Mechatronics System Design
Assignment-2

Part-1: CAD

Part-1 Overview

In this, you will design a four-bar linkage mechanism using Fusion 360. This practical exercise will help you understand the principles of mechanism design and kinematic analysis in the context of mechatronic systems.

Design Requirements

1. Design a four-bar coupling mechanism of crank-rocker-rocker configuration:
2. Each bar must be created as a separate component in Fusion 360
3. Components must be properly connected using revolute joints
4. The mechanism must satisfy Grashof's condition to ensure proper motion
5. The design should be functional and demonstrate smooth motion throughout the crank's rotation

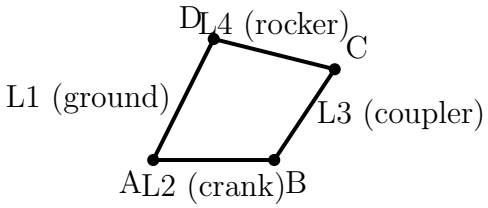

Figure 1: Schematic representation of a four-bar linkage mechanism

Design Process Guidelines

1. Create a new design in Fusion 360
2. Design each link as a separate component:
    - Ground link (L1): Fixed to the global coordinate system
    - Input link/Crank (L2): Connected to the ground at one end
    - Coupler link (L3): Connected between the crank and the rocker
    - Output link/Rocker (L4): Connected to the ground at one end
3. Use revolute joints to connect the components:
    - Joint 1: Ground (L1) to Crank (L2)
    - Joint 2: Crank (L2) to Coupler (L3)
    - Joint 3: Coupler (L3) to Rocker (L4)
    - Joint 4: Rocker (L4) to Ground (L1)
4. Test the mechanism by rotating the crank through a full 360° rotation
5. Verify that the mechanism moves as expected and does not lock up

Submission Requirements

You must submit the following:

1. The complete Fusion 360 design file (.f3d) of your four-bar linkage mechanism
2. A video (maximum 30 seconds) showing the movement of your mechanism with the crank rotating a full 360°

Important Notes

- The crank must be able to complete a full 360° rotation without the mechanism binding
- Ensure your design follows Grashof's condition: $S + L < P + Q$ where S is the shortest link length, L is the longest link length, and P and Q are the remaining link lengths

- For optimal visualization in your video, consider adding colors to differentiate the links
- You may add additional features to your design for improved functionality or aesthetics, but the basic four-bar linkage must function as required

Resources

- Fusion 360 Joint Types: https://help.autodesk.com/view/fusion360/ENU/?guid=GUID-8818AE31-958A-4A59-989B-9875A174C67A
- Fusion 360 linkage mechanisms: https://www.youtube.com/watch?v=Ct6RUt9LhTQ
- Fusion 360 Motion Study Reference: https://www.youtube.com/watch?v=JWJI8yzPUdU

.....

Part-2: Ground Vehicle Data Reconstruction Using Sensor Fusion

Objective

The objective of this part is to reconstruct the trajectory of a ground vehicle based on sensor data collected from a wheel encoder, magnetometer, and IMU. The students are expected to use the sensor data to recreate the vehicle's motion for two different paths.

Dataset Description

The dataset consists of time-series sensor readings collected during the motion of the ground vehicle. The data is in CSV format with the following columns:

Table 1: Dataset Description

Column	Description
`timestamp`	Time of data collection

Column	Description
left_encoder_count	Left wheel encoder count
left_encoder_speed	Left wheel encoder speed (pulses/s)
right_encoder_count	Right wheel encoder count
right_encoder_speed	Right wheel encoder speed (pulses/s)
accel_x	X-axis acceleration (m/s^2)
accel_y	Y-axis acceleration (m/s^2)
accel_z	Z-axis acceleration (m/s^2)
gyro_x	X-axis angular velocity (rad/s)
gyro_y	Y-axis angular velocity (rad/s)
gyro_z	Z-axis angular velocity (rad/s)
imu_temp	IMU temperature (°C)
mag_x	X-axis magnetic field (µT)
mag_y	Y-axis magnetic field (µT)
mag_z	Z-axis magnetic field (µT)
heading	Heading angle (degrees)

Task

1. **Pre-processing the Data:**
   - Load the CSV file into your preferred programming language.
   - Perform data cleaning if necessary (e.g., handle missing or noisy data).
2. **Wheel Encoder-Based Odometry:**
   - Use left and right encoder counts and speed to estimate the vehicle's displacement and orientation.
   - Note that this is a tracked robot; hence, a wheel radius is not required. Instead, each encoder pulse corresponds to approximately 0.095 mm of linear movement of the track.
   - You will also need the distance between the tracks (track width) to accurately estimate the orientation based on differential track movement. The inner tracks measure 9 cm, the outer tracks measure 12 cm, and the centre-to-centre distance is 10.5 cm.
3. **IMU-Based Motion Estimation:**
   - Use accelerometer and gyroscope data to estimate velocity and orientation.

- Integrate the accelerometer readings to estimate displacement.

4. **Magnetometer-Based Heading Correction:**
   - Use magnetometer readings to estimate heading direction.
   - Fuse the heading data with IMU readings to improve orientation accuracy.
5. **Sensor Fusion:**
   - Combine the data from wheel encoders, IMU, and magnetometer to reconstruct the complete trajectory.
   - You can use complementary filtering or other sensor fusion techniques.
6. **Visualization:**
   - Plot the reconstructed 2D path of the vehicle.
   - Include key points on the trajectory.
7. **Step-by-Step Reconstruction:**
   - First, reconstruct simple paths using the provided CSV files below
   - Path-1: **Link of first path**
   - Path-2: **Link of second path**
   - Then, reconstruct the path based on the data collected during the lab experiment(We will share this team-wise data also shortly).
8. **Report Submission:**
   - Submit your code along with a report describing the methodology, assumptions, and observations.
   - Include analysis of how each sensor contributes to the final reconstruction.

Part-3: Results on LIDAR, ToF and Camera

LIDAR Data Analysis

Tasks:

1. **Visualization:**
   - Write a code that visualizes the data collected in the two LIDAR files provided(we will provide it shortly). Your code should generate a clear 2D outline map of the scanned environment.
   - You may use your preferred programming language (e.g., Python, MATLAB) along with relevant libraries (e.g., Matplotlib, PCL, OpenCV) to process and plot the data.
2. **Observations:**
   - From the 2D outline map generated by your visualization code, provide a detailed description of your observations. Discuss any features, structures, or anomalies you can identify in the map.
3. **Detection of Black Objects:**
   - Based on the common LIDAR data provided(included in zip file), explain why black objects are not getting detected in the plotted output. In your explanation, consider factors such as the reflectivity of black surfaces and the wavelength characteristics of the LIDAR sensor.
   - Refer to the link given to access the data: **Common Data**. Note that there will be two images in the extracted folder, namely `common_1.jpeg` and `common_2.jpeg`. Both are images of the same setup but taken from different angles.

Instructions and Submission Requirements

- Include screenshots or images of the generated 2D outline map in your report.
- Provide detailed observations with technical explanations for the behavior observed with black objects.
- Submit your source code, visualizations, and report, ensuring each section is clearly labeled according to the tasks.

ZED Stereo Camera

Objective

In this part, you are provided with two sets of a color image, a depth heatmap image, and a depth matrix (saved as a `.npy` file) captured using the ZED stereo camera. Your task is to reconstruct a 3D point cloud from these data sources using the provided camera intrinsic parameters.

Data Folder Link: Click Here

Provided Data

- **Color Image:** `lab_color.png, lab2_color.png`
- **Depth Heatmap:** `lab_depth_heatmap.png, lab2_depth_heatmap.png`
- **Depth Matrix:** `lab_depth_matrix.npy, lab2_depth_matrix.npy`

Camera Intrinsic Parameters

Use the following ZED camera parameters:

- $f_x = 957.08$
- $f_y = 957.08$
- $c_x = 649.15$
- $c_y = 370.98$

Task Instructions

1. **Data Loading:** Load the color image, depth heatmap, and depth matrix using your preferred programming language (e.g., Python with OpenCV and NumPy).
2. **Point Cloud Reconstruction:** Reconstruct the 3D point cloud using the following approaches:
    - **Method 1:** Use the depth matrix directly. For each pixel (u, v) with depth Z, compute the 3D coordinates using:

$$X = \frac{(u - c_x) \cdot Z}{f_x}, \quad Y = \frac{(v - c_y) \cdot Z}{f_y}, \quad Z = \text{depth value}$$

- **Method 2:** Use the depth heatmap. Convert the heatmap to a grayscale image to approximate normalized depth values, then map these values back to the actual depth range. Specifically, if the depth normalization was done using:

$$\text{depth\_norm} = \frac{\text{depth} - \text{min\_depth}}{\text{max\_depth} - \text{min\_depth}} \times 255$$

then invert the process using:

$$\text{depth} = \frac{\text{gray}}{255} \times (\text{max\_depth} - \text{min\_depth}) + \text{min\_depth}$$

where $\text{min\_depth} = 0.1\,\text{m}$ and $\text{max\_depth} = 10.0\,\text{m}$.

3. **Visualization:** Save your reconstructed point cloud in PLY format. Visualize the point cloud using any point cloud viewer (e.g., MeshLab, CloudCompare) and include screenshots in your report.

4. **Report:** Describe your reconstruction methodology, discuss any challenges faced, and compare the outcomes if you implemented both options.

Time-of-Flight (ToF) Sensor Experiment with Multiple Objects

Objective

In this part, you are provided with:

- A CSV log file containing distance measurements from a Terabee ToF sensor.
- An image showing the setup with four objects placed in front of the sensor at 0.5m, 1m, 1.55m and 2.21m.
- **Data Folder Link** **Click Here**

The objects were removed one by one, and the sensor continued logging measurements over time. Your task is to analyze these logs and observe how the measured distance changes when each object is removed.

Provided Data

- **CSV Log File:** `tof_benchmark.csv` (includes `Timestamp` and `Distance(mm)`)
- **Setup Image:** `tof_setup.jpg` (shows initial placement of four objects in a line)

Task Instructions

1. **Inspect the Data:**
    - Open the CSV file to understand its structure (timestamp, distance).
2. **Plot Distance vs. Time:**
    - Create a plot with *time* on the x-axis and *distance* (in mm) on the y-axis.
3. **Analyze the Distance Changes:**
    - Observe how the measured distance jumps to the next object once the nearest object is removed.
4. **Discussion:**
    - Discuss any discrepancies or noise in the data.
    - Explain possible reasons for differences between measured and actual distances (e.g., sensor accuracy, object reflectivity, or environmental factors).

Submission Requirements

- **Plot:** A distance vs. time plot indicating the moments when each object was removed.
- **Short Discussion:**
    - Summarize your observations of how the distance readings change over time.
    - Comment on the sensor's accuracy relative to the known positions.
- **Code (if used):** Submit any scripts you used for data processing or visualization.

Deadline: 10th March, 11:59 PM.

If you have any questions regarding this assignment, please contact any of the TAs.