UM EECS 487

Lab 11
Inverse Kinematics and Animation

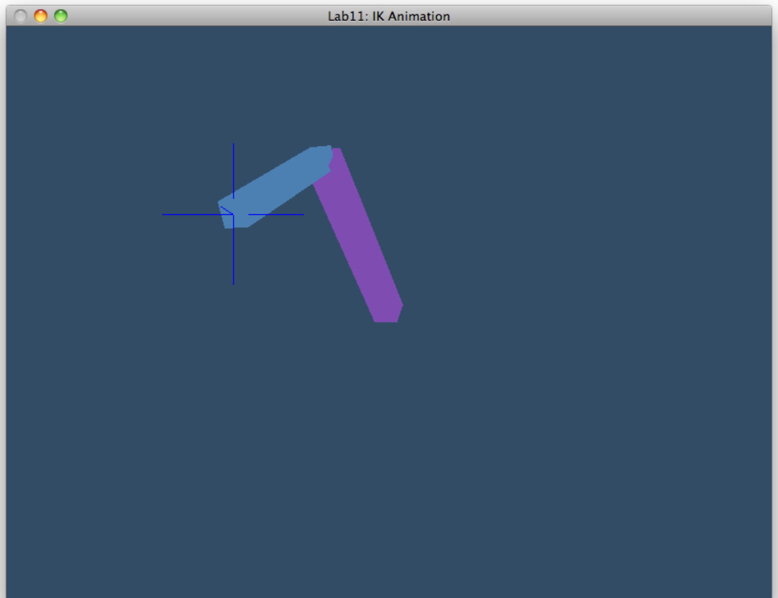

Figure 1. A simple two-bone skeletal arm (with bones depicted as rectangular prisms) with the joint angles determined by the blue locator, the end effoctor, or IK handle.

In this lab you will implement a type of inverse kinematic system often called a *rotation plane solver* (it solves the system in the plane, constrained against the axis and then rotates the whole system into the right location).

1. Inverse Kinematics: A Rotation Plane Solver

An IK handle can be derived simply if only calculated in a 2-dimensional plane. Another further simplication is to assume that the end effector is always on the x-axis. This may seem unreasonable, but the next few steps will take care of it.

1.1. **Axis-Constrained Joint Angles.** Let there be a bone of length a coupled to the origin. At the end of it let there be a second bone of length b. The first step is to find the joint rotation angles (around the z-axis) such that the end of the second bone is at a specified point on the x-axis, $(d, 0)$. This is depicted in the following picture. Observe that θ and ϕ are determined with respect to the *local* horizontal and that one of them is negative.

Search for `/* TODO 0:  */` in `animation.cpp`. Using the law of cosines (see the writeup on vectors and matrices posted on the first day of lecture), set the necessary values for `jointRotation0` and `jointRotation1`, which are θ and ϕ, respectively. The angles for ϕ

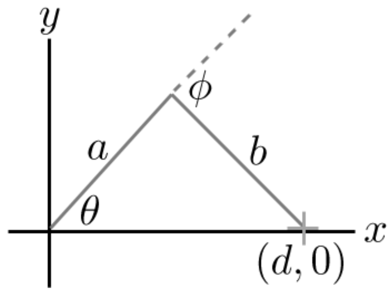

Figure 2. The basic IK setup with the end effector contrained to the x-axis, resulting in a simply geometry.

that is derived will actually need to be π minus it and with rotations being defined right-handedly, the angle will be negative. For d, you should use $||\vec{v}||$ where $\vec{v}$ is the position of the effector.

Make and run the lab. Use the h and l keys to move the ik handle left and right. The joints should rotate to keep the end of the second joint on the handle.

1.2. **Full Motion in the xy-Plane.** The next step is to allow general movement in the xy-plane. Examine the following image.

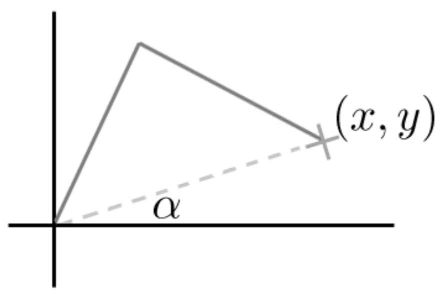

Figure 3. The entire system rotated by α.

Search for `/* TODO 1:  */` in `animation.cpp`. Set `ikInPlaneRotation` to be the α in the image above.

Now search for `/* TODO 2:  */` in `animation.cpp` (it's at the very top). Place the line somewhere inside `drawRobotArm()` such that the rotation occurs ordered properly and the end effector can be moved up k, down j, left h, and right l.

1.3. **Full Space Rotation.** The next step is to allow general movement in 3-D. Examine the following image.

Search for `/* TODO 3:  */` in `animation.cpp`. Set `ikyAxisRotation` to be the β in the image above (it is an angle in the xz-plane).

Now search for `/* TODO 4:  */` in `animation.cpp` (it's at the very top). Place the line somewhere inside `drawRobotArm()` such that the rotation occurs ordered properly and the end effector can be moved up k, down j, left h, right l, forward f, and back b.

1.4. **Pole Rotation.** The next step is to allow rotation of the system around the line between the origin and $\vec{v}$, the location of the end effector.

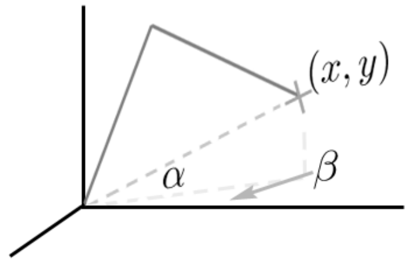

FIGURE 4. The entire system rotated around the y-axis for full 3D spatial movement.

Search for `/* TODO 5:  */` in `animation.cpp` (it's at the very top). Place the line somewhere inside `drawRobotArm()` such that the rotation occurs ordered properly and the end effector can be moved up k, down j, left h, right l, forward f, and back b.

1.5. **Limits.** There are many places in the plane that the end effector does not permit a valid solution (the system cannot *reach*). Provide some behavior of the system for the effector is too close or too far. This only involves modifying `jointRotation0` and `jointRotation1`.

1.6. **Thoughts.** Examine the behavior of the IK system. Notice what happens when the effector is close to the origin. There can be unexpected *flipping* that occurs. Think about why this occurs and if/how it might be bettered.

2. KEYFRAME ANIMATION WITH SPLINES

The lab contains a simple keyframe interpolator. It is based off of most of the code from the previous lab and is definitely worth examining.

TABLE 1. Key Bindings

ESC	Quit the application
h	Move the ik handle left
j	Move the ik handle down
k	Move the ik handle up
l	Move the ik handle right
b	Move the ik handle back
f	Move the ik handle forward
r	Rotate the ik handle air
’ ’	Take a snapshot (add a keyframe)
’.’	Play the animation.
0	Clear out the animation data (keyframes).