university of groningen

faculty of mathematics and natural sciences

Improved Detection of Faint Extended Astronomical Objects through Statistical Attribute Filtering

Bachelor's thesis

June 2015

Student: P. D. Teeninga

Primary supervisor: M. H. F. Wilkinson

Secondary supervisor: U. Moschini

Secondary supervisor: G. R. Renardel de Lavalette

Abstract

In astronomy, images are produced by sky surveys containing large numbers of objects. SExtractor is a widely used program for automated source extraction and cataloging but struggles with faint extended sources. Using SExtractor as a reference, the paper describes a Max-Tree-based method for improved extraction of faint extended sources without stronger image smoothing. Node filtering depends on the noise distribution of a statistic calculated from attributes. Run times are in the same order as SExtractor.

1 Introduction

In astronomy, images of the night sky are produced containing large numbers of objects, mostly stars and galaxies. An example is the Sloan Digital Sky Survey[9] (SDSS) where the DR7[2] catalog contains 357 million unique objects. With advances in technology more data is available and manually extracting every object is not feasible.

Current extraction software packages, including SExtractor [3], use thresholding as one of the first steps with the purpose of image segmentation while avoiding false positives. The downside is that parts of objects below the (primary) threshold are discarded (figure 1).

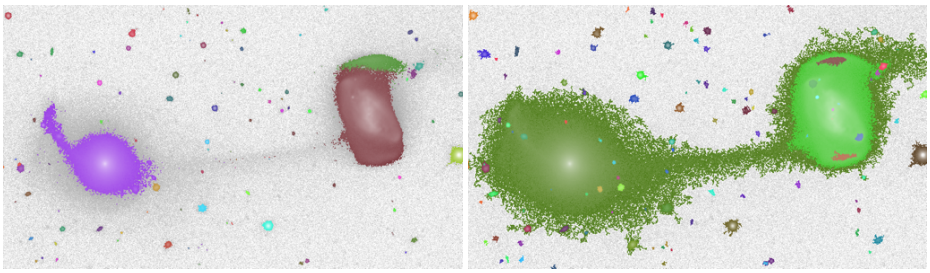

Figure 1: Results of SExtractor with default settings (top) and the presented method (bottom). Log scale. File `fpC-002078-r1-0157.fit` (cropped).

A vast literature on thresholding exists [8], including local automated thresholding. Most of the techniques do not work for the data set because the assumptions on which they are based do not hold. There is no clear distinction between foreground and background pixels and edges are not well defined. Either too much noise is picked up, generating false positives, or the (local) threshold is set too high because of bright objects. Otsu's method only detects two objects in the image of figure 1, which is similar to the number of objects seen in a linear-scale view of the image. The result of a robust automatic threshold selector (GMRATS [12]) is shown in figure 2. Apart from selecting a correct threshold, multi-thresholding is also wanted because of nested objects.

Figure 2: Result of GMRATS for the image in figure 1.

As improvement to a primary threshold, our method locally tests the threshold depending on object size in pixels in addition to statistical properties of the noise. The supporting data structure is a Max-Tree [7] (figure 3) created from the image. In a Max-Tree, every node corresponds to a connected component at the relevant threshold range. The choice is inspired by the component tree used in SExtractor. Nodes are marked significant if noise is an unlikely cause, given a significance level. Nested significant nodes can represent the same object and the choice of which is best is one of the problems. To separate nested objects (deblending), other significant branches are considered new objects.

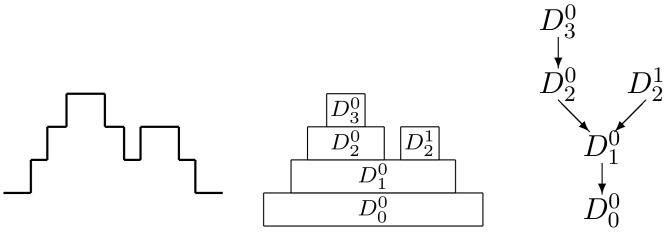

Figure 3: 1-D signal (left), corresponding peak compo- nents (middle) and the Max-Tree (right). Figure from [11].

Results are compared with SExtractor. In SExtractor, every connected component above the primary threshold is re-thresholded at N levels with logarithmic spacing, by default. The range is between the primary threshold and maximum value of the component. A tree similar to a Max-Tree is constructed. Branches are considered separate objects if the integrated intensity is above a certain fraction of the total intensity and if another branch with such property exists.

An image background, caused by light reflected and photo-chemical reactions in earth's atmosphere, is estimated and subtracted before thresholding. SExtractor's estimate shows bias from objects (figure 4), which reduces their intensities. Backgrounds in the data set turn out to be nearly flat and constant estimates per image are used instead. The method, which also reduces object bias, is described in the next section.

Figure 4: Background estimate of the image in figure 1 by SExtractor, contrast stretched.

A data set of 254 monochrome images is used for validation. It is a subset of the corrected images in SDSS DR7. Selection is based on the inclusion of

merging and/or overlapping galaxies which often include faint structures. Only images from the r-band are used, which have the best quality [5]. Images were acquired with a CCD in which photons are converted to electrons and counted per pixel. After subtracting the software bias from the corrected images, pixel values are directly proportional to photoelectron counts [1]. Noise is mostly Poissonian due to photoelectron counts. The distribution is close to Gaussian. The sky (background) typically contributes 670 photoelectrons to the counts per pixel. In our method, pixel values are assumed to be from a Gaussian distribution, with variance linearly dependent on the signal.

The paper is organized as follows: first the method for estimating background is described (section 2), secondly the object extraction (section 3), thirdly a comparison of methods including SExtractor (section 4) and finally the conclusion and ideas for future work (sections 5 and 6).

2 Background estimation

The image is assumed to be the sum of a background image B, objects image O and Gaussian noise where the variance is equal to $g^{-1}(B+O)+R$, for per-image constants g, equivalent to `gain` in the SDSS, and R, which is due to other noise sources; read noise, dark current and quantization.

A tile of the image will be called flat if the pixel values could have been drawn from a single Gaussian distribution, e.g. $B+O$ is close to constant in the tile. The background is approximated by the mean value of flat tiles (algorithm 2) and is subtracted from the image.

2.1 Searching for flat tiles

The image is split into tiles of the same size. The following statistical tests are applied to the tiles (see algorithm 1):

1. A normality test using the D'Agostino-Pearson K^2-statistic [4] which is based on the skewness and kurtosis.

2. Student's t-tests of equal means for different parts of the tile.

Student's t-tests are used since the normality test does not take positions of pixels into account. Only using the normality test could lead to situations where tiles with a near-linear slope are accepted.

Algorithm 1 `IsFlat`(T, α)

In: $w \times w$ tile T. w is even. Rejection rate α.
Out: True if T is flat. False otherwise.

1: $\alpha_1 \leftarrow 1 - (1 - \alpha)^{1/2}$
2: Perform the D'Agostino-Pearson K^2-test on the values of T with rejection rate α_1. Return false if rejected.
3: $(T_{1,1}, T_{1,2}, T_{2,1}, T_{2,2}) \leftarrow \frac{w}{2} \times \frac{w}{2}$ tiles partition of T.
4: $\alpha_2 \leftarrow 1 - (1 - \alpha)^{1/4}$
5: Perform a t-test of equal means on the pairs $(T_{1,1} \cup T_{1,2}, T_{2,1} \cup T_{2,2})$ and $(T_{1,1} \cup T_{2,1}, T_{1,2} \cup T_{2,2})$ using rejection rate α_2. Return false if the null hypothesis of equal means (and variances), in any of the two tests, is rejected.
6: Return true.

Inverses of CDFs are used to determine rejection boundaries in the tests. For example, the K^2-statistic has approximately the χ^2 distribution with 2 degrees of freedom. The CDF inverse in this case simplifies to $-2\log(1-p)$ which gives a boundary of $-2\log(\alpha_1)$.

A potential issue is that the statistics are not independent. There is a noticeable error in the actual rejection rate due to the χ^2 approximation, as seen in figure 5, however the simulated rejection rate for a 16×16 flat tile is still close to 0.05 and the error decreases for larger tiles.

The rejection rates are evenly divided between the K^2-test and the combined t-tests. Other ratios have not been tested.

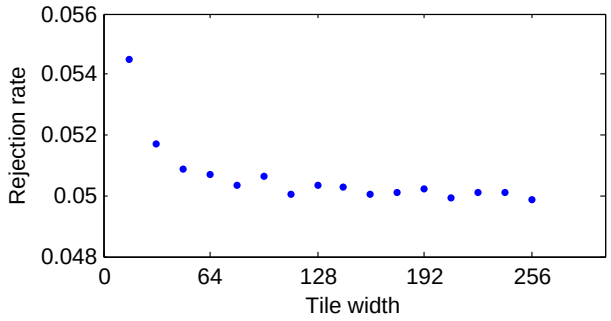

Figure 5: `IsFlat`$(T, 0.05)$. Rejection rate for square flat tiles based on 1 million simulations for each size.

Algorithm 2 `BGMeanAndVariance`(I, α, w_0)

In: Image I with at least one flat $w_0 \times w_0$ tile. Rejection rate α. Minimum tile width w_0.

Out: Background mean and variance estimates.

1: $w \leftarrow w_0$
2: **while** the $2w \times 2w$ tiles partition of I contains a tile T where `IsFlat`(T, α) is true **do**
3: $\quad w \leftarrow 2w$
4: **end while**
5: Calculate and return the mean and variance of the pixels in the $w \times w$ flat tiles.

Larger flat tiles are preferred to make detection of slopes due to objects easier. Some rows at the bottom of the image and columns at the right side of the image are ignored when the height and width are not divisible by w. Doubling w when searching for a tile size guarantees that the function runs in $\mathcal{O}(n \log n)$ time, with n the number of pixels. The noise variance is also returned because it's needed at later stages.

2.2 Value of α

Assuming the background estimate does not have a negative bias, settings that give a lower average background estimate are better, as the only bias in the background estimate is a positive bias from objects. The only possible cause of a negative bias are object-like fluctuations in the background. However, such fluctuations would not be moving with stars and galaxies and would appear as artifacts.

When α is too close to 1, the tile size decreases which results in more bias from objects. When α is too close to 0, more tiles are accepted which also results in more bias from objects. Figure 6 shows results for various settings of α.

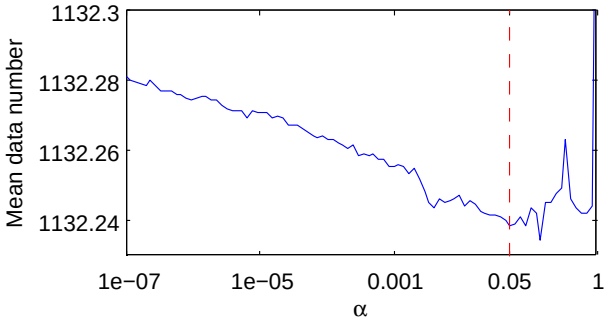

Figure 6: Average background estimate for the data set as a function of α.

Considering the standard deviation of the noise at the background is approximately 5.4 for most images, there is not much difference between $\alpha = 10^{-7}$, $\alpha = 0.05$ and $\alpha = 0.5$. Argument α is kept at 0.05.

2.3 Availability of flat tiles

All images in the data set have 64×64 flat tiles. The minimum tile width w_0 is set to 64.

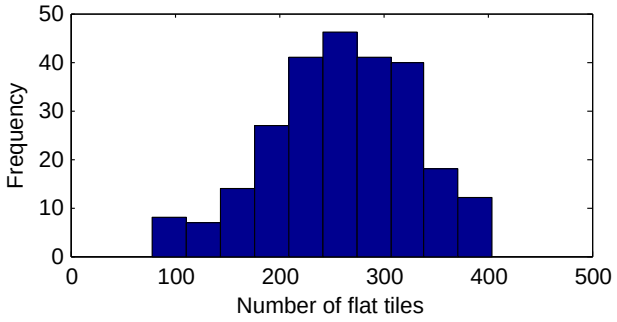

Figure 7: Histogram of the number of flat 64×64 tiles.

2.4 Is a constant a good fit?

An important question is whether a constant estimate of the background is the right choice for the data set.

Let μ_F be a statistic representing the mean of a flat tile, $\hat{\mu}_{\text{bg}}$ the background estimate (the hat indicates an estimate) and $\hat{\sigma}^2_{\text{bg}}$ the estimate of the noise variance at the background. If the background is flat, and the flat tiles are not biased by objects, $\mu_F \sim N(\mu_{\text{bg}}, w^{-1}\sigma_{\text{bg}})$, where w is the tile width, and let $\beta = (\hat{\mu}_{\text{bg}} - \mu_{F_i})\hat{\sigma}^{-1}_{\text{bg}}$. The distribution of β is approximately $N(0, w^{-1})$, with $\hat{\mu}_{\text{bg}}$ and $\hat{\sigma}_{\text{bg}}$ relatively constant. When $w = 64$, 95% of the absolute β values would be below 0.031 on average. In figure 8 this is clearly not the case. 95% of the absolute β values are below 0.14. If changes in the background are the main cause (the background is not flat), a different fit closer to the local estimates could be better. The variations in the local estimates are still small, considering most detected objects contain pixel values above 3 (times the noise standard deviation). Images with outliers are inspected to determine the cause.

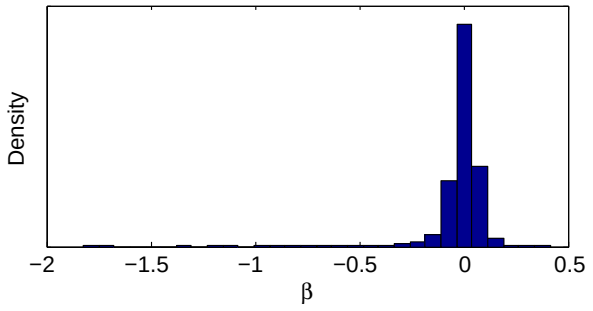

Figure 8: Distribution of β, for all images combined. Tile size is 64×64.

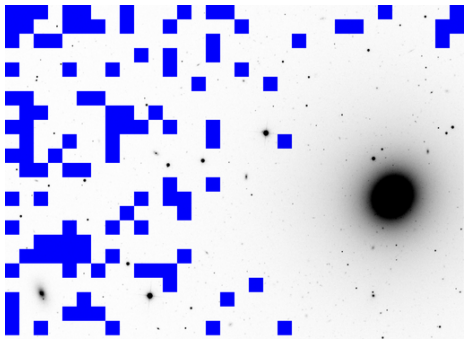

Figure 9: Image with β outliers. 64×64 flat tiles shown in blue. File `fpC-003836-r4-0249.fit`.

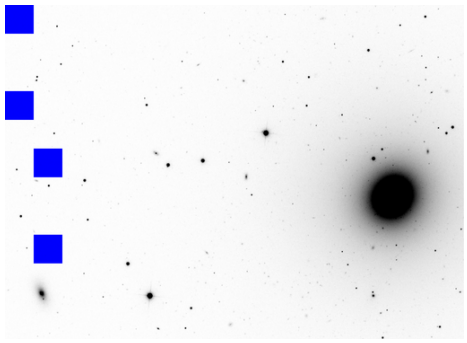

Figure 10: 128×128 flat tiles shown in blue. File `fpC-003836-r4-0249.fit`.

The background estimate in figure 9 is 1137.1 and 1135.9 in figure 10 ($\sigma_{\mathrm{bg}} \approx 5.4$), which shows the influence of the large object on the local estimates. Figures 11 and 12, which also have been picked to inspect β outliers, have a similar situation. The main cause of relatively large absolute values of β appears to be objects, not changes in the background. Attempting a non-constant background fit closer to the local estimates increases the error at locations correlating with objects.

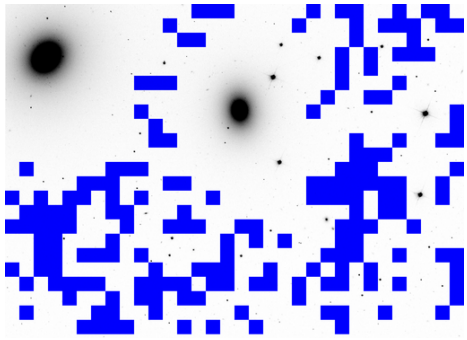

Figure 11: Image with β outliers. 64×64 flat tiles shown in blue. File `fpC-005313-r1-0067.fit`.

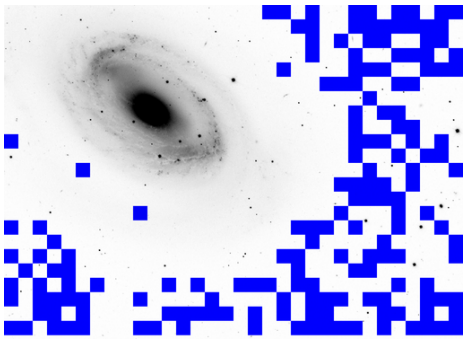

Figure 12: Image with β outliers. 64×64 flat tiles shown in blue. File `fpC-005116-r5-0148.fit`.

For the data set, and these local background estimates, a constant is a good fit.

2.5 Comparison with SExtractor

SExtractor uses bicubic interpolation between local background estimates found by iteratively clipping pixel values above $3\times$ the sample standard deviation and recalculating the sample mean. SExtractor uses 64×64 tiles by default.

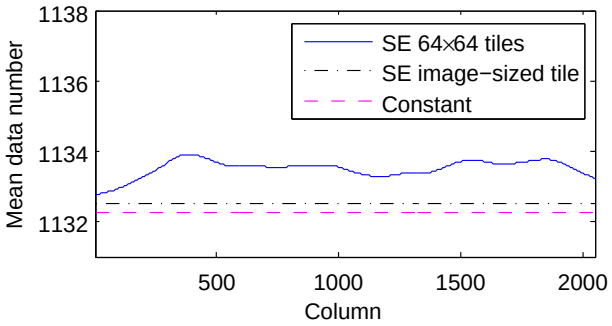

Figure 13: Average estimate of the background for all images, column wise.

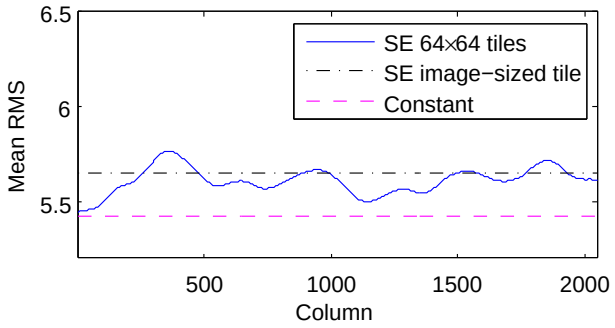

Figure 14: Average estimate of the noise standard deviation at the background for all images, column wise.

Figure 13 shows that the constant estimate suffers less from object bias on average. The background estimate by SExtractor is 1.27 higher on average which corresponds approximately to $0.23\sigma_{\mathrm{bg}}$, using $\sigma_{\mathrm{bg}} \approx 5.4$. An image-sized tile in SExtractor also reduces bias, on average, but the higher noise standard deviation, compared to the (other) constant, indicates a worse fit.

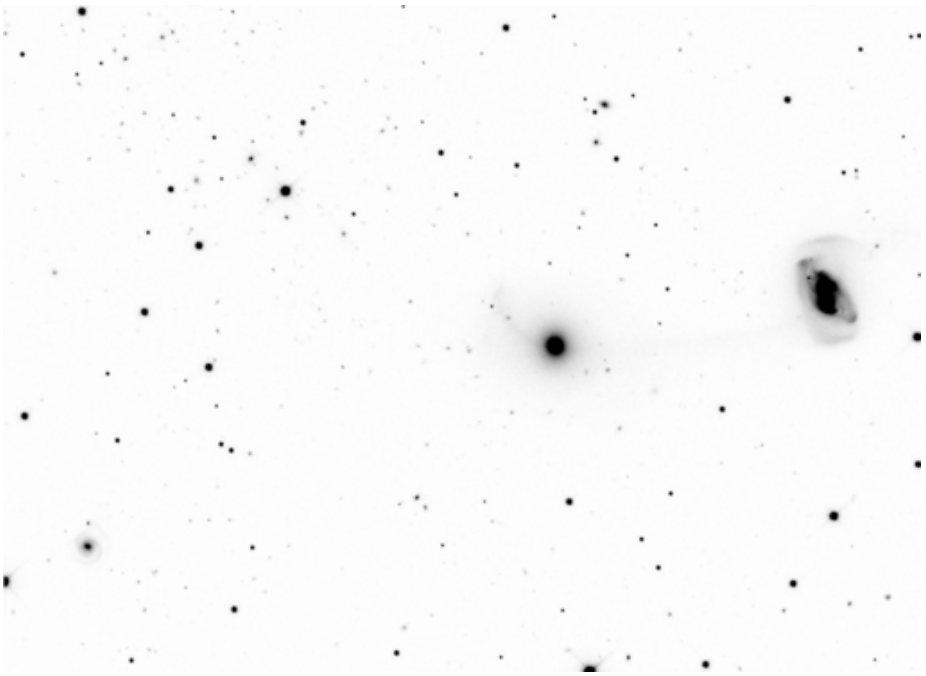

Figure 15: File `fpC-002078-r1-0157.fit`.

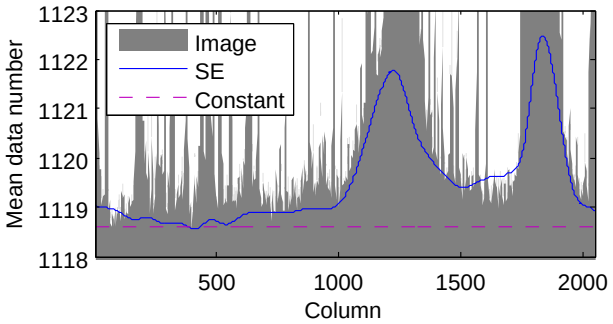

Figure 16: Background estimate by SExtractor compared to the constant estimate, averaged column wise. The image is shown in figure 15.

The background estimate by SExtractor correlates with objects (see figures 15 and 16), making it more difficult to detect (parts of) objects after subtraction. For example, the connection between the merging galaxies in figure 17 is preserved, while in figure 18 it is not.

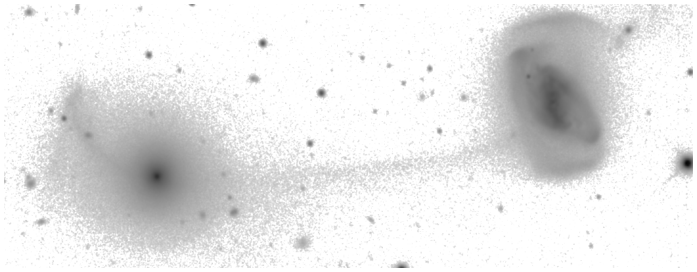

Figure 17: Constant background estimate, subtracted from the image. Smoothed, log scale and showing values above $.75\hat{\sigma}_{\mathrm{bg}}$. File `fpC-002078-r1-0157.fit` (cropped).

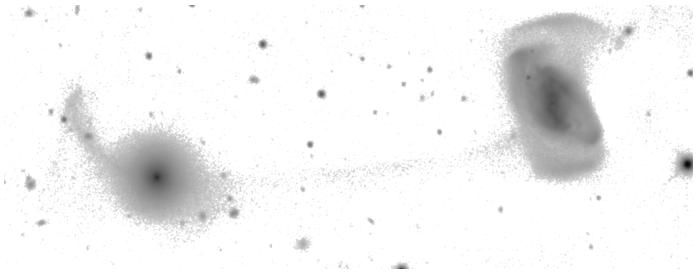

Figure 18: SExtractor background estimate, subtracted from the image. Smoothed, log scale and showing values above $.75\hat{\sigma}_{\mathrm{bg}}$. File `fpC-002078-r1-0157.fit` (cropped).

Another problem in the background estimation by SExtractor is distortion of object shapes, which appears to happen for every non-constant estimate. Figure 20 shows an example. For comparison, a constant estimate is used in figure 19.

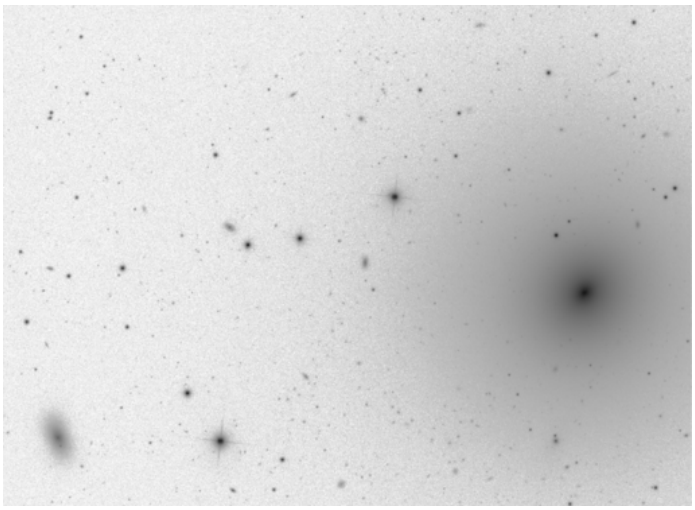

Figure 19: Constant background estimate, subtracted from the image. Log scale. File `fpC-003836-r4-0249.fit`.

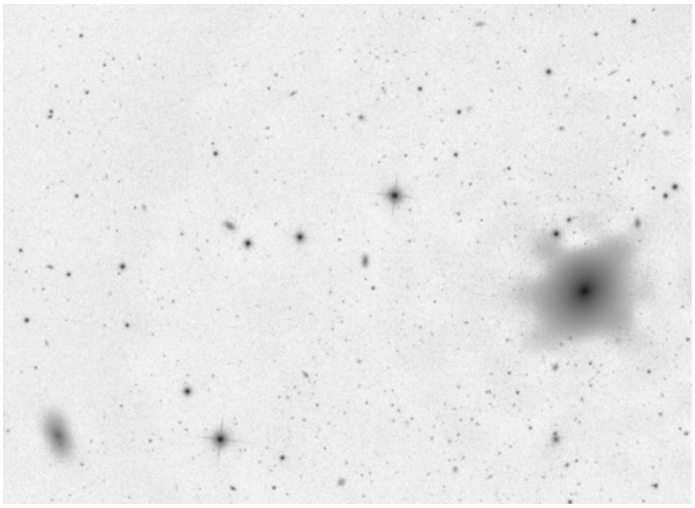

Figure 20: SExtractor background estimate, subtracted from the image. Log scale. File `fpC-003836-r4-0249.fit`.

With the goal of having the least object bias and preserving object shapes, using the constant background estimation is clearly better.

3 Object detection

With the background removed, the variance of the noise is $g^{-1}O + \sigma_{\text{bg}}^2$, where $\sigma_{\text{bg}}^2 = g^{-1}B + R \approx$ the variance given by `BGMeanAndVariance`. Negative image values are set to 0 and the Max-Tree is constructed. The next step is to identify nodes containing objects.

3.1 Significant nodes

Multiple significance tests are defined. The significance tests are designed to mark a node significant if one or more objects are represented in the pixels of the node, assuming a background value. To filter local maxima on top of objects due to noise, the assumed background can be higher than the global background. This allows nested objects to be found. After performance comparisons, one significance test is chosen to be included in the method.

Define P_{anc} as the closest significant ancestor or, if no such node exists, the root, for a node P in the Max-Tree. P_{anc} is the assumed background. P is significant if it can be shown that $\exists x \in P : O(x) > f(P_{\text{anc}})$, given significance level α. The result of $f(P)$ is the pixel value associated with P. Similarly, let $f(x)$ be the value of pixel x. The power [13] attribute of P is

$$\text{power}(P) \quad := \quad \sum_{x \in P} (f(x) - f(\text{parent}(P)))^2.$$

An alternative definition that will be used is

$$\text{powerAlt}(P) \quad := \quad \sum_{x \in P} (f(x) - f(P_{\text{anc}}))^2.$$

To normalize the power attributes, the values are divided by the noise variance.

3.1.1 Significance test 1: `power` given `area`

Let P be a random node. Define hypothesis $H_{\texttt{power}}$ as

$$H_{\texttt{power}} := (\forall x \in P : O(x) \leq f(\text{parent}(P))).$$

Assume $H_{\texttt{power}}$ is true and consider the most extreme case $\forall x \in P : O(x) = f(\text{parent}(P))$. The noise variance σ^2 equals $g^{-1} f(\text{parent}(P)) + \sigma^2_{\text{bg}}$. Statistic $(f(x) - f(\text{parent}(P)))^2 / \sigma^2$, for a random pixel x in P, has a χ^2 distribution with 1 degree of freedom. As the pixel values in P are independent, $\texttt{power}(P)/\sigma^2$ has a χ^2 distribution with $\texttt{area}(P)$ degrees of freedom. The χ^2 CDF (or inverse), commonly available in scientific libraries, can be used to test the normalized `power` given significance level α and `area`. An example of a rejection boundary is shown in figure 21. If $\texttt{power}(P)/\sigma^2 > \texttt{inverse}\chi^2\texttt{CDF}(\alpha, \texttt{area})$, $H_{\texttt{power}}$ is rejected: $O(x) > f(\text{parent}(P)) \geq f(P_{\text{anc}})$ for some pixels x in P, making P significant.

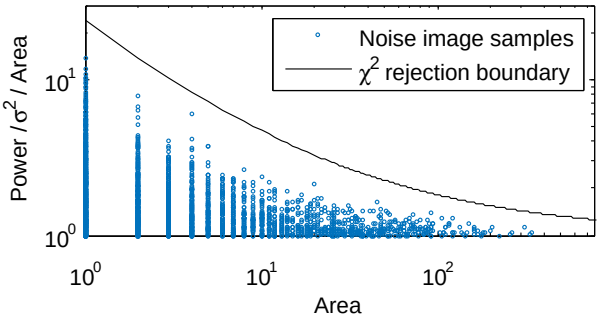

Figure 21: Rejection boundary for the χ^2 significance test given $\alpha = 10^{-6}$. Log-log scale.

3.1.2 Simulating distributions

In the next significance tests (all right-tailed) the exact distribution of `powerAlt` is not known and is simulated instead. The rejection boundary for an `attribute` and significance level α is the result of the inverse CDF, which will be denoted as `inverseAttributeCDF`(α). Let r be an integer greater than zero and $n_{\text{samples}}(\alpha, r) := r/\alpha$ with the condition that $n_{\text{samples}} \in \mathbb{N}$. n_{samples} random independent nodes are generated. On average, for r nodes the `attribute` value is greater than or equal to the rejection boundary. The best estimate of `inverseAttributeCDF`(α), without any further information about the distribution, is the average of the two smallest of the $r + 1$ largest `attribute` values.

3.1.3 Significance test 2: `powerAlt` given `area` and `distance`

Let P be a random node. Define hypothesis $H_{\texttt{powerAlt}}$ as

$$H_{\texttt{powerAlt}} := (\forall x \in P : O(x) \leq f(P_{\text{anc}})).$$

Assume $H_{\texttt{powerAlt}}$ is true and consider the most extreme case $\forall x \in P : O(x) = f(P_{\text{anc}})$. Define `distance`$(P)$ as $f(P) - f(P_{\text{anc}})$. Let X be a random set of `area`(P) - 1 values drawn from a truncated normal distribution with a minimum value of `distance`(P). The variance of the normal distribution is set to $\sigma^2 = g^{-1} f(P_{\text{anc}}) + \sigma^2_{\text{bg}}$. The distribution of `powerAlt`(P) is the same as `distance`$^2(P)$ plus the sum of the squared values in X. Let `inversePowerAltCDF`$(\alpha,$ `area`, `distance`$)$ be the function providing the rejection boundary for the normalized `powerAlt` attribute. If `powerAlt`$(P)/\sigma^2 >$ `inversePowerAltCDF`$(\alpha,$ `area`$, d))$, $H_{\texttt{powerAlt}}$ is rejected: $O(x) > f(P_{\text{anc}})$ for some pixels x in P, making P significant. Note that the minimum area of a significant node is 2.

Function `inversePowerAltCDF` is estimated for constant α, varying `area` and varying `distance`. Random samples are generated given the `area` and `distance`. $2 \leq$ `area` $\leq$ `maxArea`, with `maxArea` set to 768. Variable `distance` has a maximum of 4, referred to as `maxDistance`, and a step size of 0.25. For each rejection boundary given the `distance`, a rational function is fitted to reduce the error and required storage space, see figure 22.

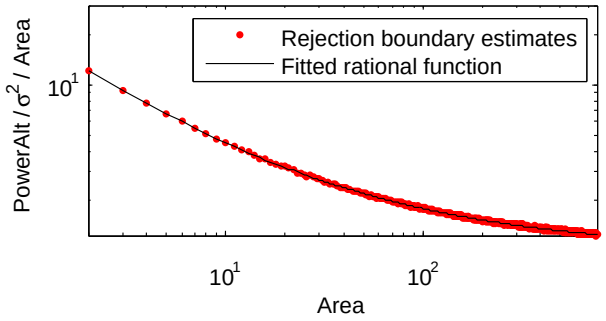

Figure 22: Simulated rejection boundary for `powerAlt` given `distance` $= 0$ and $\alpha = 10^{-6}$. The polynomials in the rational function are of degree 3. Variable $r = \lfloor \texttt{maxArea}/\texttt{area} \rfloor$. Log-log scale.

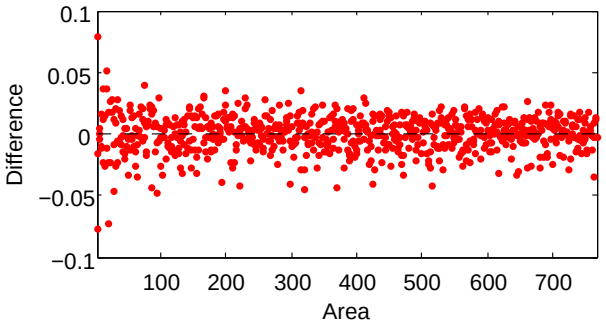

Figure 23: Difference between the rational function and estimates in figure 22. Root mean square = 0.015.

The rational function for `distance` $= 0$ appears to be a valid approximation, it is in the center of the estimates shown in figure 22. The error in this particular case is known since $\texttt{inversePowerAltCDF}(\alpha, \texttt{area}, 0)$ is equivalent to $\texttt{inverse}\chi^2\texttt{CDF}(\alpha, \texttt{area} - 1)$. Figure 24 shows the significance level error.

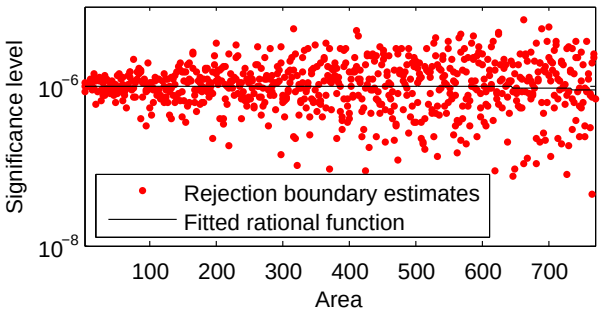

Figure 24: Actual significance level of the simulated rejection boundary in figure 22. Maximum absolute error in significance level for the fitted function is 1.04×10^{-7}. Log-linear scale.

Figure 25 shows all 17 rational functions with polynomials of degrees 3. Let `rms` be the root mean square of the differences between the rejection boundary estimates and rational function. The maximum value of `rms` is 0.019, comparable to the `rms` of the rejection boundary for `distance = 0`.

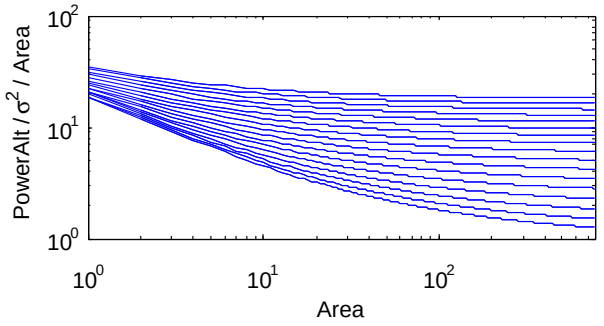

Figure 25: Approximated rejection boundaries for `powerAlt`. `distance` = 0 for the bottom curve and `distance` = 4 for the top curve. Step size is 0.25. Argument $\alpha = 10^{-6}$. Log-log scale.

When it is not possible for the expected values of the estimates to be the same as the rejection boundary, the choice is made to prefer overestimation, as it will not increase the number of false positives. Linear interpolation between rejection boundary values is used if one is not available for a `distance`, which happens in nearly all cases. The resulting value is slightly overestimated, see figure 26, as the actual function is superlinear.

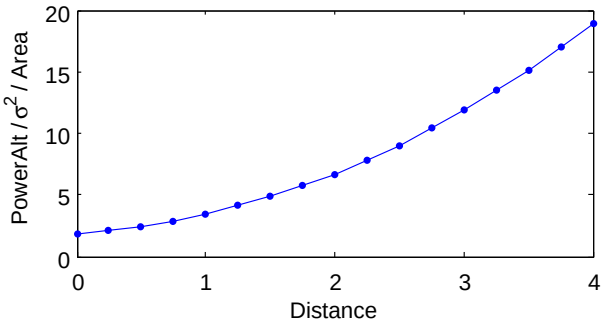

Figure 26: Approximated rejection boundary for `powerAlt`. Argument $\alpha = 0.01$ and `area` $= 100$.

If `area` $>$ `maxArea`, the significance test is changed to check if the normalized `powerAlt` attribute divided by the area is greater than `inversePowerAltCDF`$(\alpha, $`maxArea`$, $`distance`$)$ divided by `maxArea`. Since `inversePowerAltCDF` divided by the `area` decreases when the `area` increases (see figure 25), the boundary value is overestimated.

When the normalized `distance` is too large, it is set to the maximum available. A virtual significant ancestor with pixel value f_{anc} is used. To calculate the normalized `powerAlt` attribute, f_{anc} and σ^2 need to be found first. Since $f_{\text{anc}} + \sigma \cdot$ `maxDistance` $= f(P)$ and $\sigma^2 = g^{-1} f_{\text{anc}} + \sigma_{\text{bg}}^2$, solving for f_{anc} gives

$$f_{\text{anc}} = \frac{2g \cdot f(P) + m^2 - m\sqrt{4\sigma_{\text{bg}}^2 g^2 + 4g \cdot f(P) + m^2}}{2g}, \text{ with}$$
$$m := \texttt{maxDistance}.$$

Again, the boundary value is overestimated, which does not result in more false positives.

3.1.4 Significance test 3: `powerAlt` given `area`

A significance test using the distribution of `powerAlt` given α and `area`. The `distance` is not used as parameter in the inverse CDF. Using the assumptions in significance test 2, `distance`(P) has a truncated normal distribution with a minimum value of 0, the same distribution as a random non-negative pixel value. As a border case, `distance`$(P) = 0$ is excluded. To generate samples, neighboring pixels are simulated to determine `area`(P) and `powerAlt`(P) in addition to `distance`(P). When not enough samples are available for an `area`, samples from neighboring larger `area`s are merged until enough samples are available. The average `area` is used as the area being represented. The fitting weight of the sample is set to the number of merged `area`s, to avoid making the samples for larger `area`s less relevant. Figure 27 shows the rejection boundary and figure 28 shows the difference between the rational function approximation and estimates. Table 1 shows the coefficients of the rational function. In all cases 4-connectivity is used. A different connectivity possibly changes the rejection boundary.

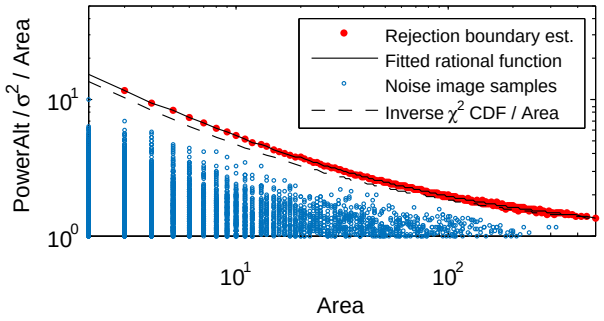

Figure 27: Simulated rejection boundary for `powerAlt`. Argument $\alpha = 10^{-6}$. Log-log scale.

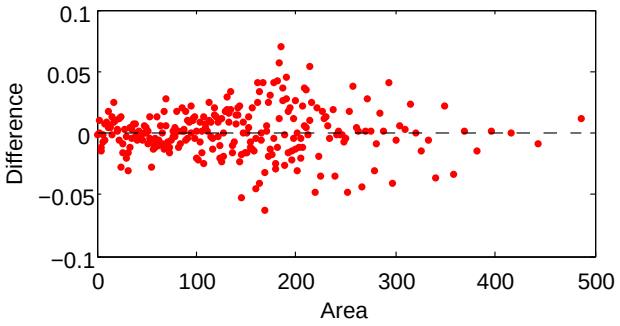

Figure 28: Difference between the rational function and estimates in figure 27.

p_1	=	1.114418454778754e+00
p_2	=	3.210124510318325e+02
p_3	=	1.031189113590674e+04
p_4	=	3.393269502056448e+04
q_1	=	1.635566553428912e+02
q_2	=	1.388490895121575e+03
q_3	=	1.775488921810826e+02

Table 1: Coefficients of the rational function approximating the rejection boundary for `powerAlt`.

3.1.5 Significance test 4: `powerAlt` given `area`, using a smoothing filter

SExtractor uses filtering to reduce noise and to detect more objects. To get similar effects, a smoothing filter is used in test 4, which otherwise is the same

as test 3. The default SE smoothing kernel is used:

$$H = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

Filtering is done after background subtraction and before setting negative values to zero. Figure 30 shows the rejection boundary and figure 29 shows the difference between the rational function approximation and estimates. Table 2 shows the coefficients of the rational function.

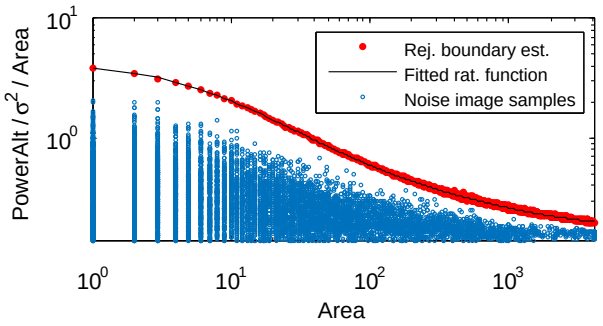

Figure 29: Simulated rejection boundary for `powerAlt` using the default smoothing filter. Argument $\alpha = 10^{-6}$. Log-log scale.

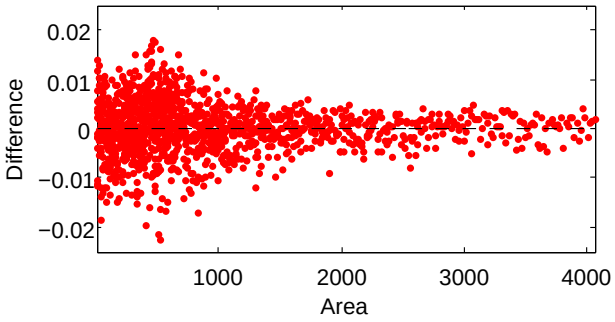

Figure 30: Difference between the rational function and estimates in figure 29.

p_1	=	1.683355084690155e-01
p_2	=	3.770229379757511e+02
p_3	=	1.176722049258011e+05
p_4	=	6.239836661965291e+06
q_1	=	1.354265276841128e+03
q_2	=	2.091126298053044e+05
q_3	=	1.424803575269314e+06

Table 2: Coefficients of the rational function approximating the rejection boundary for `powerAlt` when using the default smoothing filter.

3.1.6 Testing nodes

Algorithm 3 describes the used method for marking nodes as significant. Visiting nodes in non-decreasing order by pixel value simplifies the determination of $P_{\texttt{anc}}$, if stored for every node. There is no need if the χ^2 test is used. Function $\texttt{nodeTest}(M, P, \alpha, g, \sigma^2_{\text{bg}})$ performs the significance test and returns true if P is significant and false otherwise.

Algorithm 3 $\texttt{SignificantNodes}(M, \texttt{nodeTest}, \alpha, g, \sigma^2_{\text{bg}})$

In: Max-Tree M, significance test t, significance level α, gain g, variance of the noise at the background σ^2_B.
Out: Nodes in M that are unlikely to be noise are marked as significant.

```
for all nodes P in M with f(P) > 0 in non-decreasing order do
    if nodeTest(M, P, α, g, σ²_bg) is true then
        Mark P as significant.
    end if
end for
```

3.1.7 Value of α

The Max-Tree of a noise image after subtraction of the mean and truncation of negative values is expected to have $0.5n$ nodes, with n the number of pixels. An upper bound on the number of expected false positives is $\alpha 0.5n$ if the nodes are independent, which is not entirely the case. Given a 1489×2048 noise image, the same size of the images in the data set, and $\alpha = 10^{-6}$, the upper bound on the expected number of false positives is approximately 1.52. An estimate of the actual number of false positives for the used significance tests is shown in table 3. Argument α is set to 10^{-6} by default.

Significance test ↓	False positives
`power` given `area`	0.41
`powerAlt` given `area` and `distance`	0.72
`powerAlt` given `area`	0.94
`powerAlt` given `area`, smoothed	0.35

Table 3: Average number of false positives for 1000 simulated noise images with a size of 1489×2048. Argument $\alpha = 10^{-6}$.

3.2 Finding objects

Multiple significant nodes could be part of the same object. A significant node with no significant ancestor is marked as an object. Let `mainBranch`(P) be a significant descendant of P with the largest area. A significant node, with significant ancestor P_{anc}, that does not equal `mainBranch`(P_{anc}) is marked as a new object. The assumption of a new object is not valid in every case, it depends on the used significance test, filter and connectivity, as will be shown in the comparison section. The method is described in algorithm 4.

Algorithm 4 `FindObjects`(M)

In: Max-Tree M.
Out: Nodes in M that represent an object are marked.

```
for all significant nodes P in M do
  if P has no significant ancestor then
    Mark P as object.
  else
    if mainBranch(P_anc) does not equal P then
      Mark P as object.
    end if
  end if
end for
```

3.3 Moving object markers up

Nodes marked as objects have a number of pixels attached due to noise. The number decreases at a further distance from the noiseless image signal, which can be achieved by moving the object marker up in the tree. The first choice for an object node P is `mainBranch`(P), if such a node exists, as it does not conflict with other object markers. The second choice is the descendant of P with the highest p-value found with the relevant CDF for the `power` or `powerAlt` attribute. However, the CDF is not always available or easy to store. Instead, the descendant with the largest `power` attribute is chosen, which is one of the children, if at least one exists. The function that provides the descendant will be called `mainPowerBranch`(P). Algorithm 5 describes the method. An alternative to allowing a lower value of $f(P_{\text{final}})$ is to remove those object markers.

Algorithm 5 `MoveUp`$(M, \lambda, g, \sigma^2_{\text{bg}})$

In: Max-Tree M, factor λ, gain g, variance of the noise at the background σ^2_{bg}.
Out: For every object marker that started in a node P and ended in P_{final}: $f(P_{\text{final}}) \geq f(P_{\texttt{ancestor}}) + \lambda$ times the local standard deviation of the noise, when possible. $f(P_{\text{final}})$ might be lower if P_{final} has no descendants.

for all nodes P in M marked as objects **do**
  Remove the object marker from P.
  $h \leftarrow f(P_{\texttt{ancestor}}) + \lambda\sqrt{g^{-1}f(P_{\texttt{ancestor}} + \sigma^2_{\text{bg}}}$
  while $f(P) < h$ **do**
    if P has a significant descendant **then**
      $P \leftarrow$ `mainBranch`(P).
    else if P has a descendant **then**
      $P \leftarrow$ `mainPowerBranch`(P).
    else
      Break.
    end if
  end while
  Mark P as object.
end for

3.3.1 Value of λ

To perform quantitative tests, we need images with a known ground truth. IRAF is a program used to simulate stars. The relevant command is `mkobjects`. The magnitude is set to -5. Gaussian noise is added using the pixel value as mean and variance.

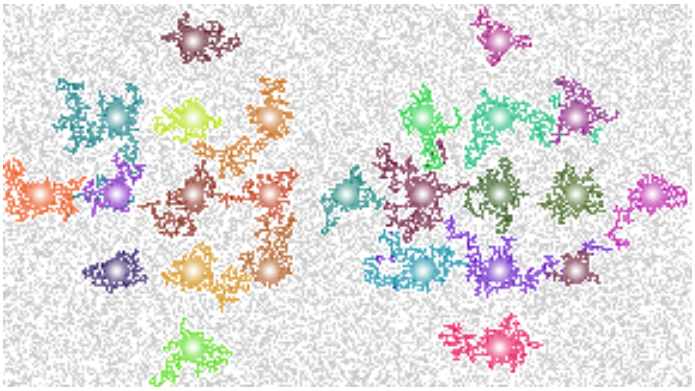

Figure 31: 25 simulated stars. Argument $\lambda = 0$. Log scale.

Argument λ is set too low in figure 31, there are too many noise pixels attached to objects.

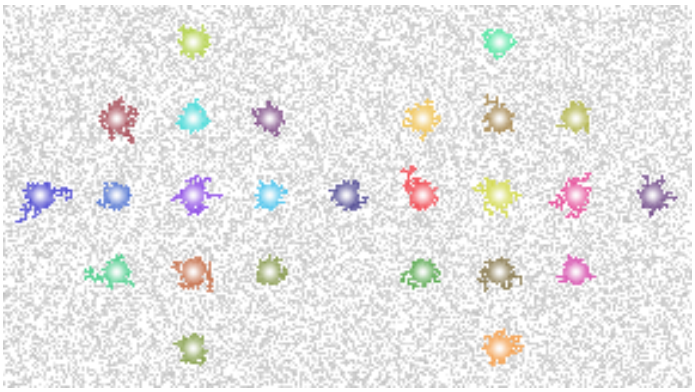

Figure 32: 25 simulated stars. Argument $\lambda = 0.5$. Log scale.

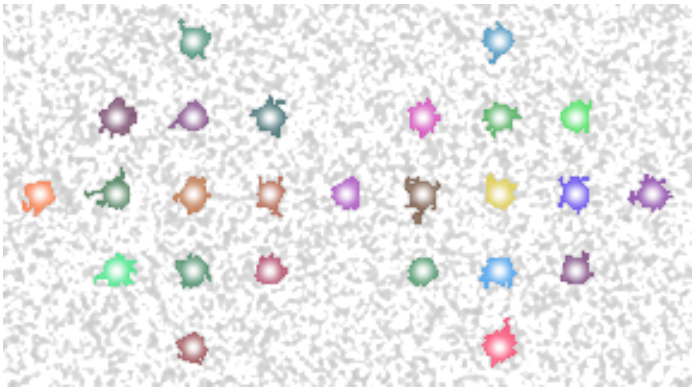

Figure 33: 25 simulated stars. Argument $\lambda = 0.5$. Image is smoothed using the default SExtractor filter. Log scale.

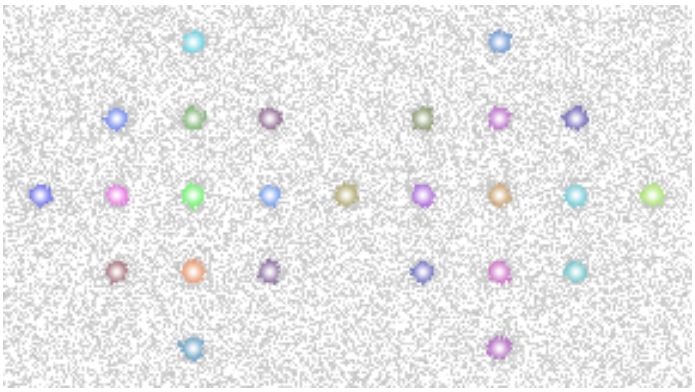

Figure 34: 25 simulated stars. Argument $\lambda = 2$. Log scale.

The object shapes in figure 34 look better. However, to be able to display faint parts of extended sources, a low value of λ is preferred. Setting λ to 0.5 is done as compromise.

3.4 MTObjects

Algorithm 6 (MTObjects) combines the methods in a single procedure.

Algorithm 6 `MTObjects`$(I, \texttt{nodeTest}, \alpha, g, \lambda)$

In: Image I, function `nodeTest`, significance level α, gain g and move factor λ.
Out: Max-Tree M. Nodes in M corresponding to objects are marked.

1: $(\hat{\mu}_{\text{bg}}, \hat{\sigma}^2_{\text{bg}}) \leftarrow$ `BGMeanAndVariance`$(I, 0.05, 64)$
2: $I_{i,j} \leftarrow \max(I_{i,j} - \hat{\mu}_{\text{bg}}, 0)$
3: $M \leftarrow$ create a Max-Tree representation of I.
4: `SignificantNodes`$(M, \texttt{nodeTest}, \alpha_{\text{nodes}}, g, \hat{\sigma}^2_{\text{bg}})$.
5: `FindObjects`(M).
6: `MoveUp`$(M, \lambda, g, \hat{\sigma}^2_{\text{bg}})$.

When g is not given, it is approximated by $\hat{\mu}_{\text{bg}}/\hat{\sigma}^2_{\text{bg}}$. The assumption is that R (variance of other noise) is small relative to $g^{-1}\hat{\mu}_{\text{bg}}$.

4 Comparison of methods

A comparison is made using SExtractor 2.19.5 as a reference. Settings are kept close to default. The constant `NSONMAX` in `refine.c` is increased to avoid overflows. We also used defaults previously [10], these are improvements found experimentally.

4.1 SExtractor settings

- Manual background and noise root mean square estimates using the values provided by `BGMeanAndVariance`. Program arguments: `-BACK_TYPE MANUAL -BACK_VALUE` <float> `-WEIGHT_TYPE MAP_RMS -WEIGHT_IMAGE` <file>.
- `DETECT_MINAREA = 3`. Default in SExtractor 2.19.5.
- `FILTER_NAME = default.conv`.
- `DETECT_THRESH = 1.575`. The default threshold of 1.5 (times the noise standard deviation) is changed to make the expected false positives similar to significance test 4 for noise-only images. Expected false positives per image is approximately 0.38 based on the results of 1000 simulated noise images, taking 8-connectivity into account.
- `MEMORY_PIXSTACK = 4000000`. To avoid overflows. The value is larger than the number of pixels in the image.

While there is no guarantee that these settings are optimal, the initial comparison gives a first impression of our method. To show exactly the strengths and weaknesses compared to SExtractor a more comprehensive comparison is required.

4.2 Object fragmentation

A potential source of false positives is fragmentation of objects due to noise. An example is shown in figure 35. Fragmentation appears to happen in relatively flat structures and the chance is increased if different parts of the structure are thinly connected. If one pixel connects two parts, the variation in value due to noise can make a deep cut. In the case of a threshold used by SExtractor, fragmentation is severe if the object values are just below the threshold.

Figure 35: Fragmentation of an 'object'; MTObjects with significance test 3 (left), test 4 (middle) and SExtractor (right). The 'object' pixels have the value 1.5, close to the SExtractor threshold, and the background has the value 0. Gaussian noise is added with $\sigma = 1$.

The expected number of false positives due to fragmentation for the given data set is unknown. Most images do not clearly show fragmented objects. One image where it does happen when using a threshold is displayed in figure 36.

Fragments detected as separate objects are called spurious detections in SExtractor. While the SExtractor parameter `CLEAN_PARAM` can be changed to prevent this from happening, it is left to the default as it has a negative effect on the number of objects detected and only one image clearly shows fragmentation.

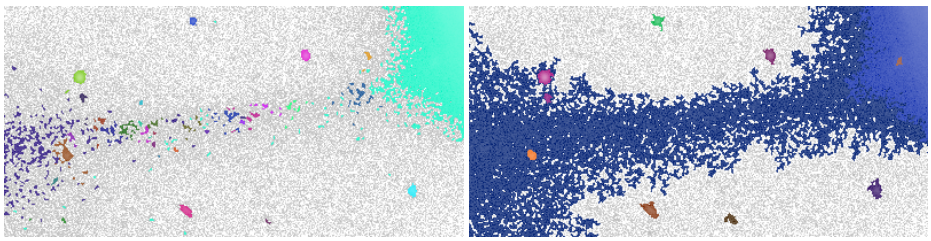

Figure 36: Fragmentation of an object in the data set by SExtractor (left) and the result of MTObjects with test 4 (right). Log scale. File `fpC-002078-r1-0157.fits` (cropped).

4.3 Object count

An object is defined as a bump in the image signal that cannot be explained by noise. Table 4 shows a comparison between the methods using the entire data set. The percentages do not add up to 100% in some cases because some images have an equal number of detected objects. Using significance test 4 gives the best results.

Test A \ Test B	1	2	3	4	SE
1		65.7	0.0	1.6	13.0
2	32.3		0.0	0.0	10.6
3	100.0	100.0		26.8	50.0
4	98.4	100.0	71.3		70.9
SE	87.0	89.0	49.2	28.0	

Table 4: Percentage of images where test A has more object detections than test B.

When inspecting the results, MTObjects detects more objects nested in larger objects, i.e. galaxies, when the pixel values of the nested objects are above SExtractor's threshold. Examples are shown in Figures 37 and 38. An explanation is that every node in the Max-Tree is used while SExtractor uses a fixed amount of sub-thresholds. A question is whether the better detection of nested objects explains the performance of significance test 4 compared to SExtractor.

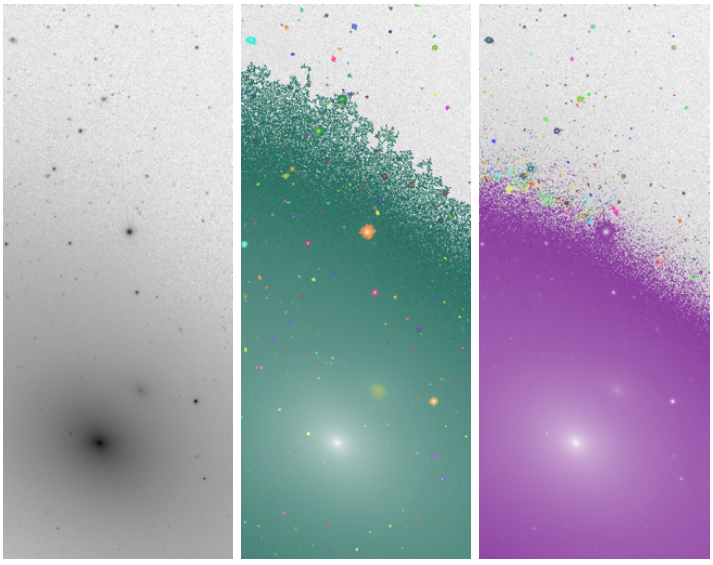

Figure 37: Original image (left), MTObjects with significance test 4 (middle) and SExtractor (right). Log scale. File `fpC-003804-r5-0192.fits` (cropped).

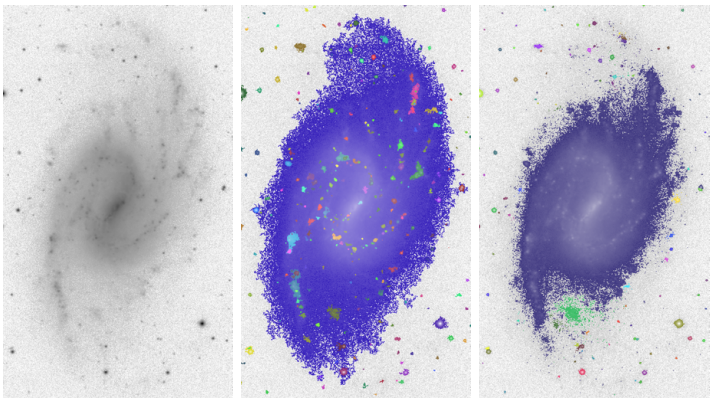

Figure 38: Original image (left), MTObjects with significance test 4 (middle) and SExtractor (right). Log scale. File `fpC-001332-r4-0066.fits` (cropped).

To answer the question, the data set is limited to more compact objects. This is done by making a sorted area list of the largest connected component of each image at the threshold used by SExtractor. The largest object is shown in in figure 39. Table 5 shows the results of the new comparison.

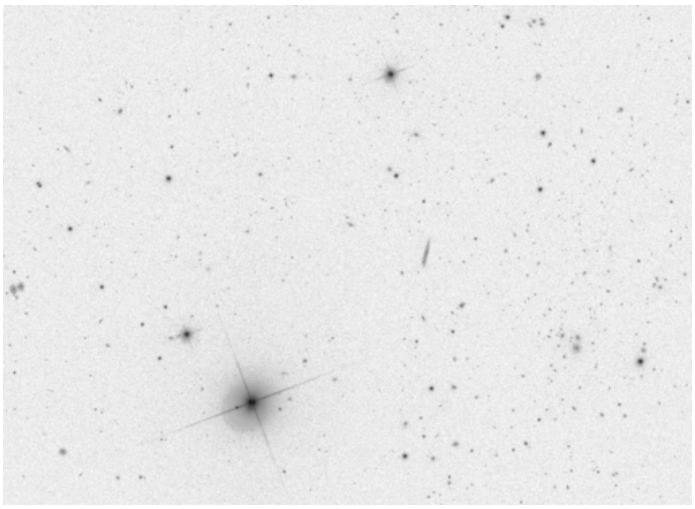

Figure 39: The largest object in the limited data set is in this image. Log scale.

Test A \ Test B	1	2	3	4	SE
1		49	0	0	0
2	47		0	0	0
3	100	100		6	6
4	100	100	94		49
SE	100	100	93	50	

Table 5: Percentage of images where test A has more object detections than test B.

The performance of significance test 4 and SExtractor is similar, meaning the difference in the previous comparison is explained by the number of nested object detections.

A question is whether significance test 4 performs as good as SExtractor in other data sets. When two identical objects overlap, one of the nodes marked as object has a lower `power` or `powerAlt` value on average. If overlapping objects are close enough to each other and at SExtractor's threshold they are still detected as separate objects, MTObjects could fail to detect separate objects.

To explore the situation of densely spaced overlapping objects, stars are simulated in a grid with the IRAF command `mkobjects` (figure 40).

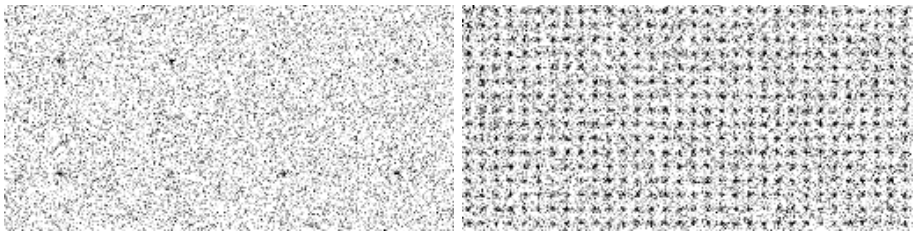

Figure 40: Part of the sparse stars grid (left) and part of the dense star grid (right).

The magnitude is set to -0.2 to make objects barely detectable when noise is added. The diameter of objects is 3 pixels (full width at half maximum). The background equals 1000 at every pixel and the gain is 1. Gaussian noise is added with the pixel value as mean and variance. Table 6 shows the number of object detections for the sparse grid and table 7 for the dense grid.

Significance test ↓	# objects detected
1: `power` given `area`	67
2: `powerAlt` given `area` and `distance`	161
3: `powerAlt` given `area`	343
4: `powerAlt` given `area`, smoothed	607
SE	540

Table 6: Sparse grid. Number of objects detected for each significance test and SExtractor. Total number of objects in the image is 1024. Size is 2048×2048.

Significance test ↓	# objects detected
1: `power` given `area`	161
2: `powerAlt` given `area` and `distance`	184
3: `powerAlt` given `area`	243
4: `powerAlt` given `area`, smoothed	268
SE	524 components, 433 extracted

Table 7: Dense grid. Number of objects detected for each significance test and SExtractor. Total number of objects in the image is 1024. Size is 256×256.

The results confirm that a threshold is better when objects are densely enough spaced. As an example of real objects, figure 41 shows a globular cluster. MTObjects using significance test 4 detects 3035 objects and SExtractor 3164.

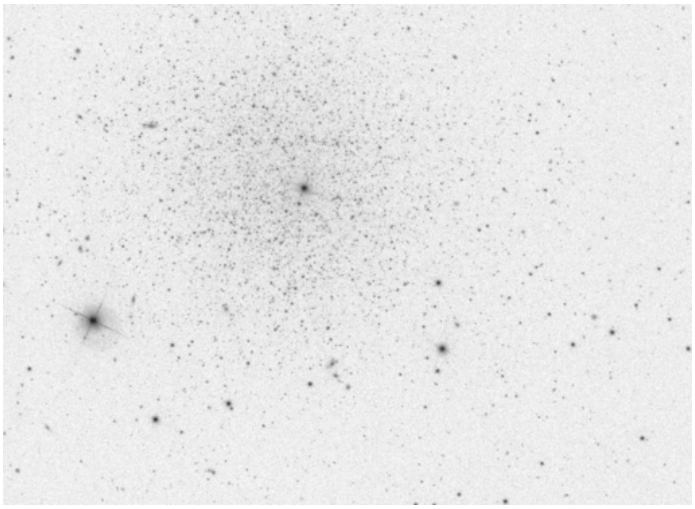

Figure 41: Image of a globular cluster not part of the original data set.

4.4 Faint structures

Effectively the threshold of 1.575 used in SExtractor is lowered to 0.5 (λ) for the object without adding false positives, which makes detecting fainter structures easier. λ can be lowered further but the side effect is more noise pixels attached to the object. Figures 42-45 show examples. Object deblending by SExtractor does not always work well (figures 43, 46 and 47).

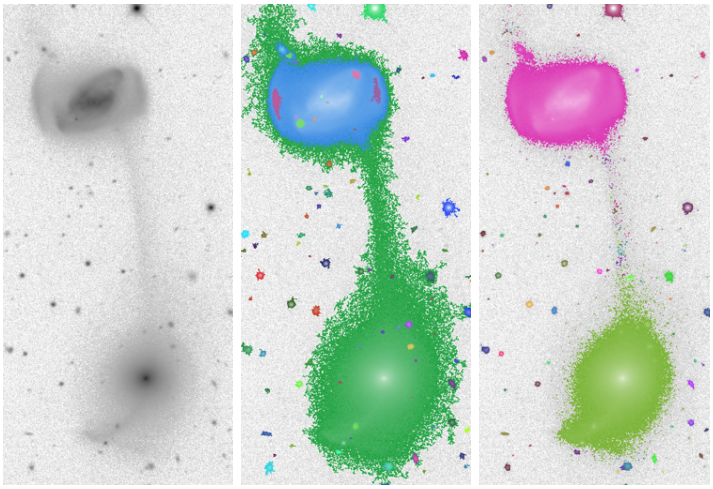

Figure 42: Original image (left), MTObjects with significance test 4 (middle) and SExtractor (right). Log scale. File `fpC-002078-r1-0157.fits` (cropped).

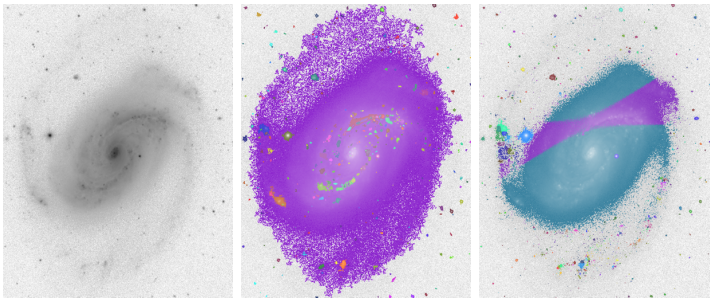

Figure 43: Original image (left), MTObjects with significance test 4 (middle) and SExtractor (right). Log scale. File `fpC-003903-r2-0154.fits` (cropped).

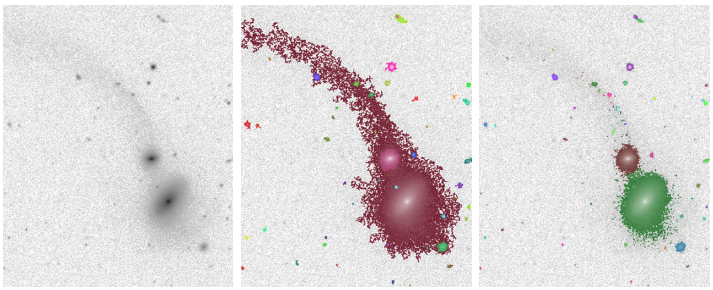

Figure 44: Original image (left), MTObjects with significance test 4 (middle) and SExtractor (right). Log scale. File `fpC-004576-r2-0245.fits` (cropped).

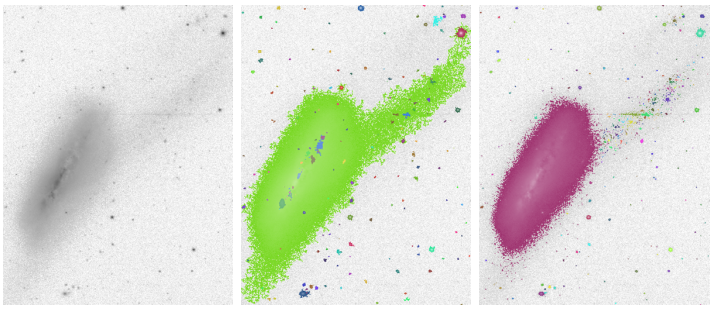

Figure 45: Original image (left), MTObjects with significance test 4 (middle) and SExtractor (right). Log scale. File `fpC-005224-r6-0255.fits` (cropped).

4.5 Dust and artifacts

The last possible sources of false positives are dust and artifacts. Examples of dust are shown in figures 46-47 and artifacts in figures 48-50. In figure 47 the galaxy core is split due to dust. The question in figure 48 is whether the 'object' is the artifact or the vertical cut-off. Refraction spikes shown in figure 50 can also cause false positives because of the wave-like shape, although not clearly visible.

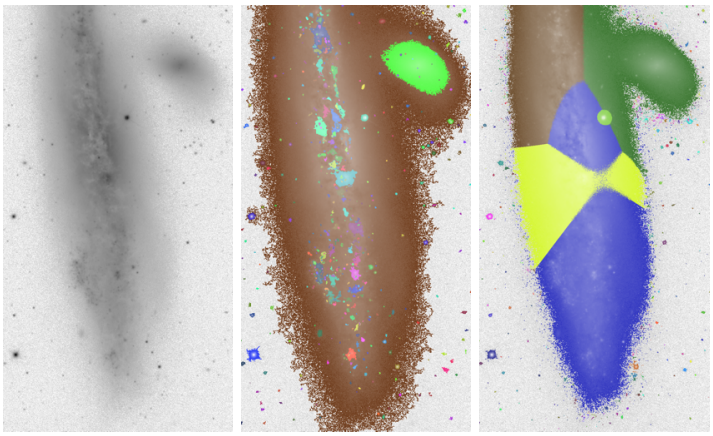

Figure 46: Dust. Original image (left), MTObjects with significance test 4 (middle) and SExtractor (right). Log scale. File `fpC-004623-r4-0202.fits` (cropped).

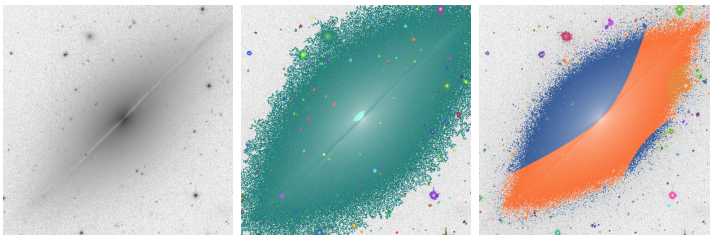

Figure 47: Dust. Original image (left), MTObjects with significance test 4 (middle) and SExtractor (right). Log scale. File `fpC-001739-r6-0308.fits` (cropped).

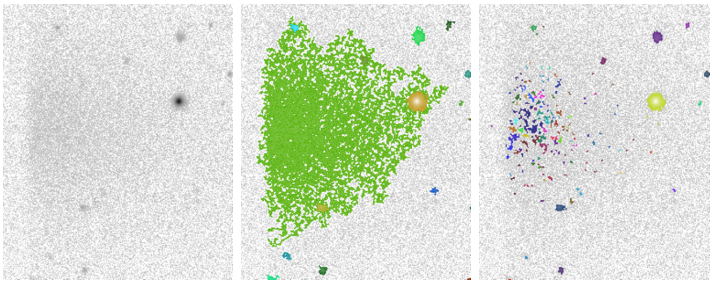

Figure 48: Artifact. Original image (left), MTObjects with significance test 4 (middle) and SExtractor (right). Log scale. File `fpC-002326-r4-0174.fits` (cropped).

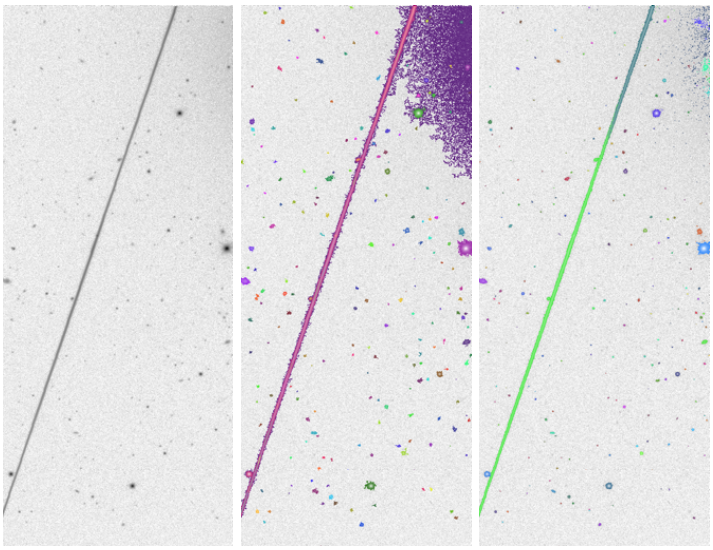

Figure 49: Artifact. Original image (left), MTObjects with significance test 4 (middle) and SExtractor (right). Log scale. File `fpC-003538-r6-0294.fits` (cropped).

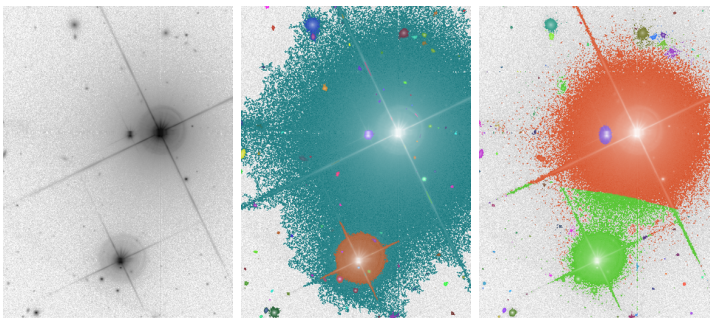

Figure 50: Artifact(s). Original image (left), MTObjects with significance test 4 (middle) and SExtractor (right). Log scale. File `fpC-005224-r6-0255.fits` (cropped).

4.6 Run times

The timer is started before background estimation and stopped after object classification in SExtractor and after `MoveUp` in MTObjects. How much time is spent on classification in SExtractor is unknown. Classification is not done in MTObjects. Tests were performed on an Intel Core i5-4460 with a single thread. SExtractor is typically 2.5 times faster than MTObjects (figure 51) using the median run time. When using the mean run time, SExtractor is 1.3 times faster. SExtractor takes longer for images that have many pixel values above the primary threshold. MTObjects is more constant in run time.

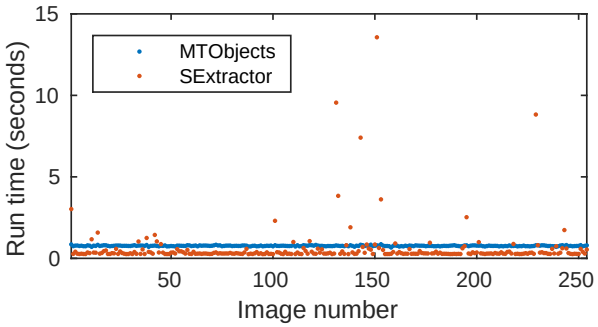

Figure 51: Comparing run times between MTObjects and SExtractor.

5 Conclusion

New methods for background subtraction and object extraction have been developed and compared to SExtractor. While the methods look promising, a more comprehensive analysis with more data sets and different SExtractor settings is required to find out more precisely which method is better in what situation and by how much.

The background estimation in the new method is less biased by objects compared to SExtractor, for the given data set with relatively flat backgrounds. On average the difference is 0.23 times the noise standard deviation.

The Max-Tree method (MTObjects) is better at extracting faint parts of objects compared to a global threshold used in current methods. The sensitivity increases with object size. MTobjects is slightly worse in the case of a globular cluster where compact objects are densely spaced and overlapping.

When defining an object as having a single maximum pixel value, not including maxima due to noise, MTObjects is better at finding nested objects. Every possible threshold is tested compared to the fixed number of thresholds in SExtractor.

Deblending objects appears to be better in MTObjects when there is a large difference in size and objects don't have a Gaussian profile. When objects are similar in size to the viewer and have a Gaussian profile, one of the objects will be seen as a smaller branch by MTObjects. SExtractor's method of fitting Gaussian profiles makes more sense in this case for an even split in pixels.

6 Future work

Many improvements to the current work are possible, which we discuss shortly in the next sections. The first section relates to background estimation and the second section to object extraction.

6.1 Background estimation

Allowing different shapes (instead of squares) of areas representing the background could give better local estimates. One possible way is to start with tiny

square flat tiles and iteratively merge similar sized areas if they pass a flatness test.

If the data set requires a non-constant background estimate, what would be the best approach? Using weighted (by range) k-nearest neighbours looks promising, it works if only a few local background estimates are available and, depending on k, is less affected by outliers than bilinear or bicubic interpolation.

6.2 Object extraction

The power attribute was initially used as convenience because in the non-filtered case it has a known (worst case) distribution: scaled χ^2. The question is whether better attribute choices exist which use more information.

Deblending similar sized objects can be improved. Currently too many pixels are assigned to one of the objects.

Nested significant connected components can represent the same object. The current choice, controlled by λ in `MoveUp`, between the components is not ideal. The threshold is still set too high for large objects and too low for compact objects. To solve, λ should be variable and depend on the used filter, connectivity and attributes of the starting node.

If other noise models are used in different data sets, new significance tests should be created. Work is in progress in [6].

Which (smoothing) filter maximizes the number of objects detected while avoiding fragmentation of objects?

The addition of object classification is ongoing.

Currently the rejection boundary is approximated by simulations which need to be redone for every filter and significance level. Knowing the exact distributions could speed up the process.

References

[1] SDSS website. Accessed June 2015. URL: `http://www.sdss2.org/dr7/algorithms/fluxcal.html`.

[2] Kevork N. Abazajian et al. "The seventh data release of the Sloan Digital Sky Survey". In: *The Astrophysical Journal Supplement Series* 182.2 (2009), p. 543.

[3] Emmanuel Bertin and S. Arnouts. "SExtractor: Software for source extraction." In: *Astronomy and Astrophysics Supplement Series* 117 (1996), pp. 393–404.

[4] Ralph B. D'Agostino, Albert Belanger, and Ralph B. D'Agostino Jr. "A suggestion for using powerful and informative tests of normality". In: *The American Statistician* 44.4 (1990), pp. 316–321.

[5] J. E. Gunn et al. "The sloan digital sky survey photometric camera". In: *The Astronomical Journal* 116.6 (1998), p. 3040.

[6] Ugo Moschini et al. "Towards better segmentation of large floating point 3D astronomical data sets: first results". In: *Proceedings of the 2014 conference on Big Data from Space BiDS14*. Publications Office of the European Union, 2014, pp. 232–235.

[7] Philippe Salembier, Albert Oliveras, and Luis Garrido. “Antiextensive connected operators for image and sequence processing”. In: *Image Processing, IEEE Transactions on* 7.4 (1998), pp. 555–570.

[8] Mehmet Sezgin et al. “Survey over image thresholding techniques and quantitative performance evaluation”. In: *Journal of Electronic imaging* 13.1 (2004), pp. 146–168.

[9] Chris Stoughton et al. “Sloan digital sky survey: early data release”. In: *The Astronomical Journal* 123.1 (2002), p. 485.

[10] Paul Teeninga et al. “Bi-variate statistical attribute filtering: A tool for robust detection of faint objects”. In: *11th International Conference "Pattern Recognition and Image Analysis*. 2013, pp. 746–749.

[11] Michael H. F. Wilkinson. “A fast component-tree algorithm for high dynamic-range images and second generation connectivity”. In: *Image Processing (ICIP), 2011 18th IEEE International Conference on*. IEEE. 2011, pp. 1021–1024.

[12] Michael H. F. Wilkinson. “Gaussian-weighted moving-window robust automatic threshold selection”. In: *Computer Analysis of Images and Patterns*. Springer. 2003, pp. 369–376.

[13] N. Young and Adrian N. Evans. “Psychovisually tuned attribute operators for pre-processing digital video”. In: *IEE Proceedings-Vision, Image and Signal Processing* 150.5 (2003), pp. 277–286.