Computational Methods for Detection, Estimation and Identification

ASSIGNMENT 5

MsC in Electrical and Computers Engineering

DEEC - University of Coimbra

Introduction

In this work proposal we aim to develop algorithms to estimate planar conic curves from a set of points $\mathbf{x_i} = (x_i, y_i)^t$ with $i = 1, 2 \ldots M$. If the conic is parameterized by a six dimensional vector $\boldsymbol{\omega} = (a, b, c, d, e, f)^t$ and $\mathbf{x_i}$ lies on the curve, then it follows that

$$ax_i^2 + bx_iy_i + cy_i^2 + dx_i + ey_i + f = 0 \tag{1}$$

Remark that $\boldsymbol{\omega}$ is defined up to a scale factor. The conic curve can also be represented by a 3×3 symmetric matrix $\boldsymbol{\Omega}$. In this case the relation of the previous equation can be rewritten as

$$\begin{bmatrix} x_i & y_i & 1 \end{bmatrix} \underbrace{\begin{bmatrix} a & \frac{b}{2} & \frac{d}{2} \\ \frac{b}{2} & c & \frac{e}{2} \\ \frac{d}{2} & \frac{e}{2} & f \end{bmatrix}}_{\boldsymbol{\Omega}} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0 \tag{2}$$

The polynomial $b^2 - 4ac$ is called the conic discriminant. The curve is an hyperbole, ellipse or parabola when the discriminant is respectively positive, negative or zero.

The measured points $\mathbf{x_i}$ are usually contaminated with bi-dimensional zero mean gaussian noise. We will assume that the noise for each direction X and Y is independent but has the same standard deviation σ. You can find enclosed several matlab functions that will help you in achieving the goals:

- **[S,$\boldsymbol{\Omega_{gt}}$,I]=CurveGenerator(σ,M,t)**

  The function *CurveGenerator* randomly generates conic curves. The input parameters are: the noise standard deviation σ, the number of data points M and the desired conic type t. The function returns both ellipses and hyperboles ($t = 0$), only ellipses ($t = 1$), or only hyperboles ($t = 2$). The output variables: s the $2 \times M$ matrix $\mathbf{S}$ with the set of data points $\mathbf{x_i}$, the ground truth conic $\boldsymbol{\Omega_{gt}}$ and a 640×480 image $\mathbf{I}$ depicting the curve.

- **$\mathbf{I_n}$=PlotConicCurve(Ω,I,c)**

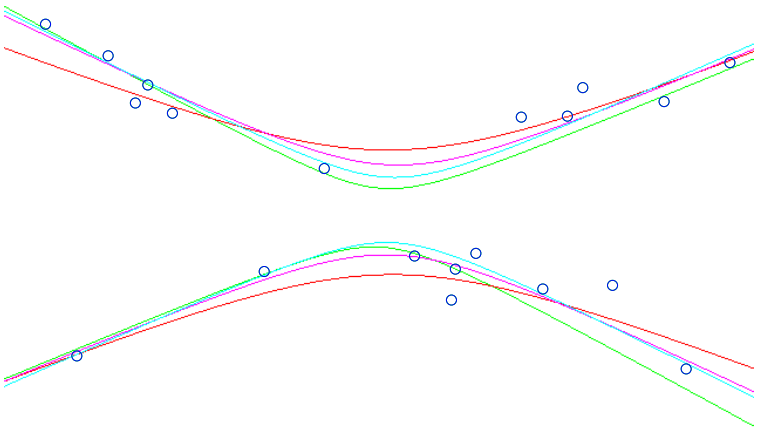

Figure 1: The figure shows the curve generated by *CurveGenerator* (green) and the corresponding data points (blue circles). The noise standard deviation is $\sigma = 20$ and the number of points is $M = 20$. The red, magenta and cyan curves are the results of using different estimation methods

This function plots the conic curve $\mathbf{\Omega}$ in the image $\mathbf{I}$. Parameter c is a 3D vector indicating the desired color in RGB (values between 0 and 255). The conic is drawn in the new image $\mathbf{I_n}$.

- **d=CompareConic($\mathbf{\Omega_1}$,$\mathbf{\Omega_2}$)**

  It receives two conics and returns the mean distance between their principal points (the four points where the major and minor axes intersect the curve).

- **[E,T]=RecursiveTest(H,k,$\mathbf{\Sigma}$, t,M)**

  RecursiveTest is a function that can be used to evaluate and compare the performance of different estimators using Monte Carlo simulation. The handles for the m estimation functions are passed through the cell array $\mathbf{H} = \{@func_1, @func_2, \ldots @func_m\}$. It is assumed that the input for each function is the set of data points $\mathbf{S}$, and the output is the estimated conic $\mathbf{\Omega}$. The noise standard deviation for each experiment is given by vector $\mathbf{\Sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_n)$. The simulation is performed k times for each one of the n experiments. Parameters t and M indicate the type of conic and number of data points as described in *CurveGenerator*. The output parameters are the $m \times n$ matrices $\mathbf{E}$ and $\mathbf{T}$. The former provides the RMS error for the mean distance between principal points (e.g $\mathbf{E}(i, j)$ is the RMS error for the estimation function $func_i$ in the experiment j with noise standard deviation σ_j), while the latter indicates the percentage of misclassification in the conic type (e.g. the estimation result is an hyperbole when the ground truth is an ellipse).

Work Proposal

1. Program a routine *TLS_estimation* to estimate a conic from a set of noisy points using Total Least Squares. Evaluate your estimator by either plotting the curve and performing Monte Carlo Simulation. Try to study the behavior when the number of data points varies.

2. Statistical bias has a strong impact in the estimation performance. As discussed in the courses, one way to deal with this effect is to pre-process the data using a linear transformation that maps the points inside an unitary circle (scaling and translation). Program an estimation function *TLS_normalize* that normalizes the data points before using total least square estimation. Compare the performance with the estimator of the previous question.

3. The correct way to balance the equations in terms of statistical bias is to normalize using the Sampson distance. The drawback is that the minimization problem has no longer a closed-form solution. Program an estimation function *Sampson* that minimizes the Sampson distance using Levenberg-Marquadt. Characterize the obtained algorithm and compare its performance with *TLS_estimation* and *TLS_normalize*.

References

1. Z. Zhang, *Parameter Estimation Techniques: A tutorial with Application to Conic Fitting*, INRIA Technical Report, 1995