Detection of Linear and Circular Shapes in Image Analysis*

by Tim Garlipp and Christine H. Müller

Carl von Ossietzky University of Oldenburg

December 30, 2004

Abstract

To estimate linear and circular shapes in noisy images we propose a two step algorithm. At first with a method originally introduced by Qiu (1997) and extended by Garlipp and Müller (2004), we estimate those pixels which belong to the edges of the shapes. Then, these edge points are regarded as coming from a mixture of (linear or circular) regression functions and we estimate the parameters of these functions.

In a simulation study, we demonstrate the immense advantage of using an outlier robust estimator for the edge points.

Keywords: M-estimation, M-kernel estimation, regression cluster, edge detection.

AMS Subject classification: 62 G 05, 62 G 08, 62 G 35, 62 H 30, 62 H 35, 62 P 99

*Research supported by the grant Mu 1031/4-1/2 of the Deutsche Forschungsgemeinschaft.

1 Introduction

For the detection of shapes in noisy images, there are various methods which estimate so called edge points. To detect edge points a (black and white) image can be regarded as two-dimensional intensity function. In areas with no edges, this intensity (or regression) function will be smooth, whereas edges will appear as discontinuities.

For one-dimensional jump detection, Qiu et al. (1991), Müller (1992), and Wu and Chu (1993) introduced similar estimators based on the difference of two one-sided kernel estimates (DKE - Difference Kernel Estimators). For some of these DKEs, results have been obtained about L^2-consistency (Qiu et al., 1991), L^p-consistency (Müller, 1992), and asymptotic normal distribution (Wu and Chu, 1993). A modification of the DKE introduced by Qiu (1994) can be used to estimate the number of jump points.

For the two-dimensional case most considerations about estimation of jump locations concern edge detection in image analysis where diverse methods are known. For example, the so-called *filter*-methods, which, like many others, use the fact that the derivative of the image function becomes very large in the vicinity of edges. Other methods use statistical tests based on the representation of the image as a Markov field. See Davis (1975) for an overview about some of the "classical" methods, or Peli and Malah (1982) for a comparison. For some newer techniques see, for example, Müller and Song (1994), Qiu and Yandell (1997), or Hou and Koh (2003).

As a generalisation of the one-dimensional DKE, Qiu introduced in 1997 the Rotational Difference Kernel Estimators (RDKE). The most important difference between the two-dimensional and the one-dimensional case is that the distinction between "left" and "right" side has now to be done along a direction (see Figure 1). To cope with this problem Qiu (1997) generalized this Difference Kernel Estimator by using rotated kernel functions. Garlipp and Müller (2004) extended this idea to

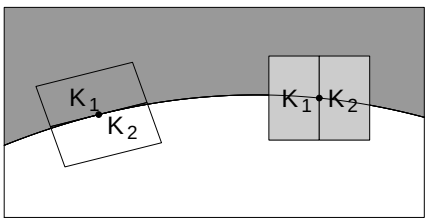


Figure 1: difference of two asymmetric two-dimensional kernel estimations

robust RDKEs by using outlier robust M-kernel estimators, which were introduced by Härdle and Gasser (1984). That way Garlipp and Müller obtained robust esti-

mators for the jump location and height and proved consistency of these estimators in the case of one jump function. Thereby one jump function means that the discontinuities are given as one function of one coordinate. Maximizing in direction of the other coordinate provides then the location of the jump.

However, real images do not consist of one jump function. Even if the image contains only one object like a circle or triangle, the edges of the object cannot be described by one jump function. And in more realistic situations, the images contain several objects. Therefore, in this paper we extend the method of Qiu (1997) and Garlipp and Müller (2004) to detect all pixels close to edges. For that we perform a statistical test based on the RDKE for every pixel to decide whether it belongs to an edge or not. For the RDKE of Qiu (1997) based on the mean, this can be done simply by the classical t-test as already Qiu (2002) noted. But for the RDKE based on an outlier robust location estimator an approximate test is needed which is described in this paper.

Since statistical tests produce level one and level two errors, both methods do not detect all true edge points and some of the detected edge points do not belong to true edges. But, as we show in this paper, the robust RDKE method provides a much smaller number of falsely detected edge points than the nonrobust RDKE method based on the mean.

Nevertheless in both cases, the detected edge points do not provide contour lines for the objects. Hence, we propose in this paper a second step to find the contour lines of the objects. For this we assume, as a first approach, simple shapes of the objects as circles and shapes given by lines. The lines and circles are found by a cluster method based on the identified edge points.

Cluster methods are often used in image analysis (see e.g. Janowitz, 1984, Krishnapuram and Freg, 1992, Granville et al., 1993, Höppner et al., 1999). Most of these methods use all pixel values and positions to identify the clusters. Therefore, they often use a certain distance, e.g. the Mahalanobis distance, which is not outlier robust. But even if robust distance measures are used, as it is done by Davies (1988), Rousseeuw and Van Aelst (1999), or Jolion et al. (1991), these cluster methods have the disadvantage, that shapes with similar gray or color values are assembled in the same cluster. Moreover these methods do not provide explicit representations of the borders of the shapes. These problems do not appear if regression clustering methods are applied to the edge points identified by the robust or nonrobust RDKE method.

For regression clustering also several methods are known (see e.g. Späth, 1979, De-

Sarbo and Cron, 1988, Wedel and Steenkamp 1991, Klesse, 1995). Morgenthaler (1990) and Meer and Tyler (1998) proposed to use M-estimators with redescending score functions to detect different regression clusters. Usually the redescending M-estimators have the disadvantage that the objective function has several local maxima (or minima, respectively). But to identify substructures in the data this is an advantage since each local maximum may correspond to a substructure. Based on this idea, Müller and Garlipp (2005) introduced a cluster method for several linear regression models including orthogonal regression and classical least squares regression and proved consistency of the method. They also shortly demonstrated without details how the method for orthogonal regression can be applied on the edge points identified by the nonrobust RDKE method of Qiu (1997).

Here we describe the method in more detail. Moreover, we apply it not only on edge points identified by the nonrobust RDKE method but also on the points identified by the robust RDKE method of Garlipp and Müller (2004) and compare the results. Since the robust method provides a much smaller number of falsely detected edge points, only the correct edge lines are found. This is not the case for the nonrobust RDKE method: due to many falsely detected edge points, many false lines are found.

Additionally, we show how the cluster method of Müller and Garlipp (2005) can be extended for finding circular structures. Although it is based on the same idea as in Müller and Garlipp for finding regression lines, it is conceptional different. For example, there exists no consistency proof for this up to now. Also the computation is more difficult.

The paper is organized as follows. In Section 2, we describe the methods: Thereby, Section 2.1 is devoted to the robust and nonrobust RDKEs and the corresponding tests for identifying edge points. Section 2.2 describes the second step and provides the cluster methods for finding linear and circular edges. It shows in particular that the problems of finding regression lines and circles are very similar. In Section 3, simulations and applications are presented. Section 3.1 provides a simulation study for the problem of finding the edges of a triangle. This simulation study demonstrates clearly the superiority of the robust RDKE method. Section 3.2 deals with an application in biology. The problem is to find the circular borders of fungi colonies. It is shown that this is well done by the RDKE method for circular edges. Section 4 contains the pseudo codes of all described methods and a link to the web page with the R library of the programs.

2 Method

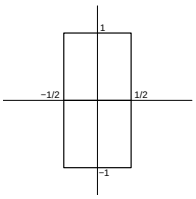
2.1 Identification of edge points

We consider $n_1 \cdot n_2$ independent observations $Z_{ij} = m(x_{ij}) + \epsilon_{ij} \in \mathbb{R}$ at equidistant design points $x_{ij} = \left(\frac{j}{n_2}, \frac{i}{n_1}\right)^\top$ $(1 \le i \le n_1,\, 1 \le j \le n_2)$.

Let $K_1(x)$ and $K_2(x)$ be two one-sided, continuous kernel functions which fulfill the following conditions:

(i) $K_1^*(x) = 0$ for $x \notin \left[-\frac{1}{2}, \frac{1}{2}\right] \times [-1, 0]$ and

$K_2^*(x) = 0$ for $x \notin \left[-\frac{1}{2}, \frac{1}{2}\right] \times [0, 1]$

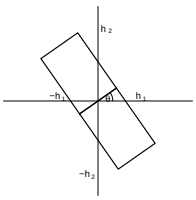


(ii) $\int\limits_{[-1,1]^2} K_j^*(x)dx = 1, \quad j \in \{1, 2\}$

(iii) $K_j^*(x) \ge 0, \quad j \in \{1, 2\}$.

With $a_{1\theta} := (\cos\theta, -\sin\theta)^\top$, $a_{2\theta} := (\sin\theta, \cos\theta)^\top$, $A_\theta := (a_{1\theta}, a_{2\theta})^\top = \left(\begin{smallmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{smallmatrix}\right)$, and $H := \left(\begin{smallmatrix} h_1 & 0 \\ 0 & h_2 \end{smallmatrix}\right)$, let

$$K_j(\theta, x) \;:=\; \frac{1}{h_1 h_2} K_j^*(H^{-1} A_\theta x),$$



for $j \in \{1, 2\}$, where h_1 and h_2 are the bandwidths.

The rotated asymmetric M-kernel estimators $\widehat{m}_j(\theta, x)$ are now defined as zeros of the objective functions $H_j(z; \theta, x)$ with

$$H_j(z; \theta, x) \;:=\; \textstyle\sum_{i=1}^n \alpha_i^{(j)}(\theta, x)\psi(Z_i - z),$$

and

$$\widehat{m}_j(\theta, x) \;\in\; \{z \in \mathbb{R} : H_j(z; \theta, x) = 0\},$$

where $\psi : \mathbb{R} \to \mathbb{R}$ is a score function and $\alpha_i^{(j)}(\theta, x)$ are rotated asymmetric Gasser-Müller weights

$$\alpha_i^{(j)}(\theta, x) \;:=\; \int\limits_{\Delta_i} K_j(\theta, u - x)\, du.$$

Then, with

$$M(\theta, x) \;:=\; \widehat{m}_2(\theta, x) - \widehat{m}_1(\theta, x),$$

$|M(\theta, x)|$ can be used as estimator for the jump height at x along the direction described by θ. It seems plausible, that $|M(\theta, x)|$ will be close to zero independently of θ for x lying in a smooth region of the regression function. But for x lying on an edge $|M(\theta, x)|$ will be close to the jump height, if the direction described by θ corresponds with the direction of the edge. Therefore, the jump height at a point x can be estimated by

$$\widehat{C}(x) \quad = \quad |M(\widehat{\theta}(x), x)|$$

where $\widehat{\theta}(x)$ is the maximizing angle

$$\widehat{\theta}(x) \quad \in \quad \underset{\theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]}{\operatorname{argmax}} \; |M(\theta, x)|.$$

For some asymptotic properties of these estimators see Garlipp and Müller (2004). Since the mean as special unrobust M-kernel estimator and the median as special robust M-kernel estimator both are asymptotically normally distributed, we can perform a test for equal expectations in both windows for a fixed angle. Therefore, it is possible to decide for every pixel if it belongs to an edge by using a multiple test for a number of different angles.

According to the Bonferoni test procedure we can reject the global hypothesis – equal expectations in both windows for every angle – at the level of α, if at least one of the single hypotheses – equal expectations in both windows for one particular angle – can be rejected at the level of α/K, where K is the number of tested angles. For a similar method concerning only the unrobust RDKE, see also Qiu (2002). Since we use equidistant designs points, we have an equal number of observations within the two windows, denoted by N. Let $Z_{11}, \ldots, Z_{1N}$ and $Z_{21}, \ldots, Z_{2N}$ denote the observations within the windows.

In the unrobust case we estimate the variance by the mean of the empirical variances within the windows

$$S_N^2 \quad = \quad \frac{1}{2N-2} \left(\sum_{i=1}^{N} \left(Z_{1i} - \overline{Z}_{1.} \right)^2 + \sum_{i=1}^{N} \left(Z_{2i} - \overline{Z}_{2.} \right)^2 \right).$$

Thus, we use the test statistic of the unpaired student t-test

$$T_N \quad = \quad \frac{\overline{Z}_{1.} - \overline{Z}_{2.}}{S_N} \sqrt{\frac{N}{2}} \tag{1}$$

with a critical value of $t_{2N-2;1-\alpha/(2K)}$, where $t_{n;\alpha}$ is the α quantile of the t-distribution with n degrees of freedom.

In the robust case we need the further assumption that the distribution function F of Z_{jn} ($j = 1, 2$, $n = 1, \ldots, N$) is differentiable at $F^{-1}(1/2)$ with $F'\left(F^{-1}(1/2)\right) > 0$. For $j = 1, 2$ let F_{jN} denote the empirical distribution function of Z_{jn} and

$$R_{jN} \;=\; F_{jN}^{-1}(1/2) - F^{-1}(1/2) - \frac{F_{jN}\left(F^{-1}(1/2)\right) - 1/2}{F'\left(F^{-1}(1/2)\right)}.$$

Then, $\sqrt{N}R_{jN}$ vanishes in probability (see for example Ghosh, 1971). Further, with

$$\begin{aligned} Q_N \;&:=\; F_{1N}\left(F^{-1}(1/2)\right) - F_{2N}\left(F^{-1}(1/2)\right) \\ &=\; \frac{1}{N}\sum_{i=1}^{N}\left(\mathbb{1}_{(-\infty, F^{-1}(1/2))}(Z_{1i}) - \mathbb{1}_{(-\infty, F^{-1}(1/2))}(Z_{2i})\right) =: \frac{1}{N}\sum_{i=1}^{N} Y_i \end{aligned}$$

we have $\mathrm{Var}Y_i = 1/2$ and hence, the central limit theorem provides

$$\sqrt{N}\frac{Q_N}{\sqrt{1/2}} \;=\; \frac{\sum_{i=1}^{N}(Y_i - EY_i)}{\sqrt{N\mathrm{Var}Y_i}} \quad\xrightarrow{\mathcal{L}}\quad Z \sim \mathcal{N}(0,1).$$

As with $\sqrt{N}R_{jN}$ also $\sqrt{N}(R_{1N} - R_{2N})$ vanishes in probability, we have

$$\begin{aligned} &\sqrt{N}\left(\mathrm{med}\left(X_{11}, \ldots, X_{1N}\right) - \mathrm{med}\left(X_{21}, \ldots, X_{2N}\right)\right) \\ =\;& \sqrt{N}\left(F_{1N}^{-1}(1/2) - F_{2N}^{-1}(1/2)\right) \\ =\;& \sqrt{N}\left(R_{1N} - R_{2N}\right) + \sqrt{N}\frac{Q_N}{F'\left(F^{-1}(1/2)\right)} \\ \xrightarrow{\mathcal{L}}\;& Z \sim \mathcal{N}\left(0, \frac{1/2}{\left(F'\left(F^{-1}(1/2)\right)\right)^2}\right). \end{aligned}$$

Assuming for example normally distributed observations, this means

$$\sqrt{N}\left(\mathrm{med}\left(X_{11}, \ldots, X_{1N}\right) - \mathrm{med}\left(X_{21}, \ldots, X_{2N}\right)\right) \xrightarrow{\mathcal{L}} Z \sim \mathcal{N}(0, \pi\sigma^2).$$

Estimating the variances within the windows by the consistent median absolute deviations $M_{jn} = \mathrm{MAD}\left(X_{j1}, \ldots, X_{jN}\right)$, by Slutzky's theorem, we finally get

$$\begin{aligned} U_N \;&:=\; \sqrt{N}\frac{\mathrm{med}\left(X_{11}, \ldots, X_{1N}\right) - \mathrm{med}\left(X_{21}, \ldots, X_{2N}\right)}{\sqrt{\frac{\pi}{2}\left(M_{1N}^2 + M_{2N}^2\right)}} \qquad (2) \\ &=\; \frac{\sqrt{\frac{\pi}{2}\left(\sigma^2 + \sigma^2\right)}}{\sqrt{\frac{\pi}{2}\left(M_{1N}^2 + M_{2N}^2\right)}} \cdot \frac{\sqrt{N}\left(\mathrm{med}\left(X_{11}, \ldots, X_{1N}\right) - \mathrm{med}\left(X_{21}, \ldots, X_{2N}\right)\right)}{\sqrt{\pi\sigma^2}} \\ &\xrightarrow{\mathcal{L}}\; Z \sim \mathcal{N}(0,1). \end{aligned}$$

Therefore, we can perform an asymptotical Gaussian test with test statistic U_N and the $1 - \alpha/(2K)$ quantile of the standard normal distribution as critical value.

2.2 Clustering of edge points

The detected edge points, now denoted by $z_1, \ldots, z_N$, are assumed as coming from an error-in-variable model. This means that they are realizations of

$$Z_i \;=\; \begin{pmatrix} X_i \\ Y_i \end{pmatrix} + \begin{pmatrix} E_{1i} \\ E_{2i} \end{pmatrix},$$

where X_i, Y_i, E_{1i}, and E_{2i} are independently identically distributed.

2.2.1 Linear

Considering a mixture of L regression lines with parameters $(\alpha_l, \beta_l) \in [-\pi, \pi] \times \mathbb{R}$, the coordinates of Z_i fulfill

$$a_l^\top \begin{pmatrix} X_i \\ Y_i \end{pmatrix} \;=\; \beta_l \qquad \text{almost sure,}$$

if Z_i is coming from the l-th regression, where $a_l = (\cos(\alpha_l), \sin(\alpha_l))^\top$.

Now, the set of parameters $(\alpha, \beta) \in [-\pi, \pi] \times \mathbb{R}$, for which

$$H(\alpha, \beta) \;=\; \frac{1}{N} \sum_{i=1}^{N} \frac{1}{s} \rho \left(\frac{\begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \end{pmatrix}^\top z_i - \beta}{s} \right)$$

has a local maximum, can be used as estimation for the regression parameters (α_l, β_l), where ρ is a redescending score function measuring the distance of the observations z_i to the regression line with parameter (α, β), and s is a scale parameter (see Müller and Garlipp, 2004).

For computing all local maxima of H, Müller and Garlipp (2005) proposed to use the Newton-Raphson method using every line through any two observations as starting values. But due to the large number of edge points, this would result in huge computation times. In order to avoid this, we used the fact that the testing method from the first step provides not only the edge points z_i themselves but also the direction $\widehat{\theta}(z_i)$ of the edge. Thereby, we have a dedicated line through each edge point providing a merely linear number of very accurate starting values.

2.2.2 Circular

In the case of circular regressions the coordinates now fulfill

$$\left\| \begin{pmatrix} X_i \\ Y_i \end{pmatrix} - \begin{pmatrix} a_{1l} \\ a_{2l} \end{pmatrix} \right\| \;=\; r_l \qquad \text{almost sure,}$$

if Z_i is coming from the l-th regression, where $a_l = (a_{1l}, a_{2l})^\top$ is the center and r_l is the radius of the l-th circle.

Corresponding to the linear case, we now use an objective function

$$H(a, r) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{s} \rho \left(\frac{||z_i - a|| - r}{s} \right),$$

where the distance of the observations z_i to a circle with center a and radius r is measured by the score function. Then, the local maxima of $H(a, r)$ can be used as estimates for the parameters of the regression circles.

For the computation of these local maxima, we again used the Newton-Raphson method. The choice of the starting values has to be carried out very carefully. The obvious idea is to use a sufficient number of starting values so that all local maxima of H are found. A heuristic method to get these starting values is the following: Use a sufficiently fine grid of equidistant center points and a number of radii corresponding to the assumed sizes of the circles.

3 Simulation and application

3.1 Linear edges

Both multiple tests of Section 2.1 were applied to a 100×100 - pixel image displayed in Figure 2 showing a triangle bounded by the regression lines $r_j(x_1) = a_j x_1 + b_j$ $(j = 1, 2, 3)$ where $(a_1, b_1) = (0, 0.3)$, $(a_2, b_2) = \left(-\frac{10}{7}, \frac{1}{7}\right)$, and $(a_3, b_3) = \left(\frac{10}{7}, \frac{11}{7}\right)$ and blurred by 30% uniformly distributed outliers.

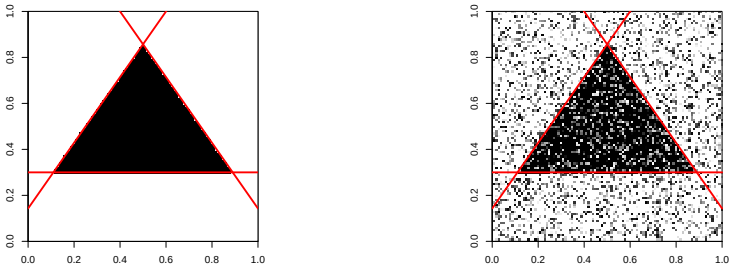


Figure 2: Original image $m(x)$ (left) and noisy observations Z_i (right) with true regression lines

If the size of the structures that are to be detected is known, the bandwidths can be chosen with respect to this size. If the bandwidth is too small, the tests have too low power to detect the edges. On the other hand, with windows too large, small structures cannot be detected, since for all design points with a distance to an edge less than the window size, the hypothesis of equal expectations in both windows does not hold. This means that we get many rejections not only exactly at the edges but in a whole environment of the edges, which therefore should be smaller than the structures that are to be identified. In this example we chose $h_1 = h_2 = 0.05$, which is approximately 1/4 of the radius of the inscribed circle of the triangle.

With these bandwidths, for every design point with sufficient distance to the margins, we performed a multiple test with 32 different angles for the global hypothesis of equal expectations in both windows. The pixels for which the global hypothesis is rejected at a 10% level with the two test statistics from (1) and (2) are shown in Figure 3. Table 1 shows type I and II errors, where the type I error concerns only pixels outside the 0.05 environment of the true edges, and for calculating the type II error pixels are regarded "true" edge points, if their distance to an edge is less than the resolution of the image, which is 0.01.

As expected, the robust test performs significantly better, what has considerable effects for the estimation of the regression lines in the next step.

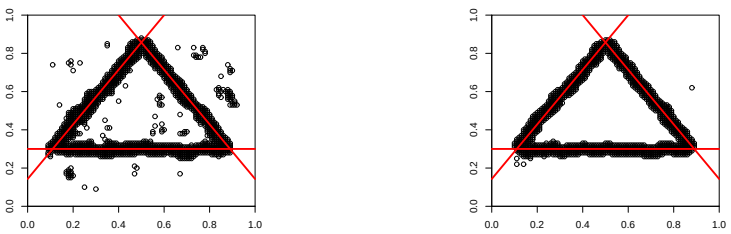


Figure 3: Detected edge points, unrobust (left) and robust (right) with true regression lines

	unrobust	robust
falsely detected	100	3
Type I error	0.024	0.001
not detected	1	1
Type II error	0.028	0.028

Table 1: Type I and II errors of the unrobust and robust version of the multiple test

For the cluster method we used the density of the standard normal distribution as score function and chose the scale parameter s with respect to the bandwidths h_j of the previous step in such a way, that points within a h_j environment of a regression line get 90% of the weight, that is $s = \frac{h}{u_{0.95}} \approx 0.03$.

Figure 4 shows again the estimated edge points from the first step but now together with the estimated regression lines. Table 2 presents their estimated parameters. Obviously, the occurrence of falsely detected edge points results in a high number of falsely detected regression lines. Thus, the robust test method provides significantly better initial values for the cluster method.

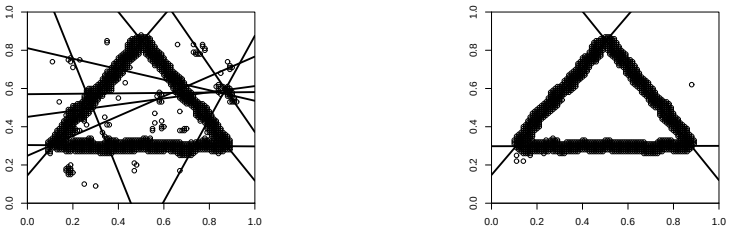


Figure 4: Estimated regression lines with edge points from Figure 3 as observations

	true		robust		unrobust	
j	a_j	b_j	a_j	b_j	a_j	b_j
1	0.0000	0.0300	0.0017	0.2981	−0.0069	0.3041
2	−1.4286	0.1429	−1.4715	1.5918	−1.4708	1.5886
3	1.4286	1.5714	1.3934	0.1478	1.3949	0.1454
4					−2.9639	1.3488
5					−1.7269	2.0991
6					−0.2751	0.8101
7					0.0109	0.5702
8					0.1611	0.4515
9					0.5187	0.2483
10					2.1645	−1.2885

Table 2: True and estimated parameters of the regression lines

3.2 Circular edges

In the following example originated from the biological research on the growth of fungi under different conditions (see e.g. Sterflinger and Krumbein, 1995), the aim is to estimate the size of colonies of black fungi on a marble surface (see Figure 5). Since these colonies grow approximately with the same rate in each direction, this is to find circles within an image of the surface and to estimate their diameter.

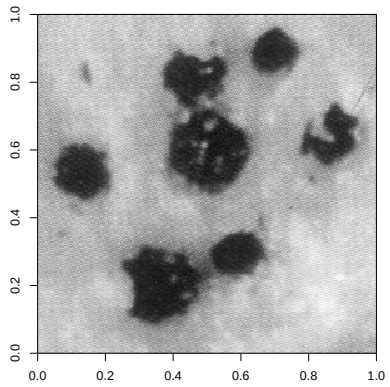


Figure 5: 272×271 pixel image, showing a marble surface inhabited by colonies of black fungi (taken from Sterflinger and Krumbein, 1995).

Since the noise of the image is less dominated by outliers but more an additive noise, we used the unrobust version of our tests from Section 2, which has a better power in this case. The estimated edge points, found with a window size of 5×5 pixels (with an image size of 272×271 pixels that is $h_1 \approx h_2 \approx 0.007$), and using 4 different angles are shown in Figure 6.

As already mentioned, the choice of the starting values for the cluster method is very important for the computation time and proper results. It is evident that not only those circles are found that represent real clusters, but for example also some larger ones, which coincide with a large number of edge points from different true circles (see Figure 6).

There exist several methods to recognize the true circles afterwards, but to reduce computation time, we tried to reduce the number of unsuitable circles already during computation. To achieve this, we aborted the maximization when the radius left the interval $[0.02, 0.15]$ or if the center point gets out of the image. Furthermore, if we start with a small radius within a true circle, it is reasonable that the Newton-Raphson methods converges to the true circle. Hence, it is sufficient to start with only one value for the radius, provided it is smaller than the smallest colony and the

grid of center points is fine enough. Therefore, we used 25×25 equidistant center points within $[0,1]^2$ and $r = 0.03$ as starting values.

The smaller the scale parameter is, the more circles are found. But with a larger scale parameter the deviation of the estimated circles becomes larger (see also Müller and Garlipp, 2005, for the case of regression lines). We chose a scale parameter with which at least all true circles are found ($s = 0.025$). Since generally also some false circles are detected (see Figure 6) the "good" estimates are determined afterwards.

There exists several methods to do this. E.g. a usual method is to select those cluster to which most of the points belong to. In the case of circular regression clusters, this should be done in relation to the estimated radius. Beside these general applicable procedures, our cluster method provides two other criteria. Especially if the number of true clusters is small, it is very accurate to choose those clusters with the largest local maxima of the objective function H. The second criterion is based on the fact, that the number of starting values is much larger than the number of clusters, so that each cluster is found several times. So it is plausible to choose the true clusters by counting, how often each cluster is reached by the maximization process. Figure 7 shows all circles which are reached more than once, which are exactly those, which describe the fungi colonies. Again, the used score function ρ was the density of the standard normal distribution.

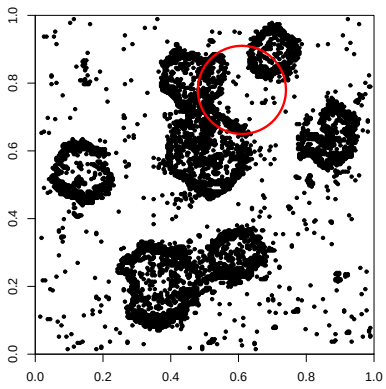


Figure 6: Estimated edge points with an example for an falsely detected circle

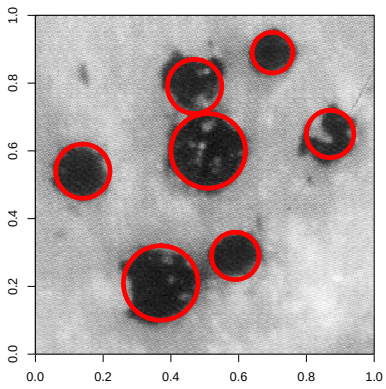


Figure 7: Original image with estimated circles

4 Pseudo codes

All programs can be found in the R library "edci", which is available at http://cran.r-project.org/.

4.1 Computational details

The observations Z_{ij} are assumed to be stored in a two-dimensional array `data[i,j]` of the size of `nrow*ncol`. $\widehat{C}(x)$ is only computed for x lying on the grid of the design points. Therefore, $x_{\mathtt{ij}} - x$ takes only values on the grid $\left\{ \left(\frac{\mathtt{l}}{\mathtt{ncol}}, \frac{\mathtt{k}}{\mathtt{nrow}}\right)^\top, \mathtt{k} = -\mathtt{nrow}, \ldots, \mathtt{nrow}, \mathtt{l} = -\mathtt{ncol}, \ldots, \mathtt{ncol} \right\}$ and since for every θ the Kernel functions $K_j(\theta, x_{\mathtt{ij}} - x)$ are zero for $x_{\mathtt{ij}} - x \notin \left[-\sqrt{h_1^2 + h_2^2}, \sqrt{h_1^2 + h_2^2}\right]^2$, it is sufficient to calculate $K_j\left(\theta, \left(\frac{\mathtt{l}}{\mathtt{ncol}}, \frac{\mathtt{k}}{\mathtt{nrow}}\right)^\top\right)$ for $(\mathtt{k}, \mathtt{l}) \in \{-\mathtt{w1}, \ldots, \mathtt{w1}\} \times \{-\mathtt{w2}, \ldots, \mathtt{w2}\}$ with $\mathtt{w1} = \left\lceil \sqrt{h_1^2 + h_2^2} \cdot \mathtt{nrow} \right\rceil$, $\mathtt{w2} = \left\lceil \sqrt{h_1^2 + h_2^2} \cdot \mathtt{ncol} \right\rceil$. Note, that this is independent of x. Additionally, only a fixed number of angles $\theta_{\mathtt{t}}$ ($\mathtt{t} \in \{1, \ldots, \mathtt{nang}\}$) is used, what means, that all occurring weights have to be computed only once and can be stored in a three-dimensional array of the size of `(2*w1+1)*(2*w2+1)*nang`.

To reduce computation time, the two kernel functions K_1^* and K_2^*, which have disjoint supports, are summed, so that only one kernel function $K(x)$ - and thus only one array `alpha[k,l,t]` - has to be used. The distinction, if a certain observation belongs to the one window or the other, is done via the x_2-coordinate of the corresponding (rotated and scaled) design point, which has to be stored in another three-dimensional array `dpx2[k,l,t]`. Although this needs still two arrays the concatenation of the two kernel functions is effective, since this reduces the number of summations to the half.

The coordinates and weights are computed by the following short routine, where `K` is the kernel function and `rot1` and `rot2` are functions calculating the rotation.

```
for (k,l) in {-w1,...,w1} x {-w2,...,w2} do
   for t in {1,...,nang} do
      theta              := -Pi/2 + Pi*t/nang
      x1                 := rot1(theta, l/ncol, k/nrow)/h1
      x2                 := rot2(theta, l/ncol, k/nrow)/h2
      dpx2[k+w1,l+w2,t]  := x2
      alpha[k+w1,l+w2,t] := K(x1,x2)
```

For all minimizations (described later in 4.3.ii, 4.5, 4.6) Armijo's Rule (see Armijo, 1966) is used for damping the step size of the Newton-Raphson method.

4.2 List of variables and functions

- Detecting edge points (4.3–4.4)

`nrow`	Number of rows of the data
`ncol`	Number of columns of the data
`data[i,j]`	Two-dimensional array of the size `nrow` $\times$ `ncol` containing the observations Z_{ij}.
`nang`	Number of equidistant angles for computing $\widetilde{C}_n(x_i)$
`h1, h2`	bandwidths h_1 and h_2
`w1, w2`	window size `w1` $= \left\lceil \sqrt{h_1^2 + h_2^2} \cdot$ `ncol` $\right\rceil$, `w2` $= \left\lceil \sqrt{h_1^2 + h_2^2} \cdot$ `nrow` $\right\rceil$
`alpha`	Three-dimensional array of the size `(2*w1+1)*(2*w2+1)*nang` containing the kernel weights
`dpx2`	Three-dimensional array of the size `(2*w1+1)*(2*w2+1)*nang` containing the x_2-coordinates of the transformed design points
`rho(z)`	score function
`drho(z)`	first order derivative of `rho`
`ddrho(z)`	second order derivative of `rho`
`pnorm(x)`	distribution function of the standard normal distribution
`pt(x,n)`	distribution function of the t-distribution with n degrees of freedom

- Clustering (4.5–4.6)

`n`	Number of detected edge points
`x`, `y`	vectors of length `n` containing the coordinates of the detected edge points
`alpha`	vector of length `n` containing the angles $\widehat{\theta}\left((\texttt{x[i]}, \texttt{y[i]})^\top\right)$
`rho(z)`	score function
`s`	scale parameter
`nx`, `ny`	number of starting coordinates for circular regression clustering
`startradius`	initial radius for circular regression clustering

4.3 Calculation of $\widehat{C}$ and $\widehat{\theta}$

The results for $\widehat{C}\left(\frac{\texttt{j}}{\texttt{ncol}}, \frac{\texttt{i}}{\texttt{nrow}}\right)^\top$ and $\widehat{\theta}\left(\frac{\texttt{j}}{\texttt{ncol}}, \frac{\texttt{i}}{\texttt{nrow}}\right)^\top$ are stored in the two-dimensional arrays `M[i,j]` and `T[i,j]`.

(i) Kernel estimators

```
for (i,j) in {w1+1,...,nrow-w1} x {w2+1,...,ncol-w2} do
   for t in {1,...,nang} do
      m1    := 0   m2    := 0        // estimates in the windows
      wsum1 := 0   wsum2 := 0        // sums of the weights
      for (k,l) in {-w1,...,w1} x {-w2,...,w2} do
         x2 := dpx2[k+w1,l+w2,t]
         a  := alpha[k+w1,l+w2,t]
         if a != 0 then
            if x2 < 0 then
               m1   := m1   + a*data[i+k,j+l]
               wsum1 := wsum1 + a
            if x2 > 0 then
               m2   := m2   + a*data[i+k,j+l]
               wsum2 := wsum2 + a
      diff := |m2/wsum2 - m1/wsum1|
      if diff > M[i,j] or t = 1 then
         M[i,j] := diff
         T[i,j] := -Pi/2 + Pi*t/nang
```

(ii) M-kernel estimators

```
DFAC := 0.7 // factor for damping (Armijo's Rule)
for (i,j) in {w1+1,...,nrow-w1} x {w2+1,...,ncol-w2} do
   for t in {1,...,nang} do
      // calculate medians as starting values
      n1 := 0   n2 := 0
      for (k,l) in {-w1,...,w1} x {-w2,...,w2} do
         x2 := dpx2[k+w1,l+w2,t]
         a  := alpha[k+w1,l+w2,t]
         if a != 0 then
```

```
      if x2 < 0 then
         n1 := n1 + 1
         window1[n1] := data[i+k,j+l]
      if x2 > 0 then
         n2 := n2 + 1
         window2[n2] := data[i+k,j+l]
z1 := median(window1[1],...,window1[n1])
z2 := median(window2[1],...,window2[n2])
// calculate M-esitmates simultanously in both windows
// by the Newton-Raphson method
step1 := 0   step2 := 0
repeat
   // value of the objective functions and their first
   // and second order derivatives at z1 and z2 resp.
   h1 := 0   dh1  := 0   ddh1 := 0
   h2 := 0   dh2  := 0   ddh2 := 0
   for (k,l) in {-w1,...,w1} x {-w2,...,w2} do
      x2 := dpx2[k+w1,l+w2,t]
      a  := alpha[k+w1,l+w2,t]
      if a != 0 then
         if x2 < 0 then
            h1   := h1   + a *   rho(z1-data[i+k,j+l])
            dh1  := dh1  + a *  drho(z1-data[i+k,j+l])
            ddh1 := ddh1 + a * ddrho(z1-data[i+k,j+l])
         if x2 > 0 then
            h2   := h2   + a *   rho(z2-data[i+k,j+l])
            dh2  := dh2  + a *  drho(z2-data[i+k,j+l])
            ddh2 := ddh2 + a * ddrho(z2-data[i+k,j+l])
   // direction and step size in window 1
   if ddh1 <= 0 then        // h1 concav
      if dh1 > 0 then
         step1 := -stepsize
      else
         step1 := stepsize
   else                     // h1 convex => Newton
      step1 := -dh1/ddh1
```

```
      // Armijo's Rule
      fac1 := 1
      repeat
         fac1 := fac1*DFAC
         hnew := 0
         for (k,l) in {-w1,...,w1} x {-w2,...,w2} do
            x2 := dpx2[k+w1,l+w2,t]
            a  := alpha[k+w1,l+w2,t]
            if a != 0 and x2 < 0 then
               hnew := hnew + a*rho(z1+step1*fak1-data[i+k,j+l])
      until hnew - (h1 + 0.5*fac1*dh1^2) <= 0.00001

      // direction and step size in window 2 as above

      z1 := z1 + step1*fac1
      z2 := z2 + step2*fac2
   until |fac1*step1| <= 0.00001 and |fac2*step2| <= 0.0001
   diff := |z2 - z1|
   if diff > M[i,j] or t = 1 then
      M[i,j] := diff
      T[i,j] := -Pi/2 + Pi*t/nang
```

4.4 Multiple Tests

Only the code for the Student t-test is given. For the asymptotical Gaussian test replace the mean and the empirical variance by the median and MAD and calculate the p-value by

```
pvalue := (1-pnorm(sqrt((n1+n2)/2)*|median1-median2|
                   /sqrt(Pi/2*(var1^2+var2^2)),n1+n2-2))
          *nang*2
```

The p-values for every pixel $\left(\frac{\texttt{j}}{\texttt{ncol}}, \frac{\texttt{i}}{\texttt{nrow}}\right)^\top$ and the values of $\widehat{\theta}\left(\frac{\texttt{j}}{\texttt{ncol}}, \frac{\texttt{i}}{\texttt{nrow}}\right)^\top$ are stored in the two-dimensional arrays `P[i,j]` and `T[i,j]`.

```
for (i,j) in {w1+1,...,nrow-w1} x {w2+1,...,ncol-w2} do
   for t in {1,...,nang} do
      n1 := 0   n2 := 0
      for (k,l) in {-w1,...,w1} x {-w2,...,w2} do
         x2 := dpx2[k+w1,l+w2,t]
         a  := alpha[k+w1,l+w2,t]
         if a != 0 then
            if x2 < 0 then
               n1 := n1 + 1
               window1[n1] := data[i+k,j+l]
            if x2 > 0 then
               n2 := n2 + 1
               window2[n2] := data[i+k,j+l]
      mean1 := mean(window1[1],...,window1[n1])
      mean2 := mean(window2[1],...,window2[n2])

      if |mean1-mean2| = 0 then
         pvalue := 1
      else
         var1  := var(window1[1],...,window1[n1])
         var2  := var(window2[1],...,window2[n2])
         if var1+var2 = 0 then
            pvalue := 0
         else
            pvalue := (1-pt(|((mean1-mean2)/sqrt((var1+var2)/2))
                              *sqrt((n1*n2)/(n1+n2))|,n1+n2-2))
                      *nang*2
      if pvalue < P[i,j] or t = 1 then
         P[i,j] := pvalue
         T[i,j] := -Pi/2 + Pi*t/nang
```

4.5 Linear regression clustering

```
DFAC := 0.7 // factor for damping (Armijo's Rule)

set_of_solutions := empty_set
for i in {1,...,n} do
   a = alpha[i]
   b = cos(a)*x[i]+sin(a)y[i]
   repeat
      // value of the objective functions and their first
      // and second order derivatives
      h   := 0
      dh  := zero_vector(2)
      ddh := zero_matrix(2,2)
      for j in {1,...,n} do
         z   := (cos(a)*x[j]+sin(a)*y[j]-b)/s
         h   := h   + rho(z)
         dh  := dh  + grad(rho(z))
         ddh := ddh + hesse(rho(z))
      det = determinant(ddh)
      if det > 0 and ddh[1,1] > 0 then    // Newtow
         stepa = -(ddh[2,2]/det*dh[1]-ddh[1,2]/det*dh[2])
         stepb = -(ddh[1,1]/det*dh[2]-ddh[1,2]/det*dh[1])
      else                                // Steepest Descent
         stepa = -dh[1]
         stepb = -dh[2]

      // Armijo's Rule
      fac := 1
      repeat
         fac  := fac*DFAC
         hnew := 0
         for k in {1,...,n} do
            z    := (cos(a+fac*stepa)*x[k]+sin(a+fac*stepa)*y[k]
                    -(b+fac*stepb))/s
            hnew := hnew + rho(z)
      until hnew - (h + 0.5*fac*(dh[1]^2+dh[2]^2)) <= 0.00001

   until fak*(|stepa|+|stepb|) <= 0.00001
   insert (a,b) into set_of_solutions
```

4.6 Circular regression clustering

```
DFAC := 0.7 // factor for damping (Armijo's Rule)
set_of_solutions := empty_set
for (i,j) in {1,...,nx} x {1,...,ny} do
   a1 := i/nx
   a2 := j/ny
   r  := startradius
   repeat
      // objective funct. and their 1st & 2nd order derivatives
      h   := 0
      dh  := zero_vector(3)
      ddh := zero_matrix(3,3)
      for k in {1,...,n} do
         z   := (sqrt((x[k]-a1)^2+(y[k]-a2)^2)-b)/s
         h   := h   + rho(z)
         dh  := dh  + grad(rho(z))
         ddh := ddh + hesse(rho(z))
      det  = determinant(ddh)
      det2 = ddh[1,1]*ddh[2,2]-ddh[1,2]*ddh[2,1]
      if det > 0 and det2 > 0 then          // Newtow
         stepa1 := (-ddh^(-1)*dh)[1]
         stepa2 := (-ddh^(-1)*dh)[2]
         stepr  := (-ddh^(-1)*dh)[3]
      else                                  // Steepest Descent
         stepa1 = -dh[1]
         stepa2 = -dh[2]
         stepr  = -dh[3]
      // Armijo's Rule
      fac := 1
      repeat
         fac  := fac*DFAC
         hnew := 0
         for k in {1,...,n} do
            z    := (sqrt((x[k]-(a1+fac*stepa1))^2+
                          (y[k]-(a2+fac*stepa2))^2)-(b+fac*stepb))/s
            hnew := hnew + rho(z)
      until hnew - (h + 0.5*fac*(dh[1]^2+dh[2]^2+dh[3]^2)) <= 0.00001
   until fak*(|stepa1|+|stepa2|+|stepr|) <= 0.00001
   insert (a1,a2,r) into set_of_solutions
```

References

[1] Armijo, L. (1962), Minimization of functions having Lipschitz continous first part derivatives, *Pacific Journal of Mathematics* **16**, 1–3.

[2] Desarbo, W. S. and Cron, W. L. (1988), A maximum likelihood methodology for clusterwise linear regression, *Journal of Classification* **5**, 249–282.

[3] Davis, L. (1975), A Survey of Edge Detection Techniques, *Computer Graphics and Image Processing* **4**, 248–270.

[4] Davies, P. L. (1988), Consistent estimates for finite mixtures of well separated elliptical distributions. In: *Classification and Related Methods of Data Analysis*, Bock, H.-H. (Ed.), Elsevier Science Publishers, Amsterdam, 195–202.

[5] Garlipp, T., Müller, C. H. (2004), Robust Jump Detection in Regression Surface, *Submitted*.

[6] Granville, V. Krivánek M., Rasson, J.-P. (1993), Clustering, calssification and image segmentation on the grid, *Computational Statistics and Data Analysis* **15** No. 2.

[7] Härdle, W., Gasser, T. (1984), Robust Nonparametric Function Fitting, *Journal of the Royal Statistical Society, Series B* **46**, 42–51.

[8] Höppner, F., Klawonn, F., Kruse, R., Runkler, T. (1999), *Fuzzy cluster analysis. Methods for classification, data analysis and image recognition*, Wiley, Chinchester.

[9] Hou, Z., Koh, T. S. (2003), Robust edge detection, *Pattern Recognition* **36**, 2083–2091.

[10] Janowitz, M.F. (1984), A cluster analysis program for image segmentation, *Statistical signal processing*, Statistics, textbooks and monographs, **53**, 399–410.

[11] Jolion, J.-M., Meer, P., Bataouche, S. (1991), Robust Clustering with Applications in Computer Vision, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **13**, 791-802

[12] Klesse, G. M. (1995), Neue Versuche der Entwicklung robuster Regressionsschätzer – Mediangerade und Cluster-Regression *Dissertation*, Universität Karlsruhe.

[13] Krishnapuram, R., Freg, C.-P. (1992), Fitting an unknown number of lines and planes to image data through compatible cluster merging, *Pattern Recognition* **25**, 385–400.

[14] Meer, P. and Tyler, D. E. (1998), Smoothing the gap between statistics and image understanding. Comments on the paper *Edge-preserving smoothers for image processing*, Chu, C. K., Glad, I. K., Godtliebsen, F., Marron, J. S., *J. Amer. Statist. Assoc.* **93**, 526-541.

[15] Morgenthaler, S. (1990), Fitting redescending M-estimators in regression. In: *Robust Regression,* Lawrence, H. D. and Arthur, S. (Eds.), Dekker, New York, 105-128.

[16] Müller, H.-G. (1992), Change-Points in Nonparametric Regression Analysis, *Annals of Statistics* **20**, 737–761.

[17] Müller, C. H., Garlipp, T. (2005), Simple consistent cluster methods based on redescending M-estimators with an application to edge identification in images, *Journal of Multivariate Analysis* **92/2**, 359–385.

[18] Müller, H.-G., Song, K.-S. (1994), Maximin estimation of multidimensional boundaries, *Journal of Multivariate Analysis* **50**, 265–281.

[19] Peli, T., Malah, D. (1982), A Study of Edge Detection Algorithms, *Computer Graphics and Image Processing* **20**, 1–21.

[20] Qiu, P. (1994), Estimation of the Number of Jumps of the Regression Functions, *Communications in Statistics - Theory and Methods* **23**, 2141–2155.

[21] Qiu, P. (1997), Nonparametric Estimation of Jump Surface, *Sankhya: The Indian Journal of Statistics, Series A* **59**, 268–294.

[22] Qiu, P. (2002), A nonparametric procedure to detect jumps in regression surfaces, *Journal of Computational and Graphical Statistics* **11**, 799–822.

[23] Qiu, P., Asano, C., Li, X. (1991), Estimation of Jump Regression Function, *Bulletin of Informatics and Cybernetics* **24**, 197–212.

[24] Qiu, P., Yandell, B. (1997), Jump Detection in Regression Surfaces, *Journal of Computational and Graphical Statistics* **6**, 332–354.

[25] Rousseeuw, P. J. and Van Aelst, S. (1999), Positive-breakdown robust methods in computer vision. In: *Computing Science and Statistics, Vol 31*, (edited by K. Berk and M. Pourahmadi, eds.), Interface Foundation of North America, Inc., Fairfax Station, VA, 451-460

[26] Späth, H. (1979), Clusterwise linear regression, *Computing* **22**, 367–373.

[27] Sterflinger, K., Krumbein, W. E. (1995), Multiple Stress Factors affecting Growth of Rock-inhabiting Black Fungi, *Botanica Acta* **108**, 490–496.

[28] Wedel, M., Steenkamp, J.-B. E. M. (1991), A clusterwise regression method for simultaneous fuzzy market structuring and benefit segmentation, *Journal of Marketing Research* **28**, 385–396.

[29] Wu, J. S. Wu, Chu, C. K. (1993), Kernel Type Estimators of Jump Points and Values of a Regression Function, *Annals of Statistics* **21**, 1545–1566.