Report from running Inverse Kinematics using PSO algorithm

Experiment 3

Course: Research project - GPU algorithms
Coordinator: Krzysztof Kaczmarski

Eryk Dobrosz
Marcin Gackowski

June 27, 2019

Description

Project tackles inverse kinematics problem and solves it using Particle Swarm Optimization algorithm implemented with CUDA. Main goals of the project are:

- real-time solving of inverse kinematics problem for arbitrarily large kinematic chains with many effectors
- solution choice that can be used for animation

Output Files

- IK-diagnostics-positions.txt - contains positions of nodes each frame during test cases
- IK-diagnostics-degrees.txt - contains rotations of nodes each frame during test cases
- IK-diagnostics-distance.txt - contains aggregated distance of all effectors from their respective targets each frame during test cases
- IK-diagnostics-frames.txt - contains number of frames required to reach acceptable solution for each test case

Report Goals

- Test implemented changes to fitness function.
- Detect whether system jittering still occurs and find which additional factors could be used by fitness function.

Computational Method

Current iteration uses the same PSO algorithm as previous one with the exception of used fitness function and simulation space constraints.

As opposed to only taking into account distance from targets, currently used fitness function uses also latest kinematic chain configuration. For given coordinates in simulation space, function evaluates differences from latest configuration in each degree of freedom and adds their absolute value multiplied by arbitrary factor to the returned value. This approach ensures that local solutions will be chosen over more distant ones, even though choosing the later could lead to slight improvements in terms of distance to targets.

Another change concerns particle simulation which now uses angle constraints specified in given kinematic chain. Each angle constraint corresponds to either maximum or minimum value possible in given dimension. After each simulation step and before using fitness function, particle position is corrected to fit in the allowed space, thus discarding all solutions that did not met defined angle constraints.

Additionally, fitness function in current iteration also is responsible for collision detection. GJK algorithm is executed for each node-collider pair and if collision is detected examined solution is be discarded entirely and will not be considered in further calculations.

Description of the results

Results were gathered through numerous executions of set test case. System configuration for testing matches test case used in previous iteration:

- Kinematic chain with 21 degrees of freedom, that consists of:
  - origin node placed in (0, 0, 0)
  - 4 consecutive nodes creating a chain with all links of length equal to 1
  - 3 effectors linked to the last node in the chain with links of length equal to 1
- 16384 simulated particles
- 3 targets in set positions that can be reached by all effectors
- aggregated error threshold equal to 0.025 (sum of distances from effectors to their targets)

At the start of each test case system was reset to its default state, providing reproducible initial conditions for the algorithm.

Gathered results allow to verify used algorithm both in terms of performance and stability in approximation of optimal solution. Results for current iteration are as follows:

- frames to reach satisfactory solution:
  - average: 33.1
  - min: 11
  - max: 171
- frame-difference for each degree of freedom (angle in radians):
  - average: 0.024
  - min: 0.00
  - max: 0.15
- frame-difference for each node position (distance in OpenGL units):
  - average: 0.022
  - min: 0.00
  - max: 0.12

Remarks

Results from test case in this iteration show significant improvements in average rotation and position difference that present as follows when compared to previous iteration:

- 8 times smaller rotation difference on average
- 5 times smaller position difference on average
- 1 additional frame needed to reach satisfactory solution on average

Significant increase in average frame count to reach solution is the result of using rotation differences to influence returned fitness function value, however current process of reaching desired solution much more resembles animation with almost no noticeable system jitters.

In this iteration even maximum rotation and position differences are comparable to average differences gathered for previous iteration of the project. This means that algorithm does not suffer from random jumps to entirely separate solution areas and is significantly better suited for use in animations.

Future works

- Use position differences as an additional factor in choosing solutions.
- Optimize fitness function to take advantage of GPU parallelism.