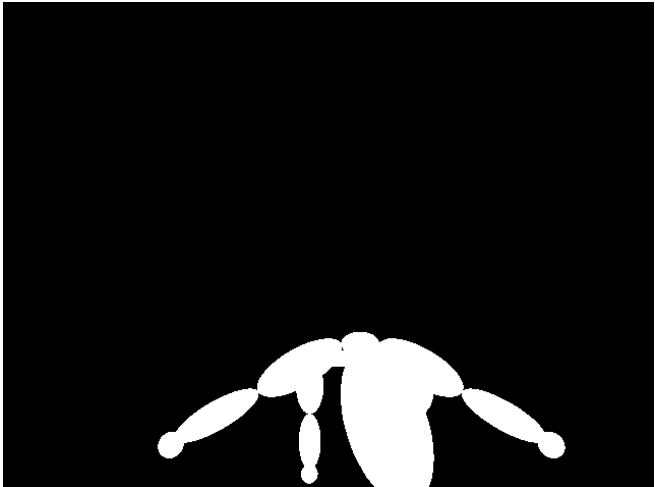


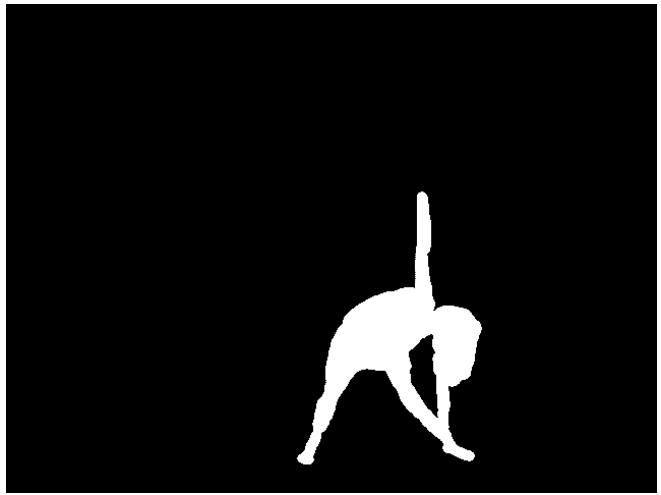
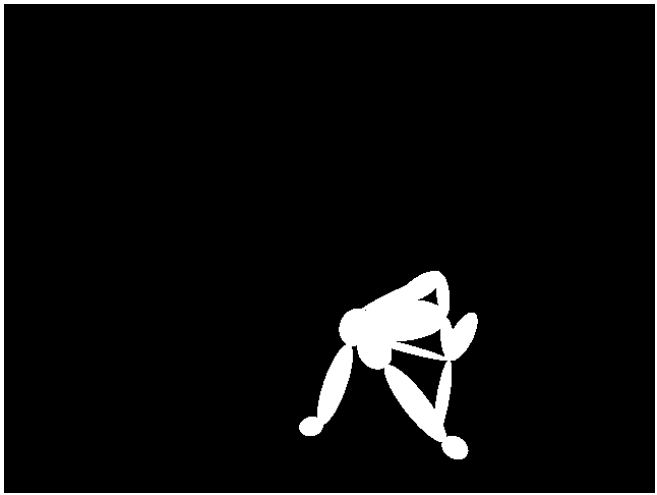
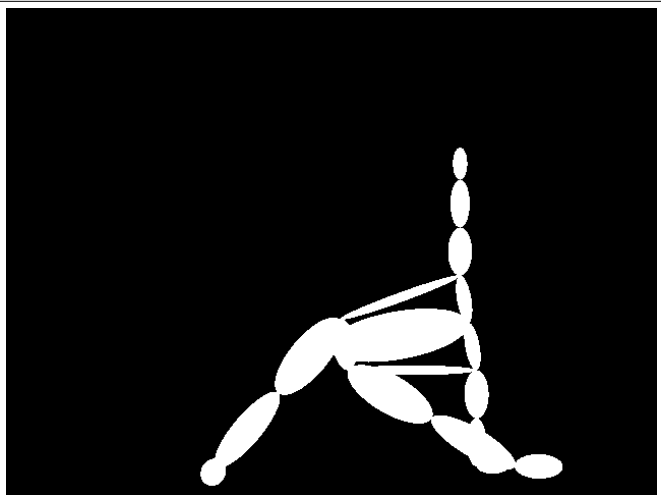
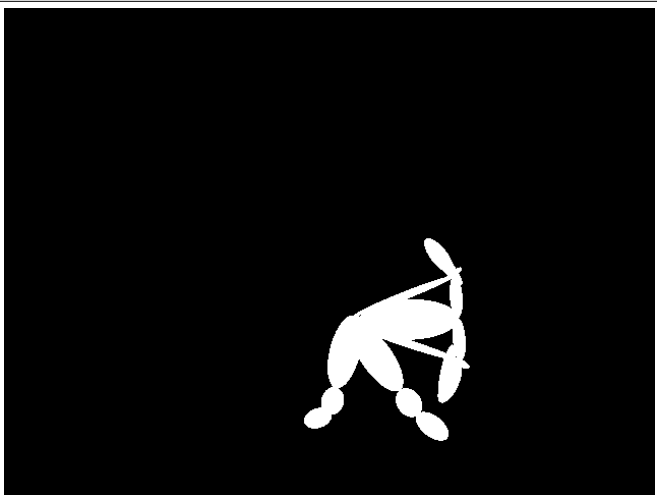
Figure 5.6: Warrior 2

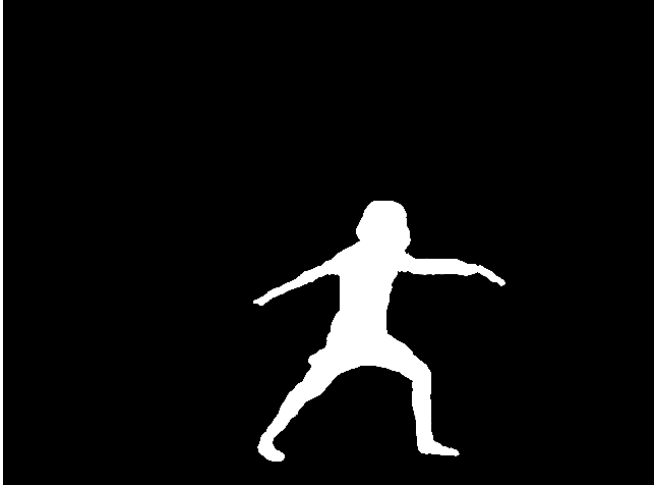
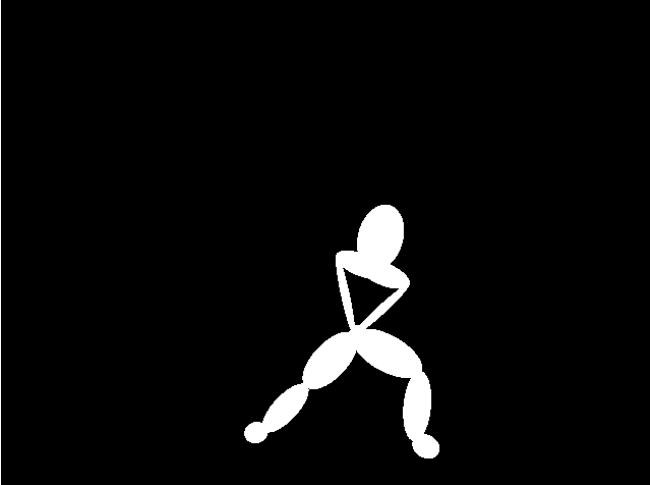
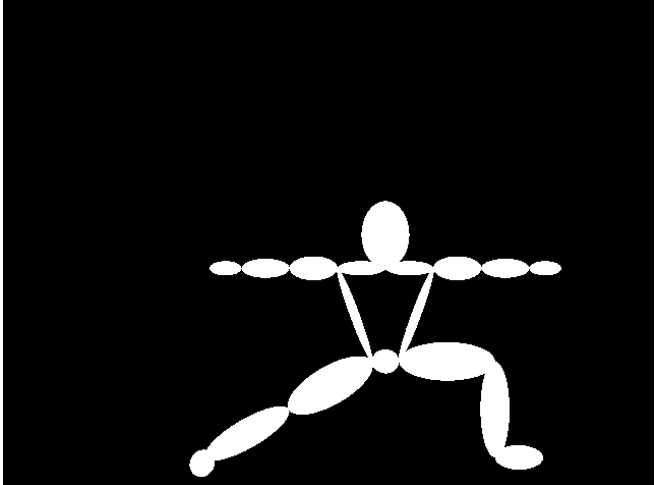
For each image, we will show the input silhouette, the best match determined objectively, the best match determined subjectively, and the aligned model that was the input to Powell's method for the


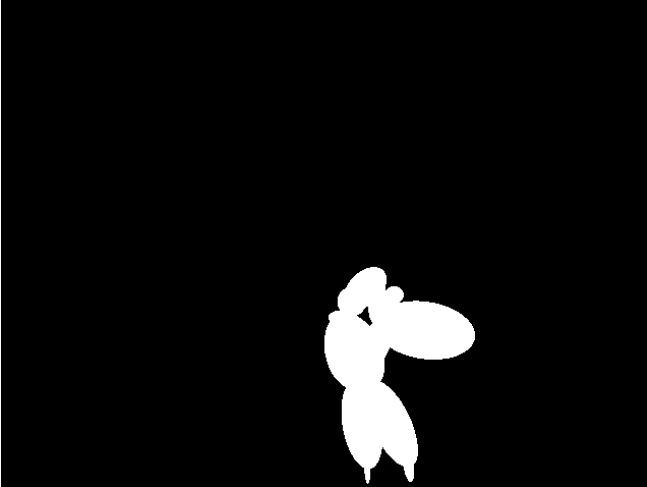
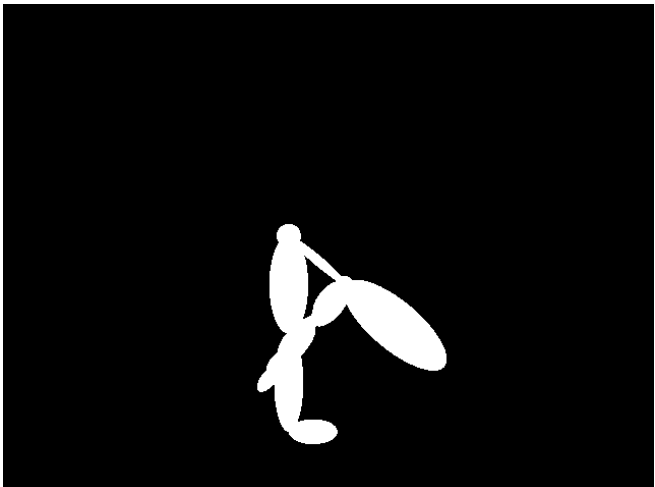
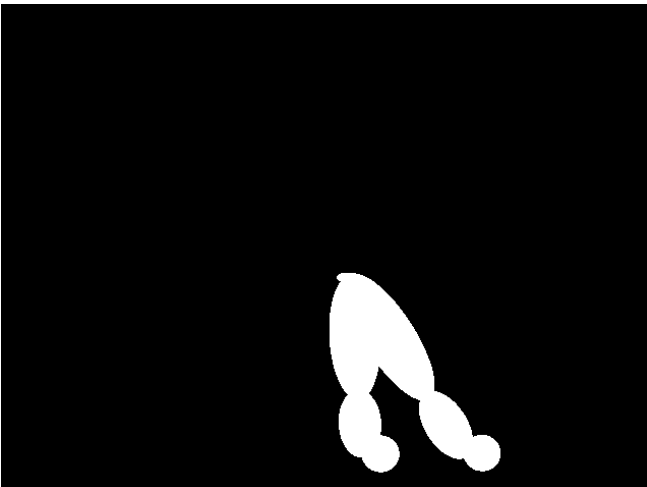
best subjective match. Only the segment length constraints affect the alignment. For the objectively best match, we use the match with the lowest error as computed by our error function with an *asl_weight* of 0 and a *floor_weight* of 0. Thus, we are simply computing the area of the symmetric difference between the image and the projected model. The subjectively best match was chosen based on several factors. First, good leg alignment was preferred as this is the most important part of many poses. Second, when matches had similar leg alignment, arm and head alignment was considered. These decisions were based not on the ideal pose, but how closely they represented the pose as performed in the image. The aligned model was computed as described in section 3.1, using the parameters for the best subjective match. Only the segment length constraints should effect this computation.


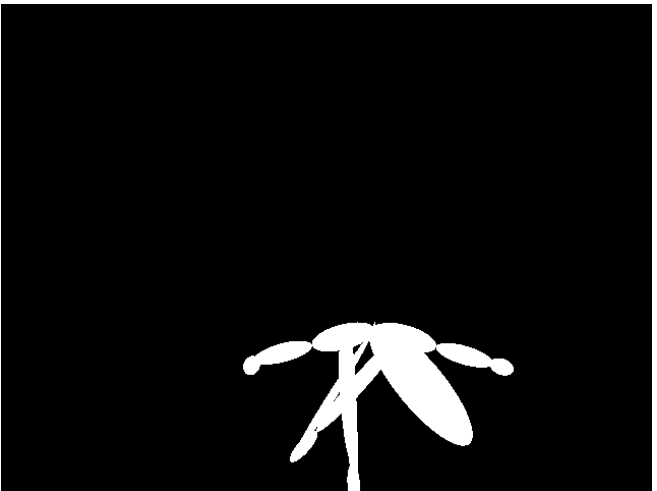
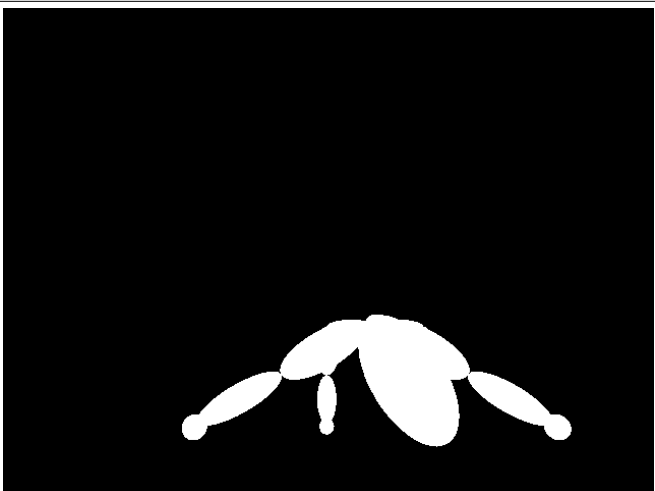
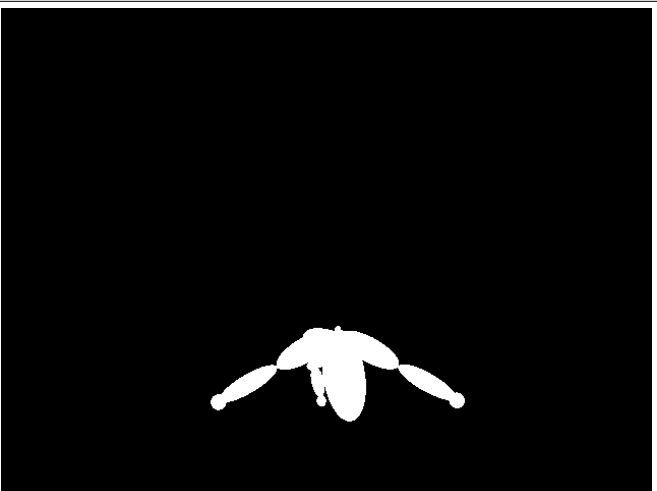
Child 1	Forward Bend
Image	Objective Match
	
asl_weight = 25000, floor_weight = 3500	
Aligned Model	Subjective Match
	
asl_weight = 20000, floor_weight = 2500	

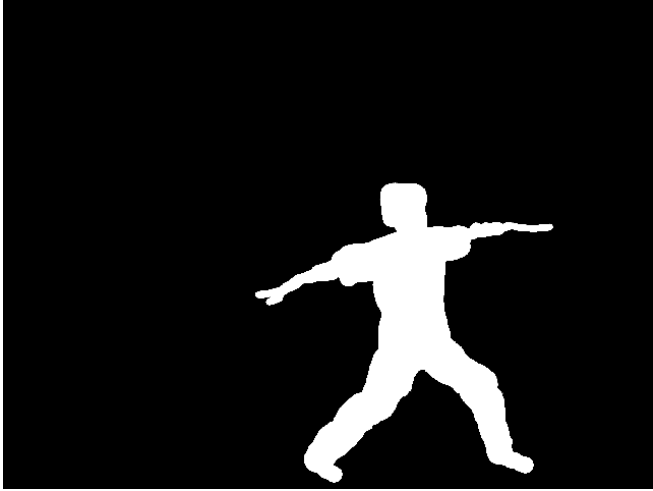
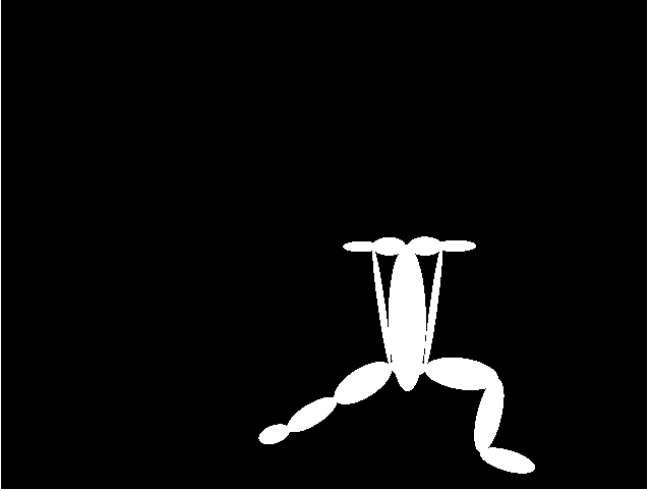
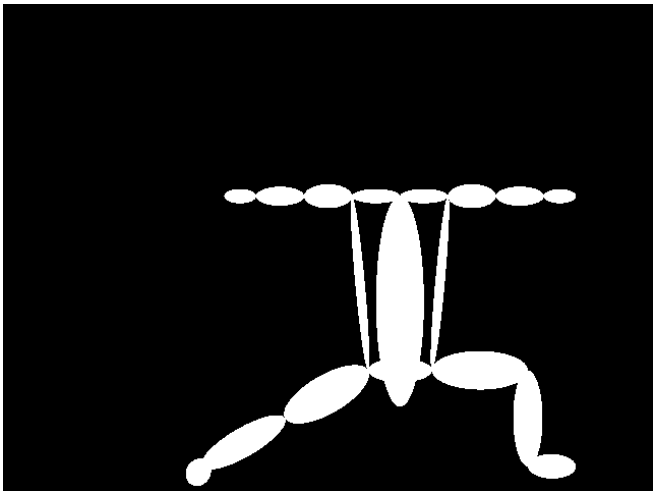
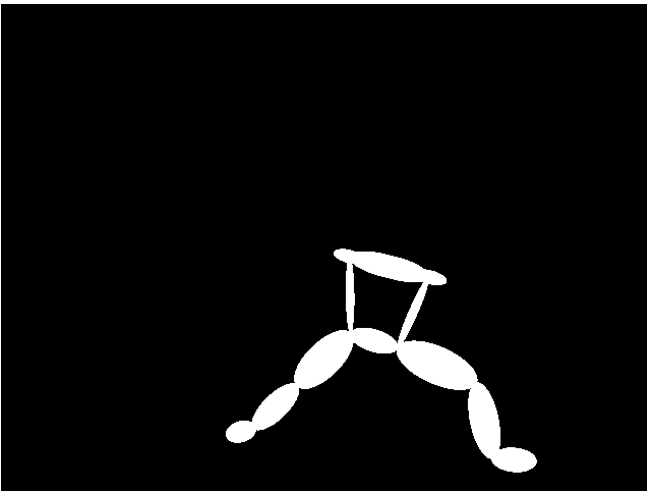
Child 1	Forward Bend (Wide)
Image	Objective Match
	 <p data-bbox="818 751 1349 785">asl_weight = 10000, floor_weight = 1500</p>
Aligned Model	Subjective Match
	<p data-bbox="818 846 1146 882">Same as Objective Match</p>

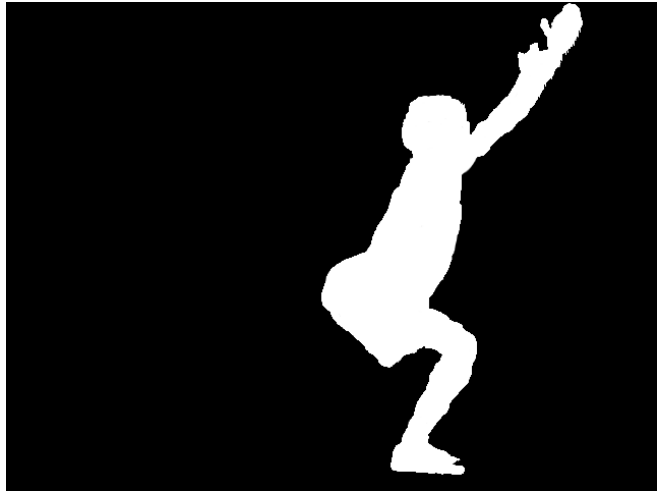
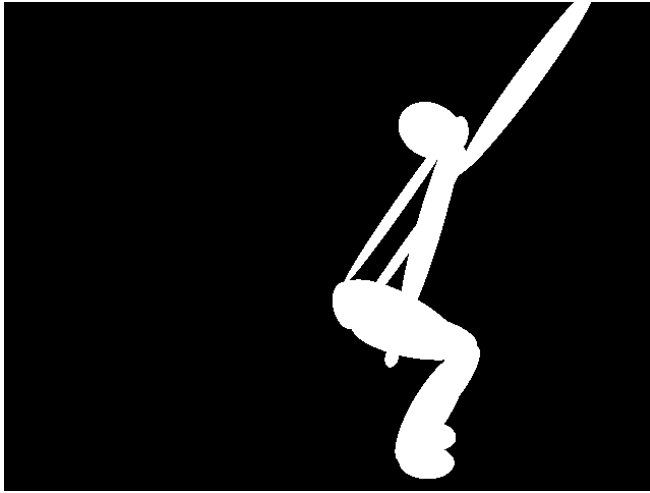
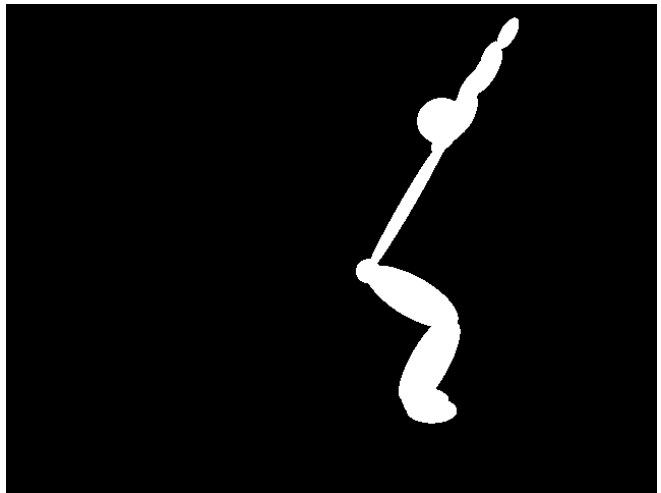
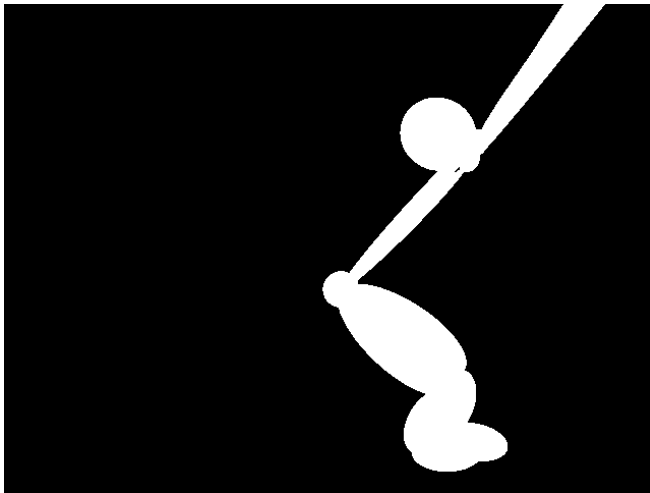
Child 1	Triangle Pose
Image	Objective Match
	
	asl_weight = 25000, floor_weight = 2500
Aligned Model	Subjective Match
	
	asl_weight = 20000, floor_weight = 2000


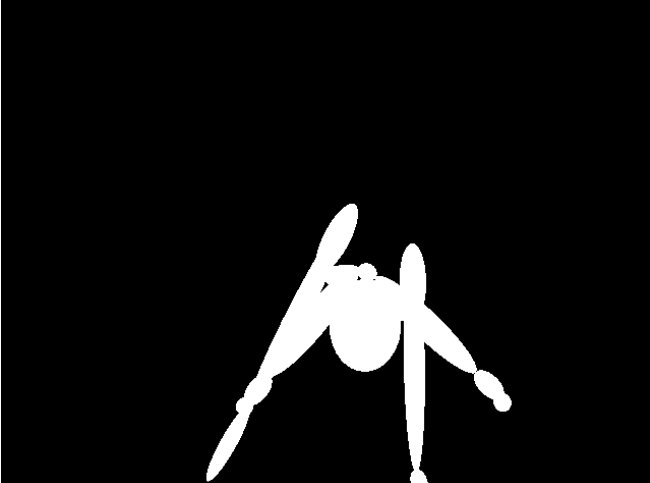
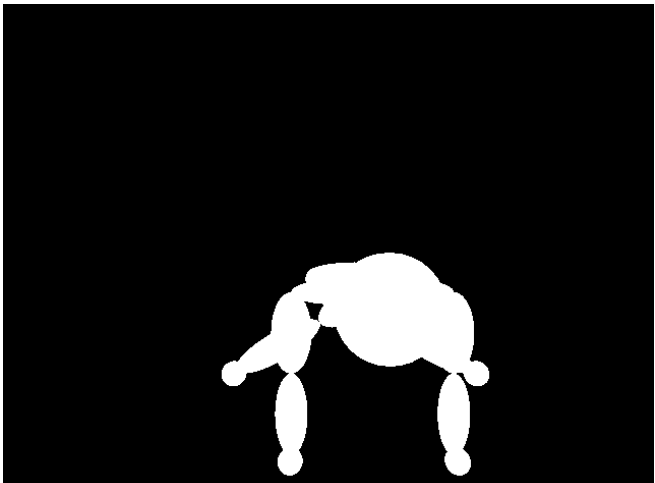
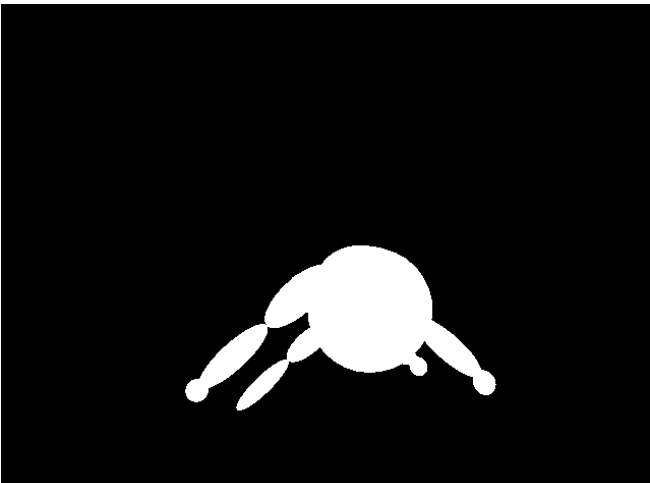
Child 1	Warrior 2
Image	Objective Match
	 <p data-bbox="818 747 1464 787">asl_weight = 0, floor_weight = 1000</p>
Aligned Model	Subjective Match
	<p data-bbox="818 848 1464 888">Same as Objective Match</p>

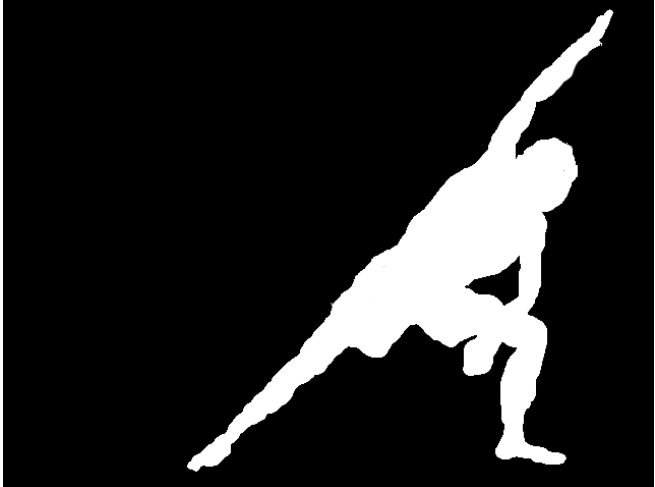
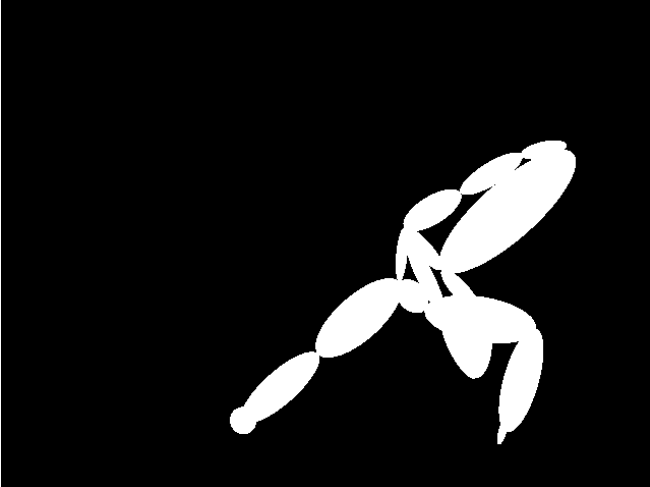

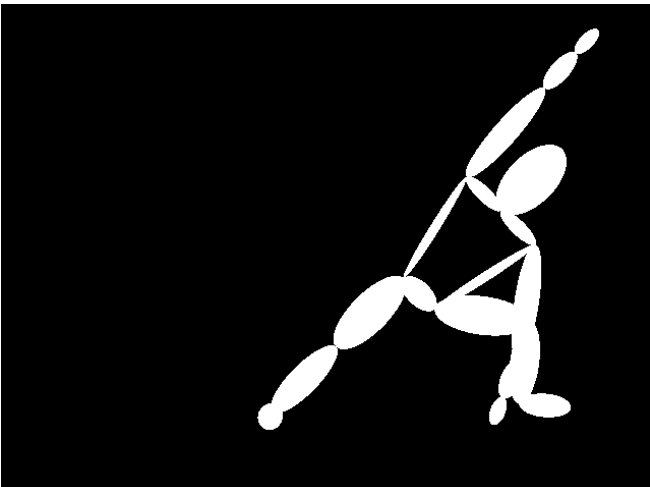
Child 2	Forward Bend
Image	Objective Match
	 <p data-bbox="818 747 1461 785">asl_weight = 0, floor_weight = 2500</p>
Aligned Model	Subjective Match
	 <p data-bbox="818 1331 1461 1369">asl_weight = 20000, floor_weight = 1500</p>

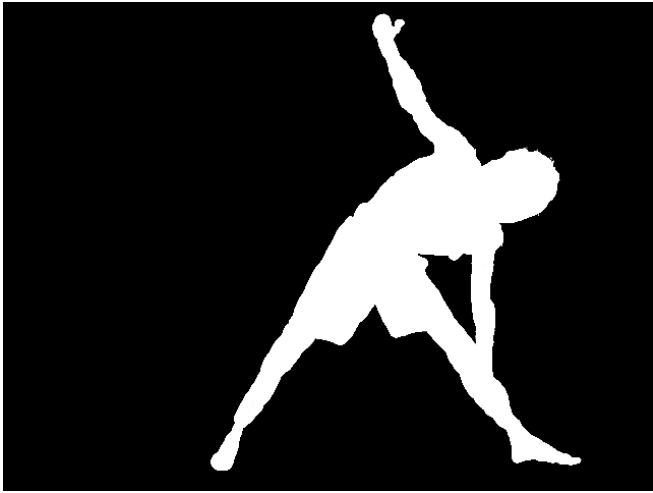
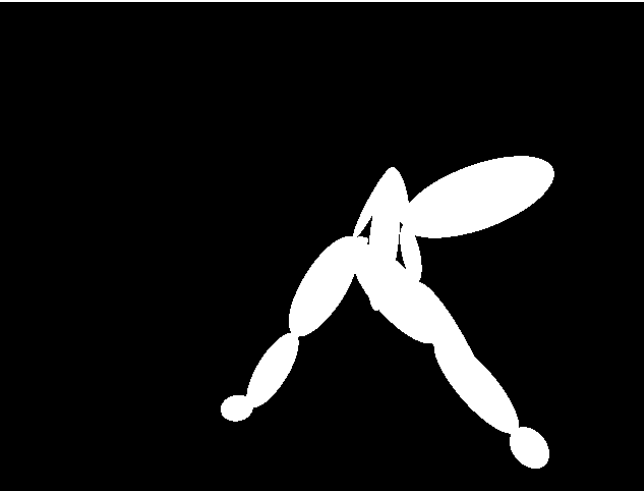
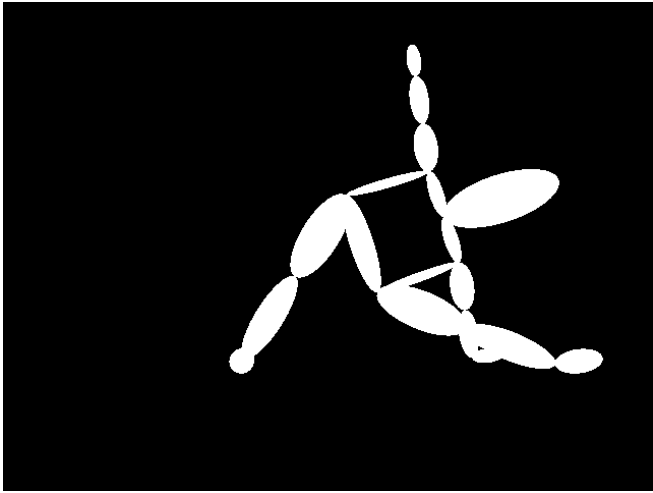
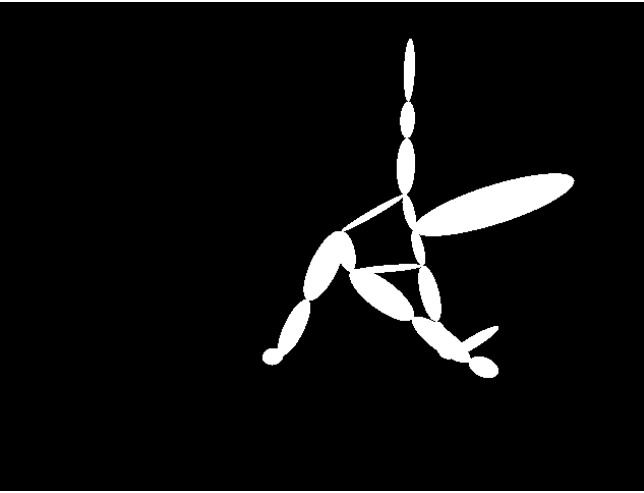
Child 2	Forward Bend (Wide)
Image	Objective Match
	 <p data-bbox="818 747 1471 785">asl_weight = 0, floor_weight = 0</p>
Aligned Model	Subjective Match
	 <p data-bbox="818 1331 1471 1375">asl_weight = 20000, floor_weight = 3000</p>

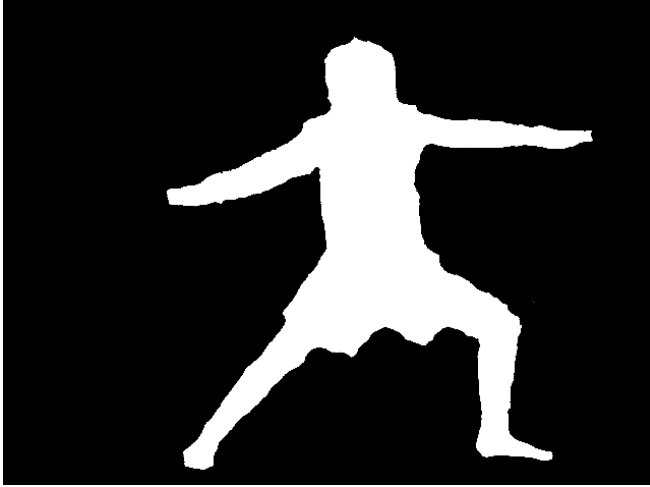
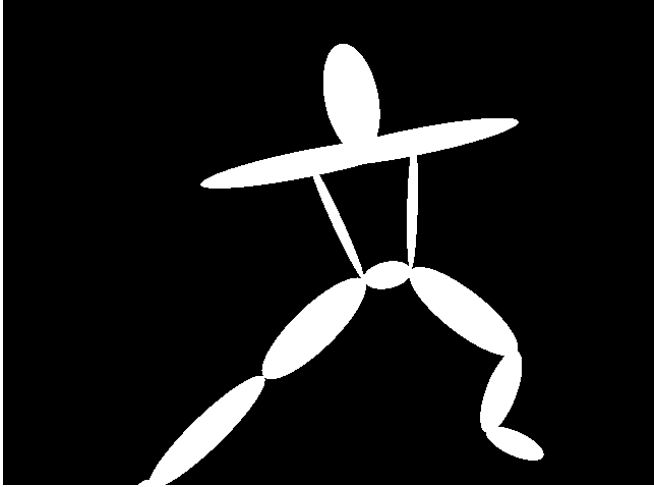
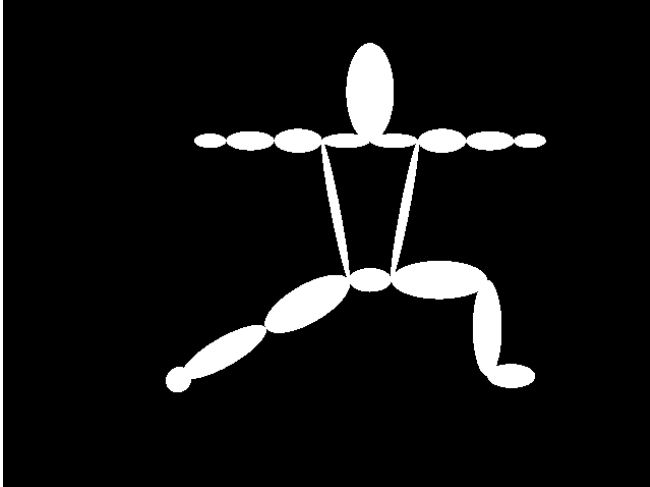
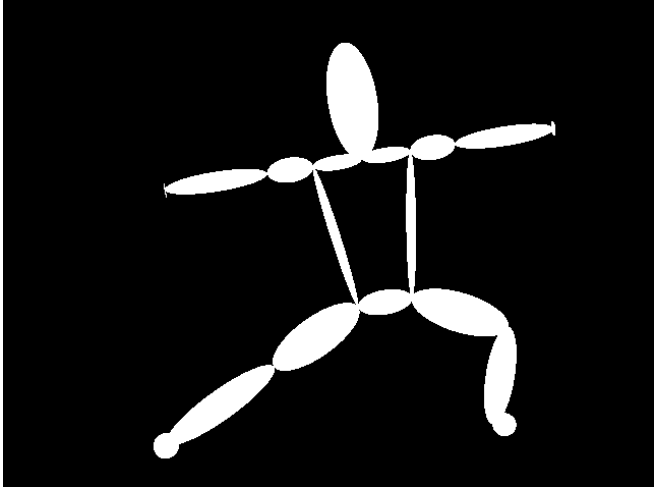
Child 2	Warrior 2
Image	Objective Match
	 <p data-bbox="818 747 1464 785">asl_weight = 0, floor_weight = 500</p>
Aligned Model	Subjective Match
	 <p data-bbox="818 1331 1464 1369">asl_weight = 10000, floor_weight = 1500</p>


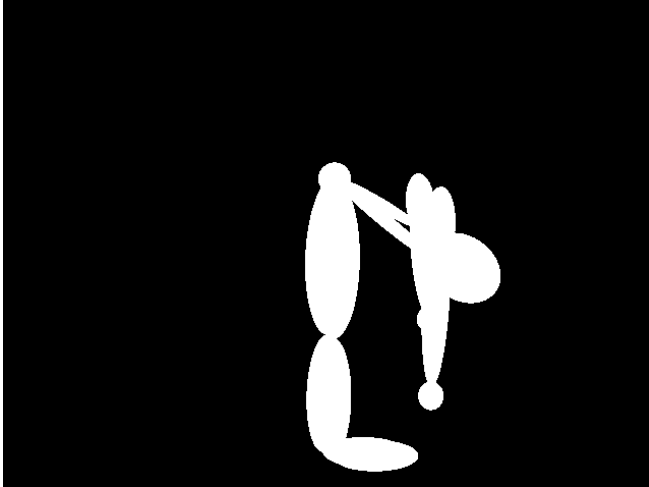
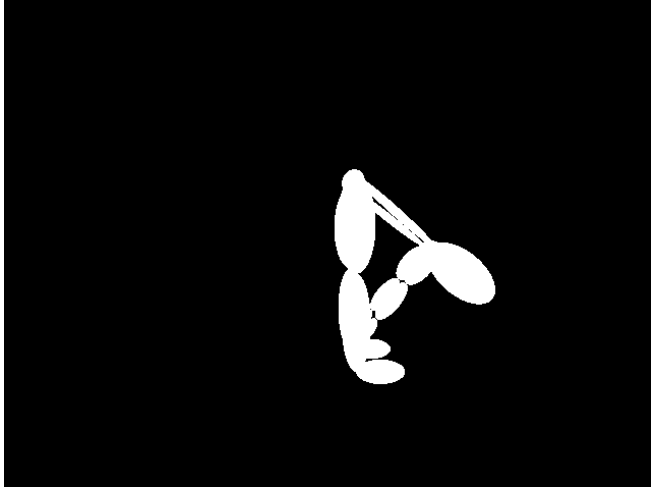
Adult 1	Chair Pose
Image	Objective Match
	
	asl_weight = 1000, floor_weight = 1000
Aligned Model	Subjective Match
	
	asl_weight = 20000, floor_weight = 2500

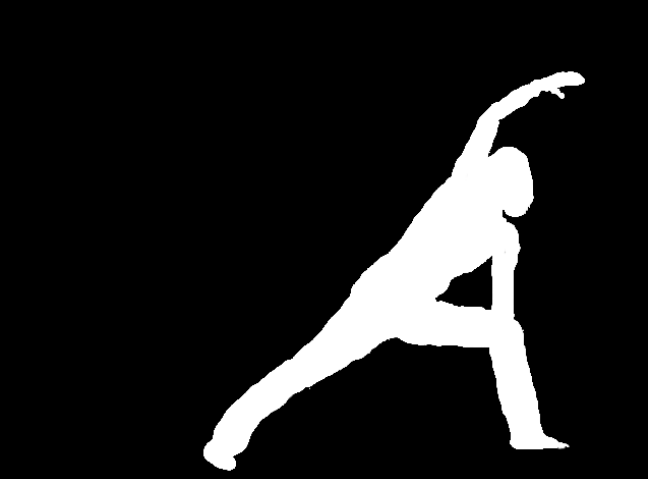
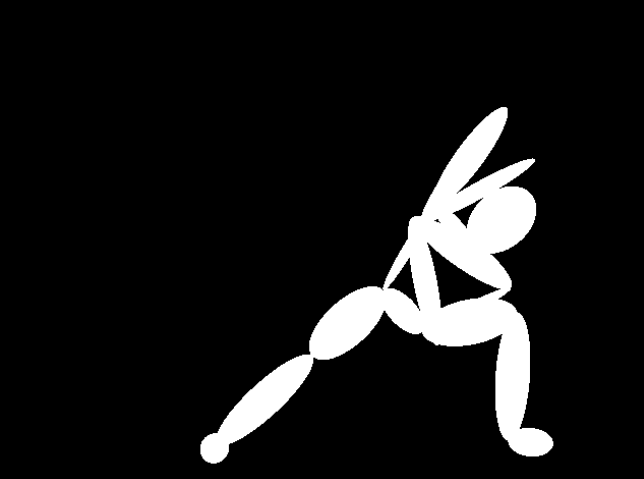
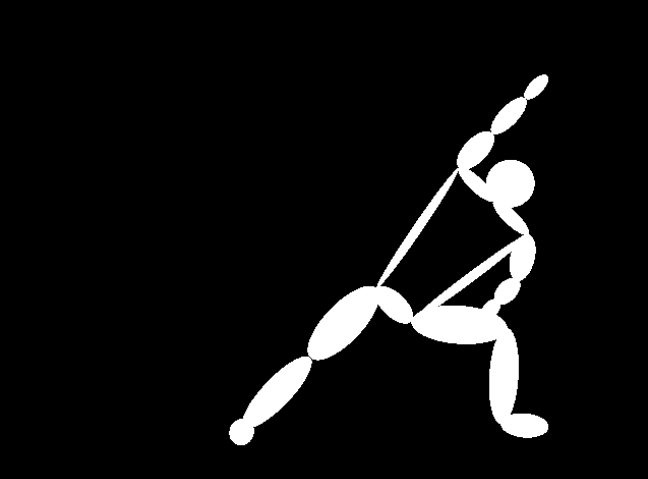
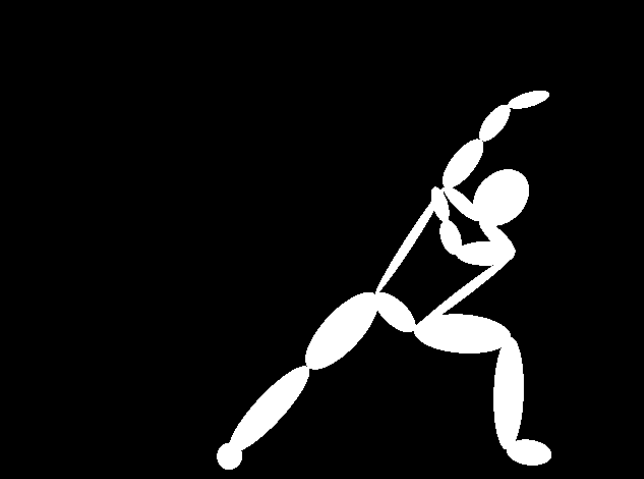
Adult 1	Forward Bend (Wide)
Image	Objective Match
	 <p data-bbox="818 747 1351 785">asl_weight = 15000, floor_weight = 2500</p>
Aligned Model	Subjective Match
	 <p data-bbox="818 1335 1351 1373">asl_weight = 25000, floor_weight = 1500</p>


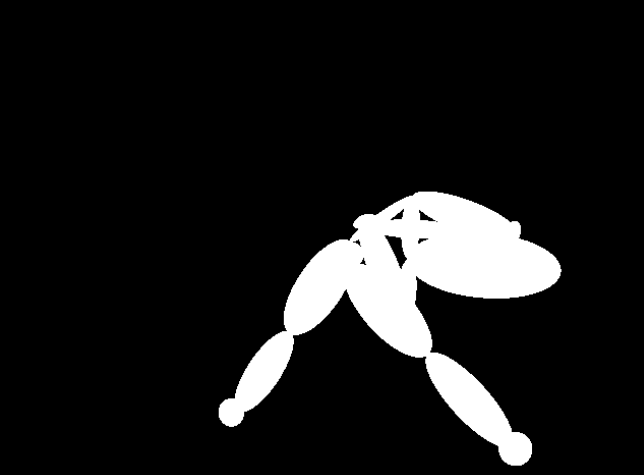
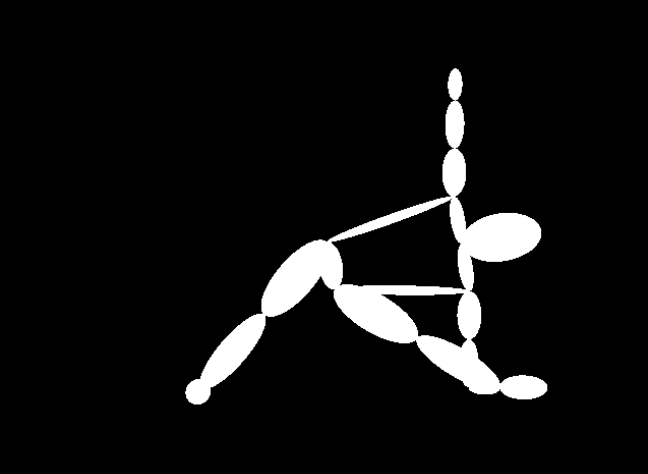
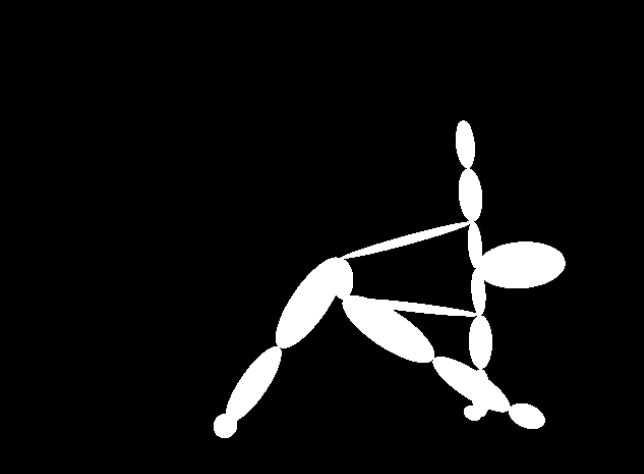
Adult 1	Side Angle Pose
Image	Objective Match
	 <p data-bbox="818 747 1464 785">asl_weight = 0, floor_weight = 1</p>
Aligned Model	Subjective Match
	 <p data-bbox="818 1331 1464 1369">asl_weight = 20000, floor_weight = 2500</p>

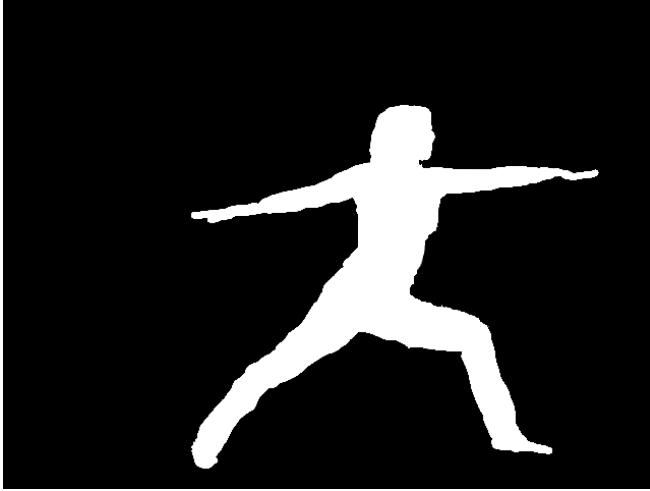
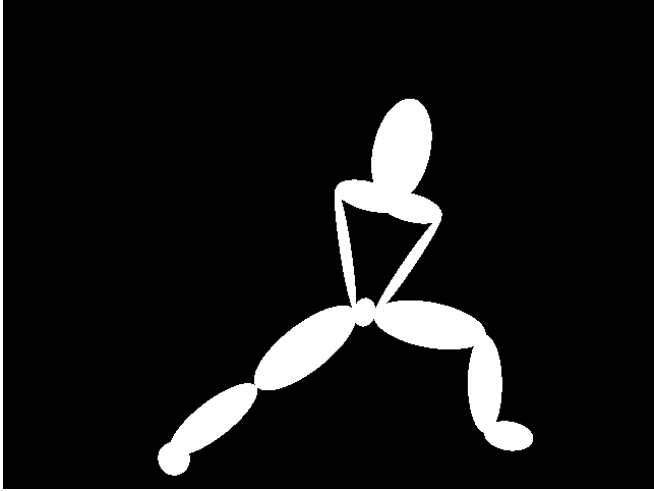
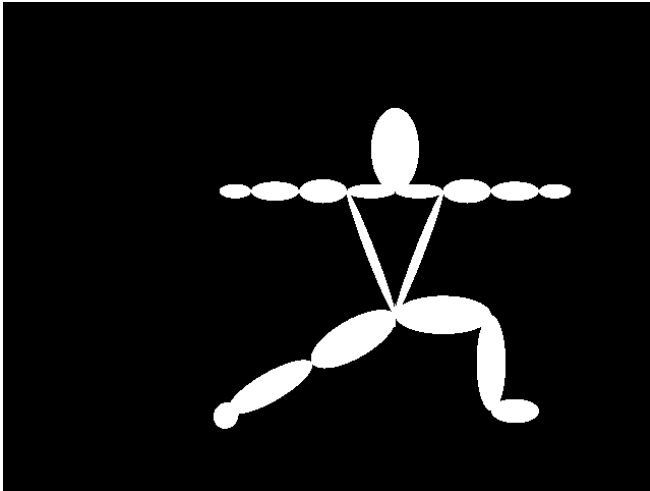
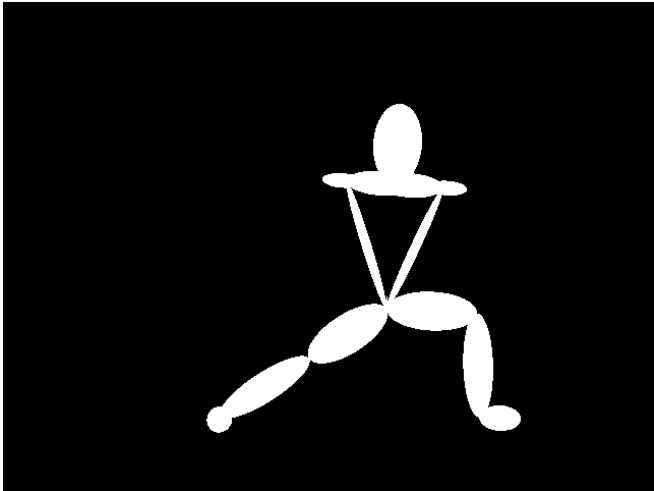
Adult 1	Triangle Pose
Image	Objective Match
	 <p data-bbox="816 747 1469 825">asl_weight = 20000, floor_weight = 500, scale during alignment</p>
Aligned Model	Subjective Match
	 <p data-bbox="816 1371 1469 1409">asl_weight = 25000, floor_weight = 2000</p>

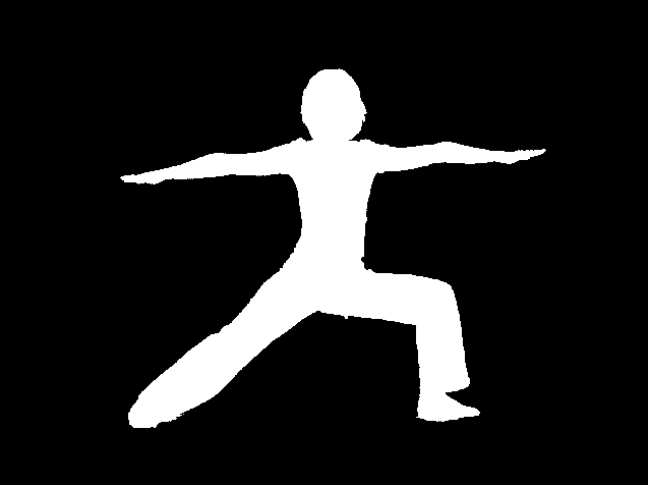
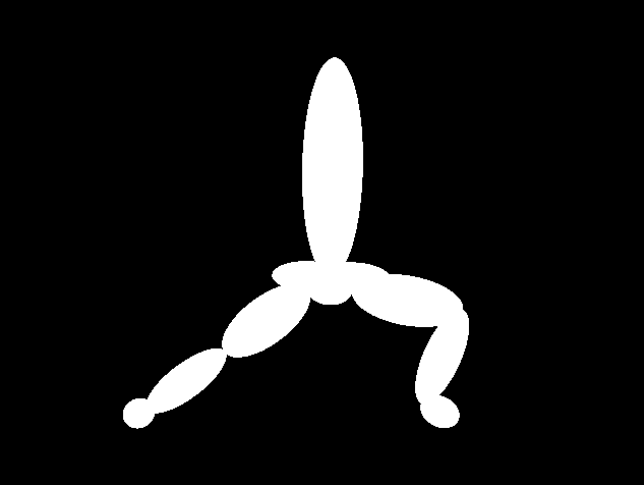
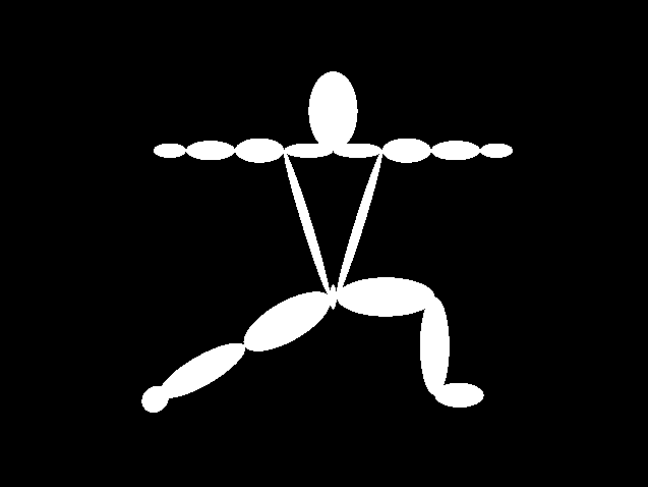
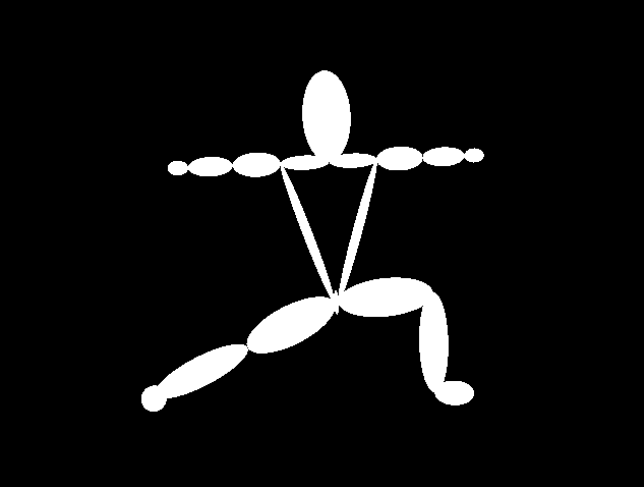
Adult 1	Warrior 2
Image	Objective Match
	 <p data-bbox="816 747 1369 787">asl_weight = 15000, floor_weight = 20000</p>
Aligned Model	Subjective Match
	 <p data-bbox="816 1333 1352 1373">asl_weight = 25000, floor_weight = 2000</p>

Adult 2	Forward Bend
Image	Objective Match
	 <p>asl_weight = 0, floor_weight = 1</p>
Aligned Model	Subjective Match
	<p>Same as Objective Match</p>

Adult 2	Side Angle Pose
Image	Objective Match
	 <p data-bbox="821 747 1461 785">asl_weight = 15000, floor_weight = 3500</p>
Aligned Model	Subjective Match
	 <p data-bbox="821 1331 1461 1369">asl_weight = 15000, floor_weight = 2500</p>

Adult 2	Triangle Pose
Image	Objective Match
	 <p data-bbox="821 747 1461 823">FLOOR500 asl_weight = 0, floor_weight = 500</p>
Aligned Model	Subjective Match
	 <p data-bbox="821 1369 1461 1402">asl_weight = 20000, floor_weight = 2500</p>

Adult 2	Warrior 2
Image	Objective Match
	 <p data-bbox="816 747 1466 785">asl_weight = 0, floor_weight = 4000</p>
Aligned Model	Subjective Match
	 <p data-bbox="816 1333 1466 1373">asl_weight = 20000, floor_weight = 1500</p>

Adult 3	Warrior 2
Image	Objective Match
	 <p data-bbox="821 751 1351 785">asl_weight = 15000, floor_weight = 2000</p>
Aligned Model	Subjective Match
	 <p data-bbox="821 1335 1416 1444">ALL205S asl_weight = 20000, floor_weight = 500, scale during alignment</p>

6 Discussion

We acquired decent matches for several of the test images, but for others the results were not as good. One thing that seemed to repeatedly cause an issue was our representation of the torso. Because there was a large hole in the center of it, our matches often tried to fill this hole. It used either the arm or the head to fill the torso in. One possible adaptation would be to represent the torso as a single segment, perhaps as a tapered cylinder as in [Lee, Cohen, 2006]. Additionally, our alignment step that we performed prior to Powell's method sometimes did not give us much to work with. There is obviously some room for improvement here.

For chair pose, we got a decent match from our test image. However, this was the only match that did not have the arm filling in some of the space the torso takes up in the image. The objectively best

match is a good example. Also, the baggy shorts also seemed to interfere with matching. This is exemplified in the subjectively best match where the scale is increased to produce larger legs to match more of the shorts. Perhaps some intelligent trimming of the silhouette could be done.

Forward bend did not have very good results. This appears to be mainly due to the torso not being aligned properly. For the wide-legged version of forward bend, although the torso seems to be aligned properly, the alignment step tends to over inflate the head to cover the arms, which are not included in the model during torso alignment. This makes it hard for Powell's method to obtain a good match because the head takes up a majority of the image.

For side angle pose we obtained pretty good results. However, the objectively best results suffered from the problem of filling in the torso using arms and head. The results for triangle pose were similar to those for side angle, although for adult 1, the head is too large and the torso is too small. Also, the upper arm is pretty far off.

The test images for warrior 2 had mixed results. For adults 1 and 3, the subjectively best match ended up pretty good. For the others, there seemed to be an issue with the arms. In most of the objectively best matches, the head and/or legs filled in the empty space in the torso.

7 Future Work

There are several areas where improvement can be made. Deriving derivative information should give us faster and more accurate matches. Extending our algorithm to use information multiple view will similarly provide improved matching. It is also likely that tweaking the way the minimization is performed may yield better results. Finally, silhouette segmentation should help with the initial alignment.

7.1 Derivative Information

By changing our minimization algorithm from Powell's method to a method which requires derivative information we can expect to see a runtime improvement of about one order of magnitude. Obtaining such derivative information for our problem would be a straightforward, but laborious task. However, with this improvement, we would expect run times much closer to what would be needed for useful real time feedback. Combining this derivative information with some of the other possible improvements below should allow this to achieve very fast and accurate matching.

7.2 Multiple Views

The current method could be extended to use multiple views of the same pose. This would allow for more accurate matches, possibly in shorter time. To accomplish this, we could directly extend our error function to take multiple images and project the model into each image. This would increase the dimensionality of our problem because we would have more camera parameters, but would only need to add 5 parameters for each camera as the field of view, and scale should be the same if we use identical cameras. Alternatively, we could perform the matching in 3-space by obtaining 3D voxel data from the various silhouettes as in [Mikić et al., 2003].

7.3 Reorganize Minimization

Some experimentation could be done with regard to the actual minimizations done. Instead of one singular minimization it could be broken up into a combination multiple lower-dimensional minimizations. For example, we could optimize just the camera parameters with all segments disabled except the torso. Then, we could add the arms and legs and not touch the camera parameters. Such an

alteration may or may not prove beneficial, but there are several such combinations that could be explored.

7.4 Silhouette Segmentation

By segmenting the silhouette using image processing techniques combined with our approximate model match, we could calculate the difference between the first moment of inertia of the projection of a segment and the section of the silhouette that it corresponds to. Using this and knowing the image plane, we could calculate how to adjust the angle to “snap” the segment into place. Additionally, if we knew what part of the silhouette was the torso, we could better align it during our initial alignment step. With a segmented silhouette, we could match the torso, and add in segments one at a time, roughly aligning each segment based on the moment of inertia or center of mass difference between the projection and the model. This would get us very close, and Powell's method could probably provide us with a very accurate final match.

8 Conclusions

While there were some promising results, overall the matches are not nearly good enough to provide good feedback to a yogi. Powell's method generally obtains good results if the initial image is close, however our alignment step does not always provide an initial match that is close enough. The representation of the torso seems to be the biggest problem. Because of the large empty space in the middle of the torso, Powell's often fills in this space with the arms or the head. Beyond improving on this, there are many other improvements that can be made. The torso/camera alignment step can not change angles, which makes it impossible to properly align the torso in some situations, such as in forward bend. It also seems there is too much interdependence in the model for a general minimization technique like Powell's Method. While some of this interdependence could be eliminated, it is impossible to completely eliminate all of it. If the feet are on the ground, changing the knee angle will always affect the position of the torso, and thus the arms and the head. Because of this, the order the angles are optimized can have a large affect. Currently, the angles are specified in arbitrary order. Putting the leg angles first, for example, should help Powell's to find the right combination of angles to change to achieve the proper change in the image.

9 References

Chen, H., Chen, H., Chen, Y., and Lee, S. 2006. Human action recognition using star skeleton. In *Proceedings of the 4th ACM international Workshop on Video Surveillance and Sensor Networks*: 171-178

Coulter, H. David. 2001, *Anatomy of Hatha Yoga* (Honesdale, PA: Body and Breath Inc.) 227-228.

Jaimes, A. 2005. Sit straight (and tell me what I did today): a human posture alarm and activity summarization system. In *Proceedings of the 2nd ACM Workshop on Continuous Archival and Retrieval of Personal Experience*: 23-34.

Lee, M. W., Cohen, Issac, 2006, A Model-Based Approach for Estimating Human 3D Poses in Static Images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume 28, 905-916.

Micilotta, A.S., Ong, E.J., Bowden, R. 2005, Detection and Tracking of Humans by Probabilistic Body Part Assembly, *Proceedings of the British Machine Vision Conference*, Volume 1, 429-438.

Mikić, I., Trivedi, M., Hunter, E., Cosman, P. 2003, Human Body Model Acquisition and Tracking Using Voxel Data, *International Journal of Computer Vision*, 53(3): 199-223.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. 1992, *Numerical Recipes in C, the Art of Scientific Computing* (2d ed. ; Cambridge : Cambridge Univ. Press) 412-420.

Salti, S., Schreer, O., Stefano, L.D. 2008, Real-time 3D Arm Pose Estimation from Monocular Video for Enhanced HCI, *Proceeding of the 1st ACM Workshop on Vision Networks For Behavior Analysis 2008*: 1-8

Appendix A- Free Software Used

Source code can be downloaded via git:

```
git clone git://github.com/yrral86/yogahero.git
```

README:

YogaHero yoga pose matching software

Written by Larry Reaves <larry@yrral.net>

make builds three executables:

findmatch:

matches an image to a pose

produces match.png (image of matching model), match.pose (saved model),
and match.data (timing/error data)

modelviewer:

view the model with sliders to change angles/segment lengths/camera params
also, can load and save models to files

test:

runs findmatch with various parameters on the given image and pose
saves match.png, match.pose, and match.data to
images/output/image-basename/ for each test

make also has a test_all target, which when invoked matches all the images
to all the poses, making some assumptions about filenames (the image will
be in images/*posename.png and the poses are in poses/posename)

If you are running Ubuntu, or any other Debian based distro,

make ubuntudeps

should get you everything you need to build/run the various YogaHero
programs

Appendix B- Free Software Used

coriander(<http://damien.douxchamps.net/ieee1394/coriander/>)- used to gather data from cameras

devhelp(<http://developer.imendio.com/projects/devhelp>)- API help for gio/glade/glib/gtk+/gtkglex

firefox(<http://www.mozilla.com/en-US/firefox/>)- research/coding help

gcc(<http://gcc.gnu.org/>)- C compiler

gio(<http://library.gnome.org/devel/gio/stable/>)- used in test for filename manipulations

git(<http://git.or.cz/>)- used for revision control of code

glade(<http://glade.gnome.org/>)- used in modelviewer GUI

glib(<http://www.gtk.org/>)- used in all programs for various purposes (string manipulations, working with gtk objects, etc.)

gnome(<http://www.gnome.org/>)- desktop environment used

gtk+(<http://www.gtk.org/>)- used in modelviewer GUI

gtkglext(http://www.k-3d.org/gtkglext/Main_Page)- used in modelview GUI, and to provide offscreen rendering for findmatch

imagemagick(<http://www.imagemagick.org/>)- used to clean up raw data to make silhouette generation easier

opencv(<http://sourceforge.net/projects/opencvlibrary/>)- used for various image manipulations in findmatch

openoffice.org(<http://www.openoffice.org/>)- writing/data analysis

perl(<http://www.perl.org/>)- used for file manipulations when working with raw data

retinex(<http://www.fmwconcepts.com/imagemagick/retinex/index.php>)- script used to improve data quality before silhouette generation

svn(<http://subversion.tigris.org/>)- used for revision control of documents/initial code

ubuntu(<http://www.ubuntu.com>)- operating system

xpaint(<http://sourceforge.net/projects/sf-xpaint/>)- preparing annotated model