# Towards a fully automated image analysis system applicable to ecological studies

*Tested on water fleas, birds and pollen*

Selwyn Hoeks, RU Nijmegen
Supervisors:

- Eelke Jongejans, RU Nijmegen
- Marjolein Bruijning, RU Nijmegen
- Chris van Turnhout, SOVON Nijmegen
- Beril Simacek, TU Delft

## Contents

## 1. Summary

In this report different methods were tested on their effectiveness for (semi)-automatically quantifying objects and organisms within images and video material. As model objects and organisms *Daphnia magna* (water fleas), *Sturnus vulgaris* (a bird species known as *Starlings*) and pollen of *Solanum carolinense* were used, since these organisms and objects require quantification in (experimental) ecology studies. Different methods for analyzing video material containing *Daphnia* individuals were developed in R, MATLAB and ImageJ. A specific method designed in R and MATLAB showed the most promising results, since it kept true detection over 90% and false detection below 5%, using video frames of lesser quality (containing 15 to 75 individuals). When using frames of a higher quality the same method was able to deliver 100% true detection and 0% false detection (tested with 70 individuals). However, it needs to be noted that this method is not yet fully automated and it needs to be validated on higher densities of *Daphnia* individuals. In addition to analyzing video frames, single images were tested as well. First a method was developed for analyzing images containing large flocks of *Starlings.* Results of R and ImageJ were compared and it was concluded that R showed the best results, although it is important to mention that ImageJ is easier in usage and thereby possibly a more practical solution. For single images the average percentages of detection appeared to be 96.4% in R and 91.4% in ImageJ, using images containing 89 up to 3564 *Starlings*. The false detection in these situations was around 5%. However, when analyzing higher densities (from approximately 5000 to 15000 individuals) false detection went up, while the total detection level decreased. Finally, pollen particles were used to test a specific method that could quantify the amount of pollen particles present; in this case only R was used to develop such an application. Here, 97.4% was the highest true detection level achieved when analyzing 10 images, each containing around 400 particles. It is concluded that the developed methods show promising results for (semi)-automatic quantification of *Daphnia* individuals, *Starlings* and pollen particles of *Solanum carolinense* up to certain densities. Additional research, maybe using more advanced methods as considered in the discussion, should be performed to see whether higher densities are possible to analyze and if these techniques can be fully automated.

## 2. Introduction

For effective conservation management, it is very important to provide accurate estimates of plant and animal population sizes, within certain time intervals. Moreover, many studies in laboratory conditions using model organisms require quantitative counts and measurements idem. Not only organisms require counting in biology, likewise different objects like pollen particles can have biological significance. So far many experimental ecology studies and field surveys are performed visually or manually which requires more time and is more prone to errors (Drake & Griffen, 2009). Since hand counting requires more effort and time, a limited area or total amount of animals can be considered for studies (Sirmacek *et al*., 2012). Therefore, ecological studies could be limited to a smaller scale. (Semi)-automatic counting techniques, that utilize image or video material, could provide new or at least in some cases partial solutions to these problems.

For example, experimental studies in which model organisms (e.g. *Daphnia*; small planktonic crustaceans which will be discussed at a later point) are used to detect a certain level of toxicity for a specific substance or abiotic variable. In these types of studies population growth or body size can be used as a proxy for health conditions. Thus, quantification and precise measurements are a must, which is mostly done by counting or measuring individual organisms by hand (Choi *et al*., 2014; Peerakietkhajorn *et al*., 2015). The same applies to ecological field surveys where reliable population counts are needed to follow population trends, predict possible declines in for the future and validate management. Some bird species occur in large single-species or multi-species flocks during certain parts of their year cycle, these flocks are, until now, regularly counted or estimated by hand. This is prone to relatively large counting errors, and should thus be improved if possible (Tou & Toh, 2012). According to Rappoldt *et al*. (1985) counting errors involving independent groups of waders can on average consist of 37% of the total amount. In the same study flying groups show an average counting error of 17%. These

counting errors will result in a counting error of 5% within the mud flat area of the Netherlands for general species (Rappoldt *et al*., 1985).
Microbial related studies rely more and more on modern techniques in order to quantify certain organisms or other objects too. In contrast to ecological studies, tools required for analyzing digital material like images automatically, in order to quantify present objects, is part of the basic toolbox in these fields (Daims & Wagnar, 2007; Schillinger *et al*., 2012).
Digital image and video analysis is a fast evolving field in computer science, thus there could be a high potential for data collection in different academic disciplines according to Burger & Burge (2008) and Eliceiri *et al*. (2012), (experimental)-ecology could be one of these disciplines. Numerous image-analyzing tools are available nowadays; many of them are free in usage (open-source) (Eliceiri *et al*., 2012; Schneider *et al*., 2012). However, in order for these (semi)-automatic counting methods to be effective, they should deliver fast, non-destructive and reliable counting methods, without requiring too much effort. It is important that even in cultures with high densities of individuals and with larger fractions of juveniles, estimates should be precise (Faerovig *et al*., 2002).

### 2.1 Report objective

Using different software applications various methods were compared to automatically count objects that differ in shape and size. The main goal was to develop a working tool that in the future could be applied in various situations to provide accurate quantification of certain objects and in some cases particle size measurements of them as well. To explore the most promising methods, three software tools most used in literature were compared, which were ImageJ, R and MATLAB. The first two tools are open-source and therefore free in usage. MATLAB however is a paid program and a paid version for professional usage will cost around 2000 dollars. Therefore it should provide sufficient advantages over R and ImageJ that are worth the investment.
The effectiveness of these tools was tested on two types of materials: digital images and video footage. Three types of objects, presented in either digital pictures or video material, were analyzed in order to make an estimate of the biological significance of the application of these tools and their effectiveness with objects from different size and shape categories. These 'objects' are discussed in more detail in paragraph 2.3 to 2.5. Using either ImageJ, R or MATLAB different pathways were tested in order to determine the most efficient and precise approach. Different methods for receiving photo and video material were used in the case of *Daphnia* in order to test the if differences in video and image quality affect the final results.

### 2.2 Reading guide

First the different model objects (*Starlings*, *Daphnia* and pollen particles from *Solanum carolinense*) will be discussed and their significance to the development of automating counting technics (paragraph 2.3 to 2.5). Second, different methods for analyzing the three model objects will be discussed in paragraph 3 'Materials and Methods'. First video material will be considered in paragraph 3.2, afterwards single images are discussed in 3.3. Then the different results of these approaches will be presented in paragraph 4 and finally these methods will be evaluated according to their specific results and practical use in the 'Discussion' (paragraph 5).

### 2.3 *Starlings* as test objects

The images of common *Starlings* (*Sturnus vulgaris*) were provided by volunteers working for SOVON (see appendix figure 4 to 8), a bird research center located in Nijmegen. In order to provide more precise population estimates, they are searching for an effective counting method for larger flocks of birds. At the moment, in these encounters with higher densities of *Starlings*, estimates are made by hand. Manual counting technics by volunteers are very difficult, because these birds in larger flocks are hard to be tracked independently by hand. Therefore, a duplicated count of the same bird that is actively moving within the group is easily made (Tou & Toh, 2012). When estimating the number of *Starlings*

using images, these duplicated counts can be avoided. These estimates can be made by hand, but by doing so the results will differ from person to person. Therefore, enormous differences in estimates of the number of birds in larger groups will be obtained. Hence automatic quantification methods can provide a more robust and fast way to deduce information about population developments. This is crucial, since effective conservation management relies on precise population estimates.

### *2.4 Daphnia* as test objects

*Daphnia* material was provided by the 'Animal Ecology and Ecophysiology' department within the RU Nijmegen. Marjolein Bruijning and Sven Kirchner were previously working on courting *Daphnia* individuals by utilizing ImageJ. *Daphnia* are planktonic crustaceans that are widely used in water toxicity/pollution experiments because of their characteristics: they have a constant fecundity across their adult life span, which starts at an average age of 11 days (depending on food conditions), they are easy to hold under laboratory conditions (Ebert, 2005) and they mostly have a fast reaction on little differences in their abiotic environment. Furthermore, *D. magna* are quite important to many fish species as they serve as one of their main food sources (Dodson,1988). In these experiments concerning *D. manga*, information about for example population size and individual body size can be crucial to either confirm or falsify a certain hypothesis. Experiments are only useful statistically speaking when the number of individuals is high enough and counting the number of individuals or measuring individual body size by hand in experiments which are in need for more statistical power, can be very time consuming and thereby expensive. Moreover, with the methods used nowadays multiple samples in time need to be harvested in order to estimate population and body size distribution, which amplifies the artificial death rate (Faerovig *et al*., 2002). Non-destructive measurements, ideally considering the whole population, could provide an answer to these problems.

### 2.5 Pollen particles as test objects

For the last example pollen particles where analyzed, these particles originated from *Solanum carolinense.* Lidewij Keser, working for the RU Nijmegen, provided pictures containing an average of 400 of these pollen particles per image. In this particular case the pollen particles were used to investigate in which proportion the genotype influences self-incompatibility, a mechanism that prevents self-fertilization (McClure *et al*., 1989), thus effecting the total number of pollen particles.

## *3. Materials and Methods*

All automatic counting technics used in this report using images or video material are based on the same few basic principles. All images can be regarded as large matrices containing the intensity values for each individual pixel; these values all represent a different intensity in a gradient from black to white. For example zero can represent black, white in this case will be represented by a value of one.
In order to create color images three of these matrices are combined into an array, each containing intensity values for either: red, green or blue. In this case two-dimensional matrices become arrays with three-dimensions. When these three matrices are overlapping one another, depending on de different intensities all colors can be created. Video material works according to the same principles. However in this case, there is not only one three-dimensional array; there are also multiple frames in a sequence in order to establish movement.

### 3.1 Simplest method; explaining basic principles

The simplest method to quantify objects would be to set a certain threshold to a single frame (grey-scale image), either using a single picture or a snapshot from video material, and afterwards count the number of remaining objects. In other words, a specific level of intensity is specified at which all the pixels for example below this value become zero and all the pixels above become one. A binary image will be the result. These pixels with the value of one now form the objects that need to quantified, the pixels with a value of zero describe the background. After determining the threshold intensity at which the image will be divided into the aforementioned new values, the objects are now represented as patches of pixels with a value of one, it is possible to group the pixels that form one single patch and count the number of formed groups. Figure 1 gives an overview of how this process will look like in two simple steps.

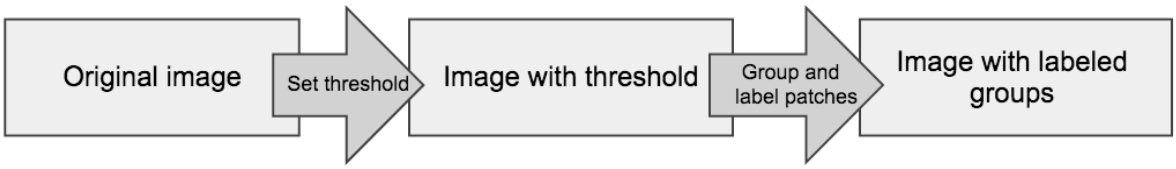



**Figure 1:** A flowchart showing the simplest analyzing method. First a certain threshold is set, than patches are grouped and labeled. These labeled groups are counted resulting in the number of objects present in the image.

In most cases however only setting a threshold is not sufficient, because of the presence of objects that should not be included in the total count of objects of interest. Moreover, not only other objects are the reason for the need of a more advanced method, also uneven lightning conditions in pictures could disrupt such a simple approach, as well as artificial background noise. In the following paragraphs some examples of how these more advanced methods work are explained.

### 3.2 More advanced method using video footage

In the first more advanced method video material was used and converted to multiple frames. Compared to the usage of a single image, which will be discussed in the paragraph 3.3 and 3.4, it is possible to detect certain moving objects, thus making it easier to eliminate other objects that are not of interest to the researcher. In order to get a more complete picture of the total process an overview of this more advanced method is shown at the end of the complete explanation (see figure 6).
Video material can be converted into image sequences in MATLAB itself; R and ImageJ however need external software in order to extract multiple frames from video material. In this example KMPlayer is used, a free and easy to use video player that converts video material into an image sequence. In the following example *Daphnia* are used to estimate how accurate this method is. In this approach, ImageJ, R and MATLAB follow roughly the same principles; therefore one method, in this case R, was selected to illustrate all individual steps. Differences in the tools used in this method are considered in the discussion. High quality frames that were made for this report containing 70 *Daphnia* individuals

illustrate all results in paragraph 3.2. In the final result, next to these higher quality frames, a series of lower quality videos ranging from 15 to 75 individuals was used. In the following method only one color layer was used for further analysis: the first color layer, the layer that represents the pixel intensities of the color red. Different approaches for dealing with color images will be discussed in paragraph 5. These higher quality frames differ from the lower quality frames in resolution (4288x2848 vs 1024x576) as well as better optimized lightning conditions (see appendix figures 1 to 3), for a higher contrast between *Daphnia* individuals and their background. Appendix figure 1 and 2 show snapshots of these lower quality frames originating from video material, Appendix figure 3 shows an example of a higher quality frame.

### 3.2.1 Creating background image

When combining multiple images in time, it is possible to create a so-called background image; this image should be an exact copy of all frames, but without the moving objects (see figure 2). Such an image can be created by calculating either the mean, minimum/maximum intensity or the median for each single pixel over multiple frames as suggested by Mallard *et al*. (2013) and Kobayashi *et al*. (2008). Since the objects, or in this case *Daphnia* are moving and therefore are not in the same place in every single frame they will be filtered out. In this report the maximum value was the most appropriate to use, because of the lightning conditions. In many cases the median or mean are recommended because using this approach will results in a more precise background image. However, a longer time span is required when using the median or mean for creating the background (Mallard *et al*., 2013). A longer time span is for example needed to eliminate the slow moving objects in the background image.

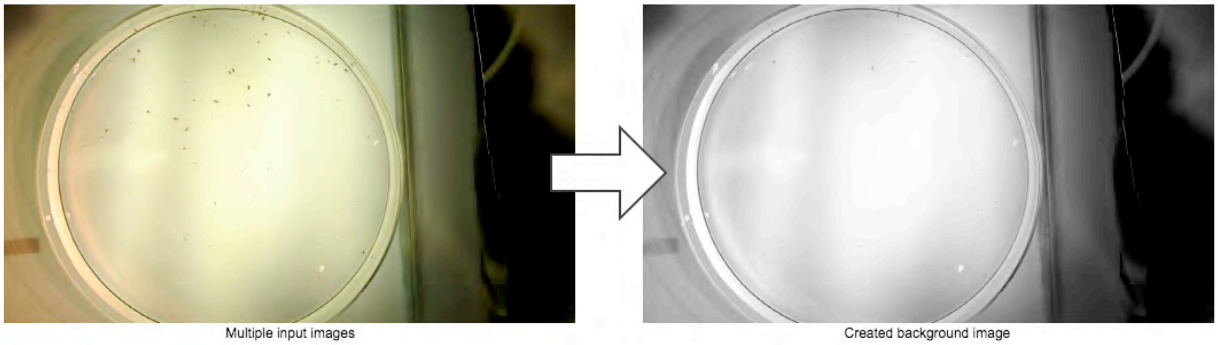



**Figure 2:** Example for creating a background images by using multiple input frames. On the left an example of one of the input images is shown, on the right the final background that was developed by using the multiple input images is presented. Only one of the RGB layers was used in order to produce the background.

### 3.2.2 Selection the region of interest

A region of interest (ROI) is an area that includes all objects of interest; in this particular example all *Daphnia* individuals are within the petri dish, meaning the ROI in this case is the area inside the petri dish. In ImageJ it is possible to select the ROI by hand, due to its graphical interface providing tools that let the user drag a selection window in various forms across the image. In R and MATLAB however this is different; these tools require coordinates and the right formulas or functions to select a certain region on one or multiple images. This leads to both disadvantages as well as useful benefits. One of these disadvantages is that it simply makes it harder and more time consuming to manually set a ROI. But on the other hand when it is possible to require the coordinates of the ROI together with the right shape characteristics, the ROI could be selected automatically. This makes the selection easier, less depended on user input and perhaps more precise. In the following example, as mentioned earlier, a circle shaped ROI needs to be selected. A method is developed that works in the case of a ROI with such a shape. The method is meant to illustrate how a specific solution can be formed in order to deal with selecting a ROI automatically over multiple images. Possible solutions for handling ROIs of different shape categories will be discussed in paragraph 5.

In this specific case, setting a ROI is necessary to eliminate reflections caused by the glass of the petri dish. Moreover, any other object moving outside of the region of interest is automatically ignored.
In order to gather the coordinates for the ROI the created background image was used. When setting the right threshold to the background image only the darker parts can be selected, including the edge of the petri dish (see image on the right in figure 2). In this example the threshold is set manually, the high contrast between the darker edge of the petri dish and lighter background makes setting a threshold manually easy. Setting a specific threshold will result in a binary image, in this case all darker parts of the former background are now represented by ones, the remainder is filled with zeroes (see image on the left in figure 3). To save time in R the functions computeFeatures.shape() (EBImage package) and PatchStat() (SMDtools package) are then used to select certain objects that remain after setting the threshold, in this case the thin white line that represents the petri dish (see figure 3). All the other objects that remained after setting the threshold have a much higher area/perimeter ratio; hence they can easily be excluded. After achieving this, the remaining pixels that represent the ROI can be used for selecting the proper ROI within the multiple input frames. There are two different approaches considered for this next step:

- A simple method could be to add the image obtained to the input frames. Because only the white circle (intensity values of 1) is present on a black background (intensity values of 0) it would not alter the input frames, only the white circle is drawn on top of the petri dish perimeter. Everything outside of this white circle will then be filled with zeroes, thereby removing the area outside the ROI.
- Because in many cases, including this specific case, the circle is not fully closed, filling up the outside could be difficult. Therefore a function is needed for determining the coordinates for the center of the circle along with information about the radius, in order to draw a perfect closed circle on top of the existing one. In each row the maximum and minimum column number for which its content is equal to 1 is calculated. The mean of these maximum and minimum values are then derived. If these coordinates for each row were to be plotted they would divide the white circle in two halves by a vertical line. When all these values for each row are averaged they represent the x-coordinate for the center of the circle. By repeating the same steps for each column the y-coordinates will get known. Now the x and y coordinates are computed, it is possible to calculate the radius by searching for the distance from the center to the outsides of the white circle. The formula: $(x-l)^2+(y-k)^2=r^2$, where *l* and *k* are the x and y coordinates and *r* is the radius, could be used to plot the circle. However, to make thing less complicated the function drawCircle() (EBImage package) in R was used the draw these circles on every frame, than floodFill() (EBImage package) filled every cell outside of the circle with zeroes.

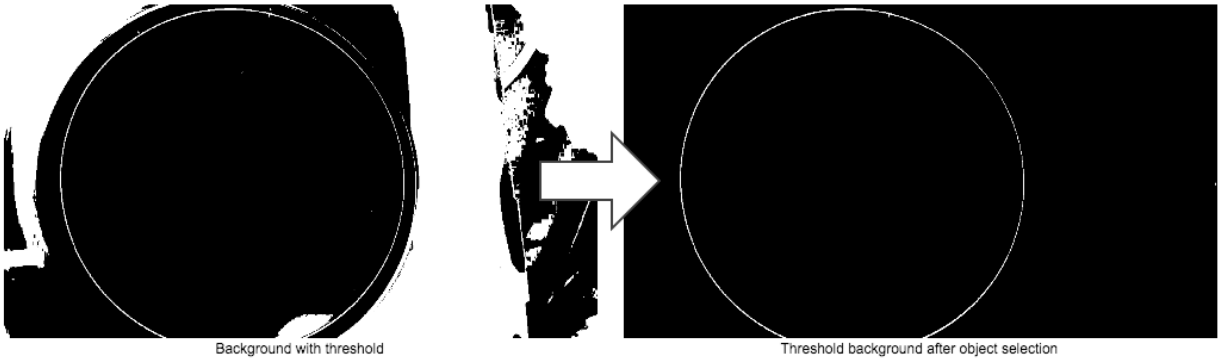



**Figure 3:** Example for locating the ROI by using first a threshold on the background image (left image) and later manipulate this image by selecting for objects with a certain shape, resulting in the image on the right.

### 3.2.3 Background subtraction and setting threshold

In this specific method randomly selected single input frames were subtracted individually from the background image without a threshold. In this way the moving objects were made negative compared the motionless objects. In the perfect condition all these motionless objects should be presented by zeroes, because they were present in the background image as well as every single frame that was used. Next the ROI coordinates discussed in the previous step were used to exclude any mirroring effects against the glass of the petri dish (see figure 4). Setting a threshold to make a binary image is easy since every cell containing a value below zero is considered as a moving object. It happens automatically for this method, but sometimes some manually adjustments can help to filter out unwanted particles.

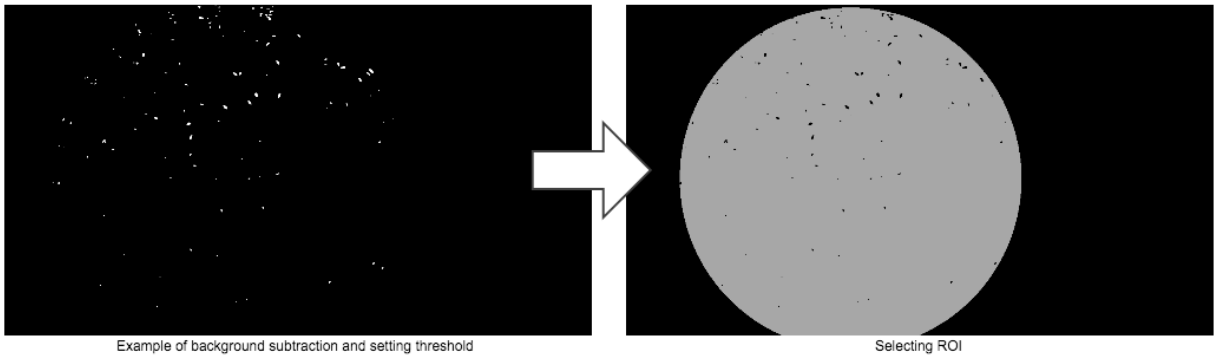



**Figure 4:** On the left an example of a background subtraction to one of the input frames is shown after setting a threshold. On the right the previously discovered ROI coordinates are used to select objects contained by the ROI only.

### 3.2.4 Final steps

The final steps consist of first calling properties of the objects that remain after the previous steps, by using computeFeatures.shape() and PatchStat(). After the shapes of objects are known it is possible to select and later exclude noise. In this rapport the selection is first made by setting a minimum and maximum size for allowed objects, afterwards a second filtering method was added which uses a perimeter/area ratio to exclude any remaining shadows casted from the petri dish (these are oblong and therefore have a higher perimeter/area ratio). The final step is to compute the number of objects remaining in the final image (see figure 5).

All steps in 3.2.3 and 3.2.4 are repeated for every single frame, resulting in multiple estimates of the total number of objects present. In some cases at certain frames objects can be close to each other, meaning that two objects could be counted as being one. By using multiple frames, thus making it possible to take the average total number of objects present in each frame, this change can be reduced. Finally a frame is selected with a total number of objects that is close to the average number of objects over multiple images, this specific frame is then used to illustrate the end result and possibly calculate other matters of interest, for example body size.

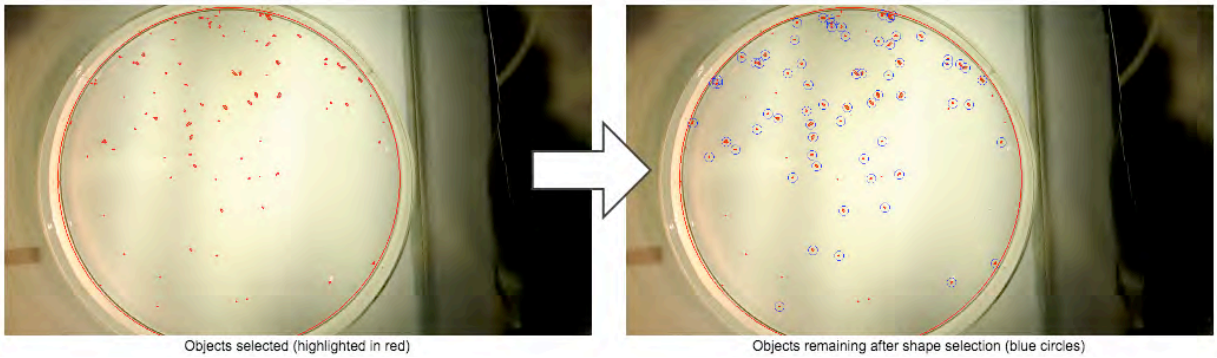



**Figure 5:** Objects selected in ROI after background subtraction (left image), Final results; objects are selected based on shape (right image). All calculations were made using only one color layer, only to visualize the final results a color background was used of the original image. In both images the red dots indicate moving particles and the selected ROI, the blue circles represent the selected moving objects that meet certain shape requirements.

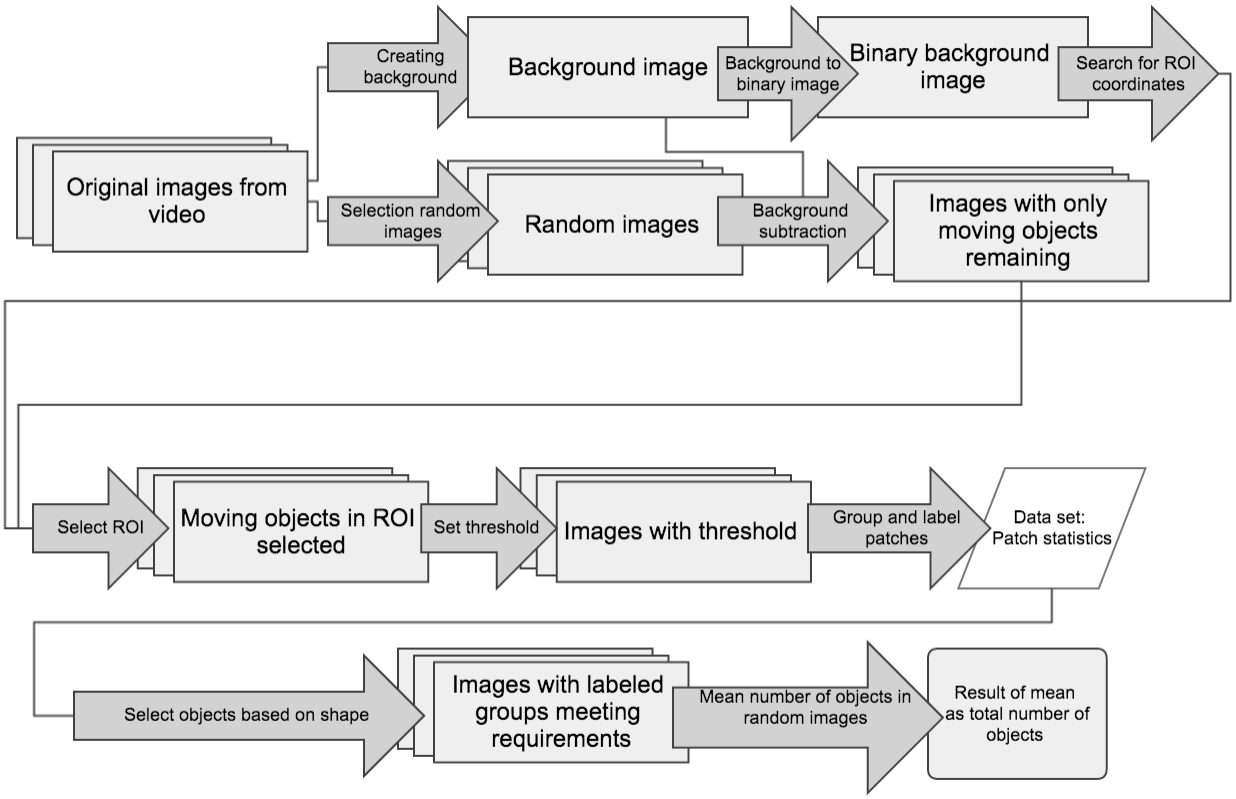



**Figure 6:** A flowchart presenting a more complex method of analyzing particles based on movement. First the background image was created by selecting the maximum value for each pixel over multiple input frames in time and creating a new image out of these maximum values (a certain amount of frames extracted from video material was used). This background image is then transformed into a binary image by setting a threshold, next this binary image can be used to search for the region of interest (ROI) by selecting for certain objects based on shape within the created binary background image, in this case the shape of the petri dish. After the coordinates of the ROI are known, they are used to select the ROI on every single input frame. Next the normal background is subtracted from every single frame, later a specific threshold is set on every frame that resulted from previous background subtraction. The last few steps are used to count every object that is contained by the ROI; again shape characteristics are used to select only the objects of interest.

### 3.2.5 Shortcomings when using ImageJ

Unfortunately, it is difficult to separate remaining patches according to size and shape in ImageJ. It is possible to select certain properties and only select patches within these properties, but it is nearly impossible to continue with these results by using only a single macro. Within R and MATLAB it was possible to create a table containing all the shape characteristics for every single patch in more detail, all these patches were also labeled with a unique value. Both these advantages made it possible to select objects by multiple size and shape characteristics and from there continue with this specific data.

## 3.3 Advanced method for analyzing single images

Because of the fact that video material of larger flocks of *Starlings* are often hard to come by or often lack in quality, the previously mentioned method is not relevant to the application on bird material. The same goes naturally for motionless objects, such as the pollen particles.
Therefore a more advanced method is necessary in order to process single images with a higher amount of precision, which will be discussed in the next paragraph.

### 3.3.1 Application in R

This particular method was done in R, but most principles were also applied in ImageJ. Both analyzing tools follow roughly the same steps, but all illustrations in this example were made by the usage of R. Differences in the tools used in this method are considered in the discussion. MATLAB and C++ programming were later used to search for future possibilities and will be reflected on in the discussion as well. In order to get a clear overview of this particular method an flowchart was made, see figure 14.

The images containing *Starling* flocks were divided into different groups, based on their level of difficulty to analyze. These groups can be distinguished by certain fixed characteristics.

- In group one pictures with almost only isolated *Starlings* are present, there is also no influence from any other objects besides *Starlings* (no presences of trees or building etc. in the background). The background itself (sky) is as clear and monotone as possible. Example images from the first group or first level of difficulty can be found in the appendix (appendix figure 4).
- In the second group *Starlings* occur clustered, and other objects besides *Starlings* such as trees or building can be present. Additionally, the background itself can consist of multiple colors for example by the presence of clouds or a setting sun etc. Example images from the second level of difficulty can be found in the appendix (appendix figures 5 to 7).
- In the last group the density is much higher, meaning it is nearly impossible to count them by hand from a photo, due to heavily clustered *Starlings* and a high amount patches containing more than two birds. The influence of other objects besides *Starlings* is much higher in this group, which means that *Starlings* can be partially in front of or behind objects. Resolution is generally lower making detection of *Starlings* difficult even by zooming in. This is due to the fact that these birds can be presented by single pixels. Example images from the third level of difficulty can be found in the appendix (appendix figure 8).

This first method is focused on images that were divided in the first and second group, because these require a less complicated approach. The last group is considered in the discussion, because these images seemed impossible to analyze using approaches that are used in R and ImageJ.

### 3.3.2 Dealing with color images

Depending on time and weather conditions the sky in the input images can have a wide variation of colors, therefore certain adjustments or selection methods have to be used to separate the background from the objects of interest. However, in some cases the background does not simply consist of a clear or disturbed sky, it is also possible to encounter objects like trees or buildings etc.
In the method involving the usage of R the clearest color layer is selected (one of the RGB-layers with the highest contrast between the background and the patches of *Starlings*), thus reducing the amount of background noise significantly. In the discussion (paragraph 5) different other methods for dealing with color images will be examined. In most of the example images used in this report the clearest layer is related to the lowest diversity in pixel intensities, due to the fact that higher diversities will be more likely to describe unevenness within an image. Therefore, in this method the clearest layer is selected by looking at certain statistics of the pixel intensities, for example certain measurements for the variation of pixels values. In the following figure (figure 7) the three histograms are made for a specific example picture, each histogram represents variation in pixel intensity within a certain color layer (red, green or blue).

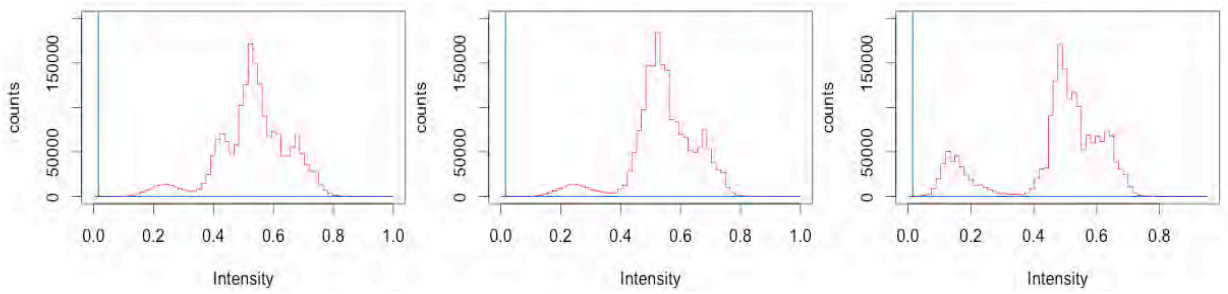



**Figure 7:** Histogram of all pixel intensities of three different color frames, with the red color layer on the left, the green layer in the middle and the blue layer on the right.

As can be seen in the figure above the green layer (second histogram) shows to lowest diversity in pixel intensity, this can also be supported by when comparing the standard deviations, the green layer or second histogram has the lowest standard deviation (0.119, 0.109 and 0.162 respectively). Therefore the green layer seems to most promising layer to continue with. When comparing these different layers image-wise this also indicates to be the most usable frame (figure 8). After selecting one layer, a grey-scale image remains.

**Figure 8:** Three different layers of a RGB-image containing a flock of *Starlings*, with the red color layer on the left, the green layer in the middle and the blue layer on the right.

ImageJ handles color images differently, due to the fact that it is possible to analyze all three frames at the same time and thereby set a threshold based on color and intensity simultaneously. However, the threshold has to be set manually as will be explained in the next paragraph, compared to R which can be applied semi-automatically, meaning some manual adjustment may be necessary in several occasions.

### 3.3.3 Setting threshold using R and ImageJ

After automatically selecting the most usable color frame a threshold is set to exclude certain objects in the background, for example a sky filled with clouds surrounding the bird flock. This threshold is selected automatically within R by searching for the most common values. The lowest most common value is used as a threshold level. In this particular example the lowest value of the 100 most common values was used. In some cases some small adjustment is necessary in order to get the perfect threshold level, which includes all objects of interest.

This lowest most common value is generally a value that represents the darkest area within the background. Objects of interest (*Starlings*) are often even lower in value (darker in color), therefore this lowest most common value can be used as a threshold level (all pixels below this value will be changed to ones and all pixels above to zeroes). Naturally, a binary image will be the result after setting a threshold on a grey-scale image (see image on the left within figure 11).

Setting the threshold in ImageJ happens manually and is different for each single image. Which of course takes more time, but due to ImageJ's interface changes can be seen in real time, meaning it is easy to adjust when necessary. When using the 'Threshold Color' option within ImageJ three sliders can be used to determine the right threshold values for each. First the 'Hue' slider can be adjusted to select the right color range, the 'Saturation' and 'Brightness' slider controls can be used to select the correct pixel intensity values.

### 3.3.4 Exclude unwanted objects

In many cases there are objects such as trees or buildings remaining within the image. In order to deal with these objects and exclude them before continuing, a layer is automatically created that contains all these unwanted objects (result shown in the image on the right within figure 10). To do so, fFirst 0.1 is added to the normal threshold level that is set to a specific image automatically. By adding 0.1 to the normal threshold level, objects within the image will become larger and more connected. This will happen due to the fact that more pixels with higher values surrounding objects, that consist of pixel with a lower value, will be included in this higher threshold. By doing so small artifacts surrounding the unwanted objects will be included in the layer that contains all unwanted objects (see figure 9)

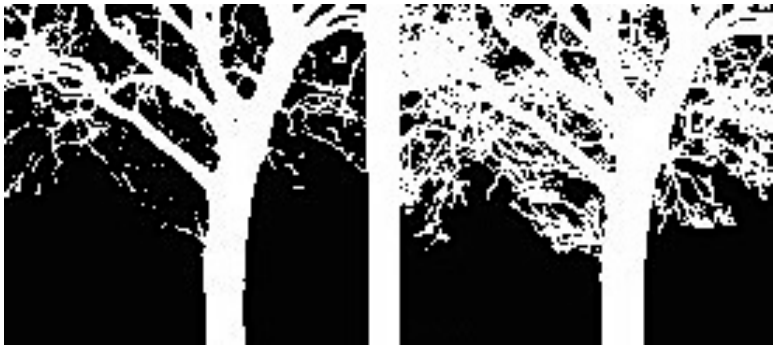


**Figure 9:** On the left a part of an image with a normal threshold level set, on the right the same part of this image is shown only the threshold level is 0.1 higher. The image on the left shows artifacts surrounding this particular object, a higher threshold will have fewer artifacts as can be seen in the image on the right

After setting the threshold, objects are selected based on shape and size, making it possible to only select unwanted objects in the background and exclude the *Starlings* for the time being (see image on the right within figure 10). This step was only developed in R, ImageJ takes a more simple approach as will be explained in 3.3.6. Within R everything explained in 3.3.4 and 3.3.5 happens automatically.

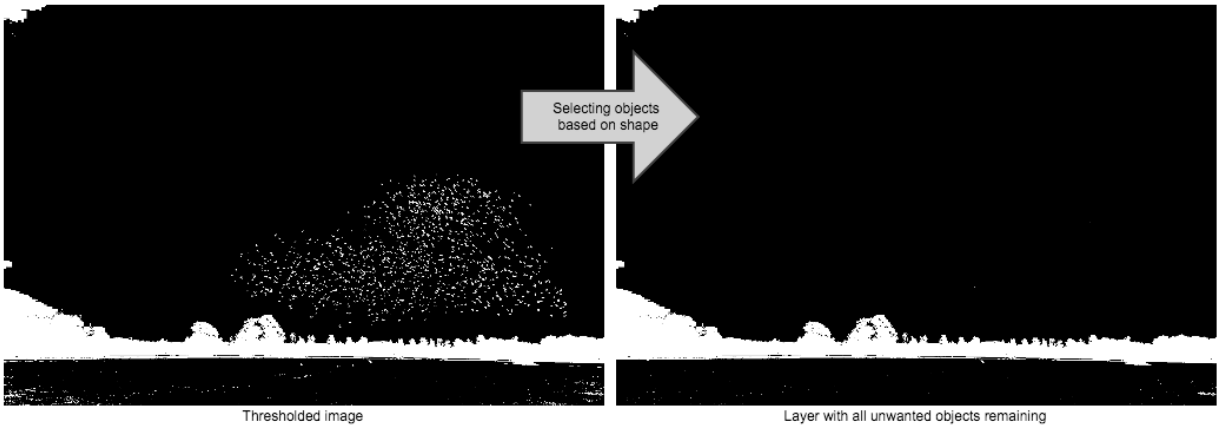



**Figure 10:** Input frame with a specific threshold level set (on the left), all unwanted objects selected based on shape and size (on the right).

### 3.3.5 Subtracting all unwanted objects

By not adding 0.1 to the automatically detected threshold level the “normal” threshold level is set on an image as discussed earlier in paragraph 3.3.3. A lower threshold, this case considered as a normal threshold, will keep objects from clustering together, something that was necessary in the creating the layer containing all unwanted objects. This previously created layer containing all unwanted objects is than subtracted from the input image with the normal threshold level set to it, resulting in an image with only *Starlings* remaining (see figure 11).

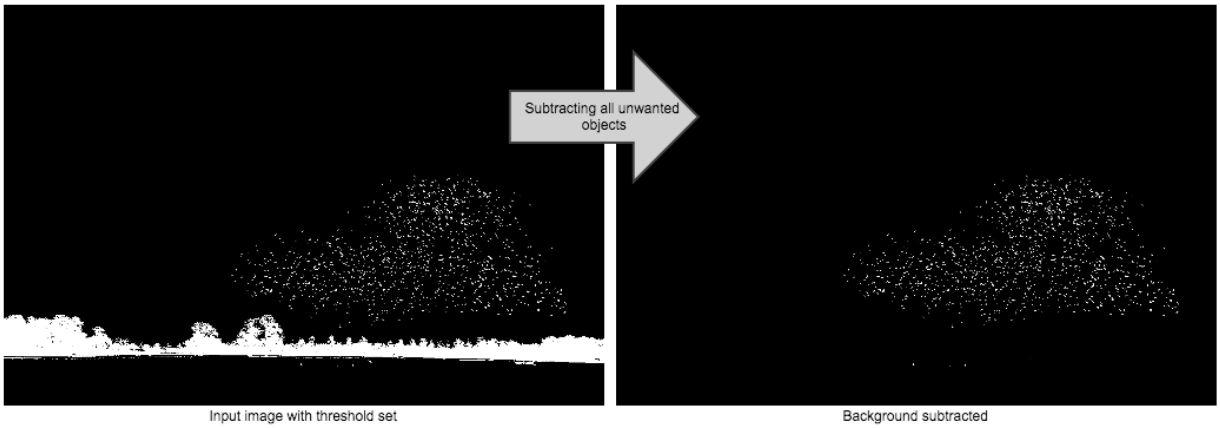



**Figure 11:** Input image with normal threshold on the left, earlier created layer containing unwanted object from input image with normal threshold on the right.

### 3.3.6 Object selection

After all unwanted objects are subtracted and the result is sufficient, remaining noise can be excluded based on size or shape characteristics. Moreover, clustering of *Starlings* is inevitable, this means that there are patches detected containing one single *Starling* and there are patches that represent two or more *Starlings*. Because there are many different poses in which a bird can be captured on the image, it is very difficult to make a precise assumption of shape and size differences between patches of single or multiple *Starlings*. Likewise it is nearly impossible to state how many *Starlings* there are in a patch containing multiple *Starlings*. Therefore a conservative assumption is made: all patches over a certain amount of pixels are on average two *Starlings* (not more), thus dividing all patches in only two groups. To detect the border between the two, knowledge about the size distribution is needed (figure 12), therefore a certain amount (in this method 35, a value that will keep the speed of processing high and provided enough information about the size distribution) of most common patch sizes are investigated, finally the maximum value of these most common values is used the define the margin boundary between patches containing only one *Starling* and patches with two *Starlings*.

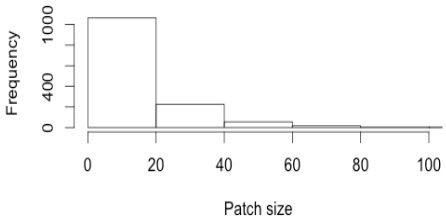



**Figure 12:** Patch size distribution of all remaining patches after subtracting all unwanted objects in the background, as shown in the previous example.

In this particular example the highest most common value appeared to be around the size of 40 pixels. When looking at the distribution of all patches within the same image this seems to be a reasonable number, while patches of single *Starlings* are way more common than patches of two or more, see figure 12 and 13. In all other images analyzed this method for distinguishing single or multiple *Starlings* in one patch deemed to be the most precise method amongst others tested. In this particular example that means that patches of single *Starlings* are described by patches smaller or equal to 40 pixels, patches larger than 40 pixels describe patches containing two *Starlings*. This process happens automatically within the R script, but there is an option, which makes it possible to adjust the size for dividing these two different groups.

**Figure 13:** End result with all recognized objects marked; objects marked red are seen as patches of single *Starlings* and objects that are marked blue are patches seen as two *Starlings*.

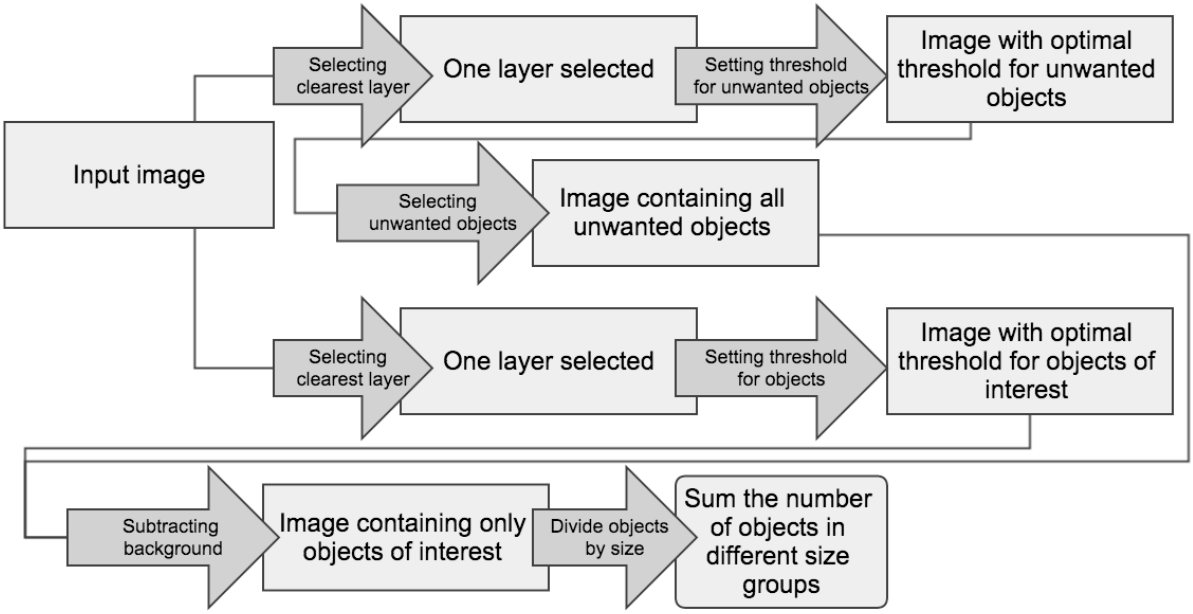



**Figure 14:** Flowchart for analyzing single pictures containing *Starling* flocks. The clearest input frame is first chosen and all the objects in the background (if there are any) are selected by their shape in this particular frame, a binary image containing all the background objects is then subtracted from the input image in a binary format. Objects remaining are grouped in patches containing single *Starlings* and patches containing two or more *Starlings*.

### 3.3.7 Application in ImageJ

The method developed in ImageJ is fairly simple and straightforward. Characterizing objects by shape or size is not possible in the way it is available in both R and MATLAB, meaning there is no possibility to continue with any collected object characteristics inside the tool itself. Consequently, objects for example in the background cannot be extracted from the original image automatically. Therefore, setting the ROI by hand was the best option to exclude unwanted objects. This requires the user to draw a line around the objects of interest; everything outside this boundary can be excluded. Secondly the image is converted into an 8-bit image (greyscale image), making further analyzing processes easier. After the image is converted into a greyscale image a tool called 'Find Edges' is used. As the name of this approach already suggests it can be used to find edges of objects in digital images, while these edges are mostly represented by changes in image brightness. Multiple methods exist for finding these edges, ImageJ uses a 'Sobel edge detector' to highlight sharp changes in intensity in an image. The Sobel edge detector tool divides the image in pairs of 3 x 3 pixel grids, called convolution masks, the first masks are used to estimate edges in the x-direction, the second masks are used for estimations in the y–direction (Vincent & Folorunso, 2009). The final image is produced by combining the two derivatives using the square root of the sum of two grids. Different methods for finding edges will be considered in the discussion. After the edges are detected the images is converted into a binary image, this makes the recognition of objects possible, the option 'Analyze Particles' is then used to count the number of objects remaining. This option also provides to possibility to export a table which includes all sizes of counted objects, therefore it deems feasible to sort these patches according to size in an external program like Excel into two groups, although this makes the total process even more complicated. One could also set a certain size beforehand and only select objects greater or smaller in size, but by doing so the user has to know something about the size distribution in advance. In the end, due to complications and time restrains, all pictures in ImageJ were not analyzed by grouping objects according to size. Therefore, all patches counted were seen as objects containing only single *Starlings*; this could lead to a major underestimation of the total amount present.

### 3.4 Application on different type of objects; pollen particles

A variation of the script that was used to analyze single frames containing *Starlings* was written to analyze different objects of interest, for example pollen particles. This variation also used some features from the script used to analyze *Daphnia*, for example the ROI was used to select an area in which the pollen particles were present. Since the ROI has a similar shapes (circular in this case), the function could be applied on these images as well.

Because it combines the same principles as discussed earlier, this variation will not be discussed in detail, a flowchart (see figure 15) of the steps taken should provide a sufficient explanation when combined with the previous explanations. However, the results will be included for the discussion on the effectiveness of these methods. An analyzed pollen particle example is shown in figure 16.

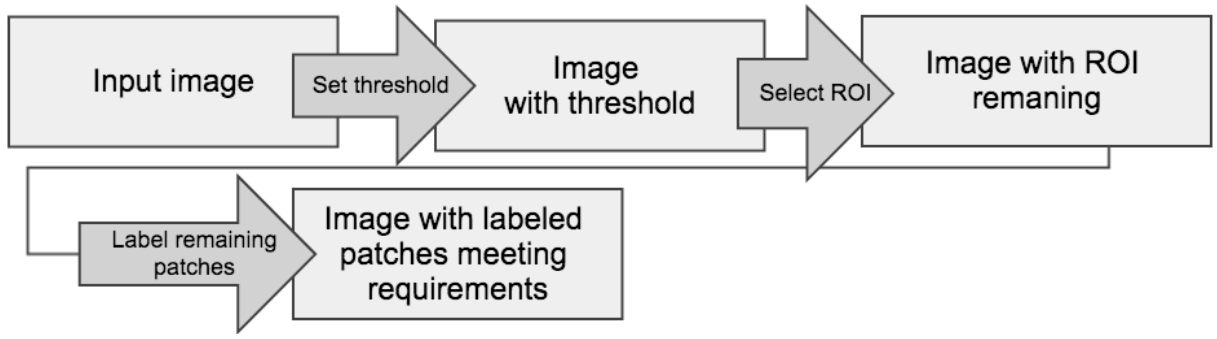



**Figure 15:** flowchart for analyzing single pictures containing pollen particles. First a threshold is set automatically to the original input image, later the ROI is selected automatically by using shape characteristics within the created binary image. This ROI is than used to select all the remaining particles within its boundary. These particular objects are than divided into shape categories, in order to exclude any particles that do not meet the shape requirement of being a pollen particle.

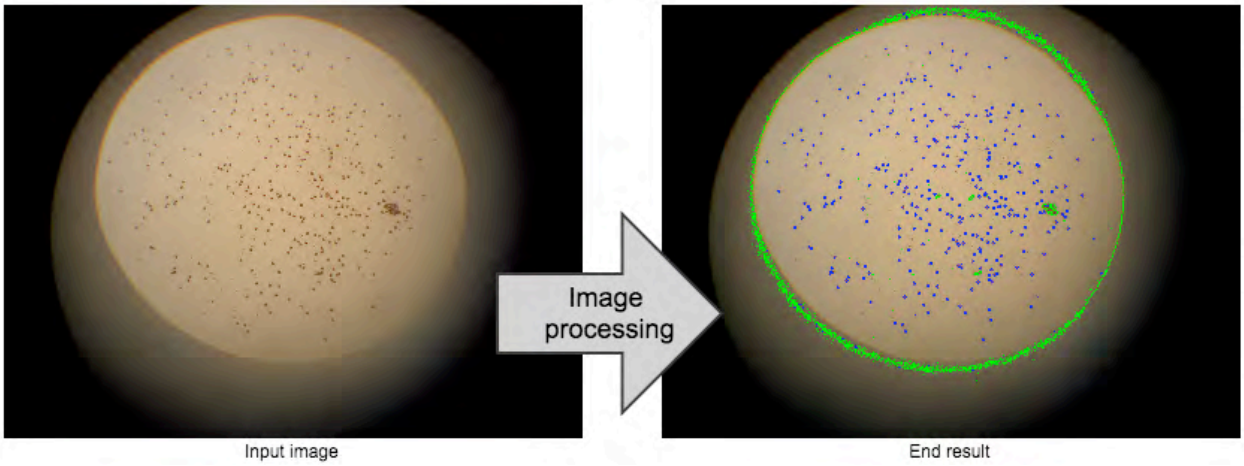



**Figure 16:** Example of an analyzed pollen particle image in R. On the right the clean input image is presented, the picture on the left shows the end result; the ROI is outlined in green, particles that do not meet the shape requirements are also painted green. Finally the recognized pollen particles are marked blue.

## *4. Results*

While different tools were able to make an almost perfect estimate of the number of objects present in an input image, it is necessary to validate the result by taking a look at the actual objects that are counted or not counted. For example, in some cases to number of objects found by using a certain method could be a perfect match with the real number of objects present, but due to background noise or other objects present this could be a misperception. Therefore, most results will be discussed by using terms such as false and true detections, meaning an object could be false detected when an object is marked but does not have the interest of the researcher or an object could be true detected when it is an object the researcher is interested in and it is marked by a certain image processing method. For example in a specific image five *Daphnia* are present, a certain script counts five objects, but when visualizing the results only four *Daphnia* are marked (true detected), the fifth object marked could be some artifact in the border of the petri dish or a dust or sand particle in the water (false detected). There are several reasons thinkable why the fifth *Daphnia* is not marked and true detected, for example it could be too small or there could be a light reflection on the water surface overlapping the contours of this fifth *Daphnia* individual. For these reasons it is necessary to know the distribution of false and true detected objects within a total count, in order to validate the method used.

### 4.1 Results *Daphnia* using video material

The results discussed in this paragraph are mostly produced by the usage of R, because of time restrictions the exact same analyzing method was done in MATLAB, providing the same results. Advantages and disadvantages between these different tools are discussed later. Due to limitations in ImageJ not every step in the method used for analyzing multiple frames of *Daphnia* footage is completed in such a way possible in R or MATLAB. Therefore, these differences will be included in the results.

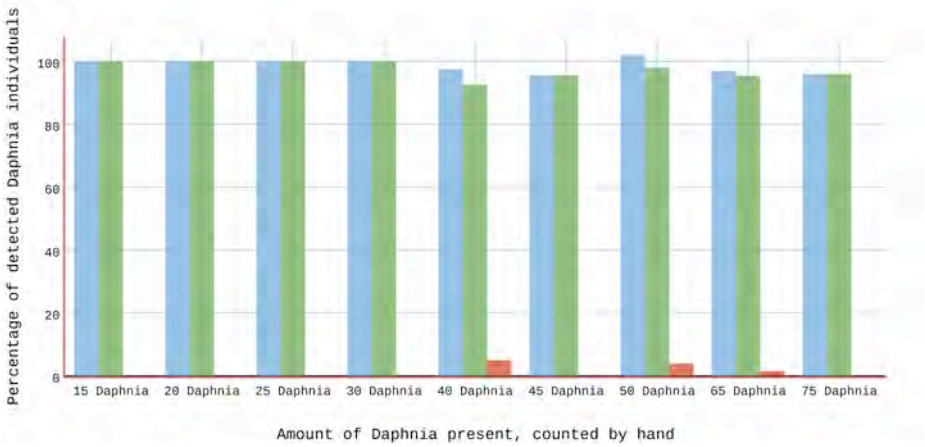



**Figure 17:** Results obtained in R by using multiple frames of Daphnia footage of a lower quality as discussed in paragraph 3.2. In blue the total percentages of *Daphnia* individuals detected. Green bars represent the percentage of true detected individuals. In red the percentage of false detected individuals is shown.

As shown in figure 17 the amount of false detected objects is generally higher when the total amount of *Daphnia* inside the petri dish is higher, the opposite is noticeable with the proportion of true detected objects. However, all levels of true detection stayed above 90%, while false detection levels did not become greater than 5%.

In ImageJ the same amount of pictures is used to create the background, the estimation of the number of *Daphnia* nonetheless is not made by using multiple frames, due to certain restrains within ImageJ (as discussed in paragraph 3.2.4). Instead of using multiple frames for this step a random frame is chosen, therefore larger divergence can be expected. Moreover the ROI could not be selected automatically, resulting in a higher deviation when setting the ROI by hand.

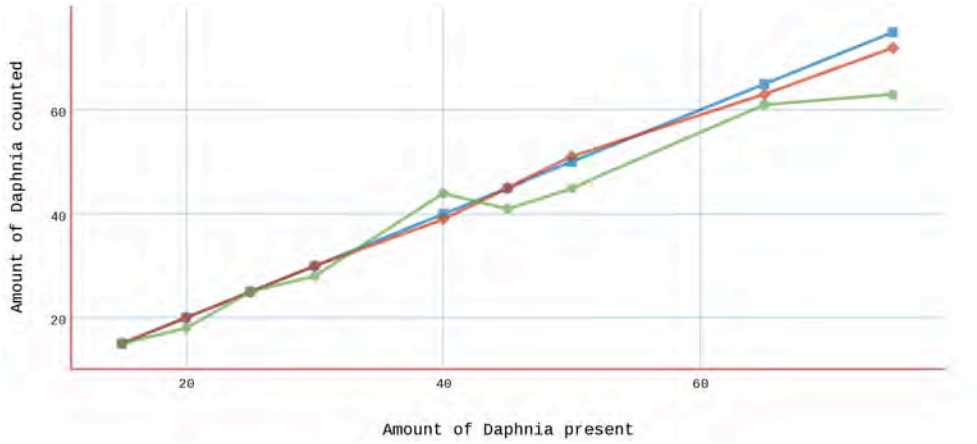



**Figure 18:** Amount of *Daphnia* counted plotted against the amount of *Daphnia* present for three different methods. First the blue line represents the results of counting *Daphnia* by hand, for this method the assumption is made the numbers obtained when counting by hand rely for 100% out of true detection values. The red line represents the method used in both R or MATLAB, only the total amount of individuals are presented in this result, false detected individuals will be discussed later. The green line shows the results achieved in ImageJ, again only the total amount of *Daphnia* individuals are shown.

As can be expected, figure 18 shows that ImageJ is less precise in estimating the total amount of *Daphnia* present, especially when it comes to greater numbers there is a case of underestimation. In the figure above only total amounts of detected individuals are presented, when comparing the false detected objects inside either ImageJ or R (or MATLAB) they do not differ much (see table 1 in the appendix). However, when trying to achieve true detected numbers closer to the real number of *Daphnia* to make up for the continual underestimation especially when using ImageJ, naturally the number of false detected objects increases, making the results obtained in some cases unusable for further research. In all different image sequences containing different amounts of *Daphnia* individuals, the same threshold value is used. When analyzing 40 individuals, a peak in the results produced by ImageJ is noticeable, in this case the number of false detected individuals was higher than average. This can be explained by the difference in lightning conditions combined with the method used, which involves setting the same threshold in ImageJ to all different sequences. Within R however, the selection of remaining objects, after setting a threshold, by shape and size characteristics will prevent the occurrence of higher false detection levels. In the case of 40 individuals the level of false detection went up to 15% when using ImageJ (6 objects were false detected), in all other cases the false detection level was below 5%. In all cases when using R the false detection level stayed below 5% as shown in figure 16. Better lighting conditions and a larger petridish, will improve the results.
More space, in this case meaning a larger petri dish makes cluttering of different individuals less likely, especially with higher numbers of *Daphnia* present. Naturally, the resolution and different other image quality aspects, such as sufficient lightning conditions in order to get enough contrast between the objects of interest and the background, did improve the end result as well. When testing new camera setups, meeting these new requirements, identifying 70 *Daphnia* individuals was not a problem; a true detection percentage of 100% could be obtained, while false detection remained at 0%. The example video frames used in paragraph 3.2 were made by the usage of this improved camera setup. Additional testing on different total amounts above 70 individual will be needed in order to know that these detection levels can be maintained with even higher densities.
The method used for these results worked almost fully automatic, selecting the ROI was automated as well as setting a threshold and counting the remaining objects. However in some cases slight adjustments were needed, especially to setting the threshold.

## 4.2 Results *Starling* flocks using single images

The first results presented are obtained with R. ImageJ was easier in usage, even though it lacks certain features, therefore the results obtained by R and ImageJ will be compared in the next paragraph. In this section only the first and second level of difficulty considering images are discussed, the third level will be reflected on in the discussion (paragraph 5.2). The images used to obtain these results were selected at random out of the aforementioned two groups; the images selected are shown in the appendix (appendix figure 4 to 7)

Figure 19 shows the true and false detected objects over thirteen different images as a percentage of the total amount of *Starlings* present in a specific image. The total amount is derived from counting the total number of birds by hand from a photo, which could also deviate from the actual real number of *Starlings* present, due to human error. As can be seen in figure 19 the percentage of true detected *Starlings* (in green) drops slightly when a larger total amount of *Starlings* is present in the images. Moreover, pictures with greater total amounts show frequently higher false detection percentages, whereas images with a lower total amount often do not show false detection at all.

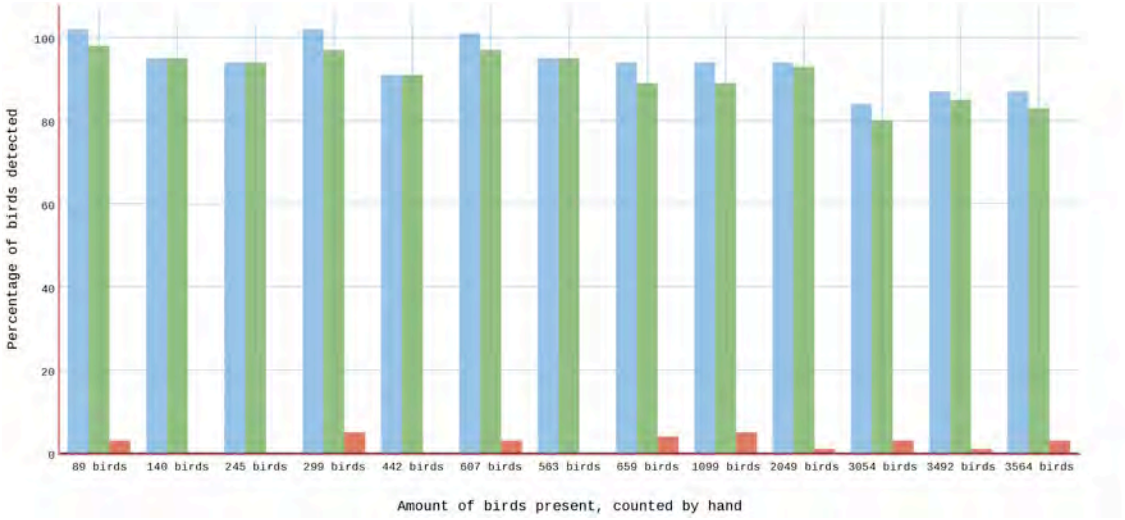



**Figure 19:** Thirteen images analyzed, each image containing a certain amount of *Starlings*. For all images the total amount the true and false amount of detected *Starlings* was determined and presented as a percentage of the total amount of *Starlings* counted by hand. Total detected percentages are presented in blue, true detected percentages are presented in green, false detected percentages in red.

In order to compare the method used in R with the more simple approach in ImageJ, more images were analyzed. Due to time restrains no separation was made in true or false detected objects, instead an attempt was made to keep the percentage of false detected *Starlings* under 5% of the total amount of *Starlings* present. This was done by evaluating multiple parts of an image for false detection and adjusting detection levels in order to keep the false detection level below 5%. By doing so, the images used in the previous example (except for one) could be combined with more images that were easier to check. This resulted in figure 20, in which only the amount of estimated *Starlings* is shown, with a false detection percentage approximately under 5%. As can be seen in figure 20 the method used in R as well as ImageJ has trouble detecting objects when higher densities are present, while keeping the number of false detected objects low. However, the method described in R seemed to have fewer struggles compared to ImageJ. Nevertheless, after 2000 individuals the method used in R drops to an average percentage of 91.8 of the total amount of *Starlings* that are present based on hand countings and ImageJ drops the an average detection of 80.8 % after exceeding the 2000 individuals. When comparing all detection levels for all images (form a total count of 89 to 3564 *Starlings*) the average percentages appeared to be 96.2 ±5.71% in R and 90.2 ±7.93% in ImageJ. Because all images were very different from each other, slight adjustments were necessary. For example in some cases the automatic threshold needed to be altered and depending on the resolution the maximum and minimum bird size needed some adjustments too.

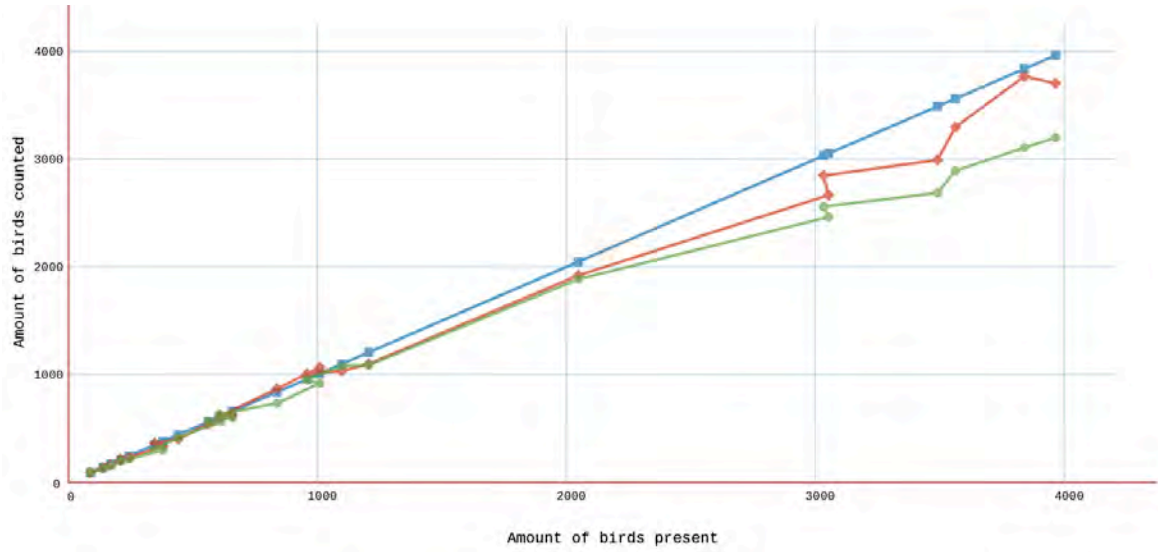



**Figure 20:** Amount of *Starlings* counted plotted against the amount of *Starlings* present for three different methods within 24 images. Blue represents the number of *Starlings* counted by hand, red the number estimated by R and green the estimated amount with the usage of ImageJ.

### 4.3 Results pollen particles using single images

The pollen particles are only analyzed by the usage of R, with functions that originate from both the method made for analyzing *Daphnia* and *Starlings* flocks combined. The true detection level was on average 94.9 ±3.53% whereas the false detection level on average was 5.02 ±3.53%, with 10 random images analyzed. These 10 images contained 402.9 pollen particles on average, calculated by averaging hand countings. For more details about the results of this section see appendix table 2.

## *5. Discussion*

First the results for the method that utilizes video material will be discussed in detail (paragraph 5.1), afterwards all methods for single images (paragraph 5.2). Paragraph 5.3 will briefly present some future recommendations and in paragraph 5.4 the final conclusion is made.

### 5.1 Counting moving objects using video material

In the first method that was designed to analyze moving *Daphnia* individuals, using video footage, ImageJ, R and MATLAB all were capable of detecting these moving objects. When evaluating ImageJ it showed to be most easy to use , due to its graphical interface. It provided fast but not always the most precise results. However, R and MATLAB showed more promising results, with more precision, and are therefore recommended for analyzing needs similar to the *Daphnia* examples. When comparing R to MATLAB, MATLAB as the advantage that it is capable of converting video material to multiple frames inside the software itself. R is not capable of reading video formats, therefore an additional tool is needed to convert video footage to single frames, making the total analyzing process less convenient. But when more familiar with the usage of R, the process of understanding MATLAB code is probably not worth the time effort, compared to its smaller advantages over R (for methods tested in this report). Moreover, MATLAB is an expansive software tool, while R is free in usage. However, MATLAB may have additional advantages; the website www.matworks.com offers a great deal of custom scripts made by people from all over the world, many of which are meant for analyzing video material. R on the contrary is not very popular yet for analyzing images or video material, and far less custom packages and examples scripts can be found on this particular subject. None of these MATLAB scripts were used in gathering results for this report, since researchers who are working with *Daphnia* were most interested by developing a tool in R. However, a script for tracking moving objects within MATLAB provided by Xiong Xiao and Diego Barragán on the “Mathworks file-exchange” was briefly tested. It was able to track

all 70 *Daphnia* individuals after some adjustments. The only drawback is that the used computer needs to meet higher hardware requirements in contrast to the more simple approach described in this report. Because these frames contained less than 100 *Daphnia* individuals counting them by hand could be a faster method in some cases compared to using one of the above-mentioned methods, for instance on average the total process inside R took 2 minutes and 37 seconds on a low-end notebook. In addition, when not setting certain parameters properly, one has to run parts of the program again in order to see what the different settings will achieve, making the process even longer. But when for example an experiment involving these types of organisms requires multiple duplicates, within several different conditions, the method described in R becomes viable. All these different populations can be recorded using specific guidelines and the video material can be used to quantify the number of individuals within each population, in a non-destructive way.
When all individual objects are successfully recognized, which was the case in the video material with a higher quality, their shape characteristic can be consulted. Meaning it is possible to measure them for instance. An easy method would be to use the function computeFeatures.shape() (EBImage package) inside R, this function is capable of providing for example the maximum radius for each patch, with the maximum radius the body size for each individual can be determined. This could save an enormous amount of time when body size is something one is interested by, because measuring every individual by hand using a binocular would not be necessary anymore.
When considering the ROI, alternative shapes next to circles are possible, as long as the border can be filtered out precisely, by using both a threshold and objects selection methods. The application can stay fairly simple if the precise outline of the ROI can be filtered out; only the pixels outside this border need to be excluded. But in a particular case when the border is not completely intact after setting a threshold and selecting certain objects, a function or a formula needs to be used in order to draw the border more precise and complete. Using this method different shapes are also possible as long as the coordinates are known (which can be found automatically as shown in this method) and the right formula that describes the desired shape. In the example shown in this report setting the ROI (a circle) could be automated, the same goes probably for different shapes in different situations. However, this requires new solutions that will be able to deal with these types of situations which will not be discussed in this report. As previously mentioned, the animal ecology and ecophysiology department (RU, Nijmegen) were most interested in finding a working method for quantifying individuals such as *Daphnia* involving R. The R script that was used in this particular example can be used for this. The function countdaphnia(), with the right input variables and directory for the input frames, analysis multiple frames containing moving *Daphnia* individuals. During the process some temporary results are shown and in the end the final result is presented, the total number of Daphnia is shown in the console window of R. The R script 'Counting Daphnia.R' contains the function as well as a simple guide on how to use this particular script with its functions. The folder called 'Test_Daphnia' provides example pictures to test the script with different settings. At this moment the function works best when using frames of a high quality, as discussed earlier, it is capable of counting densities up to 70 individuals. However, many improvements in order to make the process faster, more efficient and more precise could make this function evolve from basic test function to an actual working program. Moreover, more testing is required in even higher densities of *Daphnia* individuals to conclude whether this method is capable of dealing with higher densities.

### 5.2 Analyzing single images

For analyzing single images, R again can be more precise compared to ImageJ in the methods illustrated in this rapport. This is mainly thanks to the extensiveness and controllability inside R. ImageJ is far easier in usage, nonetheless it lacks a lot of features and certain functions cannot be controlled to the same extent as possible in R. Images with objects in the background, images with *Starlings* partly overlapping, images of different resolutions and images with different total amounts of *Starlings* present were analyzed to evaluate how different methods would deal with certain problems in single images. On

average 96.2% of the total amount of *Starlings* counted by hand were (true) detected using R. This average level of true detection is based on analyzing 24 random images within the first and second level of difficultly (explained in paragraph 3.3.1) and while keeping the false detection level below 5%. Inside ImageJ the average true detection level was 90.1% over the same 24 images. The major reason for a difference of roughly 6% could be that R has a specific step which divides the patches of *Starlings* into two groups, the first one contains patches of single birds the second of multiple clustered birds. The absence of this particular step makes ImageJ obviously more prone to under estimating the total number of *Starlings* present in an image.

As mentioned before all images containing *Starlings* were divided into three groups. Unfortunately, the second level had only a few images over 3564 *Starling* individuals. A few of these images with higher densities (higher than 10.000 *Starling* individuals) were tested, resulting in an average detection level of 69% when using R. This average detection level is not based on true detection, but on a total count of recognized objects compared to a total count present, estimated by bird professionals from SOVON and myself. In this case the level of false detection was roughly kept below 5%, using the same method as described in paragraph 4.2. In this case it is very difficult to draw any conclusions, since it could be a problem with the analyzing method in R that will result in these low estimates or a problem with the by hand estimated number of birds present.

Images of poor quality, low resolution, low contrast or images with many *Starlings* clustering together (as aforementioned) were considered as impossible to quantify using the methods described in this report. Therefore, certain quality standards need to be met in order for the developed methods to be applicable. It is however hard to exactly describe these standards. For example an image could be of a low resolution, but birds are presented by more than 10 or more pixels, making the image perhaps suitable. A different image could be of a very high resolution but birds could be represented by for example only 2 pixels, meaning it is difficult to distinguish single individuals and recognizing shape characteristics. Examples like these make it hard to set guidelines for describing precise quality standards. Images with *Starlings* behind or in front of objects were also hard (or impossible) to analyze. These characteristics belong to the third level of images, which were labeled as non-analyzable using the method described in this report. In these images, even when counting them by hand, it is impossible to distinguish individual *Starlings.* The image quality, physical clustering of individuals and presences of background objects before or behind flying *Starlings* seem to be the bottleneck in the analyzing processes described in this report.

Both the methods developed in R and ImageJ can be tested and use by reading the specially made guide for SOVON employees and other bird enthusiasts ('Handleiding voor het analyseren van spreeuwen foto's'). For each method an instruction video was created to illustrate the process in more detail. These videos can be found on you tube: https://www.youtube.com/watch?v=rvBJsCJOjqk&feature=youtu.be; https://www.youtube.com/watch?v=tHJKjWOD_kk.

In contrast to all methods used in this report there are other possibilities for dealing with color images inside R, ImageJ or MATLAB. In the methods used one of the three layers of color is chosen based on being the most clearest layer or just the first layer of the RGB-format. It is also possible to combine all three layers by taking for example the maximum, minimum, average or median value for each pixel within these three layers. These approaches did not show any improvements in these particular cases. Possibly, when color is more important in recognizing objects, a method that utilizes a different approach to deal with color images is more desirable.

### 5.3 Future improvements

Next to MATLAB, R and ImageJ there are of course numerous other tools that can be used for analyzing images. As mentioned in the first paragraph of the discussion the Mathworks file-exchange offers a great deal of functions that can be useful for analyzing images. Moreover, the Open-CV library offers even more in depth solutions for analyzing digital material. Some of these functions or scripts were

briefly tested to see whether they could be used to analyze *Starling* images, or similar images containing different objects, in the future. No results were computed to validate these methods; they are only intended to explore potential solutions for further research.

- Segmentation tool: several different options can be found on the Mathworks file-exchange for this specific approach. In this case a script provided by Micael Couceiro was used, since his script had the highest rating on the Mathworks file-exchange website. The idea behind this tool is that it utilizes Particle Swarm Optimization (a method that will not be discussed in this report, due to its complexity) to clusters the data set with minimal user interference (Omran *et al*., 2006). In this process the optimum number of clusters is also set automatically, all pixels within each cluster then get the average value within the cluster itself. An example result of this technique is shown in the appendix (appendix figure 11). When this result is compared to the original image (appendix figure 10) it shows that this approach is capable of segmentation the images correctly, while keeping all patches containing *Starlings* intact. Therefore it could be a critical step in a more precise analyzing method. It is possible to set a threshold after using the segmentation tool, by doing so better separation of birds and their background can be achieved.
- An approach that uses edge detection to separate *Starlings* from their background could also be useful in the search for a better future solution. This method was briefly discussed in paragraph 3.3.6 where one of the more simple approaches to edge detection was considered. This particular way of detection edges was the only method available in ImageJ and as mentioned before it is very prone to background noise, due to a very straightforward and simple approach. A more advance way for detecting edges would be to use edge detection that utilizes 'confidence based edge detection' (Meer & Georgescu, 2001) and 'mean shift based image segmentation' (Meer & Comanicu, 2002). This particular example utilizes a far more advanced method for both the detection of transitions in pixel intensities and grouping these pixels recognized as being edges. This particular example was briefly tested and the results are shown in appendix figure 12. The tool can be downloaded on the creator's website: http://coewww.rutgers.edu/riul/research/code/EDISON/. Although some more adjustments are needed for the tool to be used on these types of images, it showed promising results. However, clustering of birds is still a problem and it is still unclear if this technique will be helpful in future analysis for dealing with this difficulty.
- Both methods that are discussed above would help in separating the birds from their surroundings; this last future recommendation can be useful to deal with the problem of clustered birds. The method is called Skeletonization and it literally converts patches in to their skeletons, as can be seen in appendix figure 12. In this case pictures first need to be converted into binary images by setting a threshold. When all patches, after setting a threshold, are converted in to 'skeletons' in theory separating patches of single birds and multiple birds could become easier. The OpenCV tool that was used in this case can be found on this website: http://felix.abecassis.me/2011/09/opencv-morphological-skeleton/. There are also multiple script developed on the Mathworks file-exchange website for MATLAB (http://www.mathworks.com/matlabcentral/fileexchange/25865-euclidean-skeleton).

## 5.4 Conclusion

All methods used in this rapport quantify the total number of objects at least within a range of 10% around the actual number of objects present based on counting them by hand. Moreover, in all the examples presented the false detection level stays below 5%. When applying the method in R intended for moving objects (i.e. video material) on *Daphnia*, 100% true detection was possible with amounts up to 70 individuals. In order to not only quantify objects but also measure them, a true detection level of 100% is required, therefore it is important to maintain these standards when interested in for example body size. In this case certain quality standards are required. By using video frames of a lower quality an average true detection level of only 95.7% was possible, when using 65 up to 75 *Daphnia* individuals.

For single images the average percentages appeared to be 96.4% in R and 91.4% in ImageJ, using images containing 89 up to 3564 *Starlings*. For pollen particles 97.4% was the highest true detection level when analyzing 10 images, each containing around 400 particles.
Although these results are promising there are a few drawbacks; as mentioned before, images or videos need to meet certain quality standards. For example, the contrast between objects of interest and their background needs to reach a certain level. Moreover, the resolution needs to be of a specific dimension, whereby objects of interest are formed out of approximately ten or more pixels.
Not only the image qualities will influence the performance of these methods negatively, but also the density of individuals is crucial. Meaning, these methods have trouble dealing with high densities, for example images with over 10.000 *Starlings* have a detection level of 69% as mentioned before. Unfortunately in the case of *Daphnia* and the pollen particles no higher densities were tested. However, 70 *Daphnia* individuals showed a true detection level of 100%, therefore higher densities seem possible provided that the same or better quality standards are met.
In practice, especially when it comes to the *Starlings* example, these methods are only viable if even in these higher densities precise estimates are made. Therefore, more research is necessary in which the three discussed methods in 5.3 could contribute. However, it is of utmost importance to realize that not only the analyzing technique is responsible for precise estimates, but also the input images or video material.
In the end it can be concluded that these methods described are showing promising results for the in this case semi-automatic quantification of *Daphnia* individuals, *Starlings* and pollen particles of *Solanum carolinense* up to certain densities. Further research, providing more advanced methods, should be performed to see whether higher densities are possible and if these techniques can be fully automated.

## *6. References*


Choi, J.Y., Kim1, S.k., Chang, K.h., Kim M.C., La G.h., Joo G.j., Jeong K.S., (2014). *Population Growth of the Cladoceran, Daphnia magna: A Quantitative Analysis of the Effects of Different Algal Food.* Department of Biological Sciences, Pusan National University, Busan, Republic of Korea. PLOS ONE volume 9, issue 4. p 1-8

Comanicu, D., Meer, P., (2002). *Mean shift: A robust approach toward feature space analysis.* IEEE Trans. Pattern Anal. Machine Intell., 24. p 603-619.

Daims, H., Wagner, M., (2007). *Quantification of uncultured microorganisms by fluorescence microscopy and digital image analysis*. Applied Microbiology and Biotechnology, volume 75, p 237–248.

Dodson, S., (1988). *The ecological role of chemical stimuli for the zooplankton: Predator-avoidance behavior in Daphnia.* The American Society of Limnology and Oceanography, Inc. Limnol. Oceanogr., 33(6, part 2). p 1431-1439.

Drake, J.M., Griffen, B.D., (2009). *Speed of expansion and extinction in experimental populations.* Ecol Lett 12. p 772–778.

Ebert, D., (2005). *Ecology, Epidemiology, and Evolution of Parasitism in Daphnia.* Bethesda (MD): National Center for Biotechnology Information . Chapter 10, Experiments with Daphnia and Parasites.

Eliceiri, K.W., Berthold, M.R., Goldberg, I.G., Ibanez, L., Manjunath, B.S., Martone, M.E., Murphy, Peng, H., Plant, A.L., Roysam, B., Stuurman, N., Swedlow, J.R., Tomancak, P., Carpenter, A.E., (2012). *Biological imaging software tools*. Nature Methods 9. p 697–710.

Faerovig, P.J., Andersen T., Hessen, D.O., (2002) *Image analysis of Daphnia populations: non-destructive*

*determination of demography and biomass in cultures.* Department of Biology, University of Oslo, Oslo, Norway. Freshwater Biology (2002) 47. p 1956–1962.

Kobayashi, T., Hosaka, T., Mimura, S., Hayashi, T., Ostu, N., (2008). *HLAC approach to automatic object counting.* Bio-inspired Learning and Intelligent Systems for Security 4-8 Aug. 2008. p 40-45.

Mallard, F., Le Bourlot, V., Tully, T., (2013) *An Automated Image Analysis System to Measure and Count Organisms in Laboratory Microcosms.* Ecole Normale Superieure, Paris, France. PLOS ONE, volume 8, issue 5. p 1-10.

McClure, B., Haring, V., Ebert, P., Anderson, M., Simpson, R., Sakiyama, F., Clarke, A., (1989). *Style self-incompatibility gene products of Nicotiana alata are ribonucleases.* Plant Cell Biology Research Centre School of Botany, University of Melbourne. Nature volume 342 21/28 December 1989. p 955-957.
Meer, P., Georgescu, B., (2001). *Edge detection with embedded confidence*. IEEE Trans. Pattern Anal. Machine Intell., 23. p 1351-1365.

Omran, M.G.H., Salman, A., Engelbrecht, A.P., (2006). *Dynamic clustering using particle swarm optimization with application in image segmentation.* Pattern Anal Applic (2006) 8. p 332–344.

Peerakietkhajorn, S., Tsukada, K., Kato, Y., Matsuura T., Watanabe, H., (2015). *Symbiotic bacteria contribute to increasing the population size of a freshwater crustacean, Daphnia magna.* Environmental Microbiology Reports volume 7, Issue 2. p 364–372.

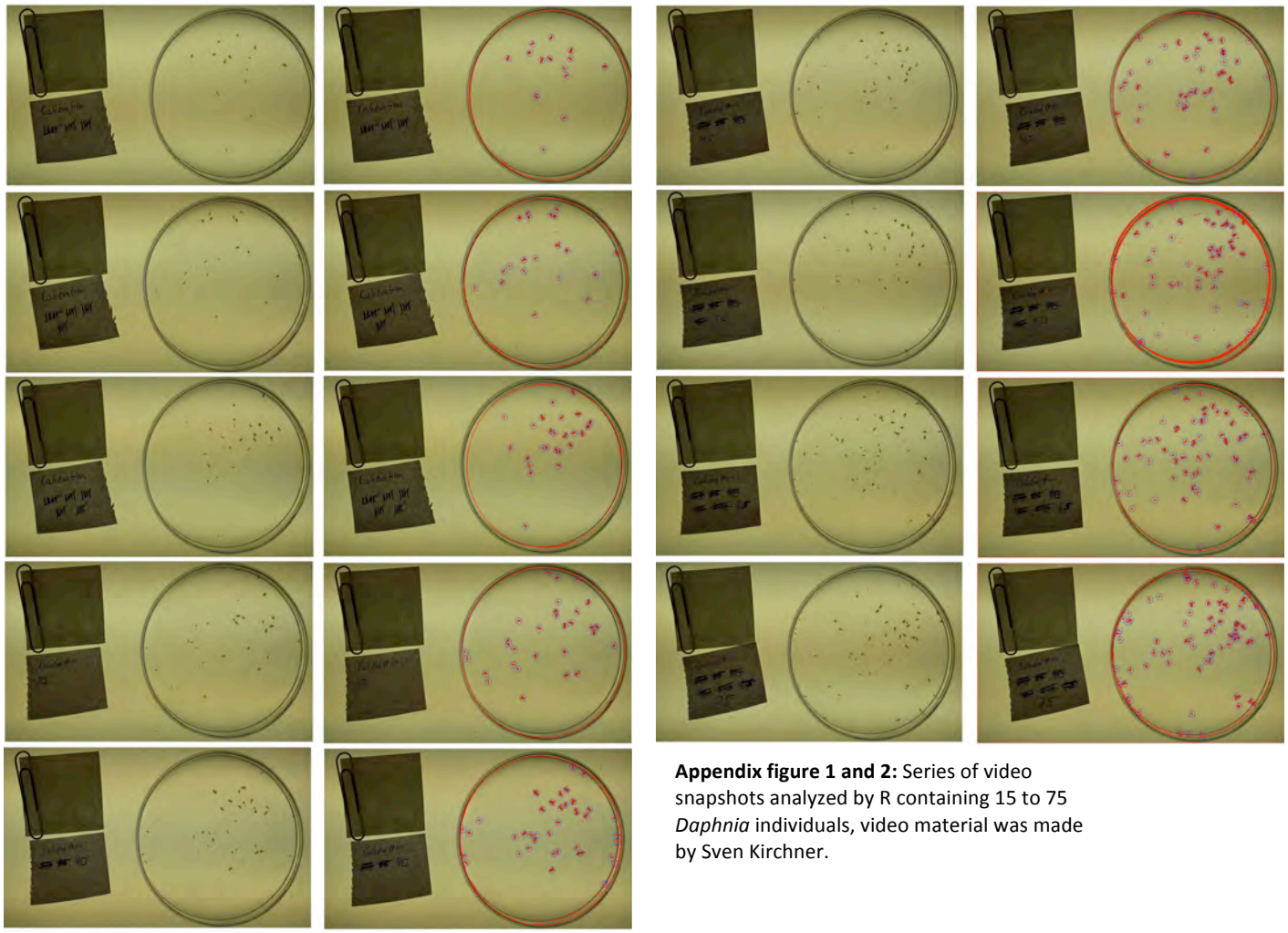
Rappoldt, C., Kersten, M., Smit, C., (1985). *Errors in large-scale shorebird counts.* Ardea 73.1 (1985). p 13-24.

Sirmacek, B., Wegmann, M., Cross, A.D.P., Hopcraft, J.G.C., Reinartz, P., Dech, S., (2012). *Automatic population counts for improved wildlife management using aerial photography.* International Environmental Modelling and Software Society (iEMSs). p 1-8.

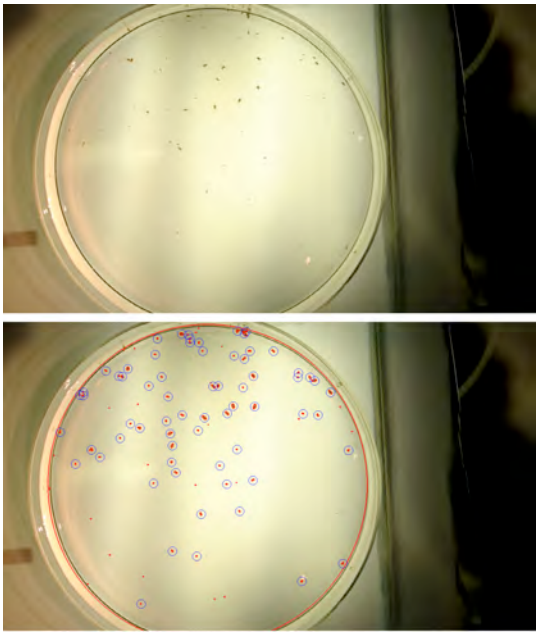
Tou, J.Y., Toh, C.C., (2012) *Optical Flow-Based Bird Tracking and Counting for Congregating Flocks.* Lecture Notes in Computer Science. Volume 7198, 2012. p 514-523.

Vincent, O., Folorunso O., (2009). *A Descriptive Algorithm for Sobel Image Edge Detection.* Proceedings of Informing Science & IT Education Conference (InSITE) 2009.

## 7. Appendix



**Appendix figure 1 and 2:** Series of video snapshots analyzed by R containing 15 to 75 *Daphnia* individuals, video material was made by Sven Kirchner.



**Appendix figure 3:** Video snapshot with higher quality for successfully analyzing 70 *Daphnia* individuals, video material was made by Selwyn Hoeks.

171 birds,
Picture made by
Ballering


960 birds,
Picture made by
vd Brink


74 birds,
Picture made by
Ballering


140 birds,
Picture made by
Ballering


89 birds,
Picture made by
Frens

**Appendix figure 4:** 5 level 1 images containing *Starling* flocks


607 birds,
Picture made by
Besters


3054 birds,
Picture made by
Bos


1099 birds,
Picture made by
KleinRouweler


2049 birds,
Picture made by
du Pon


838 birds,
Picture made by
KleinRouweler


347 birds,
Picture made by
Slegers

**Appendix figure 5:** first 6 level 2 *Starling* images

1011 birds,
Picture made by
vd Brink

211 birds,
Picture made by
Neijenhuis

approximately 12000 birds,
Picture made by
Busser

1207 birds,
Picture made by
vd Brink

442 birds,
Picture made by
Fokkelman

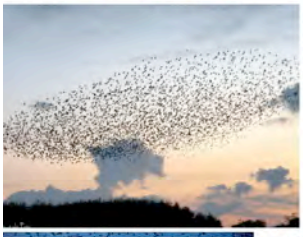
**Appendix figure 6:** 5 more level 2 images containing *Starling* flocks

3492 birds,
Picture made by
Bos

659 birds,
Picture made by
Prikkebeen

245 birds,
Picture made by
Lalkens

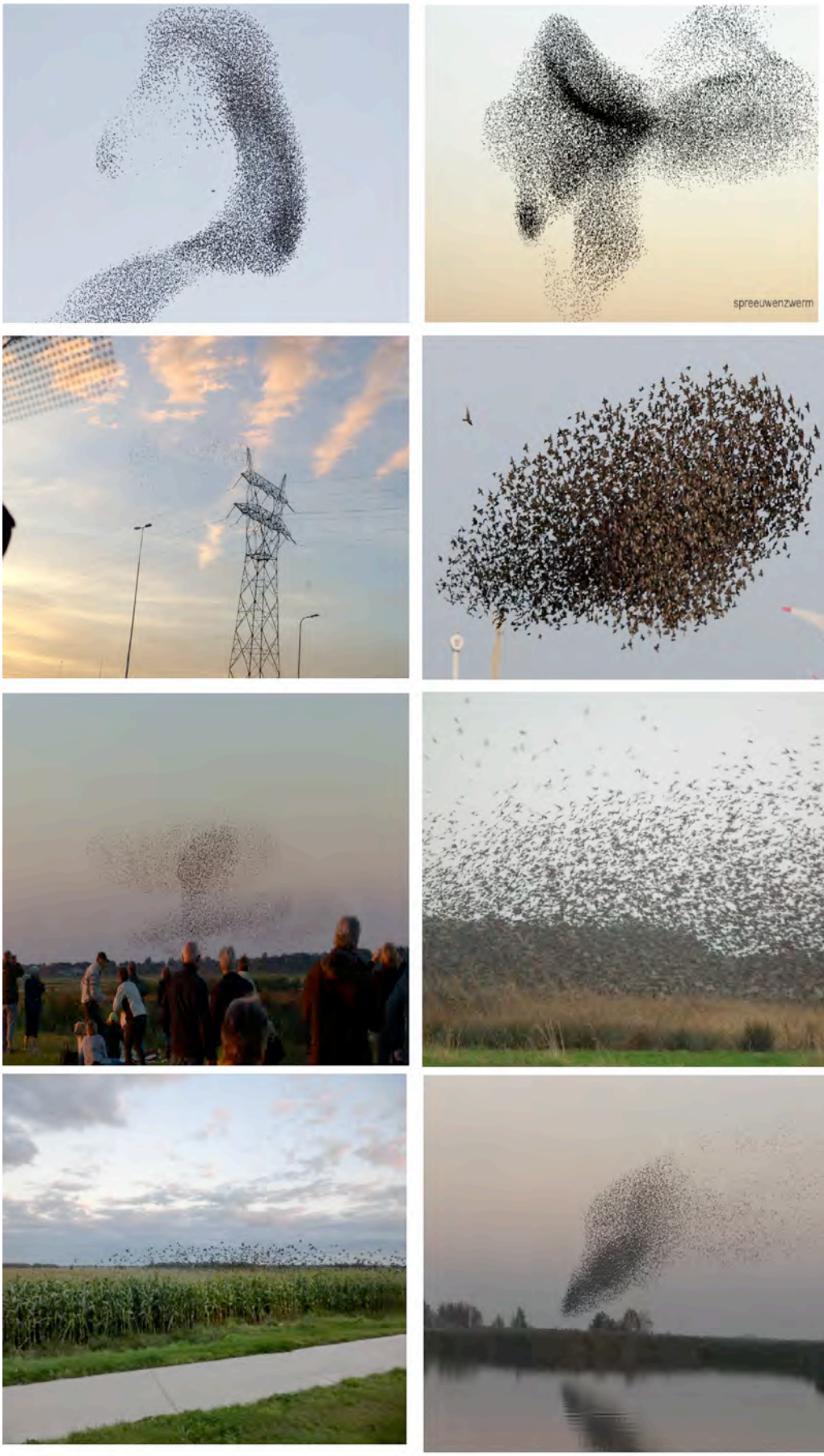
3043 birds,
Picture made by
Seitzinger



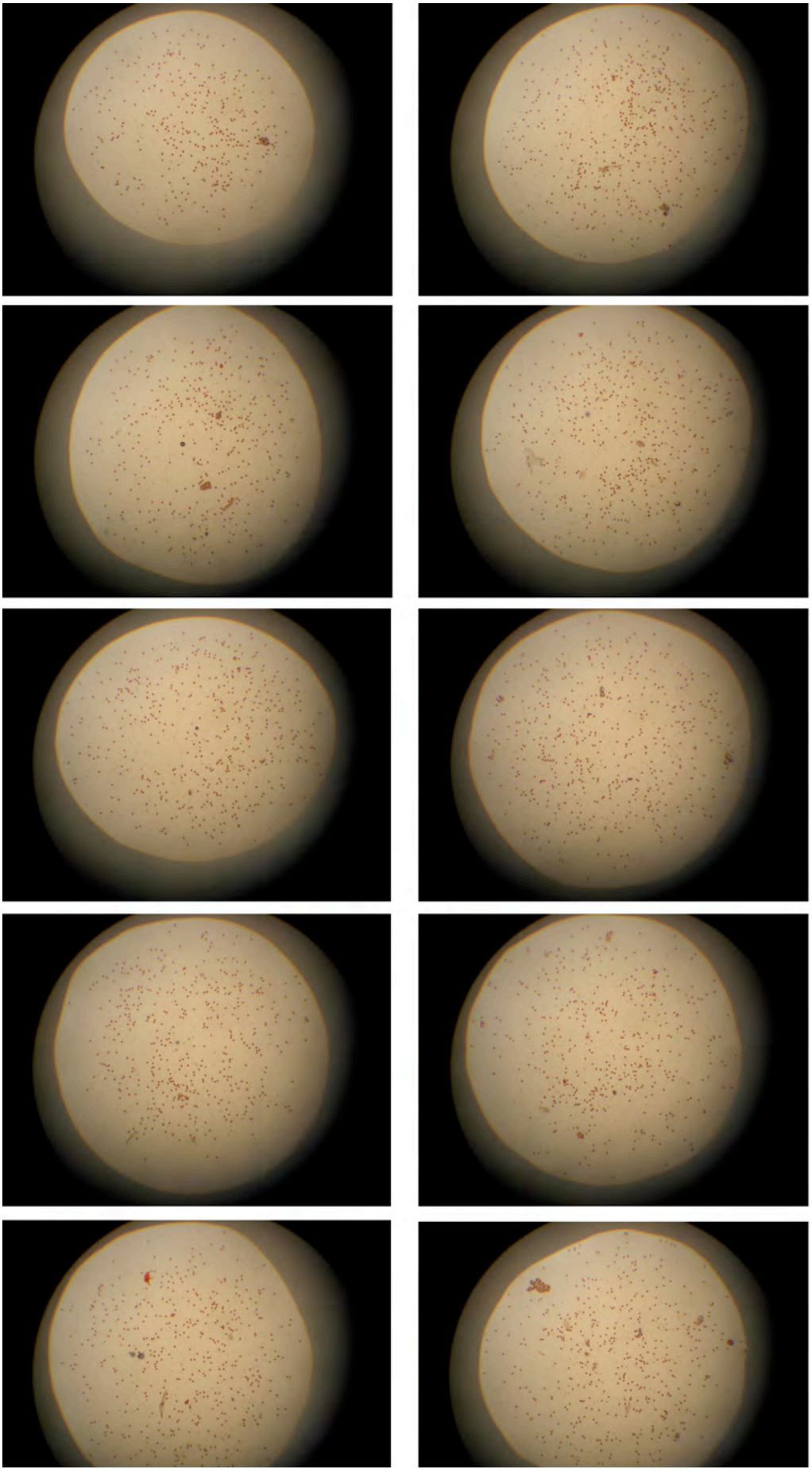
3966 birds,
Picture made by
de Pon

3840 birds,
Picture made by
Bos

**Appendix figure 7:** 6 more level 2 images containing *Starling* flocks



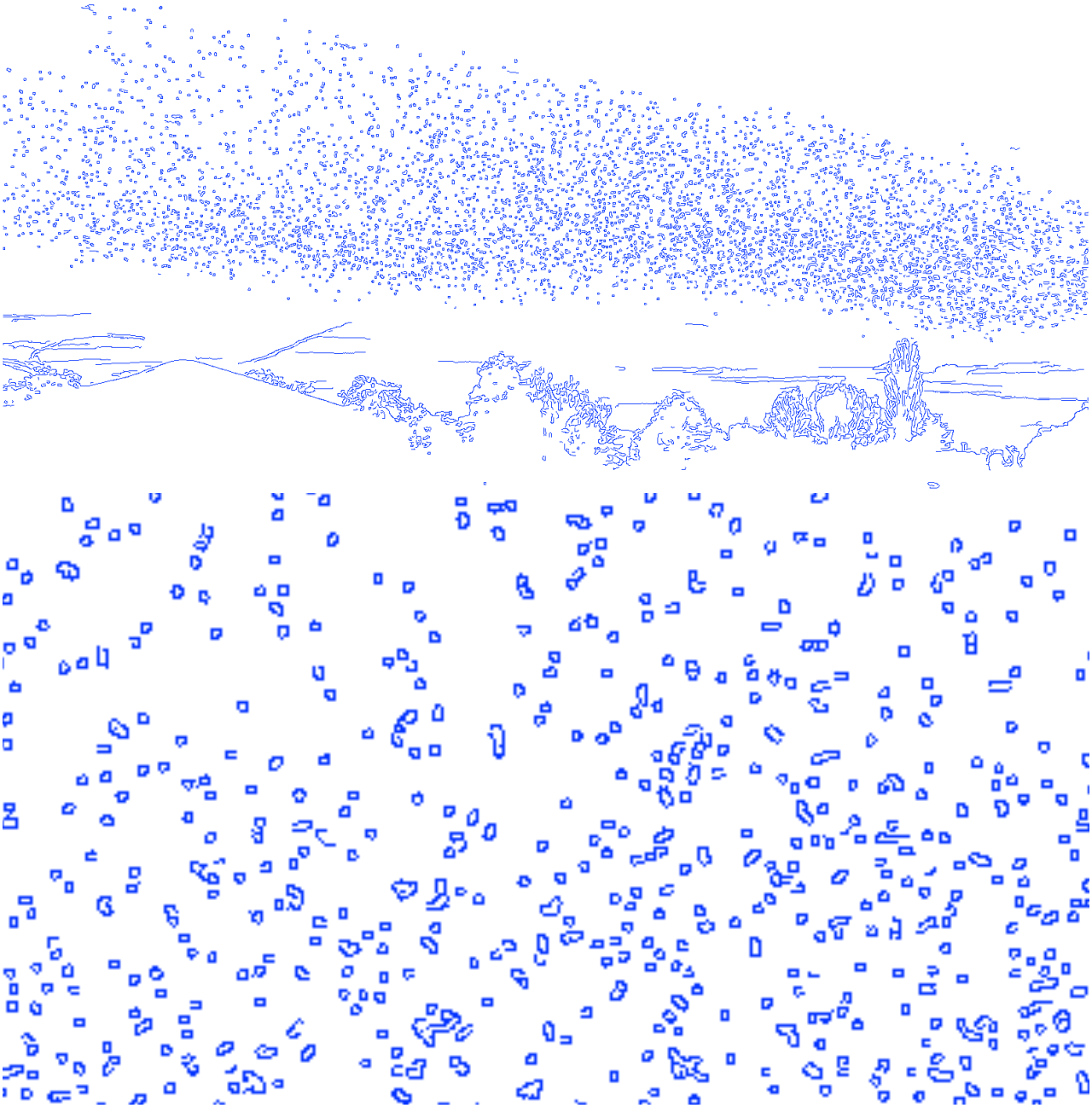

**Appendix figure 8:** Examples of some of the level 3 images containing *Starling* flocks



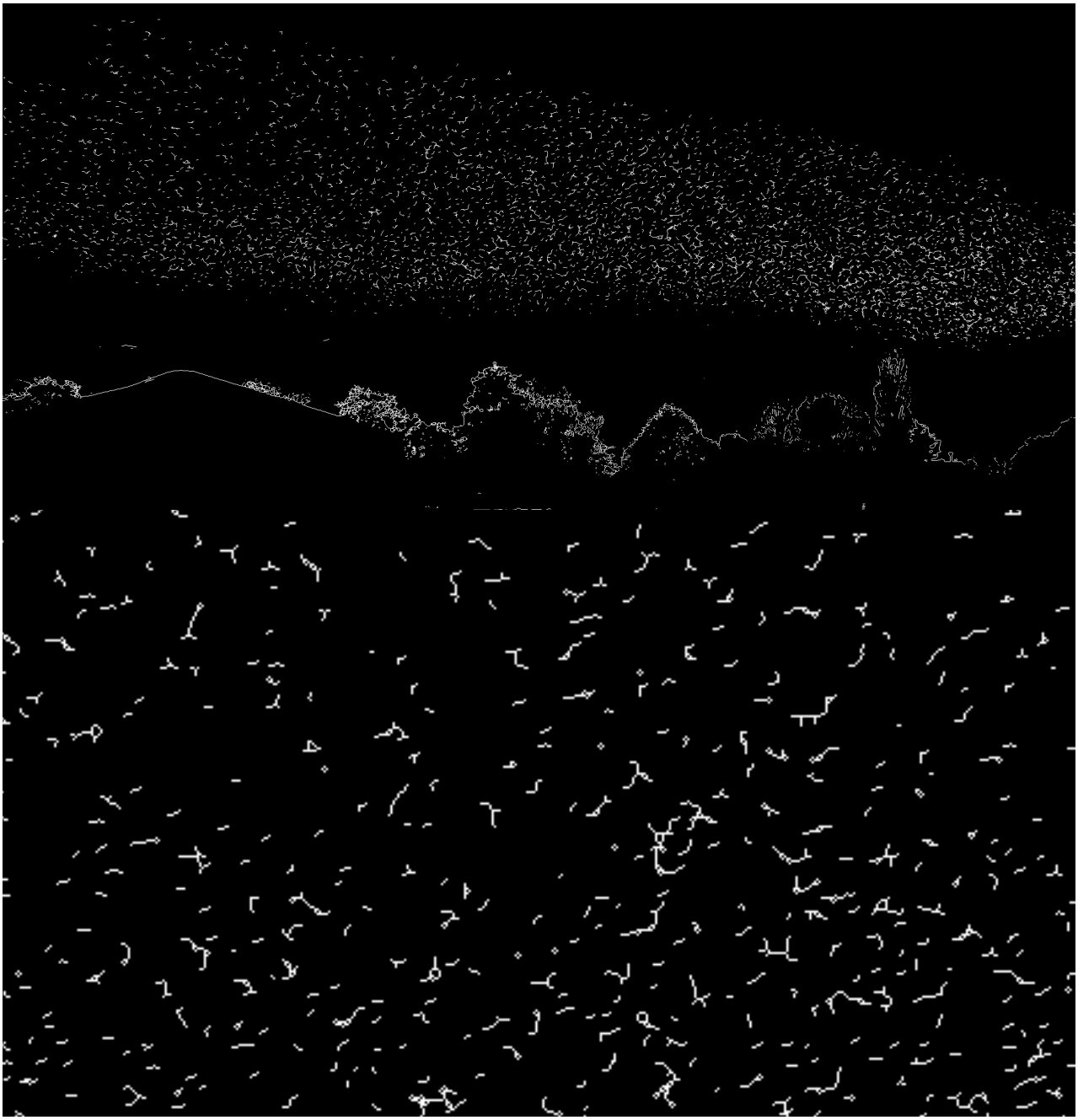
**Appendix figure 9:** 10 images used in the example of the pollen particles, picture made by Lidewij Keser

**Appendix figure 10:** Original image used for validating methods considered in the discussion. The image above is has the original dimensions, the image below shows a part of this image, in order to see how these *Starlings* look like up close. Picture made by Bos.

**Appendix figure 11:** Using a segmentation tool to separate *Starlings* from their background. The image above is has the original dimensions, the image below shows a part of the analyzed image, in order to see how these *Starlings* look like up close when analyzed. Picture made by Bos.



**Appendix figure 12:** Using an edge detection tool to separate *Starlings* from their background. The image above is has the original dimensions, the image below shows a part of the analyzed image, in order to see how these *Starlings* look like up close when analyzed. Picture made by Bos.



**Appendix figure 12:** Using a skeletonization tool to separate *Starlings* from their background. The image above is has the original dimensions, the image below shows a part of the analyzed image, in order to see how these *Starlings* look like up close when analyzed. Picture made by Bos.

**Appendix Table 1:** Total, false detected and true detected *Daphnia* individuals

| Counted by hand | Total counted by R | Total counted by ImageJ | FD individuals R | FD individuals ImageJ | TD individuals R | TD individuals ImageJ |
|---|---|---|---|---|---|---|
| 15 | 15 | 15 | 0 | 0 | 15 | 15 |
| 20 | 20 | 18 | 0 | 0 | 20 | 18 |
| 25 | 25 | 25 | 0 | 1 | 25 | 24 |
| 30 | 30 | 28 | 0 | 0 | 30 | 28 |
| 40 | 39 | 44 | 1 | 8 | 38 | 36 |
| 45 | 44 | 41 | 0 | 1 | 44 | 40 |
| 50 | 51 | 45 | 2 | 0 | 49 | 45 |
| 65 | 63 | 61 | 1 | 2 | 62 | 59 |
| 75 | 72 | 63 | 0 | 4 | 72 | 59 |

FD = False Detected

TD = True Detected

**Appendix Table 2:** Total, false detected and true detected pollen particles

| Counted by hand | Total counted by R | FD particles counted by R | TD particles counted by R |
|---|---|---|---|
| 325 | 321 | 7 | 314 |
| 453 | 463 | 21 | 442 |
| 375 | 383 | 12 | 371 |
| 451 | 440 | 11 | 429 |
| 389 | 397 | 10 | 387 |
| 429 | 436 | 9 | 427 |
| 401 | 378 | 47 | 331 |
| 398 | 404 | 28 | 376 |
| 411 | 399 | 38 | 361 |
| 397 | 383 | 18 | 365 |

FD = False Detected

TD = True Detected

# Handleiding voor het analyseren van spreeuwen foto's

Het kwantificeren van vogels (spreeuwen) in groepen op digitale foto's met behulp van R of ImageJ

**Inhoud**



## Inleiding

Voor het kwantificeren van spreeuwn die in groepen vliegen op digitale foto's, genomen door vrijwilligers, zijn twee methoden ontwikkeld (zie het verslag*: 'Towards a fully automated image analysis system applicable to ecological studies'*). De eerste methode is ontworpen in R, de tweede in het programma ImageJ. De applicatie in R is voor het grootste gedeelte zo gemaakt dat het proces zo veel mogelijk automatisch verloopt. Het uitgangspunt was dan ook om zo weinig mogelijk gebruikers input nodig te hebben om foto's te analyseren. In het geval van ImageJ was dit in niet mogelijk, deze methode vraagt dan ook meer interactie in vergelijking met de eerder genoemde methodiek, maar is daardoor wel meer inzichtelijker voor de wat nieuwere gebruiker. Resultaten voor beide applicaties zijn terug te vinden in het verslag, het betreft hier een bachelor stage verslag waarin beide methodes zijn ontwikkeld en waarbij meerdere foto's zijn geanalyseerd om de effectiviteit van beide methodes vast te stellen. Als voorbeeld zijn in de map '*Example images*' een aantal foto's toegevoegd die zowel voor applicatie in R als in ImageJ gebruikt kunnen worden om gewent te raken aan beide tools. De theorie achter beide methodieken wordt in detail behandelt in het eerder vernoemde verslag.

## Analyse in R

Voor het gebruik van de methode ontwikkeld in R is het script '*four steps and functions.R*' nodig en het package genaamd *'starling'*, hierin staan de functies en stappen waarmee het mogelijk wordt om met behulp van R verschillende foto's te analyseren. Het analyse proces verloopt volgens vier eenvoudige stappen, zoals terug te vinden in het R script en in de geproduceerde package. Bij het gebruik van de volgende methode wordt geadviseerd R studio te gebruiken. Voor een gebruiker die onbekend is met het installeren van een package en het openen van een R script, kan de video '*Analyzing bird images inside R*' hulp bieden (https://www.youtube.com/watch?v=rvBJsCJOjqk). In deze video wordt het gehele proces doorlopen een rustige en eenvoudige stappen. Niet alleen de installatie van het package en het openen van de bestanden, maar ook de hieronder volgende vier stappen worden in de video behandelt.

1. Na het installatie proces (eindigt 1 min 15 van de tutorial video) kunnen de stappen voor het analyseren doorlopen worden, beginnende bij de eerste functie: *'listbirdfiles'*. Voor dat de functie gedraaid kan worden is het nodig om de juiste '*path*' the zetten naar een map die alle te analyseren foto's bevat, dit wordt gedaan achter '*directPictures=*' .Een voorbeeld is te zien in de figuur hieronder: Vervolgens geeft de functie een lijst van alle bestanden die in de map

```
5  ################################################
6  ### step 1: setting directory and opening file
7  # set directory images
8  direcPictures="/Users/Selwyn/Desktop/Images" # path to folder, between brackets
9  # function 1: list all files in folder choosen
10 listbirdfiles(direcPictures)
```

   aanwezig zijn. In de volgende functie kan vervolgens een van deze bestanden gekozen worden om geopend te worden (zowel .jpg als .png worden ondersteund).

2. In de tweede stap is het nodig om het nummer van de gewenste afbeelding in te voeren uit de lijst die eerder werd gemaakt bij stap 1. Achter *'image_to_open'* moet het juiste nummer van de afbeelding komen, zoals bijvoorbeeld te zien in de onderstaande afbeelding:

```
### step 2: setting threshold
# settings
image_to_open=3 # choose image to open from list (state number of image in list)
Threshold=0.0 # Threshold is set automatically but can be altered for better resu
# function 2: setting threshold
setThreshold(image_to_open,Threshold)
```

   Het is tevens mogelijk om de '*threhold*' (uitgelegd in het eerder genoemde verslag) aan te passen. Dit is in de meeste gevallen niet nodig, maar soms kunnen foto's echter zo afwijkend zijn dat aanpassing nodig is. Het verhogen van de threshold zorgt ervoor dat meer objecten in de selectie worden mee genomen, een verlaging minder. Zowel de originele foto als een foto met een threshold worden geopend na het uitvoeren van de tweede stap. Door deze twee met elkaar te vergelijken kan worden bepaald of de threshold juist is bepaald door de functie.
3. In de derde stap worden objecten op de achtergrond (zoals bomen en gebouwen) van de foto verwijdert, hoe dit precies gebeurt staat beschreven in het eerder genoemde verslag. De derde stap hoeft niet perse uitgevoerd te worden en kan worden overgeslagen als er geen objecten in de achtergrond aanwezig zijn. Om objecten succesvol te verwijderen kan het soms nodig zijn om de maximale en minimale grote van de vogels aan te passen (zie de onderstaande foto; '*maxsize*' en '*minsize*'), vooral als foto's onder een bepaalde resolutie zijn (bijvoorbeeld onder 1000x700).

```
### step 3: creating layer with unwanted objects (if necessary, otherwise continue with the ne
# settings
maxsize=0 # max size bird patches, see histogram for possible adjustment (200=default, if wher
minsize=0 # min size bird patches (0=default, if when 0 is entered as value)
# function 3: creating layer with unwanted objects
backgroundObjects(maxsize,minsize)
```

   Maar in de meeste gevallen doen de default instellingen het prima. Twee foto's worden na de uitvoering van de derde stap, namelijk één met alleen de vogels resterend en één met de verwijderde objecten. Aan de hand van deze twee foto's is het mogelijk indien nodig de maximale en minimale grote aan te passen.
4. In de laatste stap worden de resterende vogels verdeelt in twee groepen, één groep met patches die bestaan uit één vogel en één groep met patches die bestaan uit meerdere vogels doordat deze zijn samengesmolten in de foto. De verdeling van deze twee groepen kan soms wat aanpassing nodig hebben. In de onderstaande afbeelding is te zien dat de vierde functie twee waarden nodig heeft namelijk een mogelijke aanpassing aan de eerder genoemde verdeling ('*adjustdivision*') en of stap 3 wel of niet is uitgevoerd ('*step3*') .

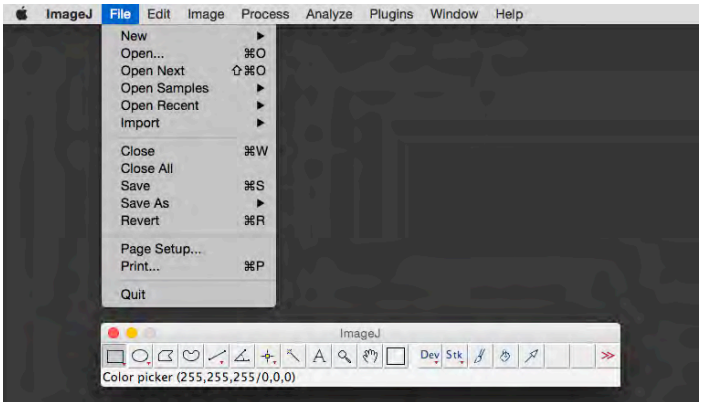
```
### step 4: dividing remaining patches in two groups and displaying results
# settings
step3=1 # 1 for if step 3 was used, 0 when skipping the 3rd step
adjustdivision=0 # may need adjustment when not choosing size catagories correct (0=default,
# function 4: dividing remaining patches in two groups and displaying results
divideResults(step3,adjustdivision)
```

   De laatste stap geeft een foto waarop het uiteindelijke resultaat zichtbaar is. In rood zijn patches weer gegeven die een spreeuw vertegenwoordigen en in blauw twee spreeuwen. Tevens geeft R een histogram weer van de verdeling van verschillende groottes van alle patches die spreeuwen vertegenwoordigen (in pixel aantallen). Hierdoor is het makkelijker om te zien of R een juiste verdeling heeft gemaakt, de gekozen verdeling door R is weergegeven in het console gebied samen met het totale aantal van aanwezig spreeuwen.

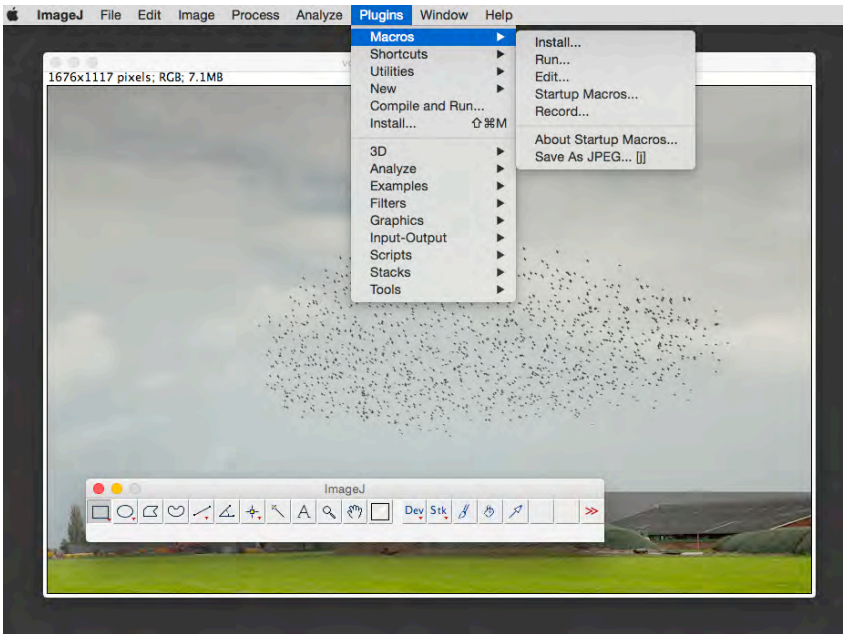
**Analyse met behulp van ImageJ**

ImageJ is een gratis software programma bedoelt om foto's te analyseren, het kan gratis gedownload worden van http://imagej.nih.gov/ij/download.html. Het is zowel beschikbaar voor Windows als Linux en Mac besturingssystemen. Het is gebaseerd op een coderingstaal in Java, vandaar de naam Image "J". Java heeft dan wellicht ook updates nodig om ImageJ goed te laten functioneren. De functies binnen ImageJ kunnen worden uitgebreid met 'packages' van het internet, deze methode is echter zo gemaakt dat geen extra functies nodig zijn binnen ImageJ. Met ImageJ is het mogelijk om zo gehete macro scripts te maken, waardoor verschillende functies binnen het programma automatisch en volgens plan verlopen. Om de analyse voor de gebruik zo simpel mogelijk te maken zijn voor deze methode in ImageJ twee macro scripts gemaakt. Deze 2 macro's zijn te vinden in de bijgeleverde folder '*ImageJ scripts*' ('*ImageJ macro part 1*' en '*ImageJ macro part 2*'). Het analyse proces verloopt volgens de volgende vier simpele stappen (waarbij de eerder genoemde 2 macro scripts gebruikt worden). De voorbeelden zijn geïllustreerd in op een Mac, maar voor Windows en Linux besturingssystemen gelden dezelfde ondernomen stappen. Het gehele proces is ook te zien in een video op you tube (https://www.youtube.com/watch?v=tHJKjWOD_kk)

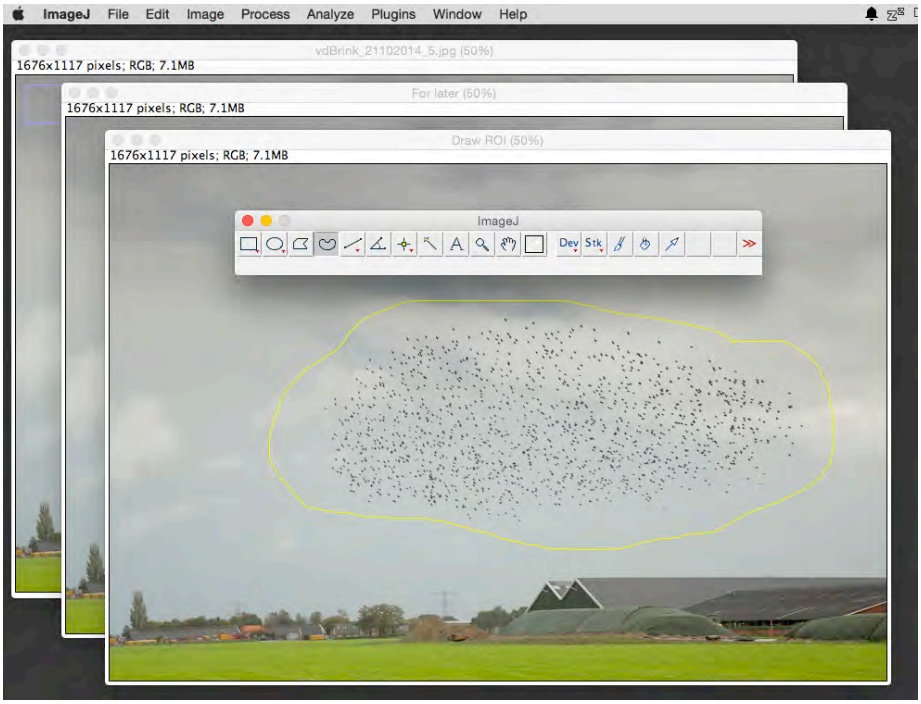
1. Na een succesvolle installatie van ImageJ, is het mogelijk een foto in te laden door te navigeren naar '*Open...*' via het menu genaamd '*File*'. Kies vervolgens de foto die geanalyseerd dient te worden. In dit geval is een van de voorbeeld foto's gebruikt om alle volgende stappen te illustreren.




2. Vervolgens is het tijd om de eerste macro te gebruiken. Dit kan worden gedaan door onder het menu '*Plugins*' voor '*Macros*' te kiezen en vervolgens de optie '*Run…*' te selecteren. Kies vervolgens de eerste macro ('*ImageJ macro part 1*').
Vervolgens openen er twee duplicaten van de eerder geselecteerde foto. Eén daarvan is nodig voor de volgende stap (met de titel '*Draw ROI*'), de tweede is nodig voor later, vandaar de titel '*For later*'.



3. Bij de derde stap is het nodig om met de hand de '*Region of interest*' (ROI) te selecteren. Dit is het gebied waar alle objecten zich bevinden die later gekwantificeerd dienen te worden. Het selecteren van de ROI gaat met behulp van de '*Freehand selection tool*' (het vierde knopje van links zoals te zien in de onderstaande afbeelding). Vervolgens is het mogelijk om met behulp van de muis het specifieke gebied te omcirkelen (zie onderstaande afbeelding).




Hierbij helpt het als de ROI zorgvuldig wordt bepaald, zo dat het een gesloten vorm is die alle vogels omvat.

5. Vervolgens kan de tweede macro ('*ImageJ macro part 2'*) gestart worden op dezelfde manier zoals beschreven in stap 2 voor de eerste macro.

Vervolgens wordt automatisch het eindresultaat bepaald en geïllustreerd (in zwart wit) zoals te zien in de onderstaande afbeelding. De tabel ‘Summary’ geeft het totaal aantal gevonden objecten weer binnen de ROI (onder het kopje ‘*Count*’), daarnaast bevat de tabel ‘*Results*’ alle oppervlaktes van de verschillende gevonden objecten (onder het kopje ‘*Area*’). Beide tabellen kunnen als Excel tabellen worden geëxporteerd.