Efficient morphological tools for astronomical image processing

Moschini, Ugo

Published as:

P. Teeninga, U. Moschini, S.C. Trager and M.H.F. Wilkinson, *"Statistical attribute filtering to detect faint extended astronomical sources"*, Mathematical Morphology - Theory and Applications, Volume 1, Issue 1, pp. 100–115. De Gruyter, 2016.

Section 3.2 is taken from: Ugo Moschini and Michael H.F. Wilkinson, *"Improving Background Estimation for Faint Astronomical Object Detection"*, IEEE Proceedings of International Conference on Image Processing, pp. 1046–1050, 2015.

Chapter 3

Improved Detection of Faint Extended Astronomical Objects through Statistical Attribute Filtering

Abstract

In astronomy, sky surveys contain a large number of light-emitting sources, often with intensities close to the noise level. Automatic extraction of astronomical objects is therefore needed. Source Extractor is a widely used program for automated source extraction and cataloguing, but it is not optimal with faint extended sources. Using Source Extractor as a reference, the chapter describes a novel max-tree based method for extraction of faint extended sources without using a stronger image smoothing. The max-tree structure is a hierarchical representation of an image, in which attributes can be computed in every node. We present an improvement on the background estimation method of Source Extractor on Sloan Digital Sky Survey images, that suffered from bias caused by presence of extended sources. After the background is subtracted, a max-tree is built. Object detection is performed on its nodes and it relies on the distribution of a statistic calculated using the power attribute, compared to the expected distribution in case of noise. A comparison with the object extraction of Source Extractor is shown and results are discussed.

3.1 Introduction

In astronomy, sky surveys contain a huge quantity of light-emitting sources, representing astronomical objects. With advances in technology, an increasing number of images and volumes at both high resolution and high bit-depths becomes available. Manually extracting every object is not feasible, due also to the low intensities of many sources, often close to the noise level. Object detection can be seen as the process of separating groups of pixels that belong to a source from those that belong to noise or background. Masias et al. (Masias et al., 2012) presented a detailed overview of state-of-the-art object detection techniques in astronomy. The authors stressed the fact that many astronomical objects do not show clear boundaries and have intensities close to the detection level of the instrument. Besides, the size of relevant objects in an image can vary greatly. To detect sources in astronomical images, two main categories of methods are prominent: thresholding and local peak search. With the former method, connected sets of pixels are considered an object if they are above a certain threshold value; with the latter, objects are identified descending to lower intensities from the pixels representing image maxima.

In recent years, methods based on component trees or max-trees have been used to process grey-scale or mono-channel images. They rely on a hierarchical representation of an image, finding connected sets of pixels at every intensity level. The tree structure can be augmented with attributes related to every node for image filtering or segmentation purposes. Such structures have been already successfully used for astronomical object detection (Berger et al., 2007; Perret et al., 2010; Ouzounis and Wilkinson, 2011).

Source Extractor (SExtractor) (Bertin and Arnouts, 1996) is a state-of-the-art software for automatic extraction of astronomical objects. It is based on thresholding and it is used on many data types, such as optical, infra-red and radio datasets. Qualitative and quantitative comparisons between SExtractor and other methods can be found in Masias et al. (2012, 2013). Its main disadvantages are two: the selection of the optimal threshold above which groups of pixels are considered as objects and the detection of the fainter structures, often faulty. Extraction of faint structures is essential to the understanding of the evolution and morphology of galaxies SExtractor first estimates the image background, caused by light produced and reflected in Earth's atmosphere, and subtracts it from the image before object detection. In SExtractor, with the default settings, to perform a correct segmentation and avoid false positives, objects are identified with the pixels with intensity at a threshold level higher than 1.5 times the standard deviation of the background estimate at that location. We refer here to such mechanism as *fixed threshold*: the threshold value relies only on local background estimates in different sections of the image and it ignores the actual object properties. To identify nested objects, larger regions are later deblended, re-thresholding at 32 quantized levels, logarithmically spaced be-

tween the threshold value and the peak intensity in the region. Deblending occurs when the integrated intensity is above a certain fraction of the total intensity and if another branch with such property exists

SExtractor's estimate shows bias from objects (see Fig. 3.1), which reduces their intensities. In the example in Fig. 3.1, part (a) shows a pair of merging galaxies with a faint tail structure linking them, part (b) shows the contrast-stretched background estimate from SExtractor. Clearly, there is correlation with the objects. Subtracting this background, shown in part (c), with pixels below the background estimate set to zero, reduces their intensities, leading to failure in detecting faint outlying regions of galaxies, or tidal tails in galaxies. Fig. 3.1(d) shows the output obtained after our background estimate. Finally, part (e) and (f) show the different object detections from SExtractor and our own detection method described in this chapter. Clearly, faint structures are detected better. In our work, we used a dataset of 254 images extracted from the Sloan Digital Sky Survey (Stoughton et al., 2002) (SDSS) Data Release 7 (Abazajian et al., 2009) catalogue, containing optical images. The whole catalogue contains 357 million unique objects, representing a perfect example of the reason why automatic object detection is needed. Our dataset contains merging galaxies, that often show faint extended structures due to their interaction and tidal forces, and overlapping galaxies, that look close to each other at the same location in the sky, but they are not interacting and they might be in fact very distant. In this chapter, we propose a solution to improve the fixed threshold approach and the quantized deblending step. In Section 3.2, we introduce our background estimation, first introduced in Teeninga et al. (2013). In Section 3.4, we present a max-tree based method that locally varies the threshold depending on object size by using a statistical test rather than arbitrary thresholds on the attributes computed in the nodes of the tree. The distribution of an attribute, the power (Young and Evans, 2003), is studied with respect to its expected distribution in case of noise components. Nodes are marked significant if noise is an unlikely cause, for a given significance level. In Section 3.5, object detection and deblending are explained and in Section 3.6 a comparison with SExtractor is presented, followed by a short analysis of performance in Section 3.7 and conclusions and future directions of research in Section 3.8.

3.2 Background estimation

Images are acquired with a CCD, photons are converted to electrons which are counted per pixel. After subtracting the software bias from the corrected images, the pixel values are directly proportional to the photo-electron counts (sdss.org, 2007). Noise is mostly Poissonian due to photo-electron counts already at the minimum intensity. The distribution is close to Gaussian; the sky (background) typically contributes 670 photo-electrons to the counts per pixel. In our method, background

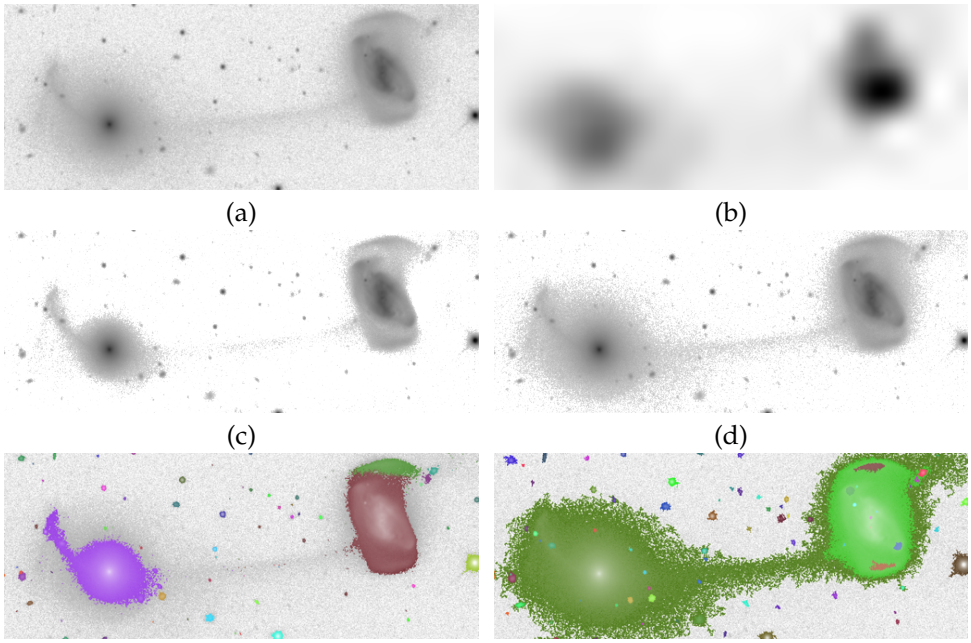


Figure 3.1: Crop of SDSS file `fpC-002078-r1-0157.fit` showing merging pair of galaxies (a) original image; (b) background estimate of the image in by SExtractor, contrast stretched; (c) difference of (a) and (b), smoothed, logarithmic scale and showing values above $.75\hat{\sigma}_{\mathrm{bg}}$; (d) as (c) but with new background estimate subtracted. Part (e) shows source detections by SExtractor with default settings, while (f) shows the same results for our proposed method.

pixel values are assumed to be from a Gaussian distribution. 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 quantisation. 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 and is subtracted from the image.

To find flat tiles, we first split the image into tiles of the same size. The following statistical tests are applied to the tiles:

1. a normality test using the D'Agostino-Pearson K^2-statistic (D'Agostino et al., 1990) which is based on the skewness and kurtosis.

2. t-tests of equal means for different parts of the tile.

The t-tests are used because the normality test does not take positions of pixels into

Algorithm 3.1. Algorithm to find flat tiles.

procedure ISFLAT($w \times w$ tile t, Rejection rate α)
    $\alpha_1 \leftarrow 1 - (1 - \alpha)^{1/2}$;
    Perform the D'Agostino-Pearson K^2-test on the values of T with rejection rate α_1: return $false$ if rejected;
    $(T_{1,1}, T_{1,2}, T_{2,1}, T_{2,2}) \leftarrow \frac{w}{2} \times \frac{w}{2}$ tiles partition of T;
    $\alpha_2 \leftarrow 1 - (1 - \alpha)^{1/4}$;
    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;
    return $true$;
end procedure

account. Only using the normality test could lead to situations where tiles with a near-linear slope are accepted. The procedure to identify a flat tile is outlined in Alg. 3.1. If a tile is flat, their dimensions are doubled to find the largest possible flat tile starting from the initially split tile. The background mean and variance is computed on all the flat tiles found in the image.

Inverses of cumulative distribution functions (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 Fig. 3.2, for $\alpha = 0.05$. 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.

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 is needed later for object segmentation. An important to discuss how to set the 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 artefacts. 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.

Considering the standard deviation of the noise at the background is approxi-

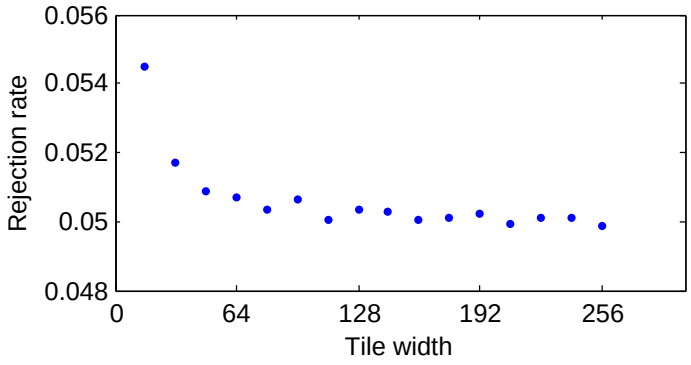


Figure 3.2: $\texttt{IsFlat}(T, 0.05)$. Rejection rate for square flat tiles based on 1 million simulations for each size.

mately 5.4 for most images, there is not much difference between $\alpha = 10^{-7}$, $\alpha = 0.05$ and $\alpha = 0.5$: α is kept at 0.05. All images in the data set have 64×64 flat tiles. The minimum tile width w_0 is set to 64.

3.2.1 Is a constant a good fit?

An important question is whether the constant background gives a good model fit. 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 $\beta = (\hat{\mu}_{\text{bg}} - \mu_F)\hat{\sigma}^{-1}_{\text{bg}}$. β approximately $\sim N(0, w^{-1})$ with $\hat{\mu}_{\text{bg}}$ relatively constant. When $w = 64$, 95% of the absolute β values would be below 0.031 on average. In Fig. 3.3 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 ($\times$ standard deviation of the noise). Images with outliers are inspected to determine the cause.

The background estimates in the image in Fig. 3.4 at 64×64 tile size and 128×128 are 1137.1 and 1135.9 respectively, with $\sigma_{\text{bg}} \approx 5.4$, which shows the influence of the large object on the local estimates. The images in Fig. 3.5, which also have been picked to inspect β outliers, have a similar situation. The main cause of the relatively large absolute values of β appears to be objects, not changes in the background. Experimentally, we verified that a non-constant background fit closer to the local estimates would increase the error at locations correlating with objects. For this data set, and these local background estimates, a constant is a good fit.

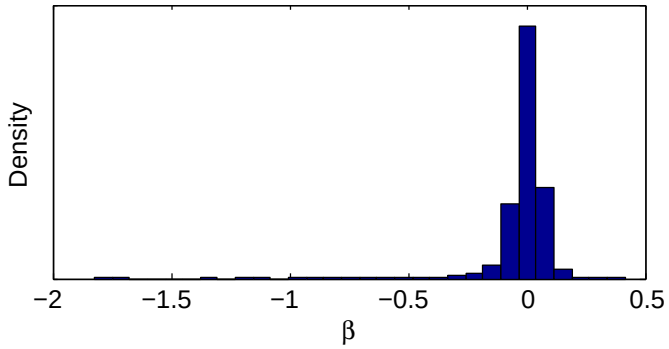


Figure 3.3: Distribution of β, for all images combined. Tile size is 64×64.

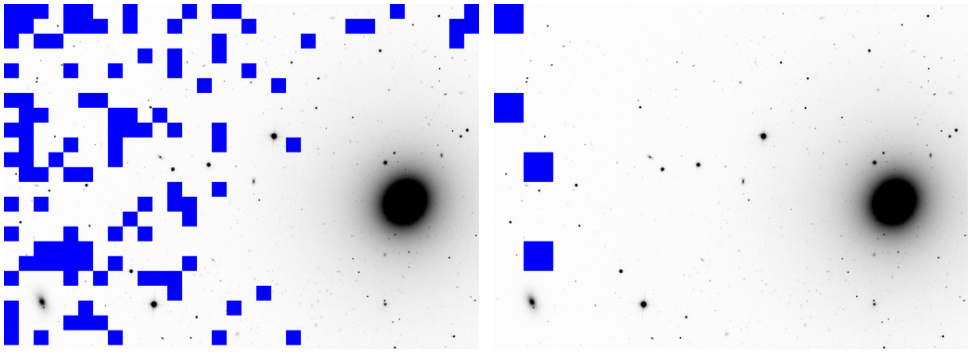


Figure 3.4: Image with β outliers. Left: 64×64 flat tiles shown in blue; right: 128×128 flat tiles shown in blue. SDSS file: `fpC-003836-r4-0249.fit`

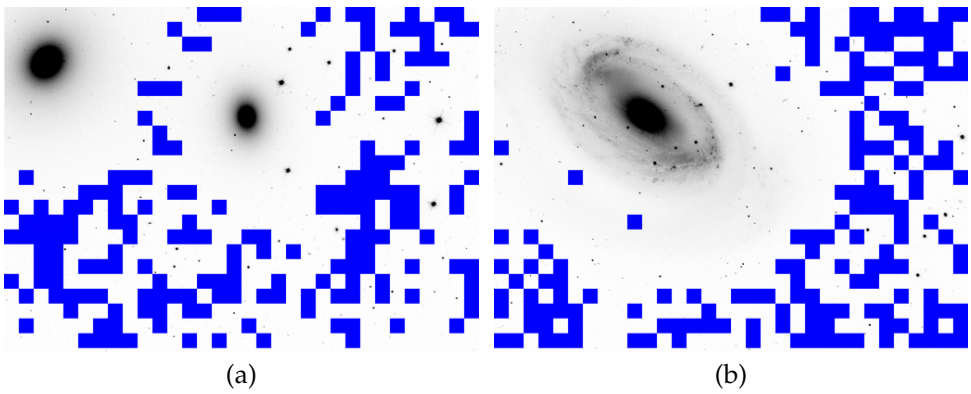


Figure 3.5: Images with β outliers. 64×64 flat tiles shown in blue. Left: SDSS file `fpC-005313-r1-0067.fit`; right: SDSS file: `fpC-005116-r5-0148.fit`

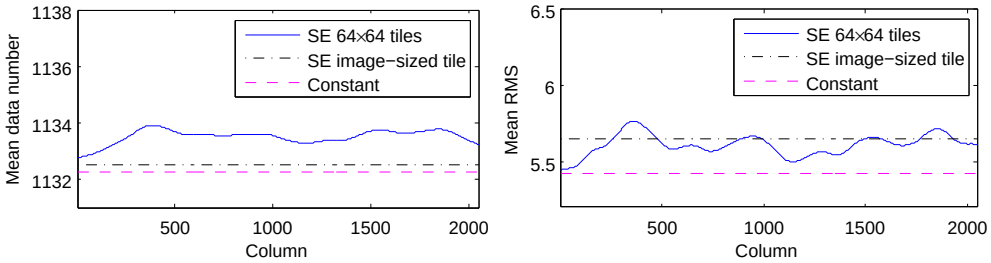


Figure 3.6: (a) Average estimate of the background and (b) noise standard deviation at the background (bottom) for all images, column-wise.

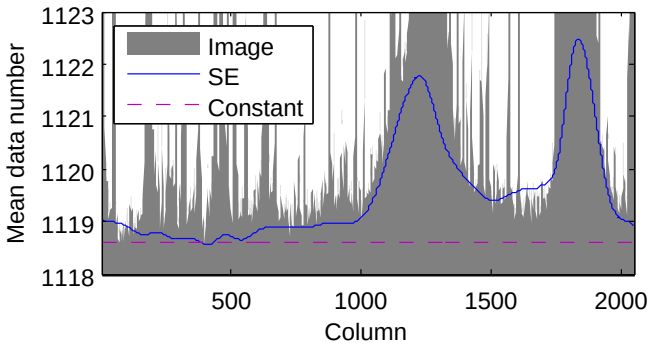


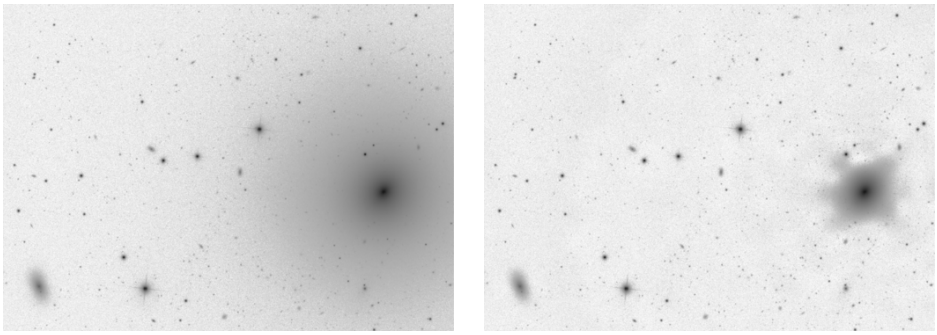
Figure 3.7: Background estimate by SExtractor compared to the constant estimate, averaged column-wise, for Fig. 3.1.

3.2.2 Comparison of the background estimate w.r.t. SExtractor

SExtractor uses bi-cubic 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.

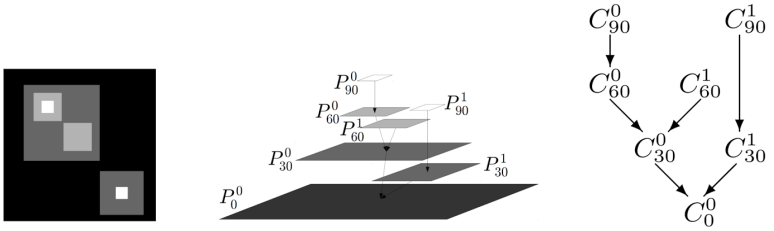
Fig. 3.6 shows that the constant estimate suffers less from object bias on average. The background estimate by SExtractor is 1.27 higher on average which corresponds approximately to $0.23\sigma_{\mathrm{bg}}$, using $\sigma_{\mathrm{bg}} \approx 5.4$. An image-sized tile in SExtractor also reduces bias, on average, but the higher noise standard deviation, compared to the (other) constant, indicates a worse fit.

The background estimate by SExtractor correlates with objects, see Fig. 3.7, making it more difficult to detect (parts of) objects after subtraction. For example the connection between the merging galaxies in Fig. 3.1(d) is more clear than in Fig. 3.1(c). Another problem in the background estimation by SExtractor due to correlation



(a) (b)

Figure 3.8: Left: Constant background estimate, subtracted from the image. Right: SExtractor background estimate, subtracted from the image. Logarithmic scale. SDSS file `fpC-003836-r4-0249.fit`.



(a) 2D image (b) Peak components (c) Max-tree

Figure 3.9: (a) a grey-scale 2D image with intensities from 0 to 90, (b) its peak components P^k_h at intensity h and (c) the corresponding max-tree nodes C^k_h.

with objects is distortion of object shapes, as seen in Fig. 3.8, which appears to happen for every non-constant estimate. With the goal of having the least object bias and preserving object shapes, using the constant background estimation is clearly better.

3.3 Connected filtering on a max-tree structure

Any grey-scale image can be represented as a set of connected components, that are groups of pixels path-wise connected and with the same intensity, according to the classical definition of connectivity (Serra, 1988). There being an ordering in the image intensities, the connected components can be nested in a hierarchical tree structure, namely a max-tree (Salembier et al., 1998). Every node in the tree corresponds to a peak component, which is a connected component at a given intensity

level in the image. The leaves of the tree represent the local maxima of the image. Fig. 3.9b illustrates the hierarchy of peak components at different intensities h for the image in Fig. 3.9a. The arrows in Fig 3.9c represent parent-child relationships that link the nested peak components. Useful measures related to the components can be computed efficiently while the max-tree is being built. As in connected mathematical morphology, all image operations are performed on the connected components: we refer to those operations as connected attribute filters (Breen and Jones, 1996). Attribute filters compute some property or attribute, like area or shape measures (circularity, center of mass, ...) for every connected component. According to the value of such attributes, the filter selects the components to be kept and those to be removed. This operation, for all connected components at every intensity level, is made fast by the use of a max-tree structure.

In the next section, we will define a *statistical* attribute filter used to detect astronomical objects. It is based on the expected noise distribution in the image compared with the distribution of the power attribute, as a function of its area. It selects which nodes of the tree are likely to belong to objects and which nodes are due to noise.

3.4 Identifying significant nodes

In our detection method, the supporting data structure is a max-tree (Salembier et al., 1998) created from the image, where every node corresponds to a connected component for all the threshold levels in the original image. The choice was inspired by the simplified component tree used in the SExtractor deblending step and was already suggested in Perret et al. (2010). Meaningful attributes allow for selecting components for a given purpose. For example, Perret et al. (Perret et al., 2010) and Berger et al. (Berger et al., 2007) defined attributes for object detection on multi-spectral data and optical data, respectively.

Four significance tests are defined in this section. Their aim is to mark a node as significant if one or more objects are represented by the pixels of the node, given our background estimate. To identify the nodes in the tree belonging to objects, we will start from the definition of the power (Young and Evans, 2003) attribute. It is a measure similar to the definition of object *flux*, often used in astronomy, or the integrated intensity, used by SExtractor. Let us define the intensity associated with a node P with $f(P)$. Similarly, let $f(x)$ be the value of a pixel x. Let us define also P_{anc} as the closest significant ancestor of a component P. If no such node exists, P_{anc} is equal to the root node. P_{anc} also represents the local background of a component. P is significant if it can be shown that $\exists x \in P : O(x) > f(P_{anc})$, for a given significance level α. Function $O(x)$ represents the hypothetical image that contains only objects and no noise, introduced in Section 3.2. We will use two different definitions of the

power attribute of P:

$$\text{power}(P) \quad := \quad \sum_{x \in P} (f(x) - f(\text{parent}(P)))^2 \tag{3.1}$$

and

$$\text{powerAlt}(P) \quad := \quad \sum_{x \in P} (f(x) - f(P_{\text{anc}}))^2. \tag{3.2}$$

To determine if a node P is due to noise or not, we will study the distribution of the power values for several area values of the components, with respect to its expected distribution in case of noise nodes. Noise scales linearly with the intensity level. To filter local maxima due to noise on top of objects, the local background of an object can be higher than our constant estimate. Therefore, to normalize the power attribute for the nodes that do not have the root (background) as parent, the power is divided by the local background variance $\sigma^2 = \hat{\sigma}^2_{\text{bg}} + g^{-1} \cdot f(parent)$. The parent component, or the closest significant ancestor in some cases, are considered as local background. To identify significant nodes, four significance tests that use the two attributes above are defined in the following.

3.4.1 Significance test 1: `power` given `area` of the node.

A node P is considered significant if it is possible to provide a statistical test to show that $O(x) > f(P_{\text{anc}})$ for pixel locations $x \in P$, given a significance level α. We use the following hypothesis:

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

This test uses the definition of power attribute in Equation 3.1. In the limit case, $\forall x \in P : O(x) = f(\text{parent}(P))$ for pixels $x \in \text{parent}(P)$. In this test, we assume that the distribution of the power attribute scaled by the variance σ^2 for noise nodes follows a χ^2 distribution. In fact, in the case of Gaussian noise, the power of noise components is a sum of squared independent variables and therefore it follows such distribution. For a random pixel x in P, the value $(f(x) - f(\text{parent}(P)))^2/\sigma^2$ has a χ^2 distribution with 1 degree of freedom. If P is due to noise, it has a χ^2 distribution with degrees of freedom equalling to the area of P. Let us define a function $\texttt{inverse}\chi^2\texttt{CDF}(\alpha, \texttt{area})$ that returns the rejection boundary given by the χ^2 cumulative distribution function (CDF), for a significance level α. The χ^2 CDF (or inverse) is commonly available in scientific libraries. An example of a rejection boundary of a χ^2 CDF is shown in Fig. 3.10a. 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 $x \in \text{parent}(P)$, making P

significant.

A precise χ^2 distribution of the power attribute holds for the nodes that have the root as parent. For the other nodes, the rejection boundary is a conservative model and minimizes the number of false positives. In significance test 1, leaf nodes are less likely to be found significant due to their small area and the low intensity difference with the parent node. Some nodes could be erroneously marked as noise even if they are not. The next three tests use the alternative definition of the power attribute in Equation 3.2 to address this issue: the power attribute has larger values than in Equation 3.1.

3.4.2 Simulating distributions

In all the next three significance tests (all right tailed) the exact distribution of `powerAlt` is not known and it is obtained by Monte Carlo simulation. Gaussian noise images are generated, with mean and variance equal to our estimates. A number n of independent values is generated. On average, given a significance level α, the number of false positive equals to $r = \alpha \cdot n$ nodes: the attribute of the false positive nodes is greater than or equal to the rejection boundary. The best estimate of the rejection boundary, without any further information about the distribution, is the average on many noise images of the two smallest of the $r + 1$ largest values of the attribute.

3.4.3 Significance test 2: `powerAlt` given `area` and `distance`.

In this test, we use the following hypothesis:

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

To make the significance level more constant for every node, independently of its height in the tree, we refer to its ancestor rather than to the parent node in the computation of the power attribute. The definition of power attribute in Equation 3.2 is used. Let us assume $H_{\texttt{powerAlt}}$ is true and consider the extreme case $\forall x \in P : O(x) = f(P_{\text{anc}})$. Let us define $\texttt{distance}(P) := f(P) - f(P_{\text{anc}})$. Let X be a random set of `area`(P) - 1 values drawn from a truncated normal distribution with a minimum value of $\texttt{distance}(P)$. The variance is set to $\sigma^2 = \hat{\sigma}^2_{\text{bg}} + g^{-1} f(P_{\text{anc}})$. Attribute $\texttt{powerAlt}(P)$ has the same distribution as $\texttt{distance}^2(P)$ plus the sum of the squared values in X. Let the function $\texttt{inversePowerAltCDF}(\alpha, \texttt{area}, d))$ return the estimated rejection boundary for the power attribute, for given α, area and distance values. Hypothesis $H_{\texttt{powerAlt}}$ is rejected if $\texttt{powerAlt}(P)/\sigma^2 > \texttt{inversePowerAltCDF}(\alpha, \texttt{area}, d))$: it means that the object image $O(x)$ at some pixels x in P is higher than $f(P_{\text{anc}})$, and P is marked as significant. The minimum area

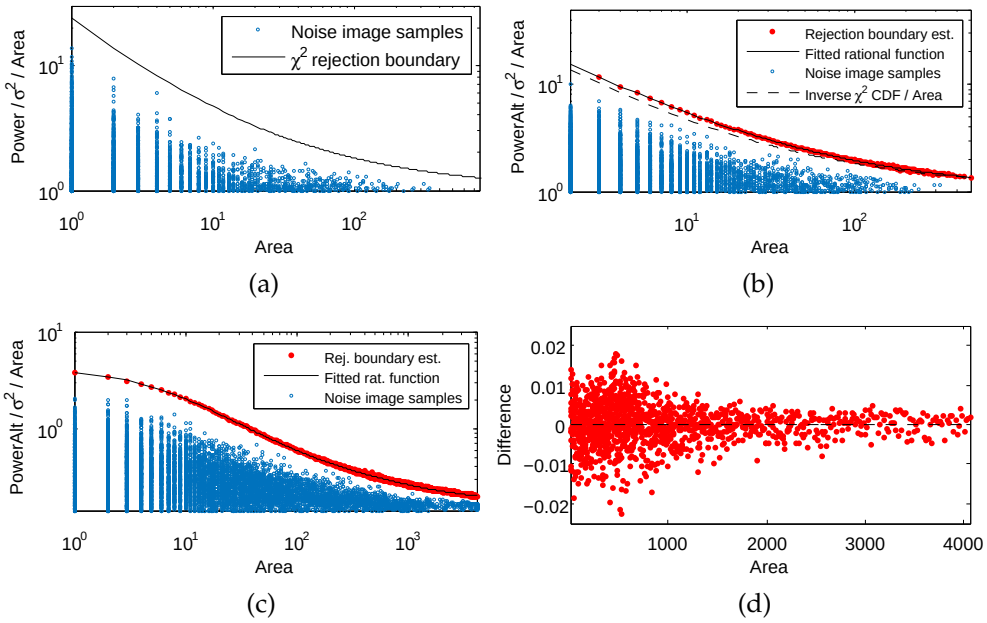


Figure 3.10: (a) rejection boundaries for significance test 1 and (b) the simulated rejection boundaries for test 3 and (c) test 4, log-log scaled; (d) shows the difference between the rational function and its estimate in (c). $\alpha = 10^{-6}$.

of a significant node is 2 pixels. An estimate is given for `inversePowerAltCDF` for constant α, varying `area` and `distance`. Random samples are generated given several values of area and distance: the range for the area values goes from 2 to 768 pixels, while distance has a maximum value of 4, with 0.25 as step size. For each rejection boundary, varying `distance`, a rational function is fitted to reduce the error and the storage space. In the tests, the rational functions appear to be valid approximations. 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 boundaries is used if a boundary is not available for a `distance`, which happens in nearly all cases.

3.4.4 Significance test 3: `powerAlt` given `area`.

Significance test 3 uses the distribution of `powerAlt` given α and area of a component. It is independent of the `distance` measure, not used as parameter in the inverse CDF. Using the assumptions from the significance test 2, `distance`(P) has a truncated normal distribution with a minimum value of 0, the same distribution

Algorithm 3.2. Nodes in M that are unlikely to be noise are marked as significant.

1: **procedure** SIGNIFICANTNODES(Max-tree M, significance test $nodeTest$, significance level α, gain g, variance of the background $\hat{\sigma}_B^2$)
2: **for all** nodes P in M with $f(P) > 0$ in non-decreasing order **do**
3: **if** `nodeTest`$(M, P, \alpha, g, \hat{\sigma}_{\text{bg}}^2)$ is $true$ **then**
4: Mark P as significant
5: **end if**
6: **end for**
7: **end procedure**

as a random non-negative pixel value. The rejection boundary is calculated through simulated noise images. Fig. 3.10b shows the rejection boundary estimate and the fitted rational function for this significance test. Four-connectivity is used. A different connectivity would possibly change the rejection boundary.

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

It is equal to significance test 3 with the only difference that the image is smoothed beforehand. Smoothing is used to reduce noise and to detect more objects. A larger number of objects is detected with this test. We use the same smoothing filter used in SExtractor:

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

Filtering is done after background subtraction and before setting negative values to zero. After smoothing, pixel values are not independent any more. Decision boundaries are determined again through Monte Carlo simulations. Fig. 3.10c shows the rejection boundary with its fitted rational function and Fig. 3.10d shows the difference between the rational function approximation and the estimates for this significance test.

3.4.6 Testing the nodes

Alg. 3.2 describes the method used for marking nodes not due to noise as significant. Visiting nodes in non-decreasing order by pixel value simplifies the identification of $P_{\texttt{anc}}$, if stored for every node. In the case of significance test 1, nodes can be visited in arbitrary order. Function `nodeTest`$(M, P, \alpha, g, \hat{\sigma}_{\text{bg}}^2)$ in Alg. 3.2 performs the significance test and returns true if P is significant, false otherwise.

Algorithm 3.3. Nodes in M that represent an object are marked.

```
1: procedure FINDOBJECTS(Max-tree M)
2:     for all significant nodes P in M do
3:         if P has no significant ancestor then
4:             Mark P as object
5:         else if mainBranch(P_anc) is not equal to P then
6:             Mark P as object
7:         end if
8:     end for
9: end procedure
```

3.4.7 Discussion on the significance level 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 $\alpha \cdot 0.5n$ if the nodes are independent. 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, given a right-tailed distribution. We performed a test on noise images and the actual number of false positives observed turned out to be lower. An estimate of the actual number of false positives is 0.41, 0.72, 0.94 and 0.35 for the four significance tests, respectively, averaged over 1000 simulated noise images. Argument α is set to 10^{-6} by default.

3.5 Finding objects

After that nodes have been marked as significant, it must be considered that multiple significant nodes could be part of the same object. A significant node with no significant ancestor is marked as an object. Let $\texttt{mainBranch}(P)$ be the function returning a significant descendant of P with the largest area. A significant node, with significant ancestor P_{anc}, that differs from the one returned by $\texttt{mainBranch}(P_{\text{anc}})$ is marked as a new object. This operation of identifying nested actual objects on top of a larger one is called deblending. The decision if a node is considered a new object depends on the used significance test, smoothing filter and connectivity, as it will be shown in the comparison section. The procedure of marking nodes as objects is summarised in Alg. 3.3.

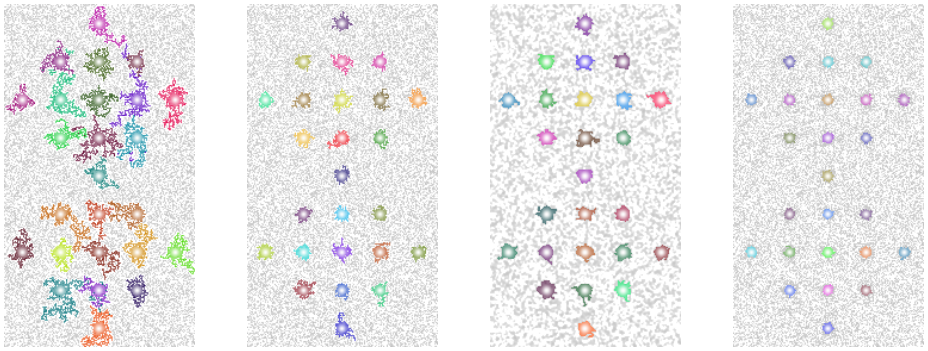
3.5.1 Moving object markers up: parameter λ

Nodes marked as objects have a number of pixels attached due to noise. The number decreases at a further distance from the background signal. Object markers can be

Algorithm 3.4. For every object marker that starts in a node P and moves to the node P_{final}: $f(P_{\text{final}}) \geq f(P_{\texttt{anc}}) + \lambda$ times the local standard deviation of the noise, when possible. $f(P_{\text{final}})$ might be lower if P_{final} has no descendants.

```
1: procedure MoveUp(Max-tree M, factor λ, gain g, variance of the noise at the background σ̂²_bg)
2:     for all nodes P in M marked as objects do
3:         Remove the object marker from P
4:         h ← f(P_anc) + λ√(σ̂²_bg + g⁻¹ f(P_anc))
5:         while f(P) < h do
6:             if P has a significant descendant then
7:                 P ← mainBranch(P)
8:             else if P has a descendant then
9:                 P ← mainPowerBranch(P)
10:            else
11:                Break
12:            end if
13:        end while
14:    end for
15:    Mark P as object
16: end procedure
```

moved up in the tree, for λ times the standard deviation of the noise. The procedure is detailed in Alg. 3.4 The obvious choice for an object node P is `mainBranch`(P), if such a node exists, since it does not conflict with other object markers. Otherwise, the descendant of P with the highest p-value found with the corresponding CDF for its `power` or `powerAlt` attribute value would be the perfect candidate. However, the CDF is not always available or easy to store. Instead, the descendant with the largest `power` attribute is chosen, if at least one exists. The function that returns the descendant is called `mainPowerBranch`(P). An alternative to allowing a lower value of $f(P_{\text{final}})$ in Alg. 3.4 is to remove those object markers. If the parameter λ is set too low, there are too many noise pixels attached to objects. However, to be able to display faint parts of extended sources a low λ is preferred. We performed tests on objects simulated with the IRAF software, generating 25 stars with low magnitude (-5) and adding Gaussian noise at every location with the pixel value as mean and variance. If parameter λ is set too low, as in Fig. 3.11(a), there are too many noise pixels attached to objects. The object shapes in Fig. 3.11(d) look better. However, to be able to display faint parts of extended sources a low λ is preferred, therefore $\lambda = 0.5$ is used as a compromise. Experimentally, such value of λ worked effectively on the SDSS data set.



(a) (b) (c) (d)

Figure 3.11: Twenty-five simulated stars, logarithmic grey scale: (a) $\lambda = 0$; (b) $\lambda = 0.5$; (c) $\lambda = 0.5$, with the image smoothed using the default SExtractor filter; (d) $\lambda = 2$.

Algorithm 3.5. Max-tree M. Nodes in M corresponding to objects are marked.

1: **procedure** MTOBJECTS(Image I, function `nodeTest`, significance level α, gain g, move factor λ)
2: $(\hat{\mu}_{\text{bg}}, \hat{\sigma}^2_{\text{bg}}) \leftarrow$ `EstimateBackgroundMeanValueAndVariance`()
3: $I_{i,j} \leftarrow \max(I_{i,j} - \hat{\mu}_{\text{bg}}, 0)$
4: $M \leftarrow$ create a max-tree representation of I
5: `SignificantNodes`$(M, \texttt{nodeTest}, \alpha, g, \hat{\sigma}^2_{\text{bg}})$
6: `SignificantNodes`$(M, \texttt{nodeTest}, \alpha, g, \hat{\sigma}^2_{\text{bg}})$
7: `FindObjects`(M)
8: `MoveUp`$(M, \lambda, g, \hat{\sigma}^2_{\text{bg}})$
9: **end procedure**

3.6 MTObjects vs Source Extractor

Alg. 3.5 summarises the whole procedure from background estimation to object identification. The proposed method is called MTObjects (Max-Tree Objects), since astronomical object detection is obtained using a max-tree structure. Our method is compared with the segmentation performed by SExtractor 2.19.5. SExtractor settings are kept close to their default values:

- Our background and noise root mean square estimates are used. This already improves the segmentation of SExtractor with respect to the original estimate of SExtractor, that correlates too much with objects.
- `DETECT_MINAREA = 3`. In SExtractor 2.19.5, it represents the minimum number of pixels for a component to be possibly detected as object.
- `FILTER_NAME = default.conv`. It is the default smoothing filter, as seen in Section 3.4.5.

- `DETECT_THRESH = 1.575`σ above the local background. The default threshold of 1.5 (times the noise standard deviation) is changed to make the expected number of 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.
- `MEMORY PIXSTACK = 4000000`. To avoid overflows: the value is larger than the number of pixels in an image of our dataset.

While there is no guarantee that these settings are optimal, our comparison gives an impression of the performance of our method. A quantitative comparison on simulated data could be an interesting follow-up to show more precisely the strengths and weaknesses of MTObjects and SExtractor. For the experiments, we used 254 images from the SDSS Data Release 7 (Abazajian et al., 2009) catalogue. For every section of the sky, five images are acquired in five different bands of the widely used photometric system (u', g', r', i', z'). We use r-band images, because they have the best quality (Gunn et al., 1998).

3.6.1 Object detection

An object is defined as a lump in the image signal that is not due to noise. All the four significance tests were compared against each other and SExtractor. The significance test 4 returns a larger number of objects in about 100% of the images in the dataset with respect to significance test 1 and 2 and in about 70% with respect to significance test 3 and SExtractor. After inspection of the results, it is clear that, in general, MTObjects preserves more the faint outer structures of objects and nested objects are deblended in a more natural way. Examples can be seen in Fig. 3.12 and Fig 3.13. The fainter parts and galactic filaments are identified by MTObjects, for example in Fig. 3.13e and Fig. 3.13b. In these two cases the difference is striking. Object deblending by SExtractor does not always work well. Sometimes, weird segmentations appear, such as the one in Fig. 3.13c. We noticed that MTObjects detects more objects nested in larger objects (galaxies), when the pixel values of the nested objects are above the SExtractor's threshold. For example, Fig. 3.12b shows a few stars on top of the galaxy segmented as separate objects, whereas in the SExtractor they are for the most part included in the same object as the galaxy, see Fig. 3.12c. This is explained by the fact that every node in the max-tree is used, while SExtractor uses a fixed number of sub-thresholds from its background level to the highest peak component in the object, without considering noise and object properties.

To understand if the improved detection of nested objects can explain the better performance of significance test 4, we limited the data set to more compact objects. This is achieved by making a list sorted by area of the largest connected component in each image at the threshold used by SExtractor. The performance of significance

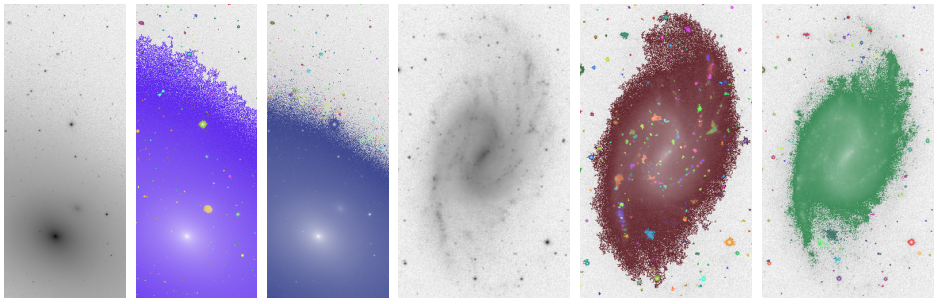


Figure 3.12: MTObjects identifies better fainter outer regions and the nested objects. Crop of `fpC-003804-r5-0192.fits`, same scale: (a) original image; (b) result of significance test 4; (c) result of SExtractor. Crop of `fpC-001332-r4-0066.fits`, same scale: (d) original image; (e) result of significance test 4; (f) result of SExtractor.

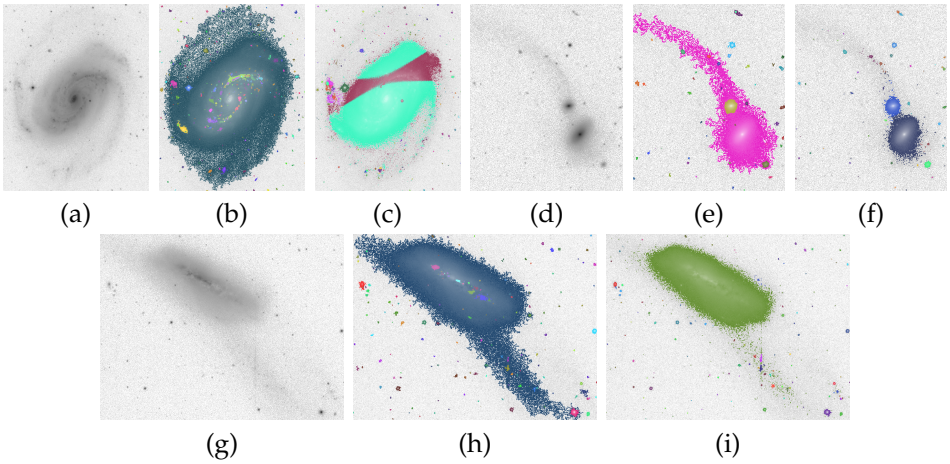
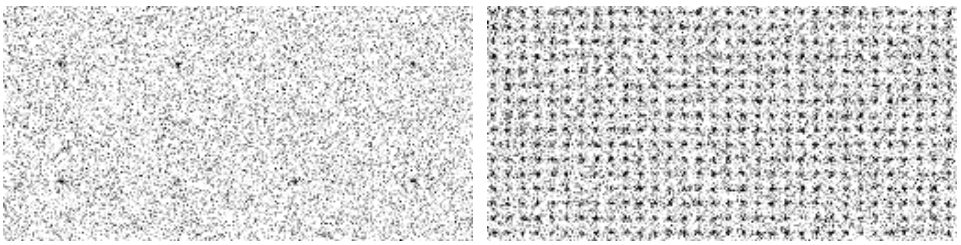


Figure 3.13: Comparison of objects with faint extended regions. Crop of `fpC-003903-r2-0154.fits`, same scale: (a) original image; (b) significance test 4; (c) SExtractor. Crop of `fpC-004576-r2-0245.fits`, same scale: (d) original image; (e) significance test 4; (f) SExtracto. Crop of `fpC-004623-r4-0202.fits`, same scale: (g) original image; (h) significance test 4; (i) SExtractor.

test 4 and SExtractor is similar on this new dataset: the difference in the total number of objects found in the images is explained by the number of nested object detections. The use of a low value (0.5) of the λ parameter used in `MoveUp`, let MTObjects identify better the faint structures, without increasing the number of false positives with respect to SExtractor. It is possible to lower λ further, but more noise would be attached to the objects and included in the segmentation.

We tested then how significance test 4 performs in the case of densely spaced



(a) (b)

Figure 3.14: Left image shows part of the sparse stars' grid; right image shows part of a dense star grid.

overlapping objects. 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 distinct objects, MTObjects could fail to detect them as separate. A grid filled with small stars is generated with the IRAF software, as in Fig. 3.14. 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 to the image, with the pixel value as mean and variance. In this case of densely spaced objects, SExtractors detects a number of stars closer to the actual number than MTObjects with significance test 4. The results show that a threshold would be better when objects are very densely spaced. We performed a further test with an actual image of a globular cluster. A cluster can be seen as a single object made of a halo caused by the light emitted from a large number of stars close to each other. The fixed threshold of SExtractor seemed to work slightly better. In a globular cluster image that we used, the total number of stellar objects identified by SExtractor is 3164, whereas MTObjects detected 3035. In SExtractor, when the halo is below its fixed threshold, it is considered background. The intensity of stars is estimated relatively to that background. In MTObjects, the estimated intensity of objects is lower, because the halo is not part of the background and considered as a large object. Therefore, some objects have too low intensity to be deblended from the halo region.

3.6.2 Object fragmentation

A source of false positives, apart from those caused by the statistical tests, is the fragmentation of objects due to noise. An example is shown in Fig. 3.15. Fragmentation appears to happen in relatively flat structures and the chance is increased if different parts of the structure are thinly connected. If only one pixel connects two parts, the variation in value due to noise can make a deep cut. In the case of the

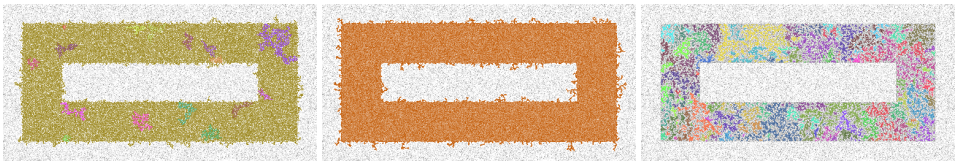
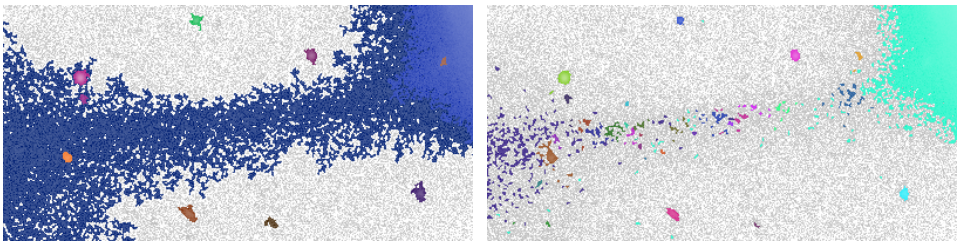


Figure 3.15: In case of a fragmented simulated object, we show three possible outputs for the significance test 3 (left), significance test 4 (middle) and SExtractor (right). The pixels of the object have the value 1.5, close to the SExtractor's threshold. The background is 0 and Gaussian noise is added with $\sigma = 1$. The image on the right shows a strong fragmentation.



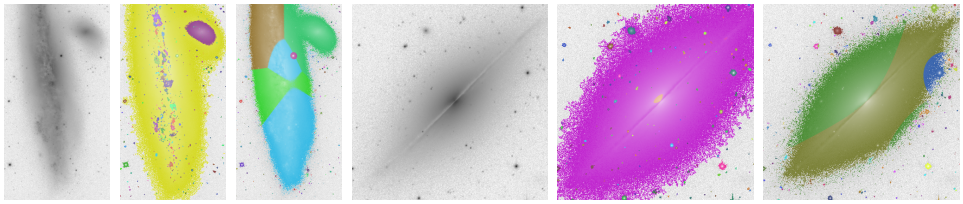
(a) (b)

Figure 3.16: Fragmentation of a thin faint filament between two galaxies, cropped section of file `fpC-002078-r1-0157.fits`. Significance test 4 (left) and SExtractor (right).

threshold used by SExtractor, fragmentation is severe if the object values are just below the threshold. The expected number of false positives due to fragmentation for the given data set is unknown. Most of the images do not show any evident fragmented objects. An image where it does happen is displayed in Fig. 3.16, when SExtractor is used. 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 fragmentation actually happens only for the galaxies in Fig. 3.16.

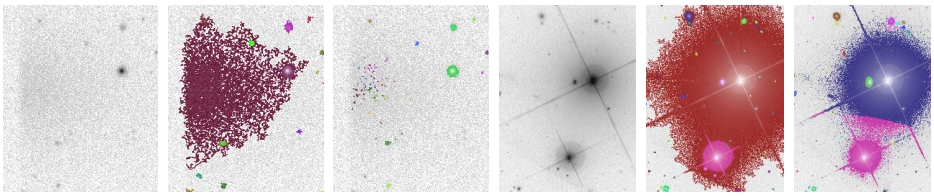
3.6.3 Dust lanes and artifacts

The last possible source of false positive is represented by dust lanes as in Fig. 3.17 and artefacts as in Fig. 3.18. In Fig. 3.17f, the galactic core is split due to a dust line. Fig. 3.18a, Fig. 3.18b and Fig. 3.18c could represent an artefact or a vertical cut-off. Refraction spikes, as the one shown in Fig. 3.18d, can also be a cause of false positives as in the wave-like shape in Fig 3.18f.



(a) (b) (c) (d) (e) (f)

Figure 3.17: Dust lanes. Crop of `fpC-004623-r4-0202.fits`, same scale: (a) original image; (b) significance test 4; (c) SExtractor. Crop of `fpC-001739-r60308.fits`, same scale: (d) original image; (e) significance test 4; (f) SExtractor.



(a) (b) (c) (d) (e) (f)

Figure 3.18: Artefacts. Crop of `fpC-002326-r4-0174.fits`, same scale: (a) original image; (b) significance test 4; (c) SExtractor. Crop of `fpC-001345-r3-0182.fits`, same scale: (d) original image; (e) significance test 4; (f) SExtractor.

3.7 Speed performance

On the SDSS dataset, both for MTObjects and SExtractor, the timer is started before background estimation and is stopped after object classification in SExtractor and after executing `MoveUp` in MTObjects. SExtractor does perform also object classification, far from perfect at lowest magnitudes, classifying objects as stars or galaxies through a neural network approach. The amount of time spent on classification is unknown. MTObjects does not perform any classification. Tests were done on an Intel Core i5-4460 with a single thread. Both methods are quite fast: the median timing on our dataset is 0.7670 seconds for MTObjects and 0.310 for SExtractor. SExtractor is typically 2.5 times faster than MTObjects, if median run-time is considered. When using the mean run-time, SExtractor is 1.3 times faster. SExtractor's execution time is affected by the number of pixels above the fixed threshold: it takes longer time for images that have many pixels above the threshold. MTObjects is more constant in run time and less dependant on the image characteristics. Further optimizations are possible and under study. First tests show that the time of MTObjects can be reduced to approximately the same as SExtractor.

3.8 Conclusions and future work

The max-tree based method (MTObjects) presented in this chapter performs better at extracting faint parts of astronomical objects compared to SExtractor, a state-of-the-art method. Our background estimate is less biased by objects than in SExtractor. MTObjects improves the fixed threshold mechanism used by SExtractor by using a statistical test based on the power attribute in each node of the max-tree representing the image. The distribution of the power is compared to its expected distribution in case of noise, according to the area of the node. MTObjects is better at extracting faint parts of objects compared to the fixed threshold used by SExtractor. The background estimate that we gave at the beginning of the chapter clearly reduces the object bias in our method, with respect to SExtractor. There are of course many other methods to which our approach should be compared before we can conclude that this is the best way forward, including recent techniques based on sparse representations and wavelets (Martínez-González et al., 2002). The whole MTObjects code is available from the author upon request.

The power attribute was initially chosen because in the non-filtered case it has a known scaled χ^2 distribution. Better attribute choices could be investigated. Deblending similar sized objects can be improved. Nested significant connected components could actually represent the same object. The current choice, controlled by λ in `MoveUp`, is not ideal. The threshold looks too high for large objects and too low for small objects. Parameter λ could be made variable and dependant on the filter, connectivity and node attributes. If other noise models are used, background mean and variance estimates and significance tests can be adjusted accordingly. The degree of smoothing applied helps to avoid fragmentation and it should be further investigated. Currently, the rejection boundaries are approximated by simulations which must be recomputed for every filter and significance level. Knowing the exact distributions would speed up this phase, but it is often not feasible.

When an object is defined to have a single maximum pixel value, excluding maxima due to noise, MTObjects is better at finding nested objects. Every possible threshold is tested in MTObjects, whereas SExtractor is bound to a fixed number of thresholds. Deblending objects appears to be better in MTObjects when there is a large difference in size and objects do not have a Gaussian profile. Otherwise, one of the objects will be considered as a smaller branch by MTObjects. A drawback is that too many pixels are assigned arbitrarily to a single object. The SExtractor method of fitting Gaussian profiles makes more sense in this case and allows for a more even split in pixels. This method could be added as post-processing step to MTObjects. MTObjects appears to be slightly worse in case of densely spaced and overlapping objects, like globular clusters.

Future work regarding the background estimate would include different shapes for the tiles representing the background, which 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. We are also considering extensions to non-flat background estimates for data sets which require this. One option is using weighted (by range) k-nearest neighbours. It works even if only a few local background estimates are available and, depending on k, is less affected by outliers than bi-linear or bi-cubic interpolation.