Journal of
open research software

Markussen, T N 2016 Parchar – Characterization of Suspended Particles Through Image Processing in Matlab. *Journal of Open Research Software,* 4: e26, DOI: http://dx.doi.org/10.5334/jors.114

SOFTWARE METAPAPER

Parchar – Characterization of Suspended Particles Through Image Processing in Matlab

Thor Nygaard Markussen
CENPERM, Department of Geosciences and Natural Resource Management, University of Copenhagen, Øster Voldgade 10, 1350 Copenhagen, Denmark
thor.markussen@ign.ku.dk

Studies of suspended particles and particle dynamics in aquatic environments increasingly rely on camera systems to characterize the particles. Numerous systems exist and all use different codes and practises to process the images from the systems. Here, a step-by-step guide to an image processing and particle characterization code for Matlab is presented with the aim of bringing the particle community towards standardized image processing techniques. The code uses morphological reconstruction and simple block processing to filter out noise, out-of-focus particles and light source inconsistencies. It has been implemented on a specific camera system but is applicable to numerous systems and on highly variable particle types due to the standardized setup.

Keywords: Image processing; Thresholding; Particle characterization; Particle size; Blob analysis; Suspended particles
Funding statement: The code design and writing was partly funded by Department of Geosciences and Natural Resource Management, University of Copenhagen, and through financial support of a larger project by the Danish National Research Foundation (CENPERM DNRF100).

(1) Overview

Introduction

The correct size characterization of particles in suspension is an important part of many scientific disciplines ranging from oil droplet spreading studies in the laboratory to sediment grain or organic marine aggregate characterization in their natural aquatic environment. Many of the studies today rely on the use of camera systems and the subsequent processing of images. Here, an image processing code is presented that has been specifically designed for the direct characterization of aquatic, suspended particles in particulate flux studies. The script has primarily been used with the non-intrusive Pcam camera system, see [1] for a recent example, but the code is applicable to different camera systems over a wide range of sizes, shapes and environments. The purpose of presenting this code is not only to give an example of a new image processing code, more importantly the purpose is to make the code freely available and open on GitHub, so that other researchers may build on this. Thus, the hope is that it can bring us closer towards one standardized method of image processing of suspended particles.

Aquatic, suspended particles differ greatly in composition and thus spectral signatures and the cameras used to capture the particles utilize many different light sources. Particles are characterized by separating the objects in the foreground from the potential noise and out-of-focus particles in the background. The code gives the possibility of correcting for inconsistencies in background illumination and to use a standardized morphological reconstruction of particles to filter out noise and is in this way not dependent on a specific light source to function. The size of particles is in most studies characterized by the equivalent spherical diameter (ESD) and the code applies this concept as well although users may decide on other ways of characterizing sizes. The code also outputs a number of other shape parameters to make way for more comprehensive characterizations of the studied particles.

Implementation and architecture

The code has been programmed in a Matlab (R2014b) working environment using the *Image Processing Toolbox* on a Windows 7 Enterprise machine. Five Matlab Code files, one Matlab Data file and a number of example figures and datasets are supplied. Three of the five codes are used to process the individual image (*ParChar*, *imoverlay_orig* and *ImRec*) and the *ParChar_mult* uses the processing code on multiple files. The fifth code (*CompatibilityTest*) is a small test script to be run before using the code, to make sure the user has the correct Matlab version and that files have been downloaded correctly. The Data file (*Bins.mat*) is necessary to run the particle characterization part of the code.

The processing code, *ParChar.m*, involves six steps of which three are optional. **Figure 1** shows a simplified flow

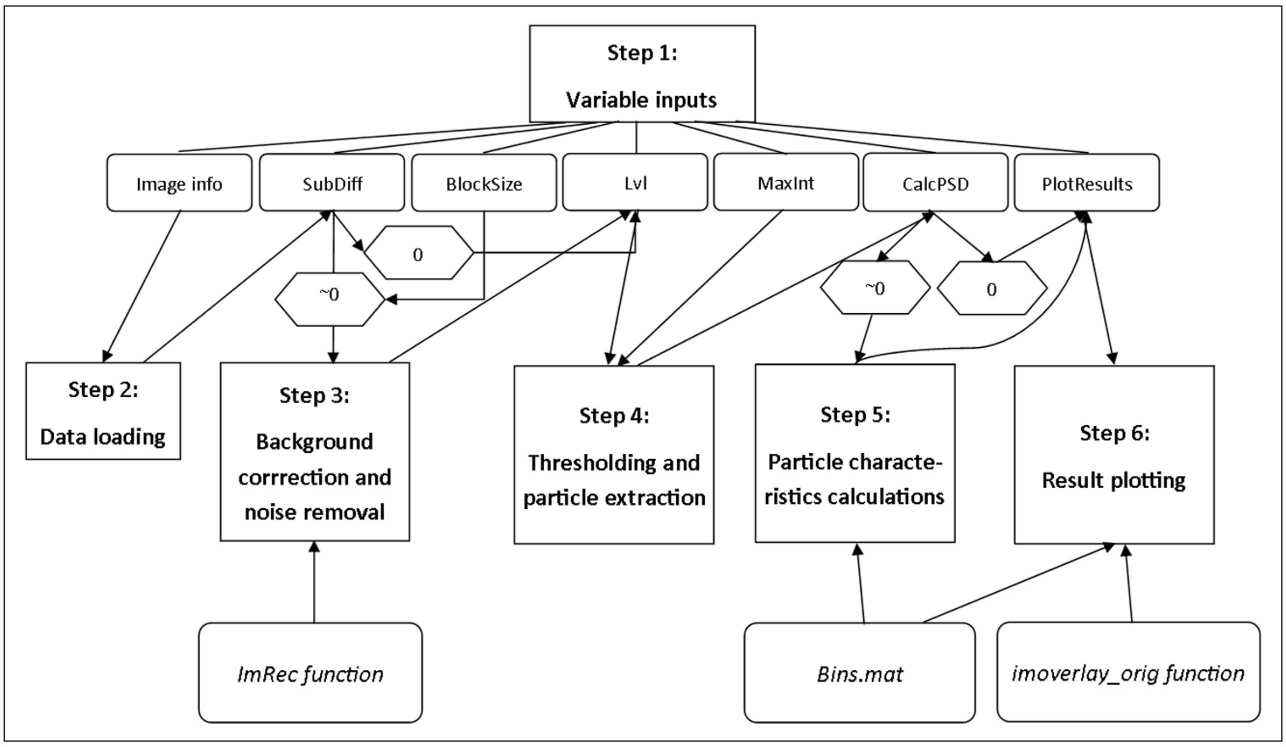

Figure 1: Simplified flow chart showing the six steps that the ParChar code is based on. Before anything can be done, the inputs have to be set in step 1. As described in the text and shown in the flow chart, some of the steps can be skipped by setting the relevant input values to zero.

chart with the six steps and their interdependencies and **Figure 2** exemplifies some of the steps. The steps are:

1. Variable inputs
2. Data loading, **Figure 2a**
3. Background correction and noise removal (optional), **Figure 2b**
4. Thresholding and particle extraction, **Figure 2c**
5. Particle characteristics calculations (optional), **Figure 2d**
6. Results plotting (optional)

Step 1: Variable inputs

The location of the file to be processed is selected by a pop-up window that allows the user to browse to the correct location. *ImHeight, ImWidth and MeasDepth* require inputs of the height, width and depth of the field of view of the image and measuring volume to be processed, in millimetres. *SubDiff* sets the amount (0–255) to extract between the marker and the mask in the morphological reconstruction in step 3). If set to 0, no morphological reconstruction will take place. *Lvl* sets the threshold level that will be used to convert the image to a black-and-white image, see step 4). *MaxInt* determines the fraction of the maximum intensity required for individual particles to be selected as described under step 4). This can also be excluded by setting *MaxInt* to 0. *CalcPSD* determines whether or not to skip step 5) of the image processing and calculate particle characteristics. If *CalcPSD* is set to 1, the characteristics will be calculated, all other values will skip step 5). *PlotResults* determines what to do in terms of plotting the results under step 6) of the image processing. *PlotResults* can be set to either 1 or 2 to enable plotting. Setting it to 1 will plot everything while setting it to 2 will only plot one figure for each processed image. Note that a small step before the data loading checks whether or not the Image Processing Toolbox is present. Preferably, this has already been assessed by running the *CompatibilityTest* script before using the *ParChar* code.

Step 2: Data loading

The image is loaded from the file location specified under step 1). Matlab reads the info from the file metadata to find the date and time that the image was taken and the height of the image, in pixels. The size of the field of view of one pixel in microns, the pixelsize, is calculated based on the *ImHeight* input in step 1).

Step 3: Background correction and noise removing (optional)

This step may be skipped entirely by setting the *SubDiff* input to 0. The *ImRec.m* function is used to correct inconsistencies in the background illumination, filter out noise and remove out-of-focus particles through morphological reconstruction. It is inspired by a code from the Matlab course "Image Processing in Matlab". The background illumination is corrected by first splitting the image into blocks and finding the minimum intensity of each block. The minimum intensity is assumed to be the intensity of the background and may in this way differ between each block, e.g. if a light source is used that yields a non-uniform intensity across the image. The width and height of each block is determined by the *BlockSize* input. The block size is an important parameter and should be related to the expected particle

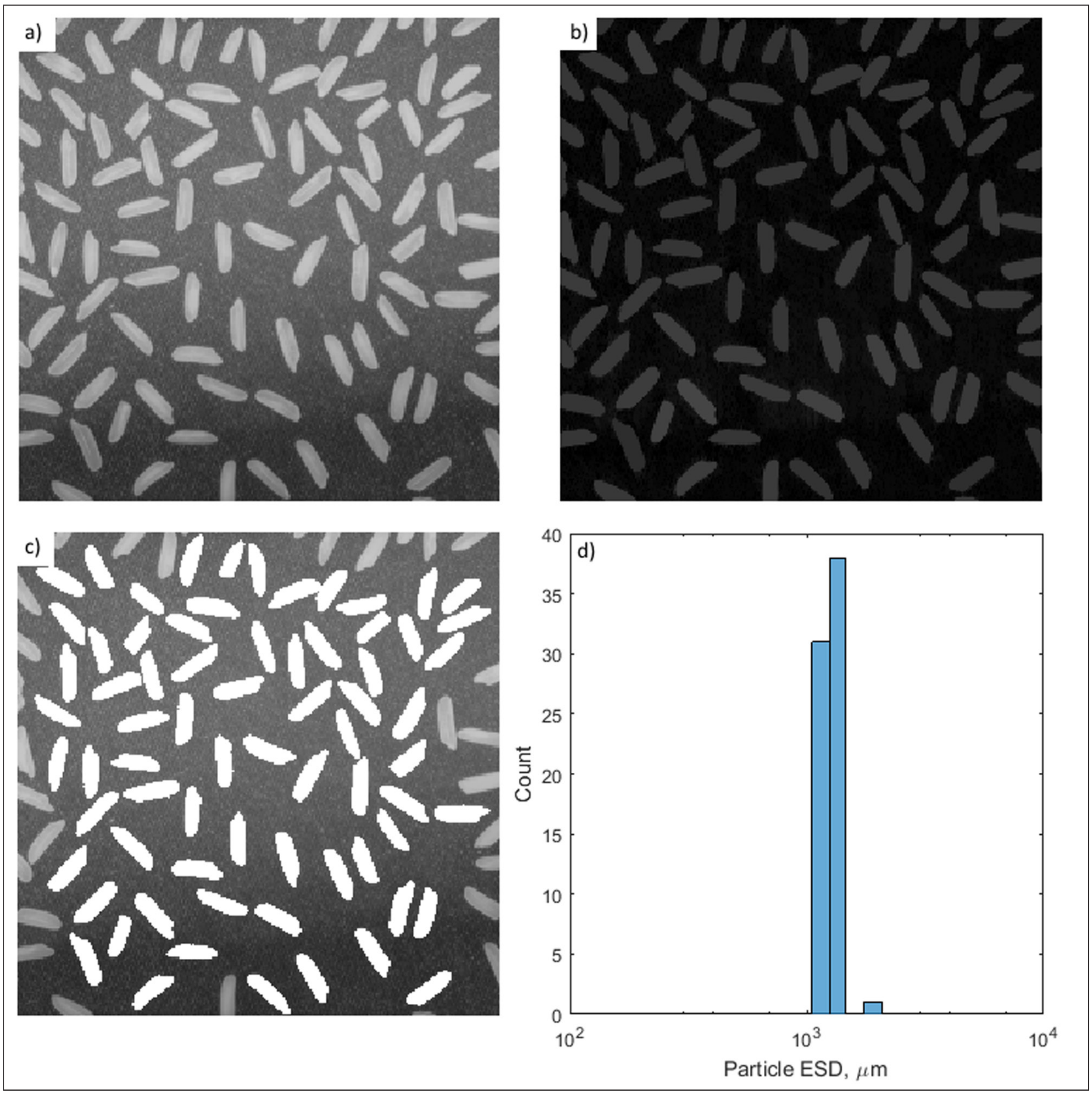

Figure 2: Four of the six processing steps using the Matlab example image Rice.png and the settings listed in **Table 1**. In Step 1, inputs are specified, in step 2, the original image is loaded (a), in step 3, the background is corrected and noise is removed (b), in step 4, thresholding is carried out and particles are identified (c) and in step 5, the particles are characterized, here exemplified by a histogram of sizes of the identified grains (d), plotted through step 6.

sizes. If, for instance, maximum particle sizes are expected to be around 1 mm, then the block size should be larger than this (e.g. 2 mm, which is the standard used here), so that individual large particles are not filling out an entire block. A new image is created in which values of each block is assigned the minimum value of this block in the original image. The resulting image is subtracted from the original image and thus the background illumination will become more uniform. In morphological reconstruction a marker image is created based on the input image, called the mask. In the marker image, all intensities of the mask image are subtracted by a constant value, specified by the *SubDiff* input. The Matlab function *imreconstruct* from the Image Processing Toolbox is then run. The marker image is dilated, meaning that the intensities are gradually increased, but the intensities of individual pixels can never be higher than they were in the original mask. This will identify peaks in the image, which in this case are in-focus particles. In this way, low-intensity objects such as out-of-focus particles or random noise can be completely removed without removing pixels that belong to particles or blobs that are a combination of low- and high-intensity features. Compare **Figure 2a+b** to see an example of background correction and image reconstruction. The corrected image in **Figure 2b** is darker but with less noise and a more uniform background than the input image.

Step 4: Thresholding and particle extraction

All functions used in this step are part of the Image Processing Toolbox in Matlab. The potentially noise-removed image has to be converted to a black-and-white image to allow the extraction of particles from the image. A black-and-white image is a logical array with nothing but zeros (black) and ones (white). The conversion is carried out using the *im2bw* function which requires an intensity threshold level to distinguish between particles in the foreground and the background. Numerous studies and automatic methods exist to find the optimal value of the threshold level, see Keyvani and Strom [2] for a good example. In the script presented here the threshold level is set to a fraction of the maximum intensity of the input image, determined by the *Lvl* input. In many cases, the maximum intensity of an image will be 255, but in some cases it might be lower due to differences in the water properties and electrical or power supply irregularities of the light source. This method has been applied after numerous tests using many different automatic thresholding methods and yields good results across environments and deployment configurations. Essentially, setting a threshold level can never be completely objective as it will be based on some degree of testing. Thus, the exact level specified in the *Lvl* input must also be based on visual comparisons of results using different values. Particles touching the border of the converted black-and-white image will be removed using the *imclearborder* function as it is unknown how big a part of these particles are outside the viewing volume. Connected components, in other words individual particles, are identified using the *bwconncomp* function and relevant particle characteristics for each individual component are subsequently calculated and stored using the *regionprops* function. See **Figure 2c** for an example of particle identifications in which grains that touch the border are also removed. The last part of step 4) is a second option for removing potentially out-of-focus particles. It removes all particles which do not have a single pixel with intensity above the *MaxInt* fraction of the maximum intensity of the image. The assumption behind this step is that particles in focus will always have a part that is illuminated by the laser sheet and get a high intensity value. If, however, the particles are loosely packed, low-density organic particles, it might not be fruitful to apply the *MaxInt* rule as such particles might not have a strong spectral signal. Thus, the maximum intensity rule can be excluded by setting *MaxInt* to 0.

Step 5: Particle characteristics calculations (optional)

This step may be skipped entirely by setting the *CalcPSD* input to 0. The size limits used for binning the particles according to equivalent spherical diameter (ESD) are loaded, ESD and equivalent spherical volume is calculated, the sizes are sorted in bins and the relative frequency of particles in each bin is calculated. The bins that are used are loaded from the *Bins.mat* file which needs to be located in the same folder as the one the code is located in. See **Figure 2d** for an example of a histogram of counts of particles in size bins after image processing. The volumes of individual particles are sorted according to the size bins and summed up to get the total volume in each bin. The volumes are recalculated to volume concentrations by dividing the particulate volume with the measuring volume obtained from the *ImHeight*, *ImWidth* and *MeasDepth* inputs. This gives the volume of particles per volume of medium, e.g. water. Mean diameters and standard deviations are calculated by converting all sizes to the phi-scale and using the statistical method of moments. The sphericity and the convexity, two shape parameters, are then calculated. Sphericity is the measure of the ratio between the major and minor axes of the best-fitting ellipsoid and is thus 1 for completely spherical particles and decreases towards 0 for increasingly rod-like or elongated particles. Convexity is a parameter often used in grain characterisations to describe the roughness of the surface of the particle. It is measured as the ratio, with a value of 0 to 1, between the perimeter of the convex hull, i.e., the smallest polygon inside which the object fits, of the particle and the perimeter of the actual particle. A convexity of 1 means that the surface of the particle is completely smooth, and convexities closer to 0 indicate longer particle perimeters in relation to the convex hull perimeter, indicating a rougher surface. Thus, the convexity is a relative but not a direct measure of the surface roughness. The solidity is calculated through the *regionprops* function in step 4). It provides an estimate of the overall fluffiness or porosity of a particle and is the ratio between the particulate area and the convex hull area.

Step 6: Result plotting (optional)

This step may be skipped entirely by setting the *PlotResults* input to anything but 1 or 2. If *PlotResults* is set to 1 this last step will produce four plots in individual figures and if it set to 2 it will produce one figure with four subplots. The four individual plots are the original image overlaid by the number assigned to each individual particle, the original image overlaid by the selected particles, a histogram of the ESD of the particles and a particle size frequency distribution. The four subplot figures are the original image, the original image overlaid by the selected particles, a histogram of ESD-values and the particle size frequency, see example in **Figure 3**.

Running the code on several images

The *ParChar_mult* code allows the user to run the particle characterization code on numerous images all from within the same file location. The location of images and the location where the results will be stored are specified through a pop-up interface. The code combines individual images into two table variables and saves these to the specified location whenever the total amount of particles that have been identified crosses 10,000, in order to save memory. This code can only run if the *CalcPSD* input is set to 1.

Influence of changing inputs

This section will give a very brief example of the influence of applying or excluding the optional image processing steps. The following examples are all based on the example image from the Pcam, available together with the

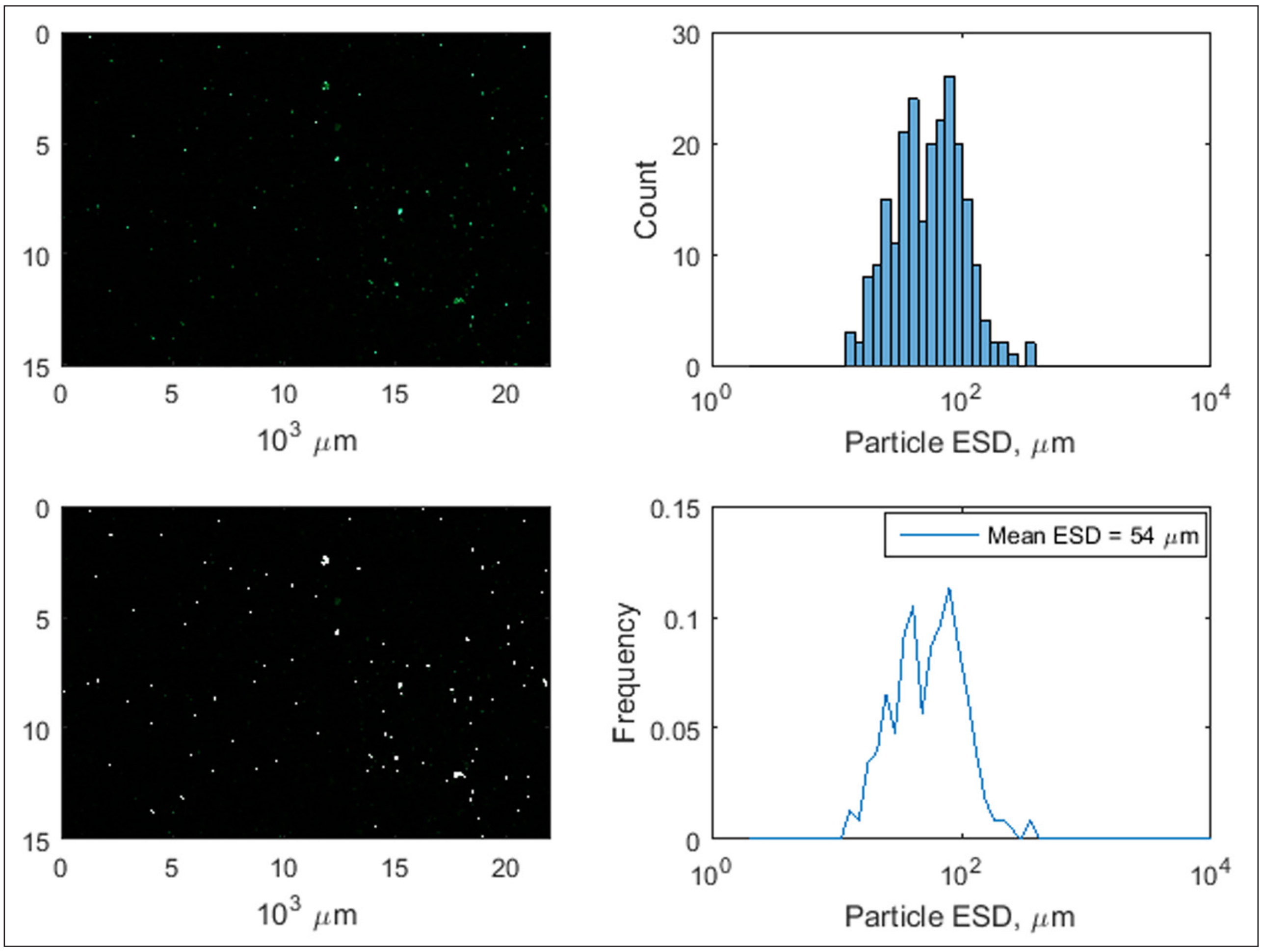

Figure 3: Example of the result of image processing using the ParChar code on an image from the Pcam of natural particles suspended in the water. The inputs for this example are shown in **Table 1**. Top left plot shows the original input image, bottom left plot shows the detected particles from the input images, top right plot shows the histogram count of particles in size bins and bottom right plot shows the frequency distribution of the sizes according to the size bins.

code. Five different setups are described and the resulting subplots and workspace variables are all available in the *Examples.zip* file which can be downloaded from github archive. For an example of the Pcam using the input values specified in **Table 1**, see **Figure 3**.

With a threshold level of 0.1 and no morphological reconstruction, no background illumination correction and no removal or exclusion of individual particles with low maximum intensity threshold, the result is rather noisy with a large number of identified particles with an area of only one pixel, see the Lvl01_Subdiff0_MaxInt0 files. The result is much less noisy and the small 1-pixel particles are removed if morphological reconstruction and background illumination correction is applied, see Lvl01_Subdiff50_MaxInt0 files. The result is also less noisy if morphological reconstruction and background noise is disabled but the maximum intensity threshold is applied, see Lvl01_Subdiff0_MaxInt05 files. The difference between the two latter cases is that a larger number of small, low-intensity particles are identified when *SubDiff* is used while more low-intensity parts of the larger particles are incorporated when *MaxInt* is used. When morphological reconstruction is carried out, background illumination is corrected and individual particles with low maximum intensity threshold are removed and the result shows no low-intensity particles and a larger exclusion of low-intensity parts of larger particles, see Lvl01_Subdiff50_MaxInt05 files. Finally, setting all parameters to the same values as the last example but using a threshold level of 0.2 will cause all found particles to be smaller, as the larger threshold means that more pixels will be excluded from the identifications, see Lvl02_Subdiff50_MaxInt05 files. The user is strongly encouraged to test the variables and the influence of applying or excluding the optional image processing steps based on the images being processed to find optimal parameter settings.

Quality control

The *ParChar* code has been tested on Matlab versions from R2012b and newer. It has not been run on other operating systems than Windows 7. The code has been used in a paper published in Scientific Reports [1]. In short, the code was found to perform very well when compared to other particle characterisation methods. Scientific Reports applies an open-access policy and I refer to the

Input variable	Rice.png	Pcam example image
ImHeight	20	15
ImWidth	20	22.5
MeasDepth	1	2
SubDiff	50	0
Lvl	0.6	0.1
MaxInt	0.5	0.5
CalcPSD	1	1
PlotResults	1	1

Table 1: Suggested code inputs to test if the code is working using either the rice grains example from Matlab or the Pcam example image supplied with the code.

supplementary information of this paper for a more thorough description of the quality control of the code, in terms of how well particle sizes are characterized in relation to other methods. The user can check if the code is working using the example image of rice grains built into Matlab. This would be done by browsing to the location of the Matlab example images (\\$Matlabdir$\$version$\toolbox\images\imdata) and selecting the rice.png file. Otherwise, the code can be run using the Pcam example image supplied with the code. The inputs shown in **Table 1** will yield particle size distributions and characterizations of the rice grains and the Pcam particles. 70 grains will be identified with a mean ESD of the grains of 1260 μm, see **Figure 2** as well. Note that this is based on an arbitrary definition of the field of view in the *ImHeight* and *ImWidth* inputs and thus does not explain the real size of the rice grains. 229 particles will be identified in the Pcam image with a mean ESD of 54.4 μm when the inputs stated in the table are used. The user may alter e.g. the *Lvl* input to see how this influences the outlines of each grain or particle and thus how it potentially can improve e.g. separation of individual grains.

(2) Availability

Operating system

Matlab 8.0 (R2012b), Matlab runs on Windows, Linux and Mac.

Programming language

Matlab 8.0 (R2012b), upward compatible.

Additional system requirements

Matlab 8.0 (R2012b), no specific computer requirements.

Dependencies

The Image Processing Toolbox in Matlab is required. Plotting requires the original version of the *imoverlay* function which is also available in the archive (*imoverlay_orig*). Note that this version is different from the imoverlay function available from the Matlab File Exchange, and the *ParChar* code only works with the original function.

A small script (*CompatibilityTest*) can be run to make sure the correct versions and files are present.

List of contributors

TNM programmed and tested the code and wrote this article.

Thorbjørn Joest Andersen, University of Copenhagen, commented on the code implementation and assisted with knowledge on other particle sizing systems.

Software location

Archive

Name: github.com
Persistent identifier: https://github.com/ThorNM/ParChar
Licence: CC-BY
Publisher: Thor Markussen
Version published: 1.0
Date published: 18/01/2016

Language

English

(3) Reuse potential

Numerous types of studies identify and characterize particles. In several of these fields camera systems are used with quite different image processing techniques. Examples are:

- Suspended particle transport in coastal aquatic environments: The correct characterization and size determination of suspended particles is important in order to understand the fluxes of particles and related substances and numerous studies have been carried out using camera systems [2, 3, 4, 5, 6, 7].
- Particle dynamics in marine environments: A very diverse range of particle types and sizes exist in marine waters and numerous studies use camera systems to describe the dynamics and particles [8, 9, 10, 11, 12, 13]
- Oil droplet studies: The size and behaviour of oil droplets is studied to understand the potential spread of oil after oil leaks. Brandvik [14] shows an example of such a study in which a combination of laser diffraction and camera systems is used for particle sizing.

The code presented here is not believed to be directly and universally applicable to all studies using camera systems, such as those referenced above. However, it can aide in getting more standardized ways of identifying and characterizing particles with camera systems. Thus, the aim has been to present a method that is freely and openly available for everyone to adapt, update and use in their applications. While the focus has been on using it for the Pcam system, users might have to adapt the code to their specific system. No guaranteed support mechanisms are in place for this code, but users are encouraged to contact the author to discuss specific application possibilities and issues.

Acknowledgements

Gabriel Herbst and Christian Winter from MARUM, University of Bremen, are greatly acknowledged for designing the Pcam system and for discussions on the applicability of this code. Steve Eddins from The Mathworks Inc. is acknowledged for creating the original imoverlay function and making it freely available on the Matlab File Exchange. The ImRec function is based on a code provided during the Matlab course "Image Processing with Matlab" and I would like to again acknowledge The Mathworks Inc. for making the code available.

Competing Interests

The author declares that he has no competing interests.

References

1. **Markussen, T N, Elberling, B, Winter, C** and **Andersen, T J** 2016 Flocculated meltwater particles control Arctic land-sea fluxes of labile iron. *Scientific Reports*. 6. DOI: http://dx.doi.org/10.1038/srep24033
2. **Keyvani, A** and **Strom, K** 2013 A fully-automated image processing technique to improve measurement of suspended particles and flocs by removing out-of-focus objects. *Computers & Geosciences*. 52. DOI: http://dx.doi.org/10.1016/j.cageo.2012.08.018
3. **Eisma, D, Schuhmacher, T, Boekel, H, Vanheerwaarden, J, Franken, H, Laan, M, Vaars, A, Eijgenraam, F** and **Kalf, J** 1990 A Camera and Image-Analysis System for Insitu Observation of Flocs in Natural-Waters. *Netherlands Journal of Sea Research*. 27(1). DOI: http://dx.doi.org/10.1016/0077-7579(90)90033-D
4. **Benson, T** and **French, J R** 2007 InSiPID: A new low-cost instrument for in situ particle size measurements in estuarine and coastal waters. *Journal of Sea Research*. 58(3). DOI: http://dx.doi.org/10.1016/j.seares.2007.04.003
5. **Syvitski, J P M** and **Hutton, E W H** 1996 In situ characteristics of suspended particles as determined by the Floc camera assembly FCA. *Journal of Sea Research*. 36(1–2). DOI: http://dx.doi.org/10.1016/S1385-1101(96)90783-2
6. **Dyer, K R** and **Manning, A J** 1999 Observation of the size, settling velocity and effective density of flocs, and their fractal dimensions. *Journal of Sea Research*. 41(1–2). DOI: http://dx.doi.org/10.1016/S1385-1101(98)00036-7
7. **Manning, A J, Friend, P L, Prowse, N** and **Amos, C L** 2007 Estuarine mud flocculation properties determined using an annular mini-flume and the LabSFLOC system. *Continental Shelf Research*. 27(8). DOI: http://dx.doi.org/10.1016/j.csr.2006.04.011
8. **Iversen, M H, Nowald, N, Ploug, H, Jackson, G A,** and **Fischer, G** 2010 High resolution profiles of vertical particulate organic matter export off Cape Blanc, Mauritania: Degradation processes and ballasting effects. *Deep Sea Research Part I: Oceanographic Research Papers*. 57(6). DOI: http://dx.doi.org/10.1016/j.dsr.2010.03.007
9. **Nowald, N, Iversen, M H, Fischer, G, Ratmeyer, V** and **Wefer, G** 2015 Time series of in-situ particle properties and sediment trap fluxes in the coastal upwelling filament off Cape Blanc, Mauritania. *Progress in Oceanography*. 137. DOI: http://dx.doi.org/10.1016/j.pocean.2014.12.015
10. **Jackson, G A, Maffione, R, Costello, D K, Alldredge, A L, Logan, B E** and **Dam, H G** 1997 Particle size spectra between 1 mu m and 1 cm at Monterey Bay determined using multiple instruments. *Deep-Sea Research Part I-Oceanographic Research Papers*. 44(11). DOI: http://dx.doi.org/10.1016/S0967-0637(97)00029-0
11. **Jackson, G A** 2005 Role of algal aggregation in vertical carbon export during SOIREE and in other low biomass environments. *Geophysical Research Letters*. 32(13). DOI: http://dx.doi.org/10.1029/2005gl023180
12. **Petrik, C M, Jackson, G A** and **Checkley, D M** 2013 Aggregates and their distributions determined from LOPC observations made using an autonomous profiling float. *Deep-Sea Research Part I-Oceanographic Research Papers*. 74. DOI: http://dx.doi.org/10.1016/j.dsr.2012.12.009
13. **Stemmann, L, Youngbluth, M, Robert, K, Hosia, A, Picheral, M, Paterson, H, Ibanez, F, Guidi, L, Lombard, F** and **Gorsky, G** 2008 Global zoogeography of fragile macrozooplankton in the upper 100-1000 m inferred from the underwater video profiler. *Ices Journal of Marine Science*. 65(3). DOI: http://dx.doi.org/10.1093/icesjms/fsn010
14. **Brandvik, P J, Johansen, O, Leirvik, F, Farooq, U** and **Daling, P S** 2013 Droplet breakup in subsurface oil releases – Part 1: Experimental study of droplet breakup and effectiveness of dispersant injection. *Marine Pollution Bulletin*. 73(1). DOI: http://dx.doi.org/10.1016/j.marpolbul.2013.05.020

How to cite this article: Markussen, T N 2016 Parchar – Characterization of Suspended Particles Through Image Processing in Matlab. *Journal of Open Research Software,* 4: e26, DOI: http://dx.doi.org/10.5334/jors.114

Submitted: 21 January 2016 **Accepted:** 01 July 2016 **Published:** 19 July 2016