Lab 13: more Eigen

In this lab we will cover some key ideas you will need for Homework 4

1. **Range Sensor Calibration Using Least Squares** [20 min]: Based on the calibration example from lecture, we're going to code up an example using the QR method in Eigen. Copy the following into `lab13_calibration.cpp` and the GSI will walk you through the implementation.

```
#include <iostream>
#include <eigen3/Eigen/Eigen>

int main()
{
    Eigen::VectorXd sensor(10);
    sensor << 0., 0.11111111, 0.22222222, 0.33333333, 0.44444444, 0.55555556, 0.66666667, 0.77777778, 0.88888889, 1.0;

    Eigen::VectorXd distance(10);
    distance << 0.48147829, 0.70278627, 0.94916103, 1.1409607, 1.38118014, 1.62292857, 1.81760639, 2.04022307, 2.27369675, 2.495571;

    // create A using sensor
    // create b using distance
    // solve for x using QR method (see lecture slides)

    // write down the values for x1 and x2
    return 0;
}
```

2. **Least Squares for Plane Fitting** [15 min]: We will go through the derivation for least squares for a plane together. Complete notes for the derivation are here. We will then write some code to test it. Create the file `lab13_plane_fitting.cpp` and copy in the following template code. We will then show how to generate the A and b matrices for least squares using eigen.

```
#include <eigen3/Eigen/Eigen>
#include <iostream>

int main()
{
    Eigen::Matrix<double, 12, 3> points;
    points << 0, 0, 0,
              0, 1, 0,
              0, 2, 0,
              1, 0, 0,
              1, 1, 0,
              1, 2, 0,
              2, 0, 0,
              2, 1, 0,
              2, 2, 0,
              3, 0, 0,
              3, 1, 0,
              3, 2, 0;

    // create the A matrix from points
    // create the b matrix from points
    // solve for x
    // convert x to our plane coefficients

    return 0;
}
```

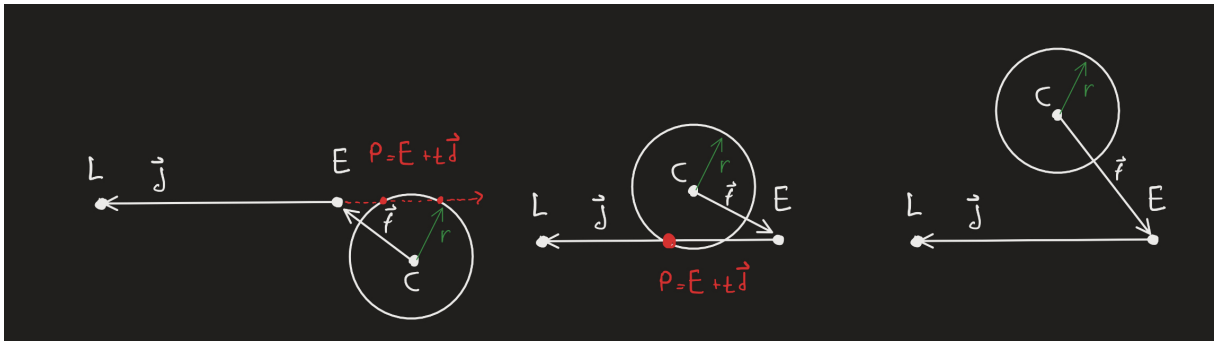
3. **Collision Checking** [15 min]: In homework 4, you'll need to implement some collision checking, which is going to require checking whether a circle and a line segment intersect. To help you with this, we're going to derive a solution to this problem together here in lab. Below is a written explanation for convenience:

To check whether an edge (line segment) intersects a disc (circle), we attempt to find a point on the segment which also lies on the circle. If we can find such a point, then the circle and segment intersect.

Let the edge be the segment from the point E to the point L, and let the center of the circle be the point C. Let $\vec{d}$ be the vector from E to L, and let $\vec{f}$ be the vector C to E (see diagram below). We can represent a point on segment as $P = E + t\vec{d}$ where $t \in [0, 1]$. We can represent a point on the circle as $(P_x - C_x)^2 + (P_y - C_y)^2 = r^2$ where $P = (P_x, P_y)$ is the point on the circle with center C and radius r. We then try to find a value of t for which P satisfies the equation of the circle, which is defined by the following quadratic equation, where $\cdot$ is the dot product.

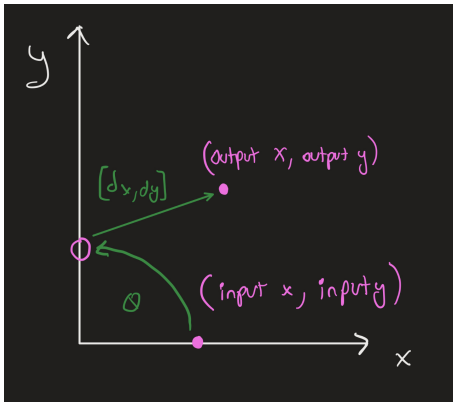
$$(\vec{d} \cdot \vec{d})t^2 + (2\vec{f} \cdot \vec{d})t + (\vec{f} \cdot \vec{f} - r^2) = 0$$

If this equation has a solution where $t \in [0, 1]$, then the circle and the line segment intersect. The diagram below shows three cases. In the first case, the line segment and circle do not intersect. In this case the value of t would be negative. In the second case, the segment and the circle intersect. In the last case, they do not intersect, and by checking the determinant we would find that the quadration equation above has no so



4. **Frame Transformations** [20 min]: In this exercise you will practice creating homogeneous transformations matrices for translating and rotating points in 2D. Create a file called `lab13_transforms.cpp` and write a function that constructs the 3x3 transformation matrix based on a rotation (θ) and translation (dx,dy): `Eigen::Matrix3d make_transform_mat(double theta, double dx, double dy)`

   Then in `main`, use this function to transform the following points using the parameters in the table below. Test your function using the inputs `input x, input y` and ensure they match the correct answers `output x, output y`. The correct answers for the final two examples are not given, as you will be asked about them on the quiz. See the diagram for clarification on how to interpret the table:



Example #	theta	dx	dy	input x	input y	output x	output y
1	0	1	0	0	0	1	0

Example #	theta	dx	dy	input x	input y	output x	output y
2	1.5707	0	0	0	1	-1	0
3	-1.5707	1	-1	0	0	1	-1
4	-2	0.56	0.11	-1	0.5	1.43	0.811
5	2	0	-0.3	1	0.2		
6	3	0.5	0.5	0.5	0.5		