Spring 2019

Exercise 3 | **Line fitting and extraction for robot localization**

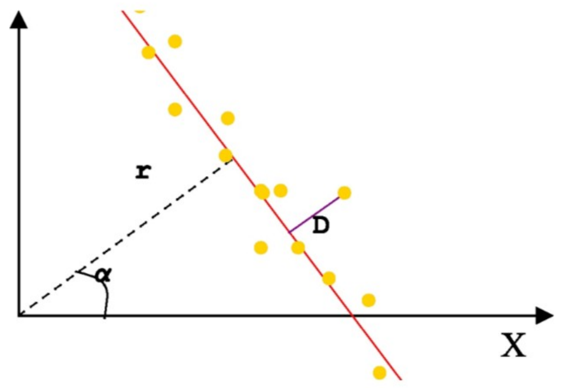
Lukas Bernreiter and Hermann Blum

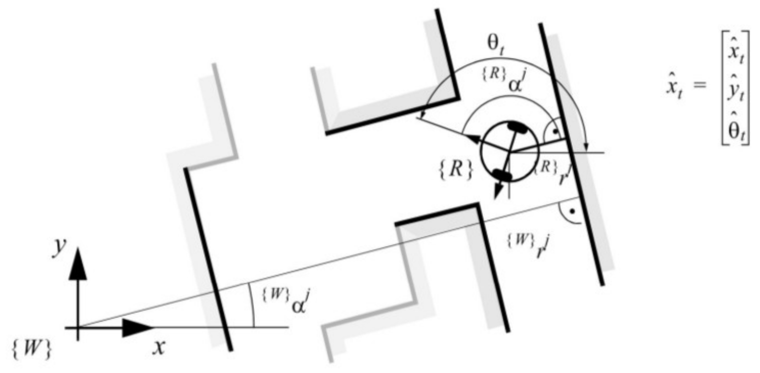
Line extraction, EKF, SLAM

Exercise 3
- Line extraction
- Line fitting

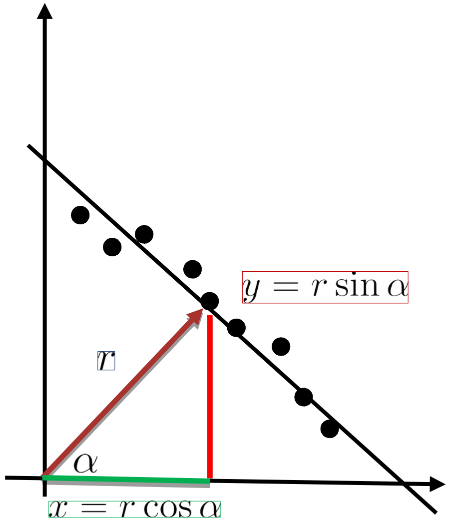
Exercise 4
- EKF
- Localization: Line extraction, given map
- Wheel odometry

Exercise 5
- Simultaneous Localization and Mapping (SLAM)
- Unknown environment (a-priori)





Describing lines using polar coordinates



- Two parameter description, i.e.

$$x\cos(\alpha) + y\sin(\alpha) = r$$

- Why switch to polar parameters?
  - Vertical lines are not representable in Cartesian coordinates (linear eq.)
  - In general simpler representation for e.g. lines or circles
- Where does the line equation expressed in polar coordinates come from?
  - Pythagorean theorem yields

$$x^2 + y^2 = r^2$$

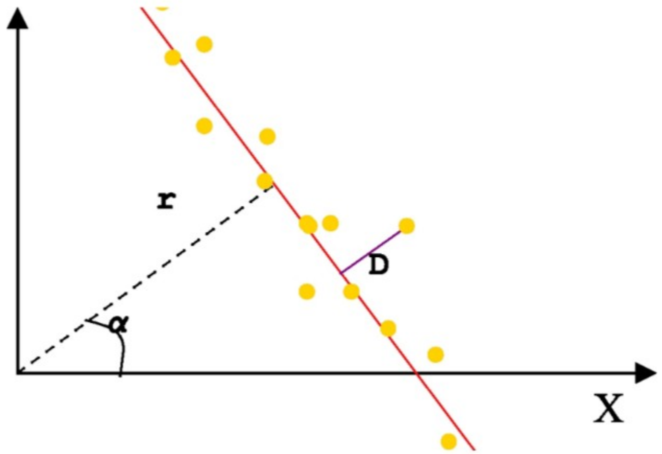
  - With $x = r\cos(\alpha)$ and $y = r\sin(\alpha)$

$$x\cos(\alpha) + y\sin(\alpha) = r$$

Line fitting / Line regression

- Squared error between the line and all points

$$S(r,\alpha) := \sum_i (r - x_i\cos(\alpha) - y_i\sin(\alpha))^2$$



- Solution for the line:

$$\nabla S = 0$$

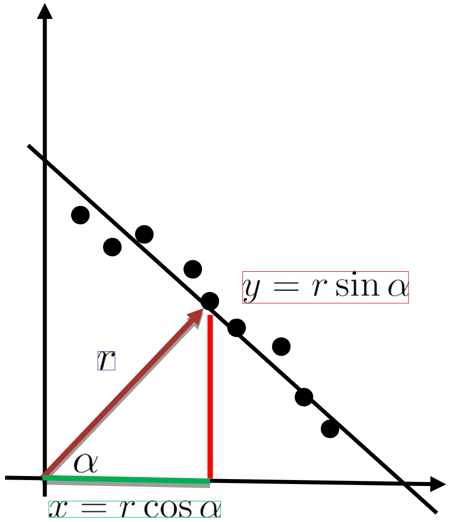
- **Task 1:**
  - Derive the line parameters using least squares, i.e.

$$\frac{\partial S}{\partial r} = 0 \qquad \frac{\partial S}{\partial \alpha} = 0$$

Line fitting / Line regression

- **Task 1:**

$$S(r,\alpha) := \sum_i (r - x_i\cos(\alpha) - y_i\sin(\alpha))^2$$



$$\frac{\partial S}{\partial r} = 2\sum_i (r - x_i\cos(\alpha) - y_i\sin(\alpha))$$

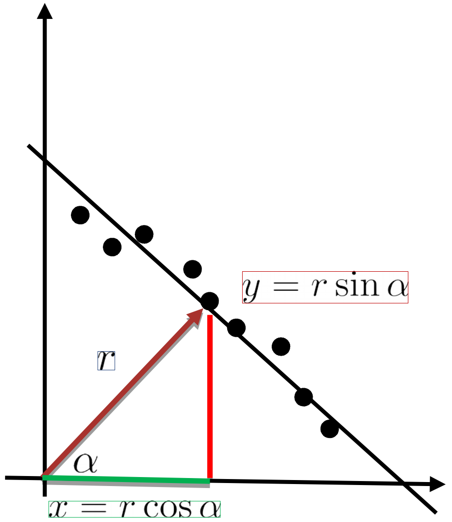
$$= Nr - \sum_i (x_i\cos(\alpha) - y_i\sin(\alpha)) \stackrel{!}{=} 0$$

$$\boxed{r = \frac{1}{N}\sum_i (x_i\cos(\alpha) + y_i\sin(\alpha)) = x_c\cos(\alpha) + y_c\sin(\alpha)}$$

Line fitting / Line regression

$$S(r,\alpha) := \sum_i (r - x_i\cos(\alpha) - y_i\sin(\alpha))^2$$

$$r = x_c\cos(\alpha) + y_c\sin(\alpha)$$



$$\begin{aligned}
\frac{\partial S(r,\alpha)}{\partial\alpha} &= \frac{\partial}{\partial\alpha}\sum_i (x_c\cos(\alpha) + y_c\sin(\alpha) - x_i\cos(\alpha) - y_i\sin(\alpha))^2 \\
&= \frac{\partial}{\partial\alpha}\sum_i (\cos(\alpha)(x_c - x_i) + \sin(\alpha)(y_c - y_i))^2 \\
&= 2\sum_i (\tilde{x}\cos(\alpha) + \tilde{y}\sin(\alpha))\frac{\partial}{\partial\alpha}\sum_i (\tilde{x}\cos(\alpha) + \tilde{y}\sin(\alpha)) \\
&= 2\sum_i (\tilde{x}\cos(\alpha) + \tilde{y}\sin(\alpha))(-\tilde{x}\sin(\alpha) + \tilde{y}\cos(\alpha))
\end{aligned}$$

$$\tilde{x} = x_c - x_i$$

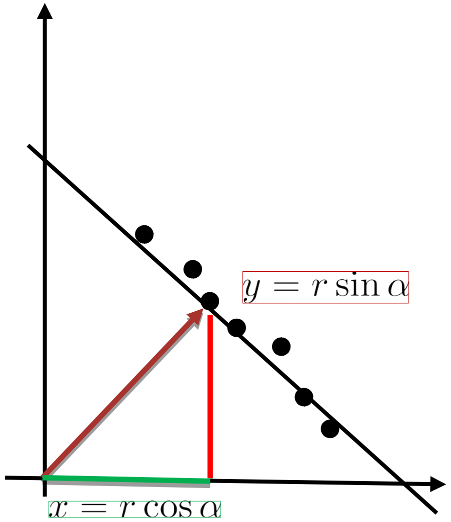
$$\tilde{y} = y_c - y_i$$

$$\sin(2\alpha) = 2\sin(\alpha)\cos(\alpha)$$

$$\cos(2\alpha) = \cos^2(\alpha) - \sin^2(\alpha)$$

Line fitting / Line regression

$$\begin{aligned}\frac{\partial S(r,\alpha)}{\partial \alpha} &= \frac{\partial}{\partial \alpha}\sum_i (x_c\cos(\alpha) + y_c\sin(\alpha) - x_i\cos(\alpha) - y_i\sin(\alpha))^2 \\ &= \frac{\partial}{\partial \alpha}\sum_i (\cos(\alpha)(x_c - x_i) + \sin(\alpha)(y_c - y_i))^2 \\ &= 2\sum_i (\tilde{x}\cos(\alpha) + \tilde{y}\sin(\alpha))(-\tilde{x}\sin(\alpha) + \tilde{y}\cos(\alpha))\end{aligned}$$



$$-\cos(\alpha)\sin(\alpha)\sum \tilde{x}^2 + \cos^2(\alpha)\sum \tilde{x}\tilde{y} - \sin^2(\alpha)\sum \tilde{x}\tilde{y} + \sin(\alpha)\cos(\alpha)\sum \tilde{y}^2 = 0$$

$$\underline{\sin(\alpha)\cos(\alpha)}\sum(\tilde{y}^2 - \tilde{x}^2) + (\underline{\cos^2(\alpha) - \sin^2(\alpha)})\sum \tilde{x}\tilde{y} = 0$$

$$\sin(2\alpha)\sum(\tilde{y}^2 - \tilde{x}^2) + 2\cos(2\alpha)\sum \tilde{x}\tilde{y} = 0$$

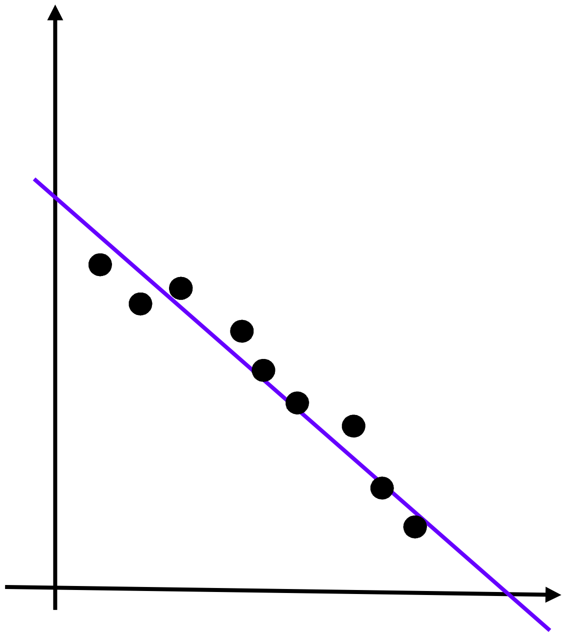
$$\frac{\sin(2\alpha)}{\cos(2\alpha)} = -\frac{2\sum \tilde{x}\tilde{y}}{\sum(\tilde{y}^2 - \tilde{x}^2)}$$

$$\boxed{\alpha = \frac{1}{2}\tan^{-1}\left(\frac{-2\sum \tilde{x}\tilde{y}}{\sum(\tilde{y}^2 - \tilde{x}^2)}\right)}$$

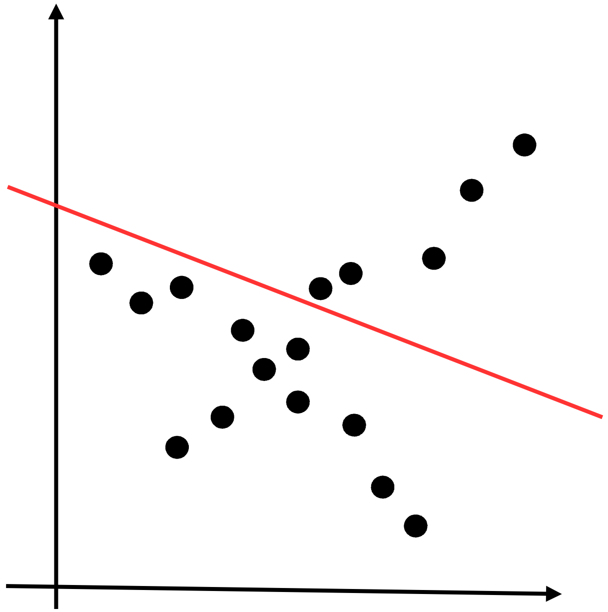
Trigonometric identities

$$\boxed{\begin{aligned}\cos(2\alpha) &= \cos^2(\alpha) - \sin^2(\alpha) \\ \sin(2\alpha) &= 2\sin(\alpha)\cos(\alpha)\end{aligned}}$$

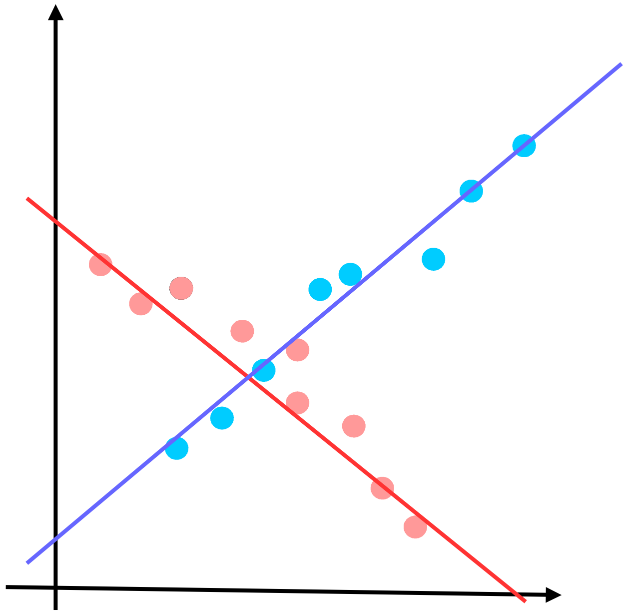
Split and Merge



Solved: Fitting a line to a set of points that should be on a line



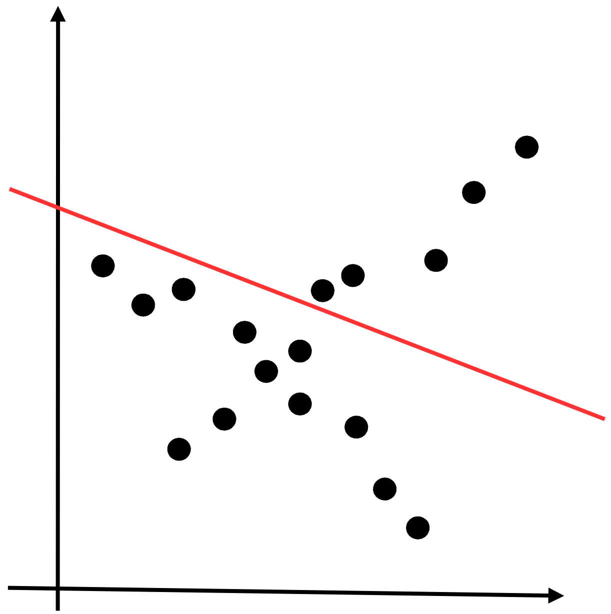
What do we do in case of multiple lines?



Find sets of points, then fit a line to the separate sets!

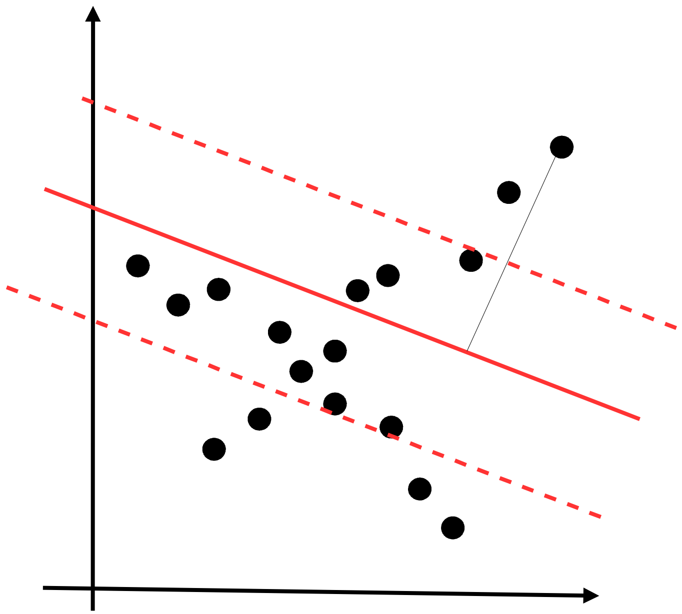
Split and Merge

```
while not finished do
    fit a line to the current set of points
```



Split and Merge

```
while not finished do
    fit a line to the current set of points
    check distance to line for each point
```



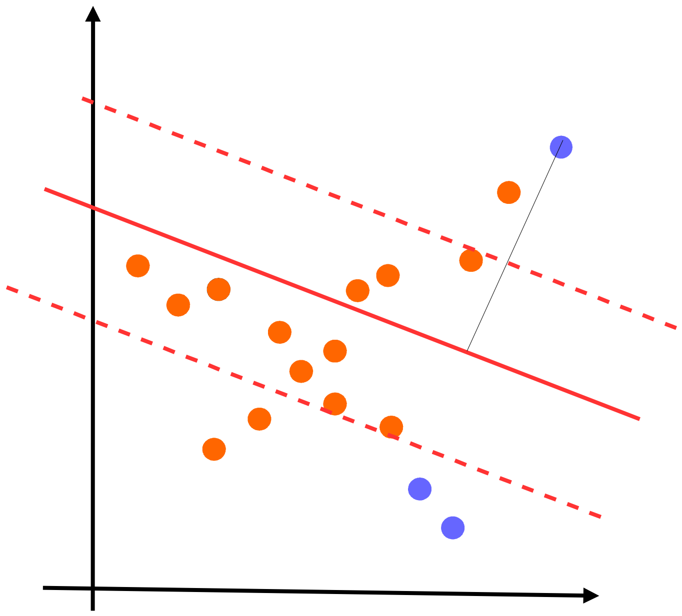
Split and Merge

```
while not finished do

    fit a line to the current set of points

    check distance to line for each point

    if max(distances) > threshold then
        split current set of points
    else
        select next set of points
    end
end
```



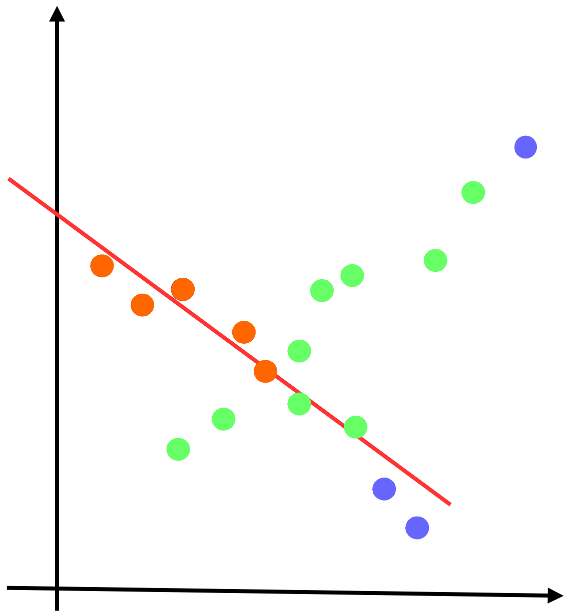
Split and Merge

```
while not finished do

    fit a line to the current set of points

    check distance to line for each point

    if max(distances) > threshold then
        split current set of points
    else
        select next set of points
    end
end
```



Split and Merge

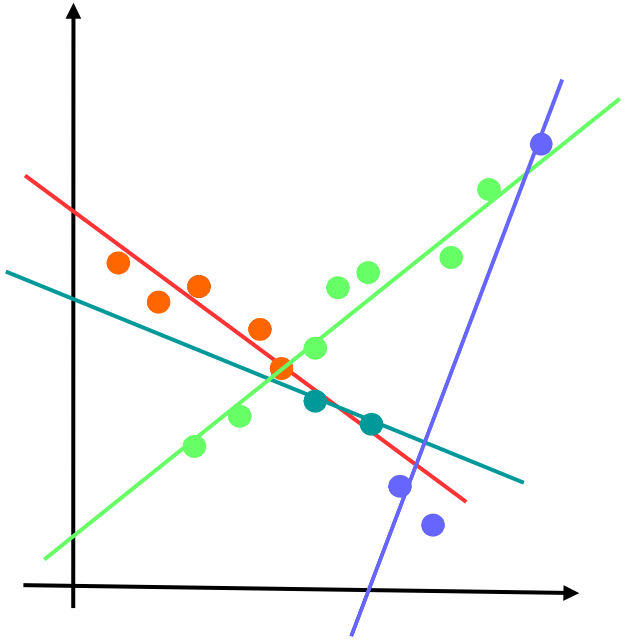
```
while not finished do

    fit a line to the current set of points

    check distance to line for each point

    if max(distances) > threshold then
        split current set of points
    else
        select next set of points
    end
end

merge collinear lines
```



Split and Merge

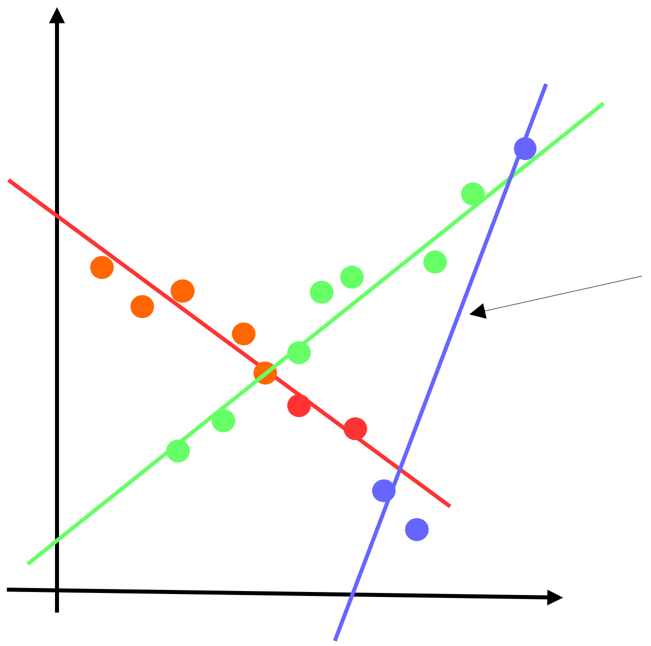
```
while not finished do

    fit a line to the current set of points

    check distance to line for each point

    if max(distances) > threshold then
        split current set of points
    else
        select next set of points
    end
end

merge collinear lines
```



good processing can prevent outliers

Line fitting / Line regression

```
function [alpha, r] = fitLine(XY)
% Compute the centroid of the point set (xmw, ymw) considering that
% the centroid of a finite set of points can be computed as
% the arithmetic mean of each coordinate of the points.

% XY(1,:) contains x position of the points
% XY(2,:) contains y position of the points

    xc = TODO
    yc = TODO

    % compute parameter alpha (see exercise pages)
    num   = TODO
    denom = TODO
    alpha = TODO

     % compute parameter r (see exercise pages)
     r = TODO

     % Eliminate negative radii
     if r < 0
         alpha = alpha + pi;
         if alpha > pi, alpha = alpha - 2 * pi; end
         r = -r;
     end
end
```

Split-and-Merge algorithm

- **Task 2:**
  - Fill in the solution for line fitting
  - Implement the split function

```
function splitPos = findSplitPosInD(d, params)
     splitPos = TODO
end
```

Line fitting / Line regression

```
function [alpha, r] = fitLine(XY)
% Compute the centroid of the point set (xmw, ymw) considering that
% the centroid of a finite set of points can be computed as
% the arithmetic mean of each coordinate of the points.

% XY(1,:) contains x position of the points
% XY(2,:) contains y position of the points

    len = size(XY, 2);
    xc = sum(XY(1, :)) / len;
    yc = sum(XY(2, :)) / len;

    % compute parameter alpha (see exercise pages)
    dX = (xc - XY(1, :));
    dY = (yc - XY(2, :));
    num   = -2 * sum(dX.*dY);
    denom = sum(dY.*dY - dX.*dX);
    alpha = atan2(num, denom) / 2;

    % compute parameter r by inserting the centroid
    % into the line equation and solve for r
    r = xc * cos(alpha) + yc * sin(alpha);
    % Eliminate negative radii
    if r < 0
        alpha = alpha + pi;
        if alpha > pi, alpha = alpha - 2 * pi; end
        r = -r;
    end
end
```

Test with ***testLineFitting.m***

```
Testing line fitting 1 : OK
Testing line fitting 2 : OK
Testing line fitting 3 : OK
Testing line fitting 4 : OK
Testing line fitting 5 : OK
Testing line fitting 6 : OK
Testing line fitting 7 : OK
Testing line fitting 8 : OK
Testing line fitting 9 : OK
Testing line fitting 10 : OK
Testing line fitting 11 : OK
Testing line fitting 12 : OK
Testing line fitting 13 : OK
Testing line fitting 14 : OK
Testing line fitting 15 : OK
Testing line fitting 16 : OK
Testing line fitting 17 : OK
Testing line fitting 18 : OK
Testing line fitting 19 : OK
Testing line fitting 20 : OK
…
```

Split-and-Merge algorithm

```
function splitPos = findSplitPosInD(d, params)
    N = length(d);
    d = abs(d);
    mask = d > params.LINE_POINT_DIST_THRESHOLD;
    if isempty(find(mask, 1))
        splitPos = -1;
        return;
    end

    [~, splitPos] = max(d);
    if (splitPos == 1), splitPos = 2; end
    if (splitPos == N), splitPos = N-1; end
end
```

```
function splitPos = findSplitPosInD(d, params)
    N = length(d);

    % Find the local maximum set (2 points)
    farOnPositiveSideB = d > params.LINE_POINT_DIST_THRESHOLD;
    farOnNegativeSideB = d < -params.LINE_POINT_DIST_THRESHOLD;

    neigborsFarAwayOnTheSameSideI = find((farOnPositiveSideB(1:N-1)
       & farOnPositiveSideB(2:N))
       | (farOnNegativeSideB(1:N-1) & farOnNegativeSideB(2:N)));

    if isempty(neigborsFarAwayOnTheSameSideI)
        splitPos = -1;
    else
        absDPairSum = abs(d(neigborsFarAwayOnTheSameSideI))
            + abs(d(neigborsFarAwayOnTheSameSideI+1));
        [~, splitPos] = max(absDPairSum);
        splitPos = neigborsFarAwayOnTheSameSideI(splitPos);
        if abs(d(splitPos)) <= abs(d(splitPos + 1))
           splitPos = splitPos + 1;
        end
    end
    % If the split position is toward either end of
    % the segment, find otherway to split.
    if (splitPos ~= -1 && (splitPos < 3 || splitPos > N-2))
        [~, splitPos] = max(abs(d));
        if (splitPos == 1), splitPos = 2; end
        if (splitPos == N), splitPos = N-1; end
    end
end
```

Line fitting implementation