VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRICAL ENGINEERING

Modular Simulation of Inverse Kinematics

Done by

16BEE0023 – Varun Singh Inda

Course Code: EEE4027

Course Name: Robotics and Control

Faculty: Prof. Rashmi Ranjan Das

Semester: WINTER'19

Contents

OBJECTIVE

The objective of the project is to simulate an inverse kinematics model consisting of a robotic manipulator that adjusts itself to point where the user drags the mouse.

The position input taken from the mouse is the target for the end-effector. The algorithm calculates the joint angles for the manipulator to reach a certain position using inverse kinematics.

PROBLEM STATEMENT

The problem statement for the project is to create a manipulator that uses the principle of inverse kinematics to reach for the point indicated by the mouse controlled by a user.

The software ‘Processing’ has been used to implement this idea and create a robot manipulator that employs inverse kinematics to reach for a defined point in the plane.

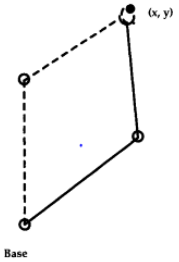
INTRODUCTION

Kinematics is a part of established mechanics that portrays the movement of focuses, bodies, and frameworks of bodies without considering the powers that caused movement. Kinematics, as a field of study, is called the "geometry of movement" and is at times observed as a part of mathematics.

Geometric changes, likewise called unbending changes, are utilized to depict the development of parts in a mechanical framework, rearranging the induction of the conditions of movement. They are additional key to dynamic examination.

The inverse kinematics problem is the problem of finding a vector of joint variables which produce a desired end effector location. If a unique vector of joint angles exists which attains the desired end-effector location, there is a well-defined inverse to the forward kinematics function and the inverse kinematics problem is well-posed. Unfortunately, the inverse kinematics problem can be ill-posed because there is either no solution or because there are many solutions.

In general, ill-posedness of the latter type can be interpreted as arising from two distinct phenomena. First, the inverse is non-unique in a global sense because of the existence of multiple solution branches. This occurs for manipulators both with and without excess DoF. For example, consider the nonredundant two-link planar manipulator with unlimited rotational joints shown in the figure below. There are two configurations which place the end effector at any given point in the reachable workspace.



Second, the inverse kinematics problem for a manipulator with redundant DoF is locally ill-posed in that each solution branch contains an infinite number of solutions. A single inverse solution branch consists of a set of configurations which have a manifold structure in the joint space of dimension equal to the number of redundant degrees of freedom. The existence of such nontrivial preimage manifolds for redundant DoF manipulators allows configuration motions to occur while keeping the end effector fixed at some desired location. Such motions are called manipulator self-motions.

In general, the set of solutions to the inverse kinematics problem for a redundant manipulator consists generically of a finite number of nontrivial self-motion manifolds in the configuration space. Any point in this set solves the task of positioning and orienting the end effector at the target. Control of a redundant manipulator involves selecting one of these branches and selecting a unique point on the branch.

MATHEMATICAL APPROACH

In general the inverse kinematic task of robots of open kinematic chain has only "differential solution" that is based on the use of some "generalized inverse" or "pseudoinverse" of the Jacobian of the arm. Let $q \in IR^n$, $n \in IN$ denote the joint coordinates of an n DoF open kinematic chain, and let $x \in IR^m$, $m \in IN$ be the array made of the Cartesian coordinates of certain points extended with the information on the pose of certain components with respect to the " workshop frame". For the prescription of the motion of the arm the function x(s), $s \in [s_i,s_f] \subset IR$ can be used in which s is a scalar parameter. In this manner a "line" is prescribed in IR^m with the initial and final points at s_i and s_f , respectively. If m > n no exact solution can be expected, but when m < n the existence of ambiguous solutions is expected for a redundant arm. In such cases dealing with the problem of redundancy resolution is of central significance.

The differential solution is provided by the Jacobian $J_{ij}(q) \stackrel{def}{\equiv} \partial x_i / \partial q_j$

$$\frac{dxj}{ds} = \sum_i \frac{\partial xj}{\partial qi} \frac{dqi}{ds} \equiv \sum_i Jji \frac{dqi}{ds}$$

and the initial condition $x(s_i) = x_i n_i$ that normally is known. In the redundant case, when m < n, some "additional idea" is needed to choose one of the possible solutions.

In the case of the "Moore-Penrose Pseudoinverse" a "cost function" $\sum_k \left(\frac{dqk}{ds}\right)^2$ is minimized under the above constraint determined leading to provided that JJ^T is invertible.

$$\frac{dq}{ds} = J^T (JJ^T)^{-1} \frac{dx(q)}{ds}$$

Other generalized inverses based on the Singular Value Decomposition (SVD) are not related to cost function minimization. The application of the "Gram-Schmidt Algorithm" also evaded the minimization of any cost function. If JJ^T is not invertible, i.e. when J^T has non-empty null space, the problem is singular. In this case with the introduction of a small parameter $\mu > 0$ the substitute / approximate solution can be obtained as

$$\frac{dq}{ds} = J^T (JJ^T + \mu I)^{-1} \frac{dx}{ds}$$

since the appropriate matrix in it always has an inverse. If μ is much smaller than the smallest non-zero eigenvalue of JJ^T this means only little deformation far from the singularities. As alternative tackling of the problem of the singularities, Pohl suggested a set of 2nd order equations instead of the linear ones near the singularities. Pohl and Lipkin in 1993 suggested complex extension of the generalized coordinates for task deformation. Both methods have some drawbacks related to mathematical difficulties, and the latter has the additional problem of the physical interpretation of the complex joint coordinate values. Iterative methods for the problem have been suggested as well.

METHODOLOGY

The project is done on a software named processing which includes a front end which is the html file and the backend which includes java. JavaScript libraries have been introduced in the project to make it more interactive and advanced.

Processing is an open-source graphical library and integrated development environment (IDE) playground built for the electronic arts, new media art, and visual design communities with the purpose of teaching non-programmers the fundamentals of computer programming in a visual context. Processing uses the Java language, with additional simplifications such as additional classes and aliased mathematical functions and operations. As well as this, it also has a graphical user interface for simplifying the compilation and execution stage.

JavaScript (p5.js) is a JavaScript port of Processing, a programming language designed to write visualisations, images, and interactive content. It allows web browsers to display animations, visual applications, games and other graphical rich content without the need for a Java applet or Flash plugin. It was discontinued in December 2018. P5.js was originally created to allow existing Processing developers and existing code to work unmodified on web.

Processing Codes

Inversekinematicsfixed.pde

```
Segment end;
Segment start;
PVector base;
void setup() {
  size(600, 400);
  start = new Segment(300, 200, 50, 0);
  Segment current = start;
  for (int i = 0; i < 4; i++) {
    Segment next = new Segment(current, 50, i);
    current.child = next;
    current = next;
  }
  end = current;
  base = new PVector(width/2, height);
}
void draw() {
  background(51);
  end.follow(mouseX, mouseY);
  end.update();
  Segment next = end.parent;
  while (next != null) {
    next.follow();
```

```
    next.update();
    next = next.parent;
  }
  start.setA(base);
  start.calculateB();
  next = start.child;
  while (next != null) {
    next.attachA();
    next.calculateB();
    next = next.child;
  }
  end.show();
  next = end.parent;
  while (next != null) {
    next.show();
    next = next.parent;
  }
}
```

Segment.pde

```
class Segment {
  PVector a;
  float angle = 0;
  float len;
  PVector b = new PVector();
  Segment parent = null;
  Segment child = null;
  float sw = 0;
  Segment(float x, float y, float len_, float i) {
    a = new PVector(x, y);
    sw = map(i, 0, 20, 1, 10);
    len = len_;
    calculateB();
  }
  Segment(Segment parent_, float len_, float i) {
    parent = parent_;
    sw = map(i, 0, 20, 1, 10);
    a = parent.b.copy();
    len = len_;
    calculateB();
  }
```

```
  void setA(PVector pos) {
    a = pos.copy();
  }
  void attachA() {
    setA(parent.b);
  }
  void follow() {
    float targetX = child.a.x;
    float targetY = child.a.y;
    follow(targetX, targetY);
  }
  void follow(float tx, float ty) {
    PVector target = new PVector(tx, ty);
    PVector dir = PVector.sub(target, a);
    angle = dir.heading();
    dir.setMag(len);
    dir.mult(-1);
    a = PVector.add(target, dir);
  }
  void calculateB() {
    float dx = len * cos(angle);
    float dy = len * sin(angle);
    b.set(a.x+dx, a.y+dy);
  }
  void update() {
    calculateB();
  }
  void show() {
    stroke(255);
    strokeWeight(3);
    line(a.x, a.y, b.x, b.y);
  }
}
```

RESULTS


inverse_kinematics
inverse_kinematics
inverse_kinematics
inverse_kinematics

CONCLUSION

The inverse kinematics problem is the problem of finding a vector of joint variables to give the end effector location as desired by the user. In our project, the desired end effector location is indicated by the mouse cursor. HTML, JavaScript and Processing codes can be employed to obtain the desired outputs.

Inverse Kinematics allows us to calculate the joint angles of a manipulator given the desired pose of their end-effector and the dimensions of the links. This method has been used in the project to come up with a robotic arm that follows the cursor at every point by readjusting its joint angles.

The same algorithm can be implemented further into hardware to create actual robotic arms that adjust themselves in order to pick and place objects at a target location. This would have a wide range of applications in warehouse automation and space missions among many others.

REFERENCES

[1] Learning inverse kinematics
A. D'Souza ; S. Vijayakumar ; S. Schaal

[2] Adaptive Solution of the Inverse Kinematic Task by Fixed Point Transformation
Hamza Khan, Aurel Galantai, and Jozsef K. Tar

[3] Inverse Kinematics of Dextrous Manipulators
David DeMers, Kenneth Kreutz-Delgado

[4] Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods
Samuel R. Bus

[5] Inverse Kinematic Solution of Robot Manipulators Using Interval Analysis
R. S. Rao, A. Asaithambi and S. K. Agrawal

MODULAR SIMULATION OF INVERSE KINEMATICS

EEE4027 – Robotics and Control

J Component Review 3

Varun Singh Inda (16BEE0023)

TEAM MEMBERS

Registration Number	Name
16BEE0023	Varun Singh Inda

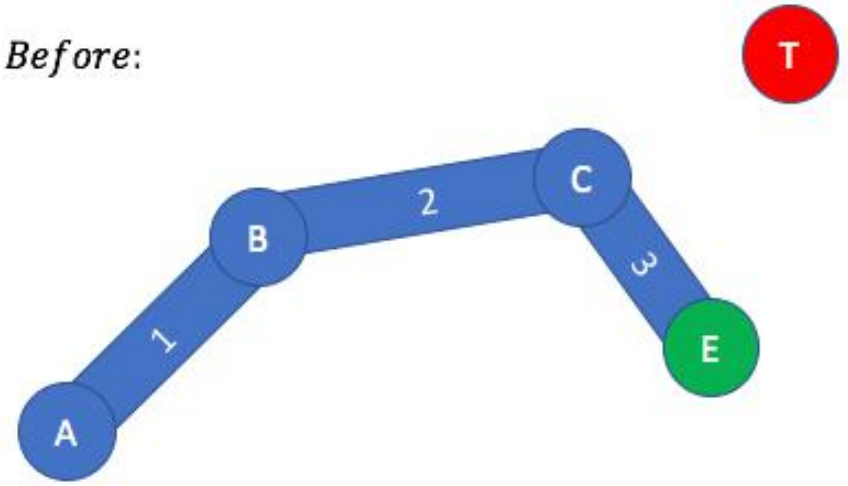
OBJECTIVE

- To simulate an inverse kinematics model consisting of a robotic manipulator that adjusts itself to point where the user drags the mouse
- The position input taken from the mouse is the target for the end-effector.
- The algorithm calculates the joint angles for the manipulator to reach a certain position using inverse kinematics.

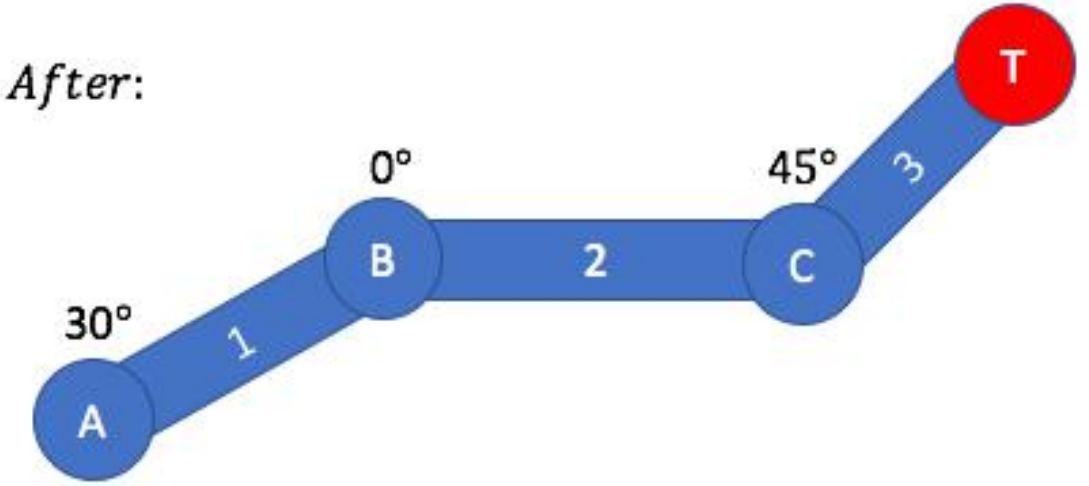
PROBLEM STATEMENT

- To create a manipulator that uses the principle of inverse kinematics to reach for the point indicated by the mouse controlled by a user.
- The software ‘Processing’ has been used to implement this idea and create a robot manipulator that employs inverse kinematics to reach for a defined point in the plane.

Before:



After:



WHAT IS KINEMATICS?

- A part of established mechanics that portrays the movement of focuses, bodies, and frameworks of bodies without considering the powers that caused movement.
- Kinematics, as a field of study, is called the "geometry of movement" and is at times observed as a part of mathematics.
- Geometric changes, likewise called unbending changes, are utilized to depict the development of parts in a mechanical framework, rearranging the induction of the conditions of movement. They are additional key to dynamic examination.

WHAT IS INVERSE KINEMATICS?

- The inverse kinematics problem is the problem of finding a vector of joint variables which produce a desired end effector location.
- If a unique vector of joint angles exists which attains the desired end-effector location, there is a well-defined inverse to the forward kinematics function and the inverse kinematics problem is well-posed.
- Unfortunately, the inverse kinematics problem can be ill-posed because there is either no solution or because there are many solutions.
- The joint angles are adjusted based on the desired final location of the end-effector.

SOLUTIONS FOR THE IK PROBLEM

- The IK problem can be ill-posed because there is either no solution or because there are many solutions.
- This is possible by two means:

1. Inverse is non-unique in a global sense because of the existence of multiple solution branches. This occurs for manipulators both with and without excess DoF.
2. IK problem for a manipulator with redundant DoF is locally ill-posed in that each solution branch contains an infinite number of solutions. A single inverse solution branch consists of configurations which have a manifold structure in the joint space of dimension equal to number of redundant degrees of freedom.

- Set of solutions to the IK problem for a redundant manipulator consists generically of a finite number of solutions in space.
- Any point in this set solves the task of positioning the end effector at the target.

MATHEMATICAL EQUATIONS

- The differential solution is provided by the Jacobian $J_{ij}(q) \stackrel{\text{def}}{=} \partial x_i / \partial q_j$
- $\frac{dxj}{ds} = \sum_i \frac{\partial xj}{\partial qi} \frac{dqi}{ds} \equiv \sum_i Jji \frac{dqi}{ds}$
- $\frac{dq}{ds} = J^T \left(JJ^T\right)^{-1} \frac{dx(q)}{ds}$
- $\frac{dq}{ds} = J^T \left(JJ^T + \mu I\right)^{-1} \frac{dx}{ds}$

METHODOLOGY

- The project is done on a software named ‘Processing’ which includes a front end which is the html file and the backend which includes Java. JavaScript libraries have been introduced in the project to make it more interactive and advanced.

- Processing is an open-source graphical library and integrated development environment (IDE) playground built for the electronic arts, new media art, and visual design communities with the purpose of teaching non-programmers the fundamentals of computer programming in a visual context.

- Processing uses the Java language, with additional simplifications such as additional classes and aliased mathematical functions and operations.

PROCESSING CODES

Inversekinematicsfixed.pde

```
Segment end;
Segment start;
PVector base;
void setup() {
  size(600, 400);
  start = new Segment(300, 200, 50, 0);
  Segment current = start;
  for (int i = 0; i < 4; i++) {
    Segment next = new Segment(current, 50, i);
    current.child = next;
    current = next;
  }
  end = current;
  base = new PVector(width/2, height);
}
void draw() {
  background(51);
  end.follow(mouseX, mouseY);
  end.update();
  Segment next = end.parent;
```

```
 while (next != null) {
  next.follow();
  next.update();
  next = next.parent;
 }
 start.setA(base);
 start.calculateB();
 next = start.child;
 while (next != null) {
  next.attachA();
  next.calculateB();
  next = next.child;
 }
 end.show();
 next = end.parent;
 while (next != null) {
  next.show();
  next = next.parent;
 }
}
```

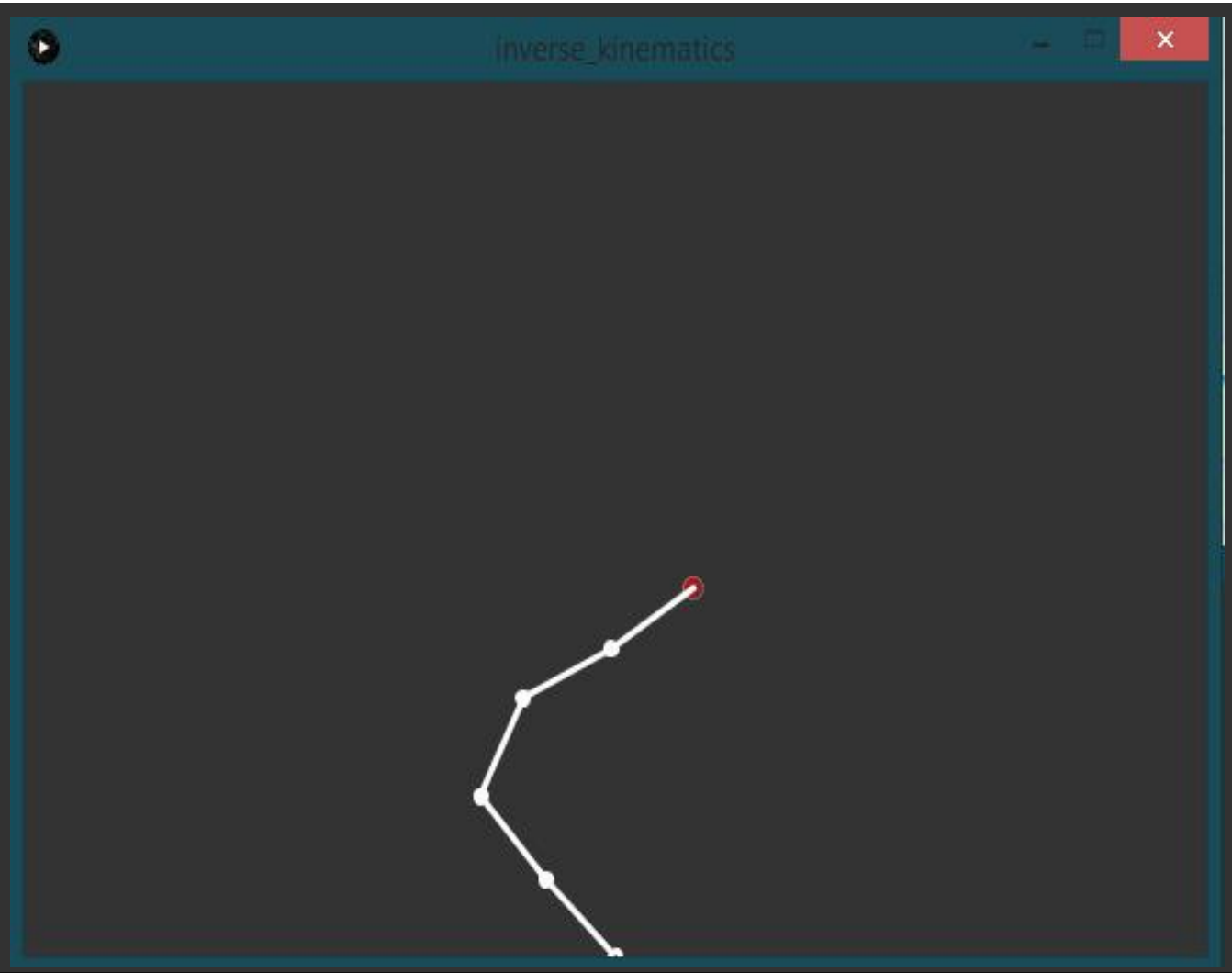
Segment.pde

```
class Segment {
  PVector a;
  float angle = 0;
  float len;
  PVector b = new PVector();
  Segment parent = null;
  Segment child = null;
  float sw = 0;
  Segment(float x, float y, float len_, float i) {
    a = new PVector(x, y);
    sw = map(i, 0, 20, 1, 10);
    len = len_;
    calculateB();
  }
  Segment(Segment parent_, float len_, float i) {
    parent = parent_;
    sw = map(i, 0, 20, 1, 10);
    a = parent.b.copy();
    len = len_;
    calculateB();
  }
```

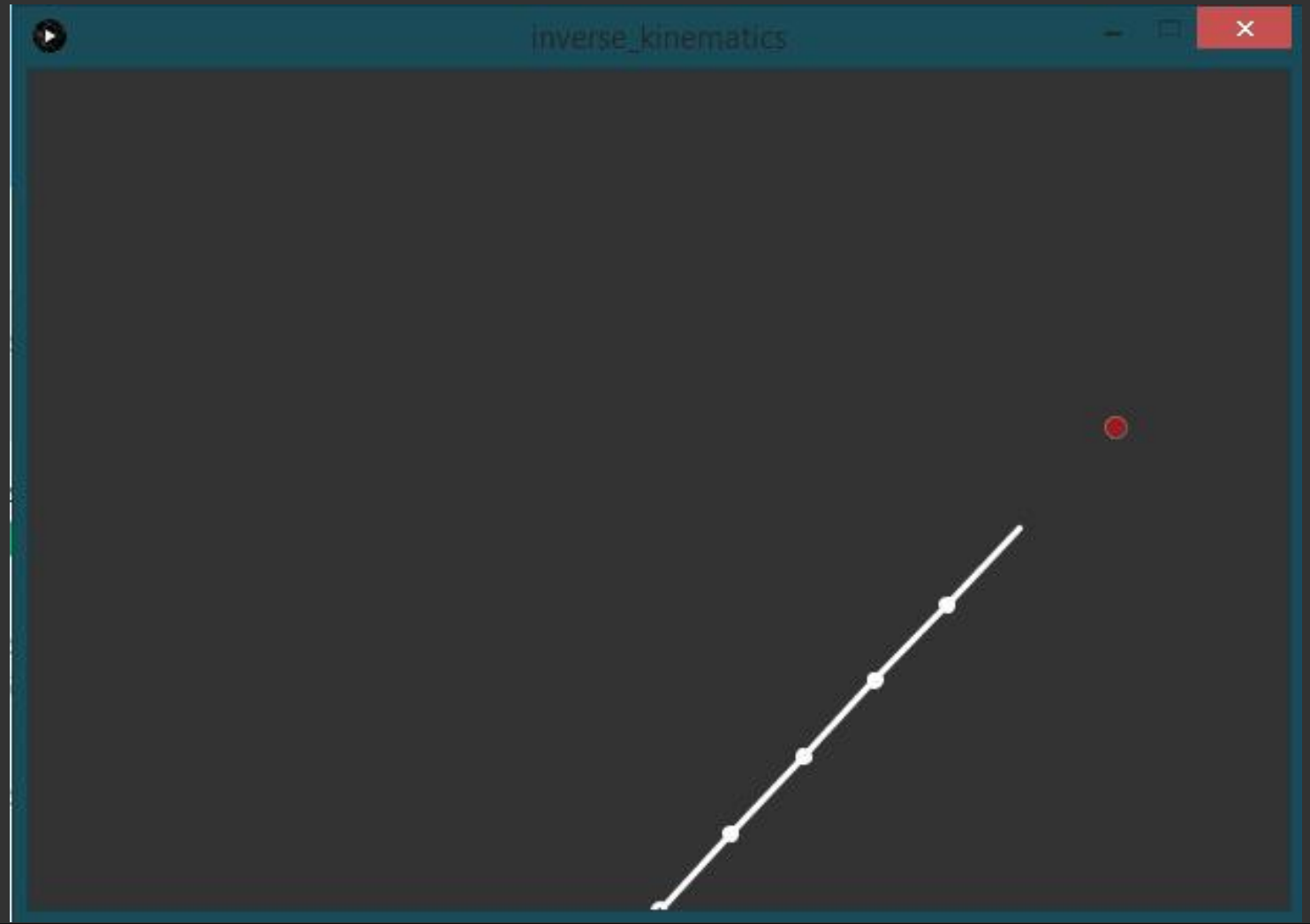
```
void setA(PVector pos) {
  a = pos.copy();
}
void attachA() {
  setA(parent.b);
}
void follow() {
  float targetX = child.a.x;
  float targetY = child.a.y;
  follow(targetX, targetY);
}
void follow(float tx, float ty) {
  PVector target = new PVector(tx, ty);
  PVector dir = PVector.sub(target, a);
  angle = dir.heading();
  dir.setMag(len);
  dir.mult(-1);
  a = PVector.add(target, dir);
}
```

```
  void calculateB() {
    float dx = len * cos(angle);
    float dy = len * sin(angle);
    b.set(a.x+dx, a.y+dy);
  }
  void update() {
    calculateB();
  }
  void show() {
    stroke(255);
    strokeWeight(3);
    line(a.x, a.y, b.x, b.y);
  }
}
```

RESULT CAPTURE 1

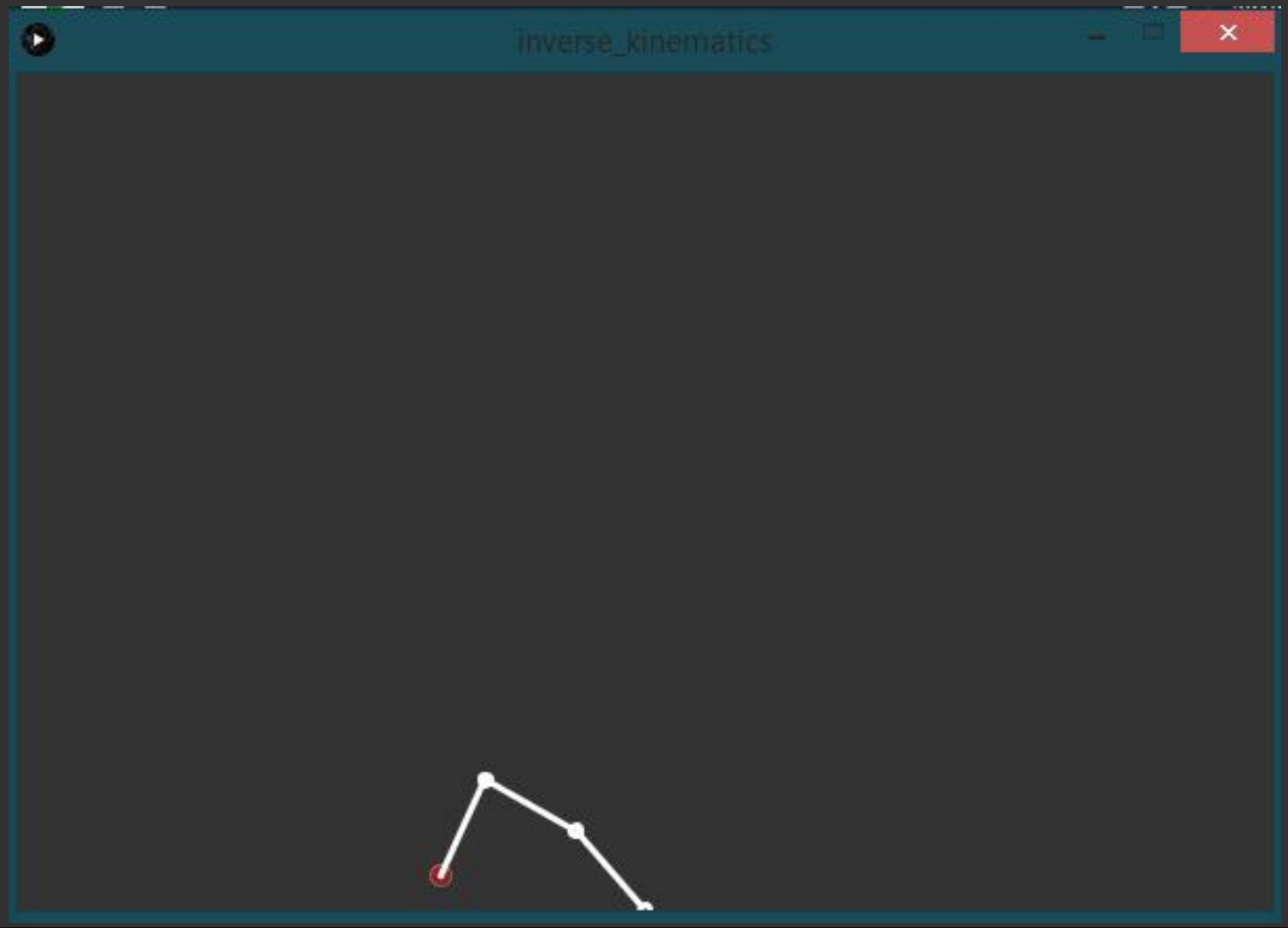


RESULT CAPTURE 2



RESULT CAPTURE 3

RESULT CAPTURE 4



CONCLUSION

- The IK problem involves finding a vector of joint variables to give the end effector location as desired by the user.
- The desired end effector location is indicated by the mouse cursor.
- Inverse Kinematics allows us to calculate the joint angles of a manipulator given the desired pose of their end-effector and the dimensions of the links.
- This method has been used in the project to come up with a robotic arm that follows the cursor at every point by readjusting its joint angles.
- The same algorithm can be implemented further into hardware to create actual robotic arms that adjust themselves in order to pick and place objects at a target location.
- This would have a wide range of applications in warehouse automation and space missions among many others.

REFERENCES

- [1] Learning inverse kinematics

  A. D'Souza ; S. Vijayakumar ; S. Schaal

- [2] Adaptive Solution of the Inverse Kinematic Task by Fixed Point Transformation

  Hamza Khan, Aurel Galantai, and Jozsef K. Tar

- [3] Inverse Kinematics of Dextrous Manipulators

  David DeMers, Kenneth Kreutz-Delgado

- [4] Introduction to Inverse Kinematics with Jacobian Transpose, Pseudoinverse and Damped Least Squares methods

  Samuel R. Bus

- [5] Inverse Kinematic Solution of Robot Manipulators Using Interval Analysis

  R. S. Rao, A. Asaithambi and S. K. Agrawal