Competitive choices on actions in motion graph

Small project GMT master by Roel Duits

This report is a collection of the algorithms used and choices made in this small project on competitive choices for animation purposes. The method is based on a paper titled "Simulating Multiple Character Interactions with Collaborative and Adversarial Goals" by Hubert P. H. Shum, Taku Komura and Shuntaro Yamazaki. In my project I have adjusted the described method to work on a standard motion graph instead of the action graph structure as they describe it. With this I run some experiments on how to model different behaviors with the various parameters.

First I will explain the original technique in some detail. Following this will be a description of choices made during my implementation as well as a short overview of the implementation. After this will be the experiment setup and results and a conclusion at the end.

1. The original technique

The method as described by Komura et al. in [1] is based on an action graph. This is a motion graph, as described in [2] and [3]. The motion graph as defined in [2] consists of directed edges that contain an animation clip and nodes where these edges connect to other edges such that transitions are possible between animations. Additional "transition" edges, short clips to go from one node to another, can be included to increase the connectivity of the graph. By having a well-connected graph, transitions between animations or "actions" becomes easier while still looking smooth. The authors of [1] call this an action level Motion Graph or Action Graph.

They analyze each animation to determine what actions it is composed of. A recorded animation may be a single action, but can also be a sequence of different actions that can then be segmented. This distinction is useful later for analyzing the choices in the game tree as described below. These animation segments are used to populate their action graph. Each segment should describe a full action and transitions will be possible between the start and end of each action, provided the poses are similar to each other.

With this graph they construct a game tree of choices made by the various characters that are interacting with each other in a competitive way as in Figure 1. The game tree is constructed by letting one character choose an action followed by his opponent. By evaluating the action choices available to each character given his current state after the previous choices the character made, a new choice will be added to the game tree. The characters choose a new action whenever they have completed their previous action and are allowed to do multiple choices in a row if their opponent has chosen a particularly long action. They limit the depth of this game tree to ensure the system will keep up. As each layer in the tree can contain several new children, the size of the tree grows exponentially. The limit ensures that the characters will look ahead at their future options, but not at the cost of real-time performance.

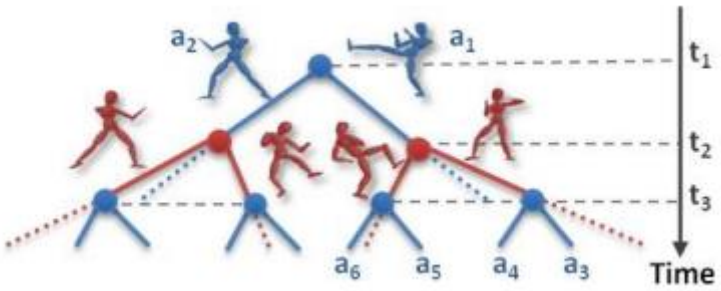


Figure 1: Example of a Game tree as found in [1]

By assigning a score to each action in the tree working to the leaf nodes they can determine the series of actions that is the most beneficial to a character. Each node will receive a score based on: a movement term, such as distance and orientation towards the opponent, an action term, based on damage done and taken, and a combination term, to exclude certain action combinations that do not make sense given what the opponent is doing. This last term can be used to fine tune the style of defense or movement for example by excluding certain actions that don't fit the 'theme' but would be valid choices in the graph. An example of a theme would be an 'agile' defense where the character will choose to dodge over blocking versus an "enduring" defense where the character will take some damage in favor of dealing more himself.

They also use a cooperative score to allow an animator to direct the characters as a group. This is useful to make characters locked in combat move along a trajectory whilst fighting. The cooperative score is evaluated such that both characters look to maximize it, while they attempt to gain the upper hand in competitive score, which should lead them to victory. This way they continue to spar with each other but will both make choices that help in moving in the right direction as well.

2. Adaptations to the technique and implementation

In order to fit this technique into RAGE the existing motion graph would have to be reworked almost completely to be an action graph as described in the paper. However it does not make much difference if the edges contain animations and the nodes contain the transitions or vice versa. As such I decided to leave the motion graph as is, as it was already functional this way and stored the animations and transitions the other way around than is described in the paper. The only change that was necessary is that when evaluating the choices in the game tree we have to follow the edge to the end node to determine the next action that will be taken in this path.

I then expand a game tree as the authors do and let both characters work on the same tree. They will be looking at the action sequences so it makes sense to have a single data structure to keep the choices in and use the data differently depending on what character is making the decision. By letting the characters pick their next choice after their current one has been nearly completed, they will choose and expand the data structure when it is their turn.

The evaluation of a competitive score is very similar to the described method in the paper.
The main difference is that I would not include the combination term. This seemed like a strange way to limit certain combinations. A good system should discourage certain choices by assigning bad scores to them based on, for example, the damage done or taken score. Such a bad choice should only be taken when the actions afterwards are significantly better than the other alternatives. By imposing an artificial limitation on such choices it probably becomes more difficult to have a counter attack behavior.

2.1. Analysis

The authors captured long segments of motion to use for their action graph and used an automated process to segment and categorize them. I opted to keep things a little more manageable by recording short animations containing a single complete action, such as walking forward or punching. Each of these animations can then be more easily processed as there is no need for segmenting the actions. The authors would analyze the accelerations of the joints to determine what type of action is taken. I have opted for a different way as described below using the ranges of orientation angles of important joints in the arms and legs. In the section 3 I report on the differences I found in how well these two methods worked for me.

When the system is started it will verify if an action graph already exists in database form. If no action graph data is present, the animations will be loaded in and run through the analysis algorithm. This procedure will walk through the animation step by step and keep track of the position and orientation of the root joint, as well as the orientations of the various other joints of the characters' skeleton. The following steps are how I implement them in my analysis algorithm. The authors have not provided their own step by step system and thus I cannot assign what is new or has changed compared to their setup.

Using this data I evaluate what range of orientation angles is used during the animation for each joint. This will allow me to determine what part of the body moved a lot during the animation. For example a punch will require rotations of the shoulder and elbow of a single arm, where the elbow will go from bended all the way to fully stretched, thus covering a large range of orientations. Similarly translations of the root joint indicate movement. After analyzing the various joints and assigning scores to the possible range of actions they can depict, the animation is assigned the action that has the highest score. By manually tuning the numbers of the ranges I have set up a system that accurately determines the right action for each animation. These number ranges can be found in Table 2 in the appendix.

With the same data the time span of the actual action, such as a punch, is evaluated. This puts a time range on when the action is taking place. This time range will then be excluded later in the graph creation process so no transitions would take place mid-action. First the frames of each animation would be compared to each other and a distance would be determined for each of these combinations. Then these distances would be compared to each other within each combination of time windows for both animations in order to determine the best possible transition.

No transitions would be allowed out of the start of a clip, nor any transitions to the end of a clip. Anything in between was not limited if a window would occur there. Most animations only had a start and end window, but some movement clips could have multiple transitions whenever both feet where on the ground. This way the actions would be played in full but would still allow a preferable transition to occur between various animations at the start and the end.

A damage done value will be calculated for all actions considered as offensive moves, such as punches. The damage done is based on the acceleration and mass of the body part that delivers the attack to calculate the force applied on the opponent. These values would then be converted to hit points by some normalization.

An action marked as defensive by the above algorithm will be assigned a number in the range (0,1] in accordance with the damage reduction it should provide. Blocks would give 50% damage reduction if they are directed to the attacking body part of the opponent and dodges would result in 100% damage reduction. These numbers were chosen for functionality and not based on any analysis of the actual defensive action.

Each of these types of actions will also have a time frame in which the offensive or defensive action will be considered 'active'. This will be used to determine if an attack is blocked or avoided and will be described in detail later in the game tree subsection

In the end the damage analysis was not implemented as the movement part of the system proved insufficient to get the characters to face each other properly for a fight. As such some values or formulas for normalization are not provided here as none have been defined for this system. In the experiments section 3.2 and 3.3 I provide a summary of my experiments with the movement scores of the game tree.

2.2. Graph creation

As mentioned above in the introduction of the section I used a different setup in the graph. As a result I could not necessarily work with the start and end poses and then line those up to connect the various actions. Instead I reworked the existing graph building algorithm of RAGE to only look for transitions in certain time windows based on the animation clip.

With motion data available from the analysis process described above the transition algorithm will be run to populate the database with transitions for the various animations. This is done by retrieving the time ranges where transitions will be allowed and iterating only over frames in these ranges. The algorithm looks for the combination of frames where the poses are the most similar and store these as a transition. The similarity in my implementation is defined as in [2] to minimize the distance between the joints of the two poses. The authors do not provide a detailed description of how they create their graph and group their poses together so no comparison can be made. They make a reference to the creation method of [2] so it could be set up similarly.

If a time range is at the very start of an animation, no transitions out of it will be stored. Similarly transitions will not be made to the very end of an animation. This way the graph should be set up such that the character will perform full actions and then choose a transition to the start of the next action, instead of moving from end to end without actually performing an action.

With these transitions in place an action graph is created by inserting nodes for animations and transitions between them as described in the database. Transitions will split a node into 2 nodes, each containing a part of the original animation clip. Each animation has some additional data about the movement or damage done or negated by choosing this animation. This data is needed later on in the evaluation process. When splitting the node this data is also split accordingly.

2. 3. Game tree

From there an initial game tree will be constructed by choosing an initial animation for each character and expanding their choices from there. They will alternate whenever the other will finish his action first. Once a parameterized depth is reached the expansion will be stopped. The characters will play their initial animation and once they have almost completed it will analyze the current game tree. This is similar to the original technique.

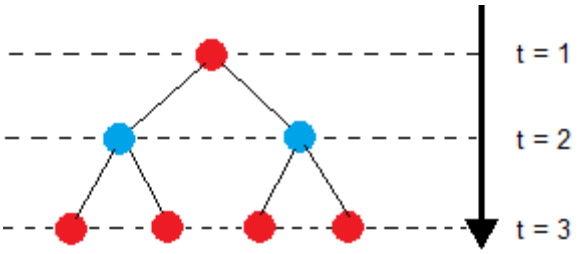


Figure 2: Example of a game tree expanded until depth level 2. Each time a decision is made between two options

The character will assign a score to each route of choices in the tree based on how much it will either benefit him or the opponent. For each node a motion score and damage score will be calculated. The action combination score term of the original technique has been excluded.
The formula for the score of an individual node is: $F^{comp} = f^{move} + f^{damage}$. The move and damage score are calculated as described in the original paper and are explained below.

The motion score equation is: $f^{move} = w_\theta(\theta - \theta_d)^2 + w_r(r - r_d)^2$, where the w_θ and w_r refer to the weight of the orientation and distance difference, the θ and θ_d refer to the relative orientation and desired orientation of the character and his opponent and the r and r_d refer to the relative distance and desired distance to the opponent. The aim is to achieve the desired relative orientation and distance to the opponent by assigning higher scores to actions that reduce the distance between the actual relative orientation and distance and the desired values.

The damage score will take into account both damage done and taken. It is calculated as follows: $f^{damage} = w_D^+ D^+ - w_D^- D^-$, where w_D^+ and w_D^- are the weights for dealing and taking damage and D^+ and D^- are the damage done and damage taken values respectively. The damage value will be calculated during the analysis for all offensive actions. Similarly a damage reduction value will be assigned to defensive actions. The calculation of these values was not defined by the paper so I came up with an intuitive definition for it. I also included a time frame in which the action has effect to determine if an attack is blocked or not.

The final result of an offensive or defensive move is dependent on the actions of the opponent. As mentioned earlier each offensive or defensive move has an 'active' window in which its effect takes place. For an offensive move this indicates in what part of the animation the damage will be dealt. Similarly it indicates in what part of the animation the damage taken is reduced for defensive actions.

The damage of an attack from character A is only mitigated if the active window of a defensive action of character B. If this the case then the damage value of A's attack is multiplied with the reduction value of B's defensive action and the result is D^+. B will calculate the damage taken D^- in the same way in his own evaluation. If the active windows do not overlap then A will deal the full damage value of the action to this opponent. See the below image for an example of the actions performed by the characters over time and the overlap of the active windows.

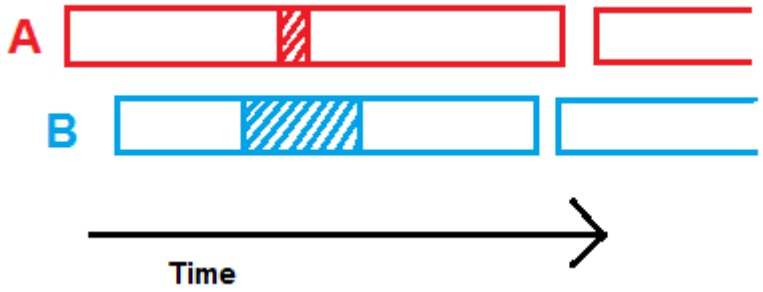


Figure 3: Example of an attack and defense combination. Character A would select an attack action with a short active window. Character B then selects a defensive action with an active window that overlaps the attack.

If a character is not near the opponent or not facing the right direction then an attack will be considered avoided. This will result in a zero value for the damage done part of the damage score and thus make it a poor candidate choice in the evaluation process.

These scores are calculated from the root of the game tree to the leaves, adding scores from your own choices and subtracting scores from actions made by your opponent and then via a min-max search the optimal route is calculated. Your own choices are considered to be max nodes and the best available choice will be taken here. The opponents' choices are considered the min nodes as they aim to reduce your chances of winning. Here the option with the lowest score will be taken as this should be the best choice for the opponent. It is best to assume your opponent will make a good decision for himself that either damages you or protects himself instead of assuming the opponent will make a bad choice. Seeing as the opponent applies the same logic the node with the minimal score will probably be the best node from the opponents' point of view. This procedure is the same as in the original technique.

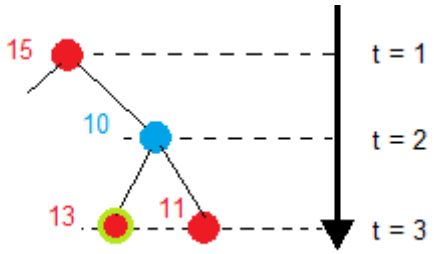


Figure 4: Example of a min-max search from the root down one subtree. The red nodes are considered max nodes and the blue nodes considered min nodes. The final choice will be made as the bottom left node in this subtree as it is the highest. If the left subtree at t = 2 has a lower score, that subtree will be chosen over this one.

When the search returns to the root node an optimal child will have been found and the next choice is stored. After this the root node is shifted one layer down and the tree is expanded as there is now a new layer available within the horizon limit of the tree. When the character has completed his current animation he will retrieve the stored choice and perform this animation. By continuously running this choice algorithm and expanding the game tree afterwards the characters will interact, until a victor emerges because he has dealt enough damage to the opponent.

3. Experiments

In order to obtain a good action graph I have made some changes to the animation analysis and scoring procedure of the technique, as described above, that would allow me to more accurately or more easily perform the required steps. The following section describes the various experiments that have taken place during the construction of the full system and have led to some of these changes. The list of actions in the system and values used can be found below in the appendix.

3.1 Animation analysis

In order to have a motion graph function like an action graph I needed additional details on the animations in it. As such I started out with an analysis system as described in the paper. This system made use of the trajectories of joints with high acceleration. This seems intuitive as the actions can be done in a short timeframe and make significant use of a certain part of the body. It should make one or several joints stand out in terms of acceleration. In practice this produced hard to interpret results and I have implemented a different method as described in subsection 3.1.1.

In order to determine the movement of the character during an animation a definition was needed for how movement was recorded. I set up local frame of reference in various ways to track the movement of the character over the course of the animation. The definition of this frame of reference would prove important to its ease of interpretation and is explained later on subsection 3.1.2.

Finally each animation would have some time frames assigned where a transition would be allowed. These frames contained a part of the animation where very little movement was being done either before or after the major action of the particular animation. This way the motion graph could be constructed so that each node contained a full action and transition only from the end of one animation to the start of the next. Therefore the detection of where the action was taking place was very important to ensure no transitions occurred in the midst of an action. This will be described in subsection 3.1.3.

3.1.1 Action analysis

The initial implementation used the squared sum of accelerations of various important joints in the arms and legs as well as the root joint. The square would make any acceleration count equally instead of having a `negative` acceleration negate a previous `positive` acceleration. However the resulting sums were often very similar between animations segments that portrayed very different actions. The calculation of a distance, velocity and acceleration would provide small numbers close to zero. Taking a square will further reduce their impact and in the end there were no significant peaks left to determine what sort of action had taken place. Changing units could solve this issue, but the system in general is aimed at very quick motions. If a defensive motion is made with a near constant speed of an arm then the sum of squared accelerations will be relatively low, even though a significant motion is made with the arm. I therefore used a measure of the angle range used during the entire motion, as this is not dependent on the speed of the motion.

In order to obtain the significant motions in a clip I changed this to work with the range of rotations used for the individual joints. A punch would have a large rotation across one of the angles of an elbow for example, as the arm has to stretch out to nearly full length. Similarly a block can be done with little to no movement except for the shoulder. By collecting the X, Y and Z angles of the root, shoulders, elbows, hands and feet, as done previously for the squared sum of accelerations and taking a min and max over the values I could determine the range that was used in the animation. By assigning scores to certain intervals for the individual joints I could then determine the type of action that was performed in the animation clip. These values can be found in Table 2 in the appendix.

This method should always provide clear information on what part of the body performed an action as the angle range should be significantly larger compared to the resting state or subtle motions made during an action of a different body part, such as slight bending of one arm when punching with the other. It is not reliant on the speed of an animation or the action being performed and is able to discern big actions just as well.

The new method does have some drawbacks however. Firstly it makes heavy use of the data on the animations themselves to determine the important intervals. As such my current parameterization for assigning the scores depends on the animations in my set. However by continuously expanding this set and refining the parameters a more general set of intervals could be determined that should accurately categorize any animation.

Secondly it is still unable to detect the action in animations where the movements are more subtle. Because any action contains small movements all over the body, it can be difficult to determine when a movement is to be considered part of the actual action or if it is just a result of movement elsewhere in the body. These motions are currently manually given an action, but a better system should be used to determine actions in more subtle motions.

Finally for each animation a movement and orientation change is stored for later use in the score calculation in the game tree. This translation and orientation change is based on the root bone but only works for animations where the actor does not return to the same position on any axis as where he started. This would result in a near zero translation or orientation change during the animation. If such animations exist a different approach is necessary, where movement is analyzed from a local point of view during the entire animation. In my set of animations using a fixed frame of reference was sufficient however.

3.1.2 Local frame of reference for movement

The definition for global movement that I used required a fixed frame of reference to determine what direction a character walked to as well as what direction the character would be rotating towards. Initially I used the local frame of the root joint from the start of the animation clip as the fixed frame of reference. However due to varying starting positions and orientations, as well as a rotational offset of the root joint due to a boxing stance the resulting translation and rotation values were difficult to interpret correctly.

As a result I included the first ten frames from the T pose at the start of every animation I had recorded to use for an initial frame of reference. As each animation was captured separately, they all had their own T pose part. During these frames I averaged out the position and orientation of the root joint. As the actor was standing such that the orientation was more like the global frame of reference, where two axes create a plane parallel to the horizontal floor and the third axis points upwards. This way a global translation and rotation were much easier to interpret relative to this new frame of reference which was aligned with the initial local frame of the root joint.

3.1.3 Transition windows

In order to build a graph with as few connections as needed to allow transitions between as many animations as possible from a single animation, I decided to limit the time window in which a transition could take place to segments before or after the major action. The definition of the action graph by the authors implied a similar split where they aligned the start and end frames of an animation with those of other motion clips. This way a character would always perform an action in full before moving on to the next clip.

In order to also automate this system I again made use of the angle ranges of the joints. After the categorization was completed and the system had decided the type of action being performed in this motion clip it was easier to define limits on the action. For each action type I would define what joints where the most significant in that action. By imposing upper and lower bounds on the angles of these joints, I could determine when the character was in a ‘rest’ state, such as at the start and end of a clip, or in the midst of an action. These can be found in Table 3 in the appendix. Any

animation not listed in there has been defined manually as the automatic process did not provide the desired time windows correctly. Graph 1 and 2 in the appendix provide a graphical overview of using the thresholds to obtain such transition windows.

Whenever the characters joint angles would move outside of the bounds a 'no transition window' would be started and whenever the angles move back into the bounds the window would be closed. I made sure that each animation would ignore the first ten frames as these were part of the T pose I included for the frame of reference creation described above and not intended to ever be part of the motion graph. This time window is manually defined and any other no transition windows would be added to the list by the analysis described here. By then taking the complement of these time frames over the course of the animation, I could determine the time windows in which a transition would be allowed.

3.2 Scoring adjustments

In the paper the authors describe their scoring formula as: $F^{comp} = f^{mov1} + f^{score} + f^{comb}$,the sum of the movement, action and combination terms. Of these terms, the movement term seems to have been incorrectly documented. The equation they have given is: $f^{mov} = w_\theta(\theta - \theta_d)^2 + w_r(r - r_d)^2$. The $w_\theta$and w_r refer to the weight of the orientation and distance difference with positive values used in their experiments, the θ and θ_d refer to the relative orientation and desired orientation and the r and r_d refer to the relative distance and desired distance.

They set the desired relative orientation to 0, meaning the character will attempt to face his opponent. They then state that by setting the desired distance to a lower value, higher scores will be giving to actions that bring the character closer to the opponent. However the difference between the relative distance and desired distance will shrink during this action, thus reducing the value of the square as well. Multiplying this with the reported positive weight will result in smaller scores given to actions that bring the character closer. The same would be true for the orientation theta. The equation would make sense as a cost function, but every description given by the authors indicates that it is a score and that higher scores will be considered as better. This equation does not provide higher scores in the way they want it to. As such I have used a negative value for the weight to circumvent the issue. In my tests I used -1 as no other variables were influencing the score and increasing the value would have no impact in that situation.

Because they do a min-max search on their game tree, where the max nodes are choices that will maximize your own score, the choice would be an action that increases the distance to your opponent, as it results in a higher score. In order to remedy this, I have used a negative score weight for the distance. This results in actions that increase the distance to be given a lower score than actions that bring the character closer to the opponent. As such the characters will make choices as expected.

The min-max search also seems a little counter-intuitive at times. For example the min nodes are described as choices where the opponent will be minimizing the score of the character, thus gaining points himself. However the opponent could do that by just constantly backing away if the movement would offset whatever the other character would do.

As such the definition an opponent trying to minimize a score seems like the wrong way to describe the issue. It should perhaps be changed to a choice that is optimal for the opponent in that situation. This would prevent the opponent from constantly backing away as this would not be optimal for himself. In my implementation I have made it so the score is calculated from the opponents' point of view. This way the choice is evaluated so that it improves the score for the opponent rather than reduce the score of the current character.

3.3 Limiting the transitions

Having run the system several times whenever a new part of the algorithm was included, I noticed that often the characters would move towards each other, but after meeting each other continue to move forward. This setup used only the distance to determine the score for a new choice by filling in zero for the other weights. I wanted to know if the character would make choices that in the long run would always make him converge to the desired distance. However the characters would meet in the middle and then continue on going forward without attempting to correct themselves by moving backwards.

After inspecting the graph I noticed the following occurrence as shown in figure 5. After performing the walk forward animation a few times by alternating (1) and (2) the characters would pass each other in the center of the environment. A character would have the option to take a long animation moving to the left (3), or a very short animation taking the character forward(1). Following the short animation was a longer animation going forward (2), whereas the walk left animation would have a transition to moving backwards (4) amongst other options not shown in the figure.

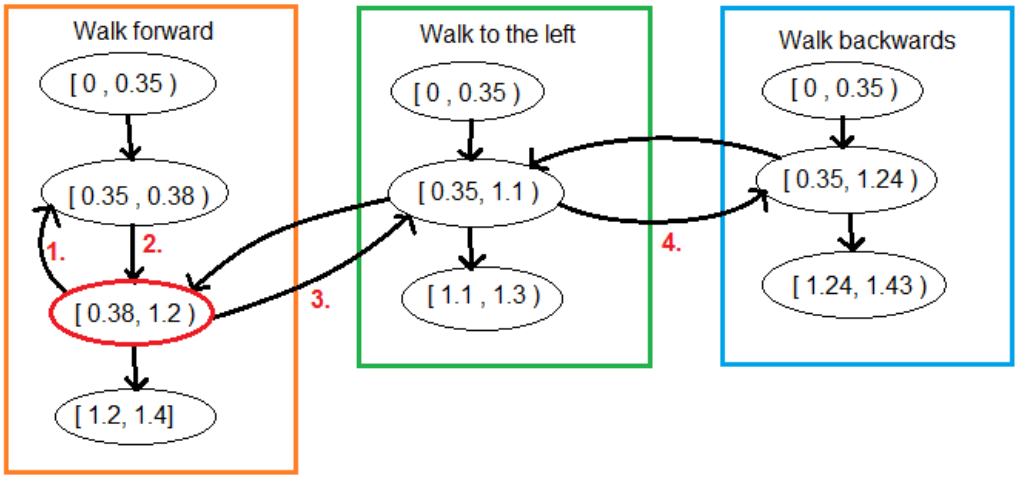


Figure 5: example of an action graph containing three movement motions

The choice for (1) followed by (2) proved to be the better choice of movement however, as the total distance increase was apparently less than doing (3) and (4). As translations were the only factor influencing the score in this experiment, I would expect the choices to be made to converge to the desired distance. That would mean going backwards instead of forwards. An option to move to the right was also available in this graph to offset the motion to the left that is required in this example graph.

Because each choice in the graph is a single animation clip, the tree would not expand enough to realize that this choice was not the best option in the long run. Increasing the depth however had a very noticeable effect on the performance, causing the system to hang for more than second running the calculations for the next choice before the rendering of the animations continued. This would also not resolve the issue completely but make it less common. As I wanted the system to run in real-time, I opted to adjust the graph further by making as many transitions as possible to start or end in the same frame of each animation. The initial setup allowed transitions in a window, but any frame in that window. By using a single frame I would obtain a graph that more closely resembles the action graph in that each animation is played in full before being able to transition to a new animation in full. I tested this out by altering the time windows to a single frame that seemed to have the best connections from in previous graphs.

Even with this change and a graph where almost each node could reach all other nodes, the same issue kept appearing. The characters circle each other for a short while before getting stuck in a loop again where they keep moving forward and away from each other. I would have thought that being able to change to any animation available from each end point in at most two steps would allow a character to decide on the best action possible to minimize the distance but it still seems like the characters avoid changing to a different animation for some reason.

From this I can conclude that the actual action or motion graph used for this system needs to be incredibly well connected, whereas the authors do not seem to emphasize this importance in their paper. If the character is not provided with transitions to animations that allow him to turn or move in any direction from each animations' end, then it is likely they cannot find the way back towards their opponent at all.

Since no damage scores were included it is possible that this issue is avoided if the whole system was operational. However since I fully intended to have the characters within range of each other and looking at their opponent when they attack, they should be able to achieve this positioning regardless of these options. For if they cannot do so, an attack action may be considered better than moving away, but ultimately do nothing either. There is then a potential that the characters will stand near each other but not close enough or not facing each other, locked in an attack sequence which scores better than any movement option available.

4. Conclusion

As my system has never become fully operational I could not run my experiments on the interaction of the various parameters for the scoring system. I have found several key elements during the implementation that are important to make this system work however. In order for the action graph method to function well, each animation needs to contain a pose similar to that of many other animations to create a graph with as many transitions as possible. It seems required that a transition is possible between nearly all animations or at least those that contain movement. Failure to do so will cause the character to be limited in his choices and select the route that it considers the best, even if it is still wrong in our eyes.

Secondly the scoring system seems to have some flaws. The aforementioned movement term which increases as the distance increases is one example of this. Similarly the min nodes where the opponent tries to minimize the characters score opens up behavior that is undesired. The expectation would then be that the opponent backs away to increase the distance. The opponent will however apply the same choice behavior on his turn and try to close the distance. This will result in both characters anticipating the opponent completely wrong.

In the paper the method is described as a good way to have characters interact in a competitive way without having to define many things as the animator. A few tweaks to the parameters such as the importance of avoiding damage or the distance from which the actions should take place is all that would be provided to the animator. However from this project I have learned that it can be difficult to obtain a motion graph that will allow this to work effectively. A better approach may be to have a very well connected graph with many transitions mid-action and devise a better approach to select a sequence of actions to perform competitive actions that are realistic, such as approaching the opponent if the distance is too large or attacking if an opportunity is provided.

Appendix

[1] Simulating Multiple Character Interactions with Collaborative and Adversarial Goals by Hubert P. H. Shum, Taku Komura and Shuntaro Yamazaki, IEEE Transactions on Visualization and Computer Graphics, Volume 18 Issue 5, pp 741-752, 2012.
[2] Motion Graphs by L. Kovar, M. Gleicher and F. Pighin, ACM Transactions on Graphics, Volume 21 number 3, pp 473-482, 2002.
[3] Motion generation from examples by O. Arikan and D. Forsyth, ACM Transactions on Graphics, Volume 21 number 3, pp 483-490, 2002.

Table 1: Animation clips and time windows

The time windows mentioned are calculated by the analysis process for animations that have been defined in table 3. The others have been manually put in as they proved difficult to assess automatically.

Animation	number of frames	Time windows for transitions
combo	169	[10, 45], [135, 169]
idle	314	[10, 280]
left block	174	[10, 40], [140, 174]
left hook	154	[10, 40], [110, 154]
left straight	159	[10, 40], [100, 159]
left uppercut	249	[10, 55], [180, 249]
middle block	149	[10, 35], [100, 149]
middle being hit	189	[10, 70], [140, 189]
move backwards long	289	[10, 45], [130, 145], [255, 289]
move backwards short	169	[10, 35], [120, 169]
move forwards long	279	[10, 60], [125, 140], [215, 279]
move forwards short	179	[10, 55], [130, 179]
move left	164	[10, 30], [115, 164]
move left curve	179	[10, 40], [120, 179]
move right	139	[10, 35], [110, 139]
move right curve	154	[10, 35], [120, 154]
right block	149	[10, 30], [115, 149]
right hook	179	[10, 40], [105, 179]
right straight	124	[10, 30], [90, 124]
right uppercut	154	[10, 25], [110, 154]

Table 2: Angle ranges used to determine the type of action.

The + signs indicate the size of the increase in score of a particular action. Hook, straight and uppercut are types of punches. The range numbers are in meters for translations and degrees for rotations. The axes are based on what was outputted by the system and are dependent on the system definition for local frames. The middle block, idle and being hit animations were defined manually based on the animation name as they proved too difficult to determine by the system.

Joint / type	axis / name	range	effect
Root joint position	X	> 0.4	movement to the right +++
		< -0.4	movement to the left +++
	Z	> 0.4	movement backwars +++
		< -0.4	movement forwards +++
Root joint rotation	Y	60 < Θ < 90	curved movement ++
		> 80	combo +
Left shoulder rotation	X	30 < Θ < 40	left block ++
		40 < Θ < 50	left straight ++
			combo +
		50 < Θ < 60	left hook ++
		> 60	left uppercut ++
	Y	50 < Θ < 70	left block +
			left straight +
		70 < Θ < 90	left hook ++
		> 90	left uppercut ++
	Z	> 40	left punches and combo ++
Left elbow rotation	Y	> 50	left block and uppercut +
			combo ++
			left hook and straight +++
Right shoulder rotation	X	40 < Θ < 50	right block ++
		50 < Θ < 60	right uppercut ++
		> 60	combo +
			right hook and straight ++
	Y	50 < Θ < 65	right straight ++
		65 < Θ < 80	combo +
		80 < Θ < 110	right block +
		> 110	right hook ++
			right uppercut +++
	Z	40 < Θ < 50	right block and hook +
		50 < Θ < 70	combo and right straight +
		> 70	right uppercut +
Right elbow rotation	Y	50 < Θ < 80	right block +
		> 80	combo +
			right straight +++
			right hook +++
			right uppercut +++

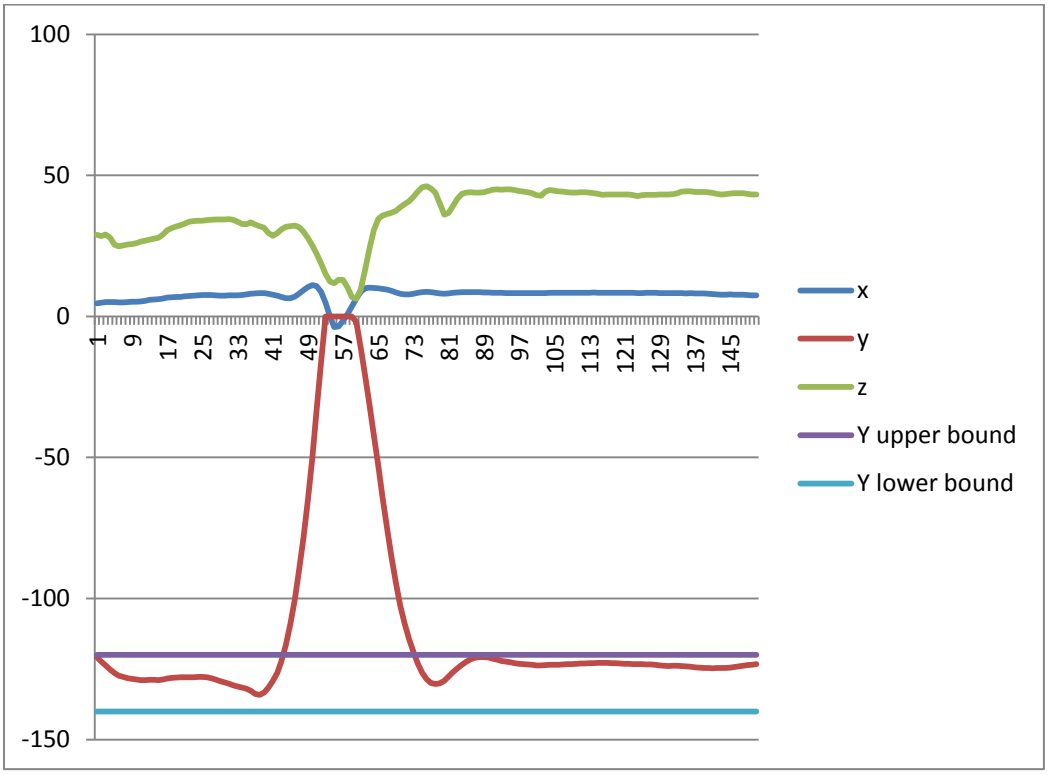
Table 3: Joints and angle ranges used for Transition window detection.

The range numbers are degrees for rotations. The axes are based on what was outputted by the system and are dependent on the system definition for local frames. Any action not listed here has been defined manually as it proved difficult to obtain the right timeframes for them.

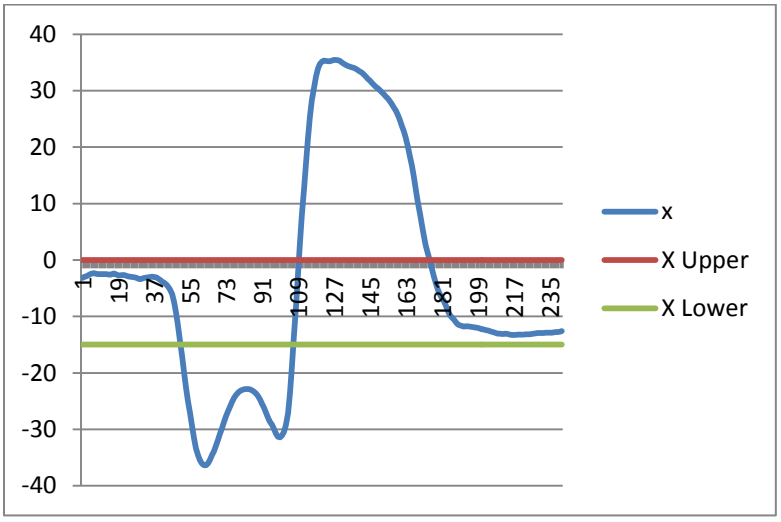
Action	Joint	Axis	Range
left straight	left elbow	Y	-140 < Θ < -120
Left uppercut	left shoulder	X	-15 < Θ < 0
		Y	-52 < Θ < -45
		Z	-27 < Θ < -20
Left hook	left shoulder	X	-40 < Θ < -30
		Y	-15 < Θ < -10
		Z	-50 < Θ < -40
left block	left shoulder	Y	-53 < Θ < 40
		Z	-40 < Θ < 30
right straight	right elbow	Y	110 < Θ < 120
right uppercut	right shoulder	Y	65 < Θ < 70
right hook	right elbow	Y	115 < Θ < 122
right block	right elbow	Y	93 < Θ < 105

Graph 1: Left straight punch, left elbow

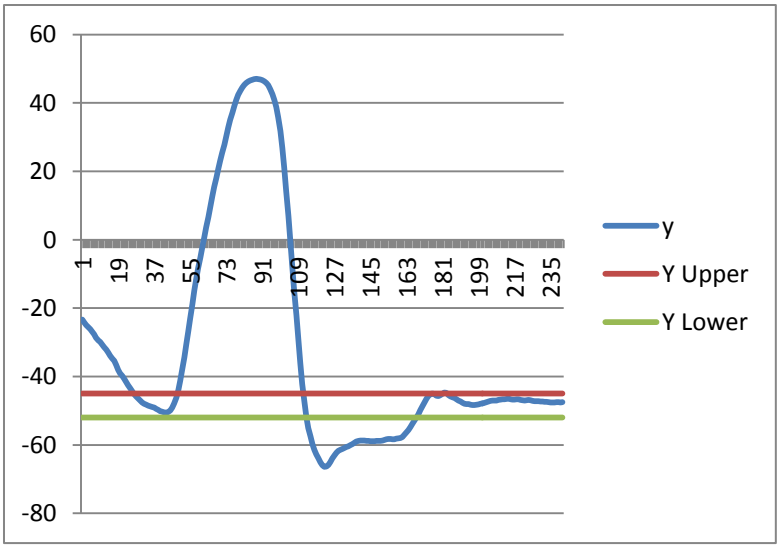
Example of orientation angles of the left elbow during a left straight punch including upper and lower bounds for time frame analysis.



Graph 2.1: Left uppercut, left shoulder, X rotation



Graph 2.2: Left uppercut, left shoulder, Y rotation



Graph 2.3: Left uppercut, left shoulder, Z rotation