CSci 5611
Project - Stick Solo
Report

Yashasvi Sriram Patkuri - patku001@umn.edu
Rishab Girdhar - girdh005@umn.edu
Sreevatsa Saurabh Mylavaram - mylav008@umn.edu

1 Introduction

Inspired from rock climbing, the goal of this project is to animate human-like and animal-like characters climbing walls in realistic fashion. To achieve this, we model our agent as a skeleton with joints and arms. We employ probabilistic road map for global planning and use inverse kinematics for local planning. For simplicity, we stick to 2D motion.

2 Related work

As discussed in [1] analytical solutions for inverse kinematics, although fast and exact become difficult to solve as the degree of freedom increases. As in the case of simulating humans, they have more than 6 DoFs and the analytical solvers are useless in this case. Although Jacobian methods which we have used can be used to solve for cases as complex as 70 DoFs, they start producing oscillations or jerky and unnatural motion especially when the clamping for the linear approximation allows large steps between the target and the end position, or when the limbs are in such a position that a singular matrix is obtained.

There exist solutions like FABRIK(Forward and Backward Reaching Inverse Kinematics) which are faster and do not suffer from singularity conditions. However they have also become outdated due to the recent trends.

For Human-like models, the most popular solution today is to use data driven and deep learning models. One advantage of using data-driven models is that they do are not required to know the constraints in the model and they produce a feasible model based on the data they have. The generated pose is natural and according to the anatomy and constraints of the human skeleton. However, these methods require an offline learning stage and the quality is heavily dependent on the volume of motion data used. In cases where training postures differ greatly from the true solution, the results can have awkward poses. Although in recent years the motion clips are available abundantly, data-driven methods are still not a clear choice for human-like simulation. There are many other factors that affect the performance of data-driven methods, such as a maintaining consistency across time and the smoothly transitioning between the poses.

Deep learning methods can help us overcome these spatial and temporal correlation obstacles. They use pre-captured motions and even though these clips can be erroneous, deep learning methods acquire knowledge of motion many times and through convolution smooth out the errors, thus reconstructing natural and plausible movements. However, learning a CNN model is also time-consuming and the quality of the resulting motion is heavily dependent on the amount of data used for training. When motion is smoothed through the convolution process, it refines the motion and produces fine details. Both learnt and deep learning methods seem to be among the most popular approaches for reproducing human pose currently. Even when working on human-like characters, data-driven methods are not always suitable.

Data-driven methods are not efficient enough for real-time simulation. In such cases, the alternative is to use hybrid models which combine numerical and analytic techniques in a hierarchically structured model. They may be used for tasks where low computation cost is desired and a well constrained human-like agent is needed.

3 Stick-figure agent

A stick-figure agent contains rigid links joined by revolute joints. The simplest stick-figure agent is thus a line agent. A simple family of stick-figure agents is the serial family which contain n-links attached serially. A human-like stick-figure agent can have nine rigid links with a neck and a tail. These are illustrated in the Figures 1, 2, 3. An agent can hold on to a hold in the wall using one of its link ends which which does not have a joint.

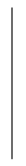


Figure 1: Simplest stick-figure agent

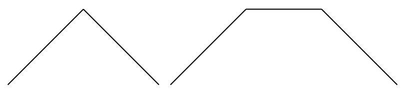


Figure 2: 2 link, 3 link serial stick-figure agents

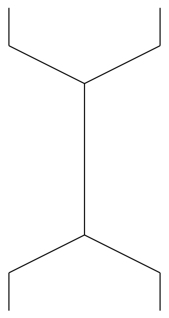


Figure 3: A human-like stick-figure agent

4 Wall environment

A 2D wall with holds which can be used to place the free end of a link. An example is illustrated in the Figure 4.

5 Approach

In this section, we incrementally present our approach. Given a wall with some holds and a finish hold, the agent has to plan a path and climb to reach the finish hold. We break down this task into a global planning and local planning task. The global planning is done using probabilistic road map [4]. Once a probabilistic road map is built we use A* search for path

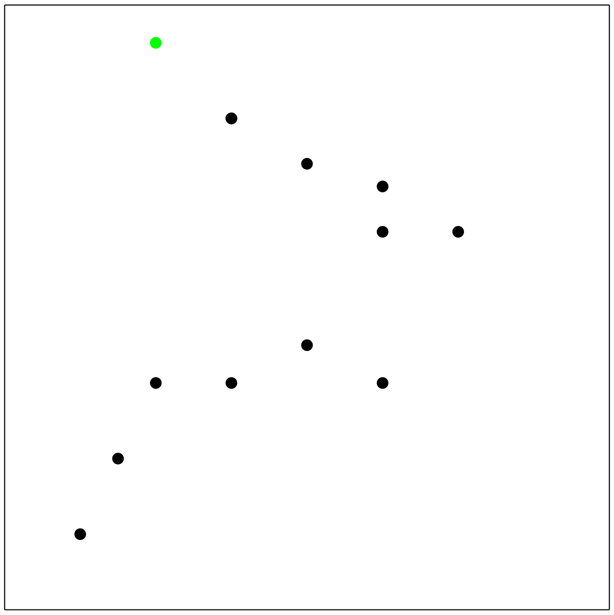


Figure 4: A wall with holds, green is finish hold

finding. In case the agent needs to re-plan we start from the current position to find a new path in spirit of D*Lite [5]. We implement line segment and circular obstacles. We keep some tunable margin around them where we do not sample milestones to take into account the extent of the agent. The path generated by PRM is essentially a list of milestones. We use inverse kinematics for moving from one milestone to another as local planning method.

5.1 Two link serial stick-figure agent

We model a two link serial stick-figure agent as illustrated in 5.

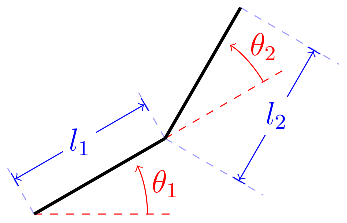


Figure 5: A two link agent.

Each link is associated with an angle as shown in the Figure 5. If the pivot end is at origin then the free end shall be at

$$p \equiv \begin{bmatrix} l_1 cos(\theta_1) + l_2 cos(\theta_1 + \theta_2) \\ l_1 sin(\theta_1) + l_2 sin(\theta_1 + \theta_2) \end{bmatrix}$$

All variables except the angles are constant and known. So we can write,

$$p \equiv f(\Theta), \Theta \equiv (\theta_1, \theta_2)$$

This is called forward kinematics. But the problem here is to find the angles given a free end position. For a given free end position (x, y) the angles θ_1, θ_2 are given by

$$\theta_1 \equiv atan2(y, x) - arccos(\frac{l^2 + l_1^2 - l_2^2}{2l_1 l})$$

$$\theta_2 \equiv atan2(y - l_1 sin(\theta_1), x - l_1 cos(\theta_1)) - \theta_1$$

where

$$l^2 \equiv x^2 + y^2$$

Therefore we place one end of agent (the pivot) at first milestone and solve for angles such that free end is at next milestone. Once the goal angles are found we interpolate linearly the angles to produce smooth animation. This solution fails when $l \equiv 0$ i.e. free end and pivot co-incide. This is only possible when $l_1 \equiv l_2$. This can also happen while interpolating. In such cases we give a random jerk to the agent so that it doesn't get stuck. This is illustrated in Figure 6.

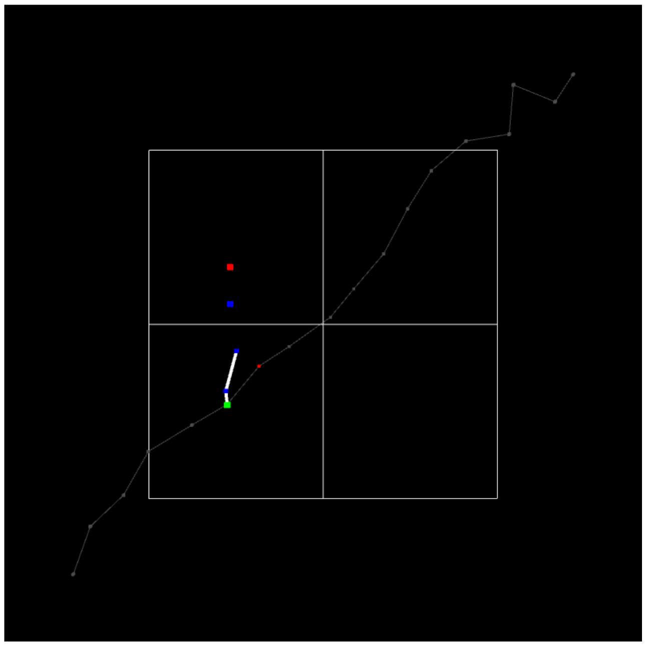


Figure 6: A 2 link agent using closed form inverse kinematics. The angle space is visualized in the back. The blue dot represents current pose in terms of angles and red dot represents goal pose which takes free end to the next milestone.

5.2 N link serial stick-figure agent

The setup for N link serial stick-figure agent is same as that of 2 link one. But for a general N link agent there is no available closed form solution. So we opt for iterative methods like Jacobian transpose [3] and Pseudo inverse [2].

Jacobian Transpose In this method we minimize the squared error loss function

$$L \equiv \frac{1}{2}(p_{goal} - p)^T(p_{goal} - p)$$

$$L \equiv \frac{1}{2}(p_{goal} - f(\Theta))^T(p_{goal} - f(\Theta)), \Theta \equiv (\theta_1, \theta_2, ...\theta_n)$$

Differentiating it w.r.t. angle tuple we have,

$$\frac{\delta L}{\delta \Theta} \equiv -((p_{goal} - f(\Theta)^T)\frac{\delta f(\Theta)}{\delta \Theta})^T$$

$$\frac{\delta L}{\delta \Theta} \equiv -\frac{\delta f(\Theta)}{\delta \Theta}^T(p_{goal} - f(\Theta))$$

$$\frac{\delta L}{\delta \Theta} \equiv -J^T(p_{goal} - f(\Theta)), J \equiv \frac{\delta f(\Theta)}{\delta \Theta}$$

Therefore a step in direction of negative gradient shall be

$$\frac{\delta L}{\delta \Theta} \equiv \alpha J^T(p_{goal} - f(\Theta)), J \equiv \frac{\delta f(\Theta)}{\delta \Theta}$$

where α is a small real number, J represents the jacobian of the free end position function w.r.t. the angle tuple. For a 2D N link agent connected using revolute joints the jacobian is

$$J \equiv \begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times p_{0,f} & \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times p_{1,f} & ... & \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \times p_{n-1,f} \end{bmatrix}$$

where $p_{i,f}$ represents the displacement from ith end to free end. Here all $p_{i,f}$ have z-component 0.

This is the simplest gradient descent iterative inverse kinematics solve. It has the advantages of being simple and fast and numerically stable, but might need many iterations to converge especially when the difference b/w goal and current free end positions is low. This method looks more natural than its pseudo random counter part, which looks especially robotic. This method is illustrated in the Figure 7.

Pseudo Inverse In this method we minimize the loss function

$$L \equiv \lambda^T(\Delta p - \frac{\delta f(\Theta)}{\delta \Theta}\Delta\Theta) + \frac{1}{2}\Delta\Theta^T\Delta\Theta$$

where λ is a variable. Minimizing L w.r.t. and $\Delta\Theta$ simultaneously we have the gradient step as

$$\Delta\Theta \equiv \alpha J^T(JJ^T)^{-1}\Delta p$$

This produces shortest path b/w milestones. It has the advantage of being a second order method but might needs matrix inversion and therefore can be numerically unstable. This method is illustrated in the Figure 8.

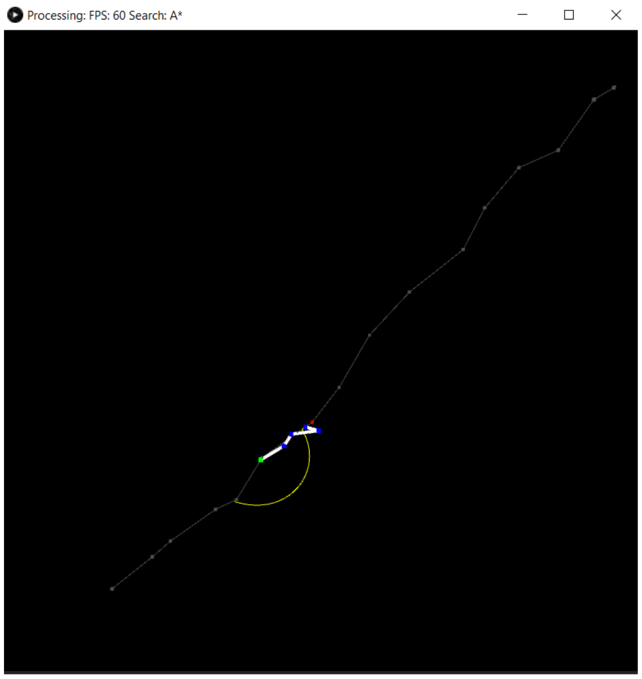


Figure 7: A 4 link agent using jacobian transpose method for inverse kinematics. The path of free end is shown in yellow.

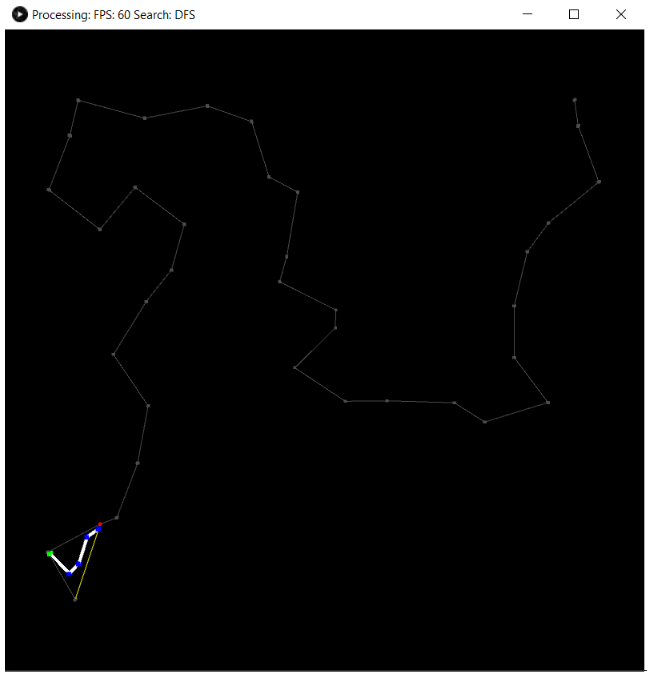


Figure 8: A 4 link agent using pseudo inverse method for inverse kinematics. The path of free end is shown in yellow.

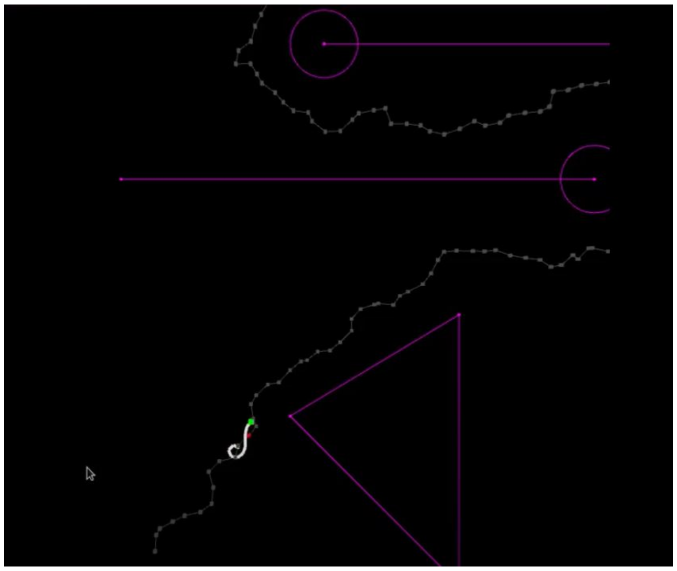


Figure 9: A 10 link agent simulating a worm.

5.3 Modelling worms

An N link serial stick-figure agents with large enough N and small enough link lengths can model worm-like creatures, especially if one end leads and other follows instead of switching pivot in a cart-wheel fashion. This is illustrated in Figure 9.

5.4 Two arm agent

A two arm agent has two independently functioning arms. We model this using two N link serial stick figure agents connected to common point (the neck). We use the Jacbian transpose iterative method for inverse kinematics. We follow a simple heuristical climb cycle as follows,

1. Move a random arm to the next milestone.
2. Move neck below the latest moved arm.
3. Move the other arm.
4. Go to step 1.

While moving arms only one N link serial body part is active and pivoting on neck. While moving neck both N link serial body part are active and pivoting on holds. A subtlety here that while moving the neck we need to synchronize both N link serial agents such that their free ends are at same position all the time. This is done by setting small incremental goal positions for both arms and iterating both arms enough number of times so that they reach the goal. This is illustrated in Figure 10.

5.5 Four limb agent

A four limb agent has four independently functioning limbs, two arms and two legs. We model this using four N link serial stick figure agents, two connected to upper common point (the neck) and two of them connected to lower common point (the tail). We use the Jacobian

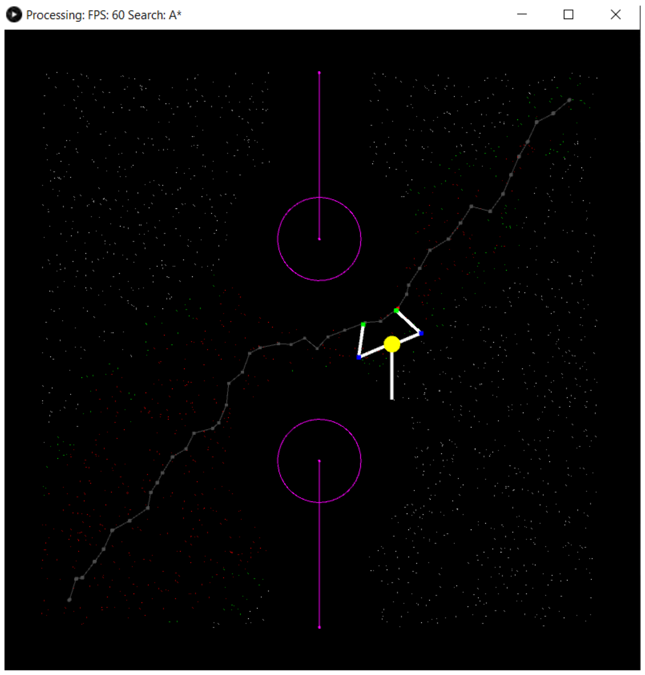


Figure 10: A two arm agent in environment with obstacles. The PRM milestones are visualized. Some margin around each obstacle is left for collision avoidance.

transpose iterative method for inverse kinematics. We follow a simple heuristic climb cycle as follows,

1. Move a random arm to the next milestone.
2. Move neck and tail below the latest moved arm simultaneously.
3. Move the opposite leg to a certain tuned displacement from tail.
4. Move the other arm next milestone.
5. Move neck and tail below the latest moved arm simultaneously.
6. Move the opposite leg to a certain tuned displacement from tail.
7. Go to step 1.

While moving arms or legs only one N link serial body part is active and pivoting on neck/tail. While moving neck and tail all N link serial body part are active and pivoting on holds. This is illustrated in Figure 11. We added an energy level to the agent, such that the agility of agent depends on energy it has remaining which decreases at it climbs. The energy of the agent is visualized as its color. After the agent has exhausted all its energy it stops for a break to recharge and then continues.

5.6 Multiple solutions

A subtlety with N link serial stick-figure agents is that there can be multiple (even infinite) solutions or no solutions for a given free end goal position. The no solutions case generally occurs when the distance b/w current and goal positions is greater than sum of lengths of links of robot, so that even at maximum stretch reaching goal is not possible. This is mitigated in the global planning step by limiting the maximum and minimum edge lengths

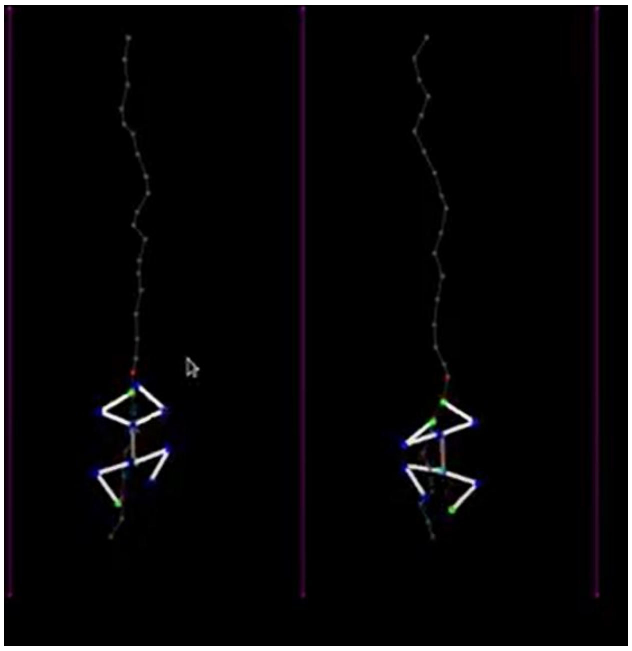


Figure 11: Couple of four arm agents in a race.

of probabilistic roadmap. Still sometimes a different solution can be achieved which causes weird artifacts. This is illustrated in Figure 12.

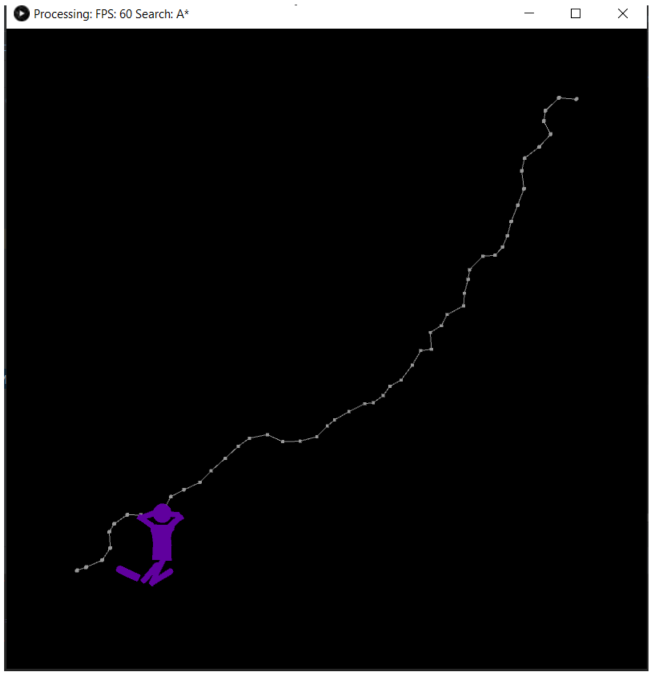


Figure 12: A bad local minima.

5.7 Slippery environment

We modelled a slippery environment by randomly marking some milestones on the path as slippery ones. The agent slips down a bit when it reaches such a milestone and then replans its path that does not include the observed slippery milestone. This is illustrated in the Figure 13.

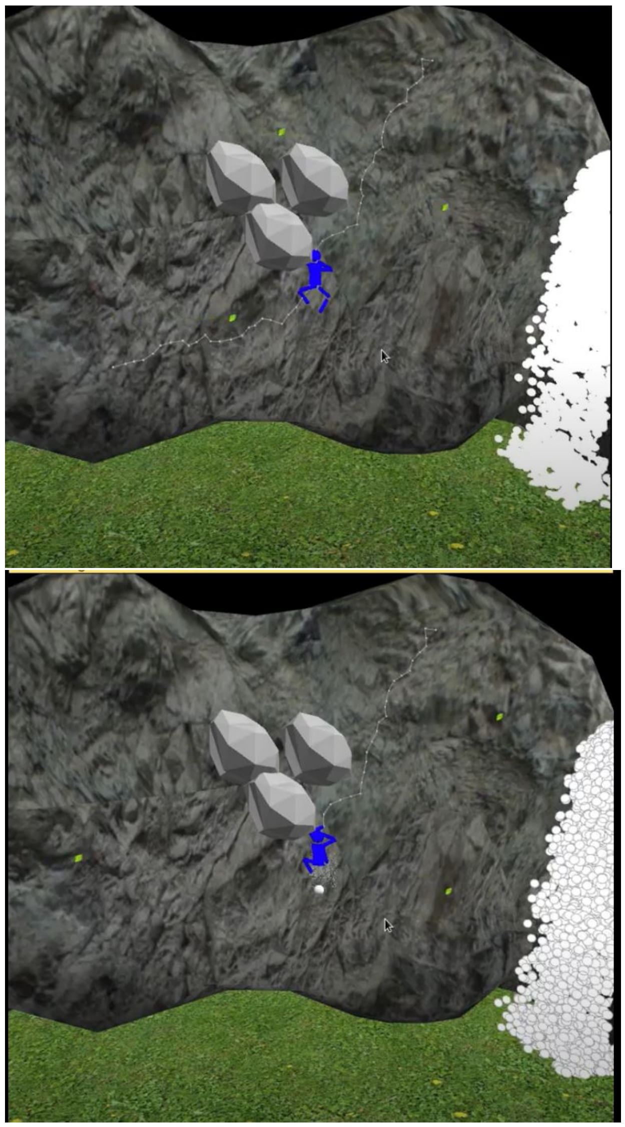


Figure 13: Agent path before and after slipping.

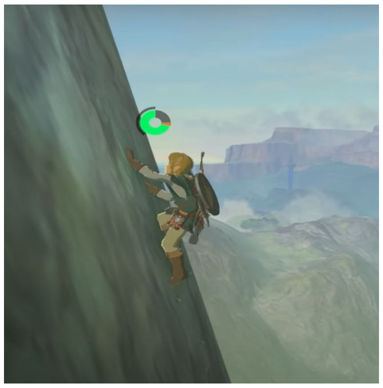
6 Conclusion

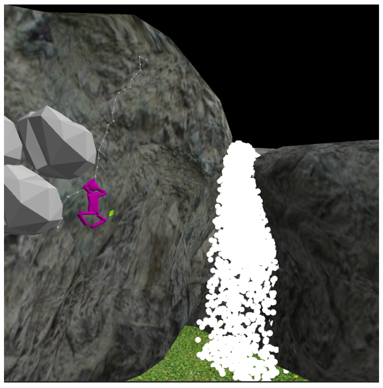
What worked Our agents are able to climb up, down, left, right and diagonally in a robust and realistic fashion. Our agents are able replan a better path when they slip and avoid obstacles. Our agents are able to model human-like and worm-like agents. We implemented a race b/w two climbing agents, wind effects, sound effects, a particle system waterfall, particle system slippery holds. We created and used our own 3D model for each of the body part. Due to the modular and simple architecture and control of our agents, they are can be easily extended to several types of other creatures and motions with relatively small changes.

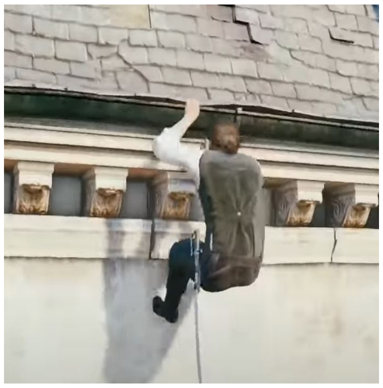
Computational bottlenecks The biggest computational step is calculating the inverse kinematics step. This is because of matrix manipulations and sometimes even inversions. Moreover the size of matrices involved scale linearly with the number of links of each body part. Although not an issue at a single agent level or even at the scale of tens of agents, as the scale reaches thousands this step will be the computational bottleneck.

Comparison of our model with state of the art The human agent in our case is simulated by attaching a pair of two link agents to a straight line whose ends act like the neck and navel of humans. The global path produced is the path to guide the neck of the agent towards the goal. The hands or the two link agents attached to the neck guide it on the path while the legs are positioned according to the navel in a natural-looking way. For our case we only have 2 degrees of freedom in each limb. Hence, using a model like FABRIK or a model based on deep learning like [6] was not feasible or necessary. Jacobian methods and analytical methods are sufficient in this case and they will work well as any other model. Even the slow motion produced by jacobian methods can be dealt with by fixing the speed of the movement of the arm.

Comparison with implementation of IK in games The state of the art implementation today can be seen in the latest games like Zelda:Breath of the Wild 14a and Assassins Creed. These animations although they use inverse kinematics to solve for the body poses, there are some notable differences from our implementations. One, they are implemented in the 3D world while ours just works in 2D. One can see the change in poses due to IK more clearly when on 3D terrain. Also, the characters in these games adapt their body postures, hands and legs according to the structure they are climbing while our agent has the same posture for any type of climb.


(a) Zelda:Breath of the Wild


(b) Our system


(c) Assassins Creeed

What did not work The biggest problem we have is the possibility of reaching bad local minima and its high dependence on initial pose and tuning parameters.

Future work Restricting angle associated with each link would be a good place to start. A belay rope system can be implemented when agent runs out of energy. A multi-agent system can be created. More heuristics can be explored. The limbs can be made springy. Prismatic joints can be used instead of just revolute joints. The whole system can be extended to 3D.

References

[1] A. Aristidou et al. “Inverse Kinematics Techniques in Computer Graphics: A Survey”. In: *Computer Graphics Forum* 37.6 (2018), pp. 35–58. DOI: `10.1111/cgf.13310`. eprint: `https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13310`. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13310`.

[2] Zeeshan Bhatti et al. *Forward and Inverse Kinematics Seamless Matching Using Jacobian.* 2014. arXiv: `1401.1488 [cs.GR]`.

[3] Ignacy Dulundefinedba and Michał Opałka. “A Comparison of Jacobian-Based Methods of Inverse Kinematics for Serial Robot Manipulators”. In: *Int. J. Appl. Math. Comput. Sci.* 23.2 (June 2013), pp. 373–382. ISSN: 1641-876X. DOI: `10.2478/amcs-2013-0028`. URL: `https://doi.org/10.2478/amcs-2013-0028`.

[4] Lydia Kavraki et al. *Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces.* Tech. rep. Stanford, CA, USA, 1994.

[5] Sven Koenig and Maxim Likhachev. “D*lite”. In: *Eighteenth National Conference on Artificial Intelligence.* Edmonton, Alberta, Canada: American Association for Artificial Intelligence, 2002, pp. 476–483. ISBN: 0262511290.

[6] S. Phaniteja et al. “A deep reinforcement learning approach for dynamically stable inverse kinematics of humanoid robots”. In: *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO).* 2017, pp. 1818–1823.