Cobot Simulator

Damian Dhesi, Reza Mousakhani, Shiv Panchal

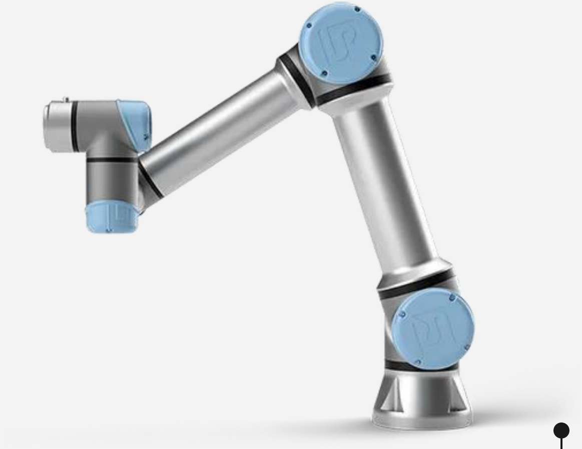


Table of contents

01

Introduction

02

Demo

03

Software Design

04

Code Metrics

05

Deployment

06

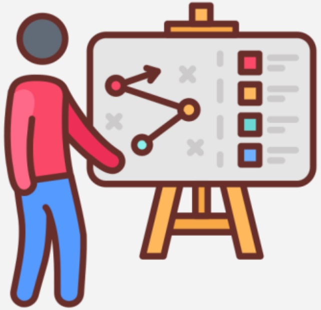
Conclusion

01

Introduction

Project Proposal

- Goal
  - Image --> outline (contours) --> x,y points --> 6 angles for MoveJ() --> Cobot moving
- Python for image processing
- Java for contours --> x,y points --> angles
- TCP for connecting to Cobot
- Seamlessly transition from image --> Cobot drawing it



02

Demo

03

Software Design

Architectural Design / UML Diagram

Architecture

- MVC
- Blackboard
- Client – Server

Design

- Delegate

Lucidchart

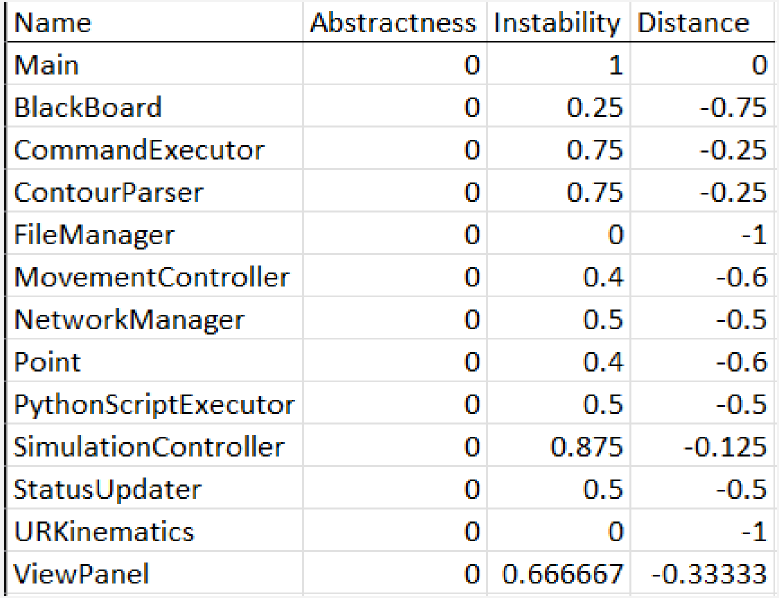
We used Lucidchart to create UML diagram for collaboration purposes

UML Diagram

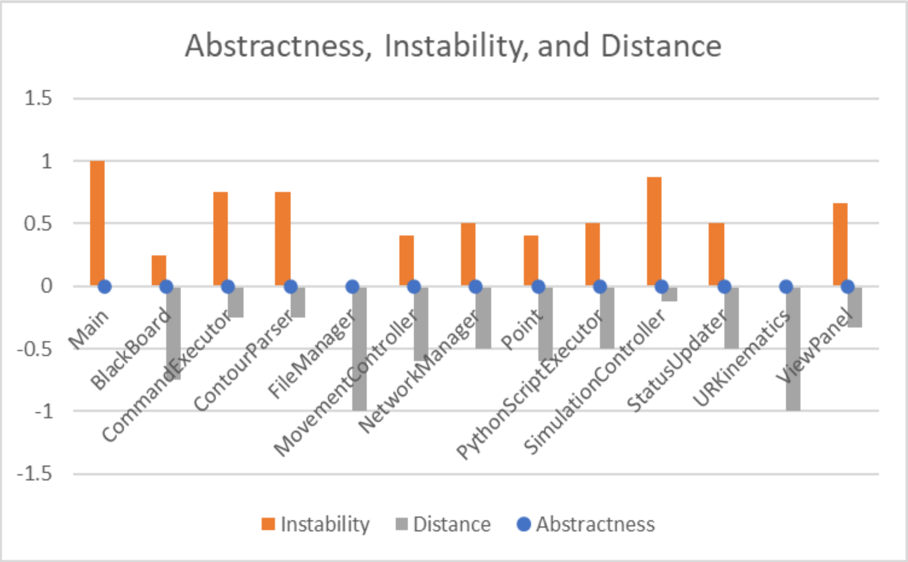
04

Metrics

Metrics: Abstractness, Instability, Distance



Name	Abstractness	Instability	Distance
Main	0	1	0
BlackBoard	0	0.25	-0.75
CommandExecutor	0	0.75	-0.25
ContourParser	0	0.75	-0.25
FileManager	0	0	-1
MovementController	0	0.4	-0.6
NetworkManager	0	0.5	-0.5
Point	0	0.4	-0.6
PythonScriptExecutor	0	0.5	-0.5
SimulationController	0	0.875	-0.125
StatusUpdater	0	0.5	-0.5
URKinematics	0	0	-1
ViewPanel	0	0.666667	-0.33333

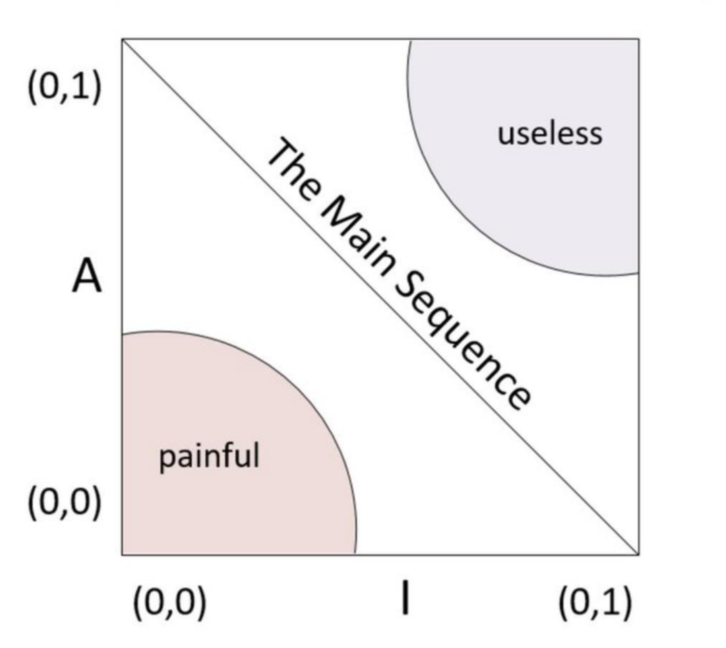


Metrics: Abstractness, Instability, Distance

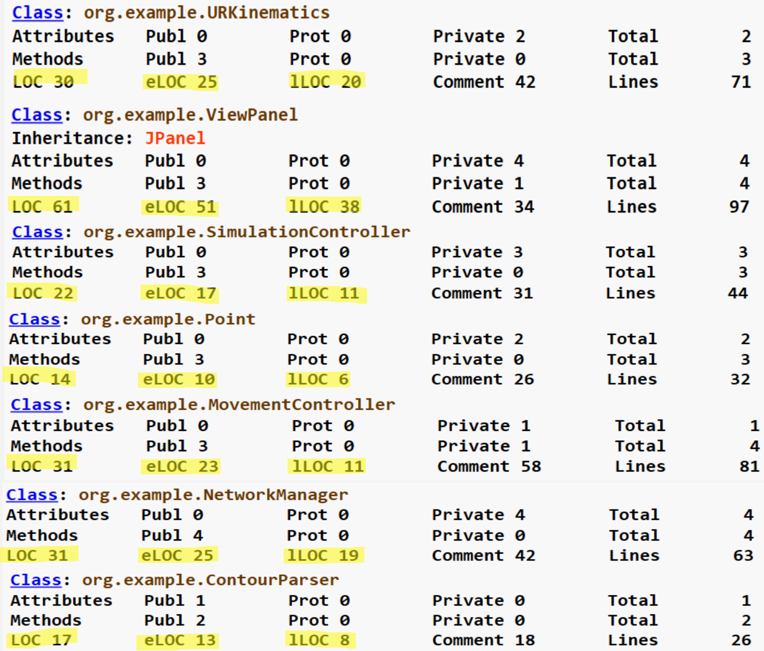
- Majority of classes are within the bounds of "The Main Sequence"

  Abstractness was 0 for all classes. An improvement would be more abstractness for certain portions of the design

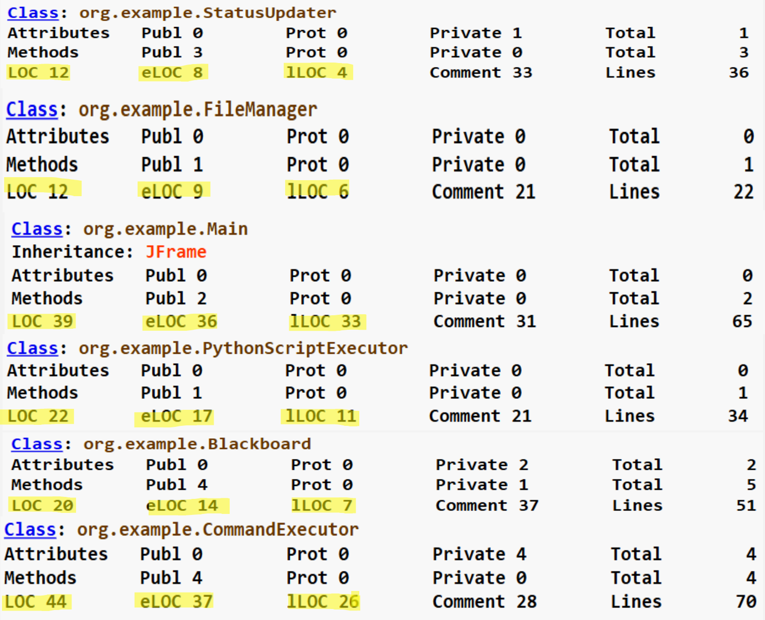
- FileManager & URKinematics were the only classes in the "Painful" zone



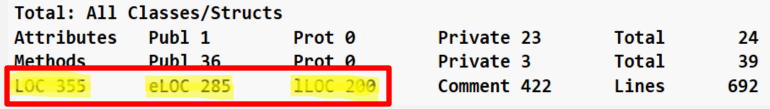
Code Metrics



```
Class: org.example.URKinematics
Attributes    Publ 0      Prot 0      Private 2      Total      2
Methods       Publ 3      Prot 0      Private 0      Total      3
LOC 30        eLOC 25     lLOC 20     Comment 42     Lines     71

Class: org.example.ViewPanel
Inheritance: JPanel
Attributes    Publ 0      Prot 0      Private 4      Total      4
Methods       Publ 3      Prot 0      Private 1      Total      4
LOC 61        eLOC 51     lLOC 38     Comment 34     Lines     97

Class: org.example.SimulationController
Attributes    Publ 0      Prot 0      Private 3      Total      3
Methods       Publ 3      Prot 0      Private 0      Total      3
LOC 22        eLOC 17     lLOC 11     Comment 31     Lines     44

Class: org.example.Point
Attributes    Publ 0      Prot 0      Private 2      Total      2
Methods       Publ 3      Prot 0      Private 0      Total      3
LOC 14        eLOC 10     lLOC 6      Comment 26     Lines     32

Class: org.example.MovementController
Attributes    Publ 0      Prot 0      Private 1      Total      1
Methods       Publ 3      Prot 0      Private 1      Total      4
LOC 31        eLOC 23     lLOC 11     Comment 58     Lines     81

Class: org.example.NetworkManager
Attributes    Publ 0      Prot 0      Private 4      Total      4
Methods       Publ 4      Prot 0      Private 0      Total      4
LOC 31        eLOC 25     lLOC 19     Comment 42     Lines     63

Class: org.example.ContourParser
Attributes    Publ 1      Prot 0      Private 0      Total      1
Methods       Publ 2      Prot 0      Private 0      Total      2
LOC 17        eLOC 13     lLOC 8      Comment 18     Lines     26
```

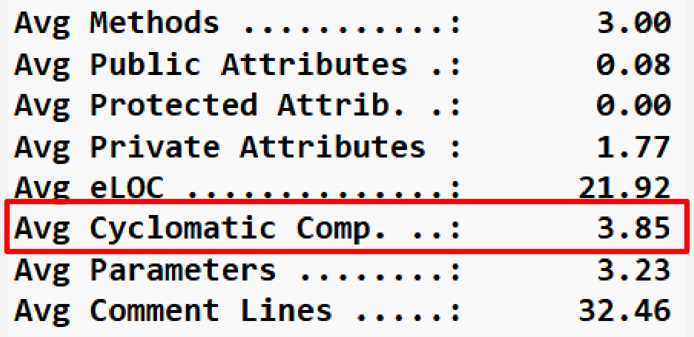


```
Class: org.example.StatusUpdater
Attributes    Publ 0      Prot 0      Private 1      Total      1
Methods       Publ 3      Prot 0      Private 0      Total      3
LOC 12        eLOC 8      lLOC 4      Comment 33     Lines     36

Class: org.example.FileManager
Attributes    Publ 0      Prot 0      Private 0      Total      0
Methods       Publ 1      Prot 0      Private 0      Total      1
LOC 12        eLOC 9      lLOC 6      Comment 21     Lines     22

Class: org.example.Main
Inheritance: JFrame
Attributes    Publ 0      Prot 0      Private 0      Total      0
Methods       Publ 2      Prot 0      Private 0      Total      2
LOC 39        eLOC 36     lLOC 33     Comment 31     Lines     65

Class: org.example.PythonScriptExecutor
Attributes    Publ 0      Prot 0      Private 0      Total      0
Methods       Publ 1      Prot 0      Private 0      Total      1
LOC 22        eLOC 17     lLOC 11     Comment 21     Lines     34

Class: org.example.Blackboard
Attributes    Publ 0      Prot 0      Private 2      Total      2
Methods       Publ 4      Prot 0      Private 1      Total      5
LOC 20        eLOC 14     lLOC 7      Comment 37     Lines     51

Class: org.example.CommandExecutor
Attributes    Publ 0      Prot 0      Private 4      Total      4
Methods       Publ 4      Prot 0      Private 0      Total      4
LOC 44        eLOC 37     lLOC 26     Comment 28     Lines     70
```

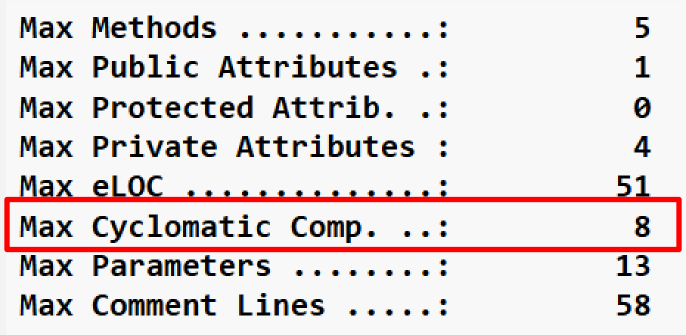


```
Total: All Classes/Structs
Attributes    Publ 1      Prot 0      Private 23     Total     24
Methods       Publ 36     Prot 0      Private 3      Total     39
LOC 355       eLOC 285    lLOC 200    Comment 422    Lines    692
```

Code Metrics: Cyclomatic Complexity



```
Avg Methods ...........:        3.00
Avg Public Attributes .:        0.08
Avg Protected Attrib. .:        0.00
Avg Private Attributes :        1.77
Avg eLOC ..............:       21.92
Avg Cyclomatic Comp. ..:        3.85
Avg Parameters ........:        3.23
Avg Comment Lines .....:       32.46
```



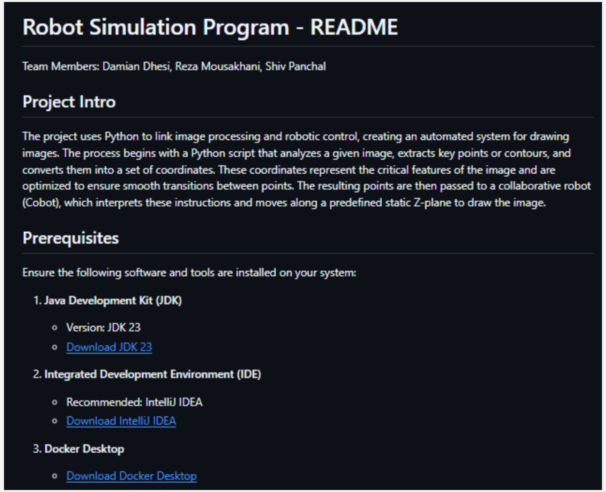
```
Max Methods ...........:           5
Max Public Attributes .:           1
Max Protected Attrib. .:           0
Max Private Attributes :           4
Max eLOC ..............:          51
Max Cyclomatic Comp. ..:           8
Max Parameters ........:          13
Max Comment Lines .....:          58
```

05

Deployment

Deployment

README.md



Robot Simulation Program - README

Team Members: Damian Dhesi, Reza Mousakhani, Shiv Panchal

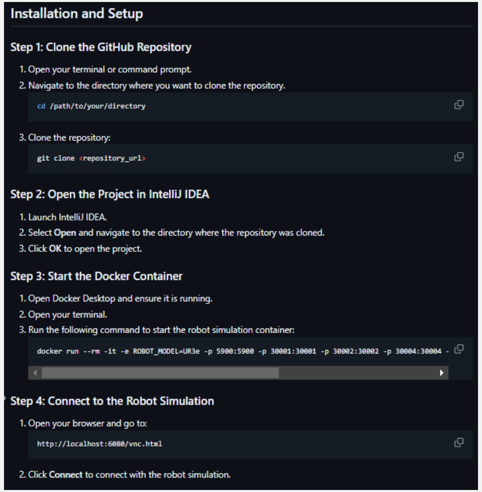
Project Intro

The project uses Python to link image processing and robotic control, creating an automated system for drawing images. The process begins with a Python script that analyzes a given image, extracts key points or contours, and converts them into a set of coordinates. These coordinates represent the critical features of the image and are optimized to ensure smooth transitions between points. The resulting points are then passed to a collaborative robot (Cobot), which interprets these instructions and moves along a predefined static Z-plane to draw the image.

Prerequisites

Ensure the following software and tools are installed on your system:

1. **Java Development Kit (JDK)**
   - Version: JDK 23
   - Download JDK 23
2. **Integrated Development Environment (IDE)**
   - Recommended: IntelliJ IDEA
   - Download IntelliJ IDEA
3. **Docker Desktop**
   - Download Docker Desktop



Installation and Setup

Step 1: Clone the GitHub Repository

1. Open your terminal or command prompt.
2. Navigate to the directory where you want to clone the repository.

```
cd /path/to/your/directory
```

3. Clone the repository:

```
git clone <repository_url>
```

Step 2: Open the Project in IntelliJ IDEA

1. Launch IntelliJ IDEA.
2. Select **Open** and navigate to the directory where the repository was cloned.
3. Click **OK** to open the project.

Step 3: Start the Docker Container

1. Open Docker Desktop and ensure it is running.
2. Open your terminal.
3. Run the following command to start the robot simulation container:

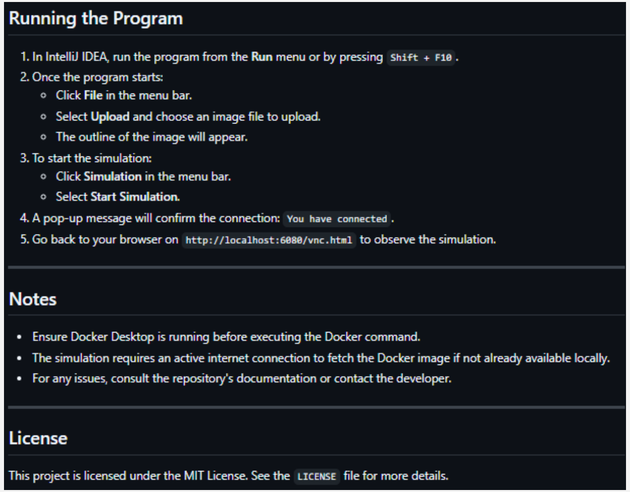
```
docker run --rm -it -e ROBOT_MODEL=UR3e -p 5900:5900 -p 30001:30001 -p 30002:30002 -p 30004:30004 -
```

Step 4: Connect to the Robot Simulation

1. Open your browser and go to:

```
http://localhost:6080/vnc.html
```

2. Click **Connect** to connect with the robot simulation.



Running the Program

1. In IntelliJ IDEA, run the program from the **Run** menu or by pressing `Shift + F10`.
2. Once the program starts:
   - Click **File** in the menu bar.
   - Select **Upload** and choose an image file to upload.
   - The outline of the image will appear.
3. To start the simulation:
   - Click **Simulation** in the menu bar.
   - Select **Start Simulation.**
4. A pop-up message will confirm the connection: `You have connected`.
5. Go back to your browser on `http://localhost:6080/vnc.html` to observe the simulation.

Notes

- Ensure Docker Desktop is running before executing the Docker command.
- The simulation requires an active internet connection to fetch the Docker image if not already available locally.
- For any issues, consult the repository's documentation or contact the developer.

License

This project is licensed under the MIT License. See the `LICENSE` file for more details.

06

Conclusion

Future Developers

- The code demonstrates high quality
- We achieved a low cyclomatic complexity, ensuring that functions are simple and easy to follow
- Majority of our lines of code are kept under 100 characters, with a significant portion being effective lines of code (eLOC), focused on delivering functionality rather than clutter
- The code is highly readable, easily modifiable, and extendable
- We did this to ensure that maintainability and adding enhancements are possible