Assignment 2: Utilising Abstraction for a Range of Sensor Classes

Intent: Skills in utilising, classes, functions, pointers and utilising abstraction, encapsulation, inheritance, polymorphism with appropriate documentation will be assessed.

Individual Task

Weight: 20%

Task: Write a program in C++ using object oriented paradigms that embodies a range of Mechatronics sensors, utilising abstraction, encapsulation, inheritance and polymorphism. Supply appropriate auto-generated documentation utilising inline source mark-up.

Rationale: In a Mechatronics System, sensors producing data of similar nature are often abstracted from a base sensor (ie. sensor that produce range to the sensor surrounding such as a radar/laser/sonar). This allows treating a suite of sensors in an abstract manner when processing the data, the remaining of our processing code can be agnostic to the sensor type.

Your task is to:
(a) Create a base sensor class and expand it to a range of sensors
(b) Create a class to process this data that is agnostic to the sensor type and process that data to produce data that is fused from sensor output.

Due: Sunday 22nd April 23:59.

Specifics

The physical sensors are collocated, the spatial separation of sensors can be disregarded.

A) Create a Base Sensor Class (called Ranger) and three derived Sensor Classes (Laser and two Radar's) that contain:

1. X Offset of Sensor from Origin
2. Orientation Offset of Sensor from Origin
3. Field of View of Sensor
4. Angular Resolution
5. Number of Samples
6. Data
7. USB Port for Connection

B) Each Sensor will need to

1. Initialise all the required variables to enable connecting to the sensor
2. Enable obtaining all the hardware specific fixed parameters of the sensor
3. Enable setting configurable parameters of the sensor
4. Inform if the values to be set are sane, use default values otherwise
5. Obtaining sample sensor data and sensor resolution within specific sensor sensing range: generated as random value from a normal distribution [refer: http://www.cplusplus.com/reference/random/normal_distribution/] (mean 6m and standard deviation 5m)

C) The RangerFusion Class will need to

1. On creation be set to a fusion method (min/max/average)
2. Accept a STL container of Sensors
3. Produce a fusion of sensor readings at the resolution of the laser with specified fusion method and dealing with readings that are on the boundary of the sensing range (max range).
4. Return an STL container of fused data
5. Return an STL container of raw unfused data
6. You will need to base your classes (inherit from) the **RangerFusionInterface class** and implement its virtual functions.

D) Create a Main that

1. Initialises the sensors (1 Laser, 2 Radar)
2. Queries the fixed sensor parameters
3. Sets sensor parameters as specified by the user
4. Continuously calls (each second) the RangerFusion class, and obtains the fused data output

Assessment Criteria Specifics

Criteria	Weight (%)	Description / Evidence
Sensor classes exploits abstraction (encapsulation, inheritance and polymorphism) to cover a range of sensors	40	Inheritance from base class, common data stored and generic functionality implemented solely in base class (no duplication). Classes that should not be available for instantiation are aptly protected.
Proper code execution	30	User input of parameters works as expected. Data values reported as per sensor description (values and timing).
Documentation	10	ALL classes contain comments to understand methods, members and inner working (ie border case handling of fusion)
Modularity of software	20	Appropriate use class declarations, definitions, default values and naming convention allowing for reuse. No implicit coupling between classes that disables reuse. Sensor classes interface in ways allowing use of class in others contexts and expansion (adding more sensors). No dependency on ordering of sensors, no “hard coded” values or assumptions of sensor order in fusion class.

Sensor 1 – Laser Rangefinder Scanner

Specifications

Model	UTM-XXL
Baud	38400 or 115200
Port	USB (typically /dev/ttyACMX) …where X=0,1,2
Field of View	180 degrees
Angular Resolution	10 or 30 degrees
Max Distance	8.0m
Min Distance	0.2m

Sensor 2 – Radar

Specifications

Model	RAD-001
Baud	38400 or 115200
Port	USB (typically /dev/ttyACMX) …where X=0,1,2
Field of View	60 degrees
Angular Resolution	20 degrees
Max Distance	16.0m
Min Distance	0.2m

Sensor Configurations

The sensors can have a spatial separation. Two sensor configuration examples are presented below.

Sensor Configuration 1 (Collocated Sensors)

Sensor	Field of View [degrees]	Angular Resolution [degrees]	Orientation offset [degrees]	X offset [m]
Laser	180	30	0.0	0.0
Radar1	60	20	-30.0	0.0
Radar2	60	20	+30.0	0.0

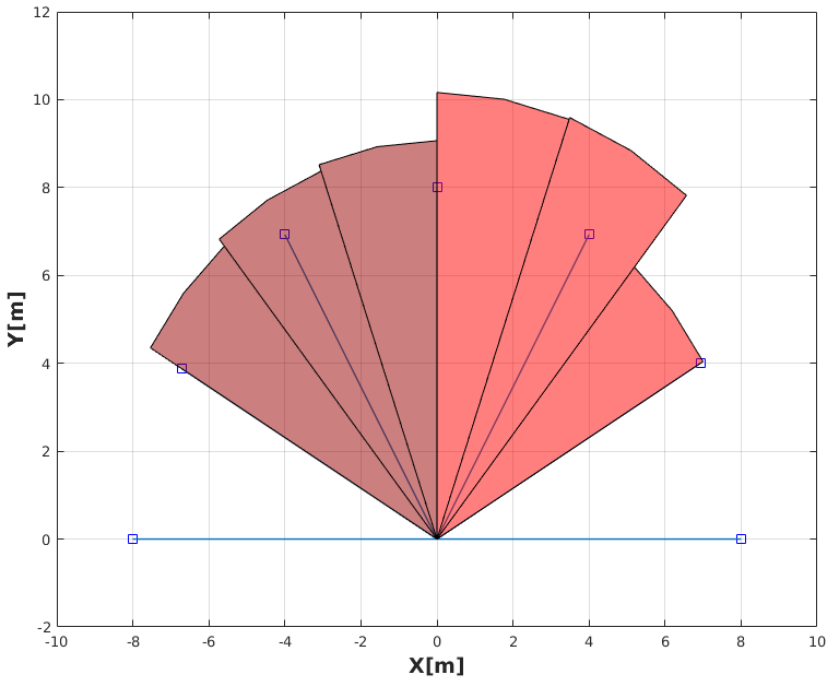

Figure 1 - Configuration 1 with sample readings; blue - laser; light red - radar 1; dark red - radar 2

Sensor Configuration 2 (Spatially Separated Sensors)

Sensor	Field of View [degrees]	Angular Resolution [degrees]	Orientation offset [degrees]	X offset [m]
Laser	180	10	0.0	0.0
Radar1	60	20	0.0	1.0
Radar2	60	20	0.0	-1.0

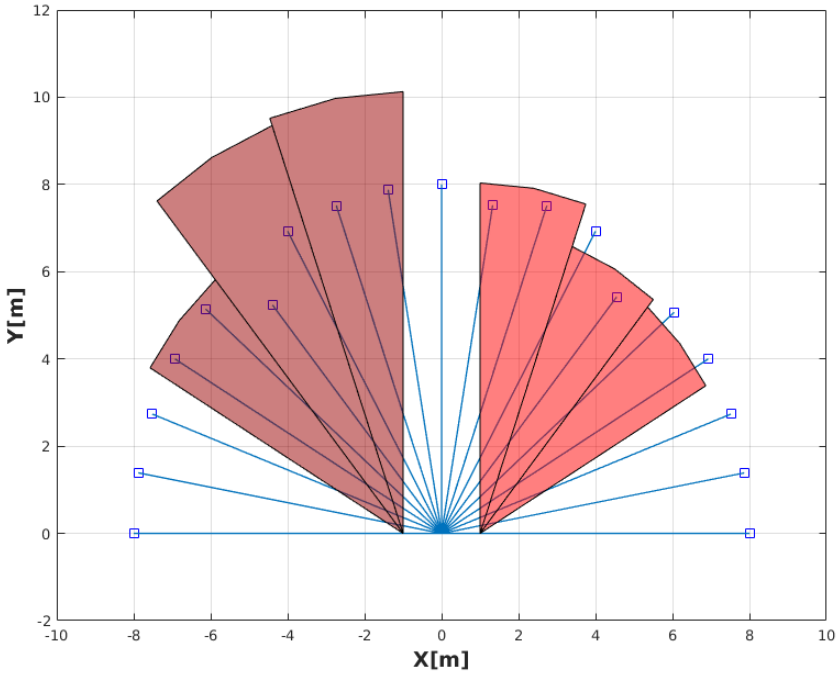

Figure 2 - Configuration 1 with sample readings; blue - laser; light red - radar 1; dark red - radar 2