RYERSON UNIVERSITY

Faculty of Engineering, Architecture and Science

Department of Electrical and Computer Engineering

Course Number	CPS 843
Course Title	Introduction to Computer Vision
Semester/Year	F2023

Instructor	Dr. Guanghui Richard Wang

ASSIGNMENT No.	3

Assignment Title	Homework 3

Submission Date	November 5th, 2023
Due Date	November 6th, 2023

Student Name	Abdulrehman Khan
Student ID	500968727
Signature*	A.K.

**By signing above you attest that you have contributed to this written lab report and confirm that all work you have swung the lab contributed to this lab report is your own work.*

Part 1:

Problem 1:

(1):

The equation of the first line is $y = -0.5x + 2$. We can rewrite it as $line\ 1\colon\ 0.5x + y - 2 = 0$.

The equation of the second line is $3x + 6y = 5$. We can rewrite it $line\ 2\colon\ 3x + 6y - 5 = 0$.

Where $line\ 1 = (0.5,\ 1,\ -2)$ and $line\ 2 = (3, 6, 5)$
So the matrix X is $X = 1 \times 1'$

Now, we have two equations in the form $Ax + By + C = 0$, where $A,\ B,\ and\ C$ are coefficients.

We can use the cross product method to find the intersection point. The intersection point (x, y) satisfies both equations simultaneously, so we can calculate it as follows:

i	j	k
x_1	x_2	x_3
y_1	y_2	y_3

i	j	k
0.5	1	-2
3	6	-5

Thus, A_1 = 0.5, B_1 = 1, C_1 = -2 and A_2 = 3, B_2 = 6, C_2 = -5

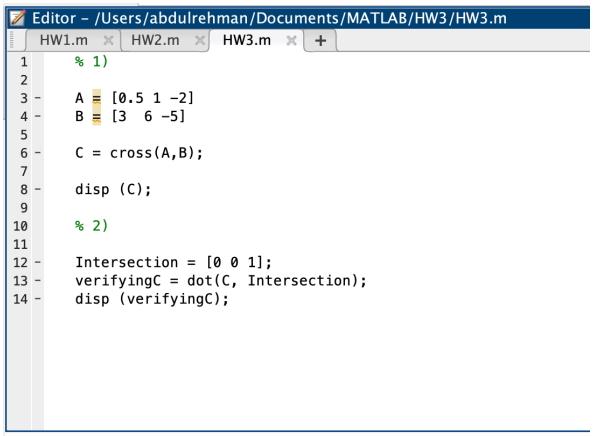
(2):

The intersection of the two lines $line\ 1$ and $line\ 2$ is the point $x = line\ 1 \times line\ 2$.
$X = (x, y)^T$ lies on the $line\ 1 = (a, b, c)^T\ if\ and\ only\ if\ ax + bx + c = 0$

$(x, y, 1)(a, b, c)^T = (x, y, 1)l = 0$ The point x lies on the $line\ 1$ if and only if $x^{T1} = 1^{Tx} = 0$

The resulting cross product of X is $(7, -3.5, 0)$ as $(x_1, x_{2,} x_3)^T$ respectively. The line at infinity is: $l(infinity) = (0, 0, 1)^T$. To confirm whether the hypothetical point is on the line at infinity, we can compute the cross product of $line\ 1$ and $line\ 2$ and then perform a dot product with the line at infinity. If the outcome is 0, it indicates that the point lies on the line.

The ideal point is determined by taking the cross product of the line with the line at infinity. Therefore, when we compute the cross product of [7, -3.5, 0] and [0, 0, 1], we obtain the result of 0. Consequently, this implies that the ideal point is indeed situated on the line at infinity.



Thus through the utilization of matlab we can deduce that the ideal point is on the line at infinity (7, -3.5, 0) due to it abiding by $(x1, x2, 0)^T$ coordinates.

(3):

The equation of a conic in (Assuming the cross product of the line 1 and line 2 is to be substituted into the following equations):

1. ***Homogenous:*** $ax + by + c = 0.7x - 3.5y = 0$
2. ***Inhomogeneous:***

   $ax^2 + bxy + cy^2 + dx + ey + f = 0.7x^2 - 3.5xy + 0 + 0 + 0 + 0 = 0$
3. ***Matrix form*****:** $x^{TCx} = 0$
4. C =

i	j	k
a	$b/2$	$d/2$
$b/2$	c	$e/2$
$d/2$	$e/2$	f

Which is

i	j	k
7	$-3.5/2$	0
$-3.5/2$	0	0
0	0	0

Problem 2:

Abdulrehman Ichaq
50096872 7

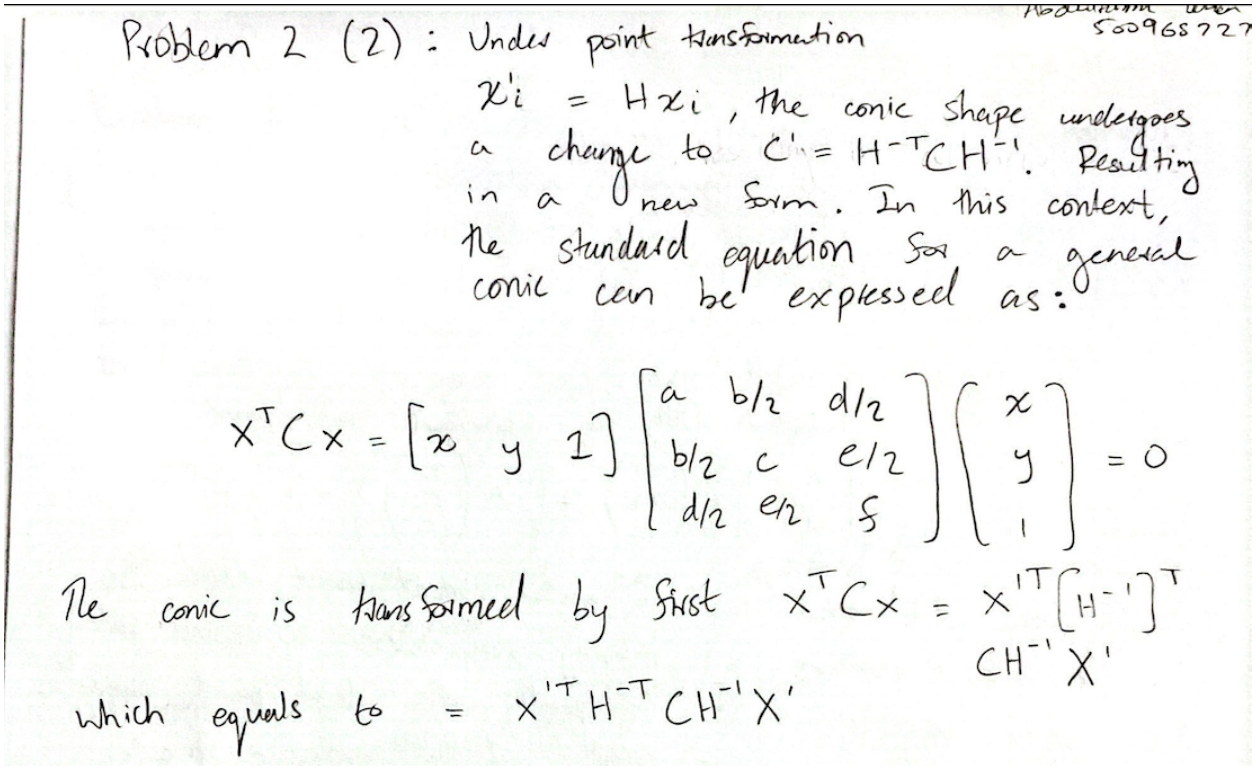
Problem 2 (1): Generally when dealing with points represented by homogeneous coordinates, they do not directly relate to finite points in the two-dimensional space, R^2. In this scenario, the concept of an ideal point is connected to numerous parallel lines, yet only one of these lines intersects with the specified finite point:

$$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix} = \begin{pmatrix} A \begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix} \end{pmatrix} = 1' = H^{-T} 1$$

Initially the ideal point maps to another ideal point, where the line at infinity remains at infinity. Nevertheless, the points along this line undergo a shift. Consequently, the ideal point undergoes transformation into a finite point, and the infinite line:

$$\begin{bmatrix} A & t \\ v^T & v \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix} = \begin{pmatrix} A \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ v_1 x_1 + v_2 x_2 \end{pmatrix}$$

Thus verifying that under a general 2D transformations, an ideal point is mapped to a finite point and allowing the user to observe a horizon and the phenomenon of vanishing points.



Problem 3:

(1):

Projective	
General Form	$\begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$
Degree of Freedom	A projective transformation has 8 degrees of freedom due to the 3x3 matrix representation, but it's typically normalized to have 9 elements, where one element is set to 1. Therefore, there are 8 degrees of freedom.
Invariant Properties	Projective transformations preserve collinearity, which means that if three points are collinear before the

	transformation, they will remain collinear after the transformation. This preservation of collinearity is a fundamental property of projective geometry. One important concept related to projective geometry is the cross ratio. The cross ratio of four points on a line is an invariant property under projective transformations. This means that no matter how you transform the four points using a projective transformation, the cross ratio will remain the same. In other words, the cross ratio is a projective invariant. So, in summary, projective transformations preserve collinearity, and the cross ratio of four points on a line is an invariant property in projective geometry.

Affine	
General Form	$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$
Degree of Freedom	For 2D transformations, an affine transformation has 6 degrees of freedom. These degrees of freedom are divided into components contributed by the matrix A and the translation vector b: Matrix A contributes 4 degrees of freedom: 2 for scaling, 2 for rotation. The translation vector b adds the remaining 2 degrees of freedom for translation. Additionally, when considering non-isotropic scaling, there are 2 extra degrees of freedom introduced: 2 degrees of freedom for non-isotropic scaling include the scale ratio and the orientation of scaling.

	So, in total, an affine transformation in 2D has 6 degrees of freedom, consisting of 2 scales, 2 rotations, and 2 translations. If non-isotropic scaling is considered, it includes 2 extra degrees of freedom related to scaling. In 3D transformations, the total degrees of freedom would be 12, with a similar breakdown but in the 3D space.
Invariant Properties	Affine transformations have invariant properties that include the preservation of parallel lines, ratios of lengths along lines, and angles between lines. These properties make affine transformations suitable for approximating local geometric distortions. The invariance of parallel lines ensures that lines that are parallel before the transformation remain parallel after the transformation. Ratios of lengths along lines are preserved, which means if you have two line segments with a certain ratio of lengths, that ratio will remain the same after the affine transformation. In addition to length ratios, ratios of areas are also preserved, meaning that if you have two regions with a certain area ratio, that ratio remains unchanged under affine transformation.

Similarity Transformations	
General Form	$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{bmatrix} s\cos\theta & -s\sin\theta & t_x \\ s\sin\theta & s\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$
Degree of Freedom	A 2D similarity transformation is characterized by 4 degrees of freedom. These degrees of

	freedom are typically represented as follows: 1. Scale factor (s) 2. Rotation angle (θ) 3. Translation in the x-direction 4. Translation in the y-direction These transformations are considered equi-form, as they preserve shape, meaning that they maintain the relative proportions and angles within the transformed object. Furthermore, they preserve the metric structure up to similarity, which implies that the relative lengths and angles between objects are preserved, making similarity transformations a useful tool in various geometric applications. The term "similarity" signifies that this transformation is equi-form, meaning it preserves the shape of objects. It retains the metric structure, which refers to the structure up to similarity, and ensures that lengths and angles between points are maintained.
Invariant Properties	Similarity transformations exhibit several invariant properties. They preserve angles and ratios of distances, making them well-suited for modeling transformations that maintain the shape and orientation of objects while allowing for changes in size and position. These transformations also maintain the ratios of lengths, angles, and areas, and they preserve the parallelism of lines, making them a versatile tool for various geometric and spatial applications.

(2) and **(3)**:

Problem 3 (2): For verifying that an affine transformation maps a point at infinity to a point at infinity we know:

Abdulrahman Ichou and 500968727 (3)

$$x' = H_A x = \begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix} x$$

$$\begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 0 \end{pmatrix} = \begin{pmatrix} A \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ 0 \end{pmatrix}$$

Thus, the affine transformation maps ideal points to ideal points

Furthermore, under projective transformation H, the line at infinity is a fixed line if and only if H is an affinity: $x' = H_A x = \begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix} x$

$$l'_\infty = H_A^{-T} l_\infty = \begin{bmatrix} A^{-T} & 0 \\ -t^T A^{-T} & 1 \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = l_\infty$$

Thus verifying that an affine transformation maps a line at infinity to a line at infinity.

Problem 4:

(1), **(2)**, and **(3)**:

Sw968727

Problem 4 (1): The circular points observe all circles intersecting the infinite line, whereby the circle is defined as:

$x_1^2 + x_2^2 + d x_1 x_3 + e x_2 x_3 + f x_3^2 = 0$

and $x_3 = 0$ and the line at infinity is $x_1^2 + x_2^2 = 0$

Let $I = (1, i, 0)^T$ and $J = (1, -i, 0)^T$. I changes as the circular points are identified; the orthogonality is now determined $1 = (1,0,0)^T + i(0,1,0)^T$

I and J are circular fixed points if and only if H is a similarity. So if $I' = H_S I$ then

$$\begin{bmatrix} s\cos\theta & -s\sin\theta & tx \\ s\sin\theta & s\cos\theta & ty \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 1 \\ i \\ 0 \end{pmatrix}$$

Thus: $se^{-i\theta} \begin{pmatrix} 1 \\ i \\ 0 \end{pmatrix} = I$

The whole process can be repeated for J and thus verifying that any circle in a 2D plane intersects the line at infinity at the two circular points.

Problem 4 (2): Abdulrahman Khan

The dual conic is fixed under the projective transformation H if and only if H is a similarity. Under point transformation $X' = H_s X$ in turn generating $C_\infty^{*\prime} = H_s C_\infty^* H_s^T = C_\infty^*$

Thus, the line at infinity is a null vector of the dual conic, represented by: $I^T 1_\infty = J^T 1_\infty = 0$ and $C_\infty^* 1_\infty = (IJ^T + JI^T) 1_\infty = I(J^T 1_\infty) + J(I^T 1_\infty) = 0$ and $C_\infty^* 1_\infty = (IJ^T + JI^T) 1_\infty = I(J^T 1_\infty) + J(I^T 1_\infty) = 0$

Problem 4 (3): Lines l and m are orthogonal if $l^T C_\infty^* m = 0$. Where $l = (l_1, l_2, l_3)^T$ and $m = (m_1, m_2, m_3)^T$.

Identifying the conic on the projective plane: $\cos\theta = \dfrac{l^T C_\infty^* m}{\sqrt{(l^T C_\infty^* l)(m^T C_\infty^* m)}}$. Once the conic is identified then the angles on the Euclidean space are measured:

$$\cos\theta = \frac{l_1 m_1 + l_2 m_2}{\sqrt{(l_1^2 + l_2^2)(m_1^2 + m_2^2)}}$$

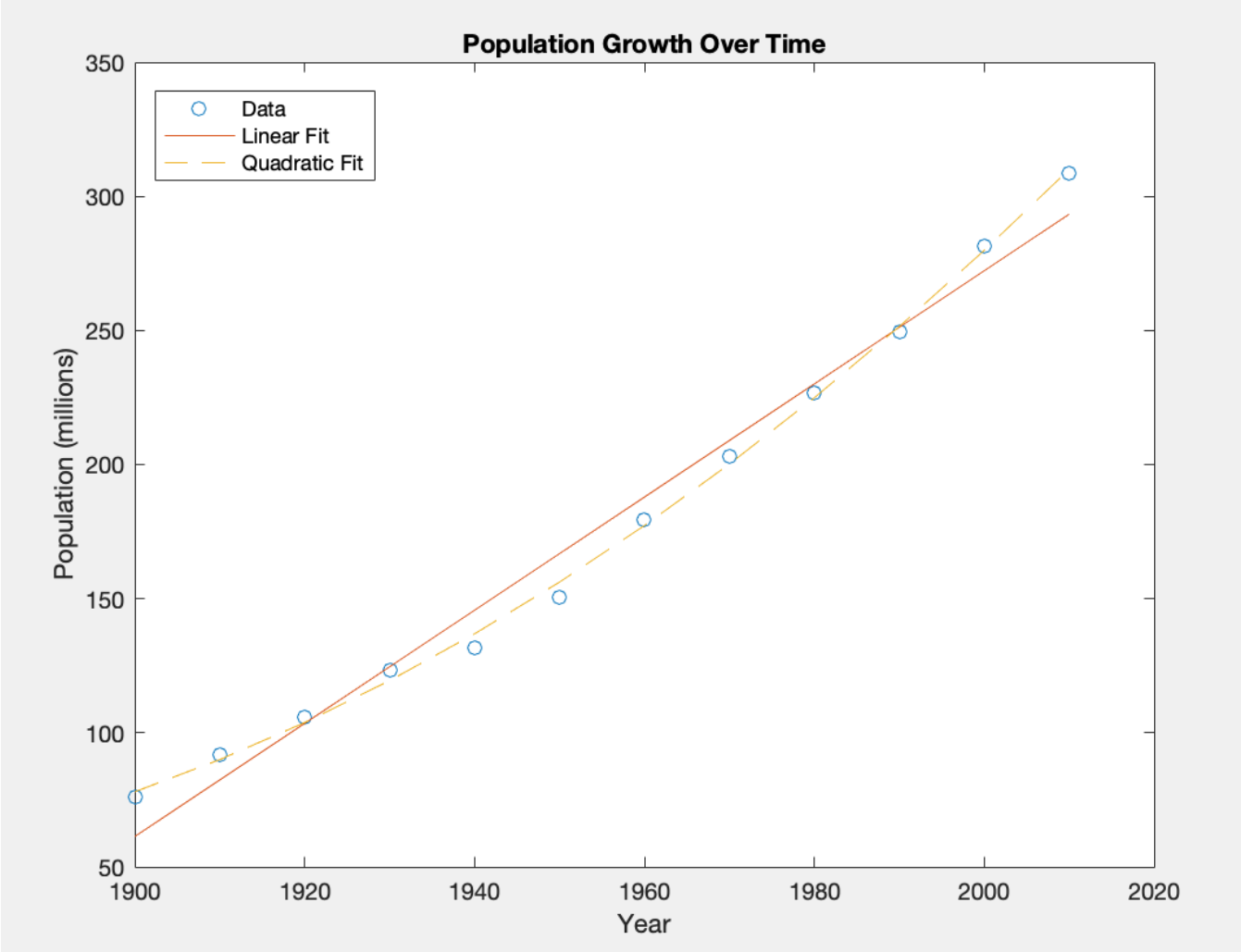
$\therefore$ verifying that two orthogonal lines are conjugate with respect to the dual conic

Problem 5:

Code:

```
% Problem 5

% Times: x-axis
year = (1900:10:2010)';

% Population: y-axis
population = [75.995, 91.972, 105.711, 123.203, 131.669, 150.697, 179.323, 203.212, 226.505, 249.633, 281.422, 308.748]';

% Linear model
linear_coefficients = polyfit(year, population, 1);
linear_fit = polyval(linear_coefficients, year);
linear_population_2020 = polyval(linear_coefficients, 2020);

% Quadratic model
quadratic_coefficients = polyfit(year, population, 2);
quadratic_fit = polyval(quadratic_coefficients, year);
quadratic_population_2020 = polyval(quadratic_coefficients, 2020);

% Plot the data and the fitted models
figure;
plot(year, population, 'o', year, linear_fit, '-', year, quadratic_fit, '--');
xlabel('Year');
ylabel('Population (millions)');
legend('Data', 'Linear Fit', 'Quadratic Fit');
title('Population Growth Over Time');

% Display the predicted populations for 2020
disp(['Linear Model Prediction for 2020: ', num2str(linear_population_2020)]);
disp(['Quadratic Model Prediction for 2020: ', num2str(quadratic_population_2020)]);
```

Result:



Part 2:

MLESAC (Maximum Likelihood Estimation Sample Consensus) is an influential algorithm in computer vision, addressing the robust estimation of geometric relations between data points. In computer vision applications, data is often corrupted by outliers, necessitating the need for robust techniques.

MLESAC combines Maximum Likelihood Estimation (MLE) and Random Sample Consensus (RANSAC) to achieve robust estimation. It aims to find model parameters that best fit the inliers while accounting for outliers.

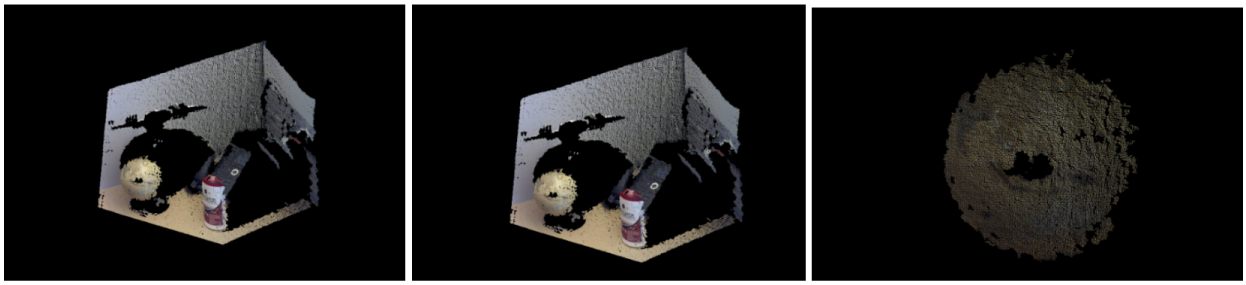
The algorithm begins by randomly selecting a minimal set of data points to estimate the geometric model. It then computes the likelihood of each data point being an inlier under this model, classifying them accordingly.

MLESAC distinguishes inliers from outliers based on their conformity to the model, and outliers are systematically rejected during the estimation process. This ensures that the presence of outliers does not adversely affect the model.

Non-linear optimization techniques are employed to refine the model estimation, enhancing parameter accuracy and resulting in more robust model fits.

The algorithm offers different parametrization methods for various geometric models, such as fundamental matrices and homographies, ensuring minimal and consistent representations.

Experimental results demonstrate that MLESAC excels in reducing variance in parameter estimates and effectively handling noisy data, establishing itself as a valuable tool in addressing challenges in image geometry estimation within computer vision applications.

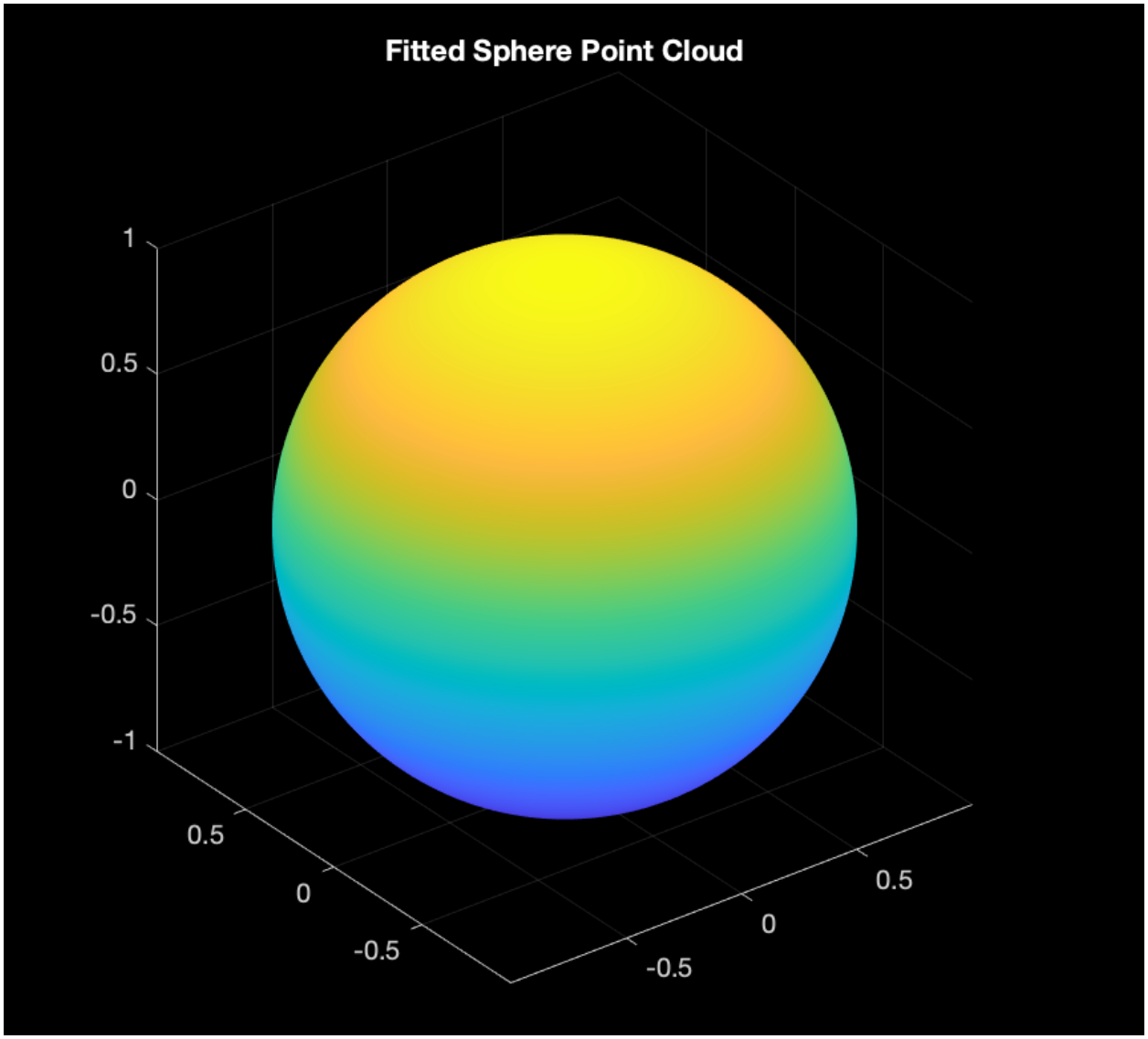


Code:

```
% Part 3:
% Generate a point cloud of a sphere
radius = 1;
numPoints = 1000;
theta = linspace(0, 2*pi, numPoints);
phi = linspace(0, pi, numPoints);
[THETA, PHI] = meshgrid(theta, phi);
x = radius * sin(PHI) .* cos(THETA);
y = radius * sin(PHI) .* sin(THETA);
z = radius * cos(PHI);

% Create a point cloud object
ptCloud = pointCloud([x(:), y(:), z(:)]);

% Save the point cloud to 'object3d.mat'
save('object3d.mat', 'ptCloud');

% Load the point cloud from 'object3d.mat'
load('object3d.mat');

% Define parameters for sphere fitting
maxDistance = 0.01;
roi = [-inf, 0.5; 0.2, 0.4; 0.1, inf];
sampleIndices = findPointsInROI(ptCloud, roi);

% Fit a sphere to the point cloud
[model, inlierIndices] = pcfitsphere(ptCloud, maxDistance, 'SampleIndices', sampleIndices);
```

```
% Plot the extracted globe (fitted sphere)
figure;
pcshow(globe);
title('Fitted Sphere Point Cloud');
```

Result:



Part 3:

Groups	Members	Assignment	Discussions	Email	Actions
Project Group					Expiry Date: Nov 6, 2023 11:59 PM
Group 96	2	Project Report	Project Group ...		Leave Group

Group Members - Group 96

Group 96

Last Name ▲ , First Name
Iqbal, Hamza
Khan, Abdulrehman

Considering the essential role of edge detection in computer vision, image processing, and machine learning, our team made the decision to explore this concept more deeply. The primary objective of edge detection is to emphasize the retrieval of valuable information from images, a critical aspect in various computer vision procedures.

Our final project centers around geometric and other image features and techniques, with a particular emphasis on extracting compressed image features. Our strategy involves applying edge detection to compressed images, specifically examining various types such as adaptive, canny, color, edge types, and energy function-based edge detectors. We aim to assess how these tools can effectively identify edges and curves within digital images that exhibit discontinuities.

References:

MathWorks. (n.d.). pcFitSphere. MathWorks.
https://www.mathworks.com/help/vision/ref/pcfitsphere.html

Toronto Metropolitan University. 2023. Course Content: Week 6. In Computer Vision, CPS 843. Toronto Metropolitan University's Learning Management System.
https://courses.torontomu.ca/d2l/le/content/797209/viewContent/5384725/View

Torr, P. H. S., & Zisserman, A. (2000). MLESAC: A new robust estimator with application to estimating image geometry. Computer Vision and Image Understanding.
https://www.robots.ox.ac.uk/~vgg/publications/2000/Torr00/torr00.pdf