Geometric morphometrics

Geometric morphometric data extraction

Manuel F. G. Weinkauf

26–27 August 2022

Contents

1 Introduction

The extraction of geometric morphometric data from 2D and 3D images is time intensive and requires specialized software. Some software packages are available that allow to extract geometric morphometric data to at least some degree. These include:

- FIJI for landmark data and technically outline data in 2D and 3D. For outlines, however, FIJI does not support choosing a defined starting point and does not support normalizing the number of outline points by default. Both requires either plugins or secondary data manipulation later. Exporting the raw outline coordinates is also not possible by default. FIJI also does not support to export any data in a native geometric morphometric format like .tps or .nts, but rather requires data to be reformatted later. FIJI is available here.
- tpsDig2 is a specialized program for both landmark and outline extraction. The program supports templates for landmark extraction and has a rudimentary thresholding for outline extraction, which may work with unaltered pictures if the contrast between object and background is large enough. It may be a good program for users who prefer a graphical user interface over command line, but at the moment it is only maintained by James Rohlf and the long-term future of the program is unknown. tpsDig2 is available here.
- SHAPE as an all-in-one package for elliptic Fourier analysis (EFA). SHAPE is not a pure data extraction software, but a complete but methodologically limited software to perform EFA. It automatically extracts outline data from images, which may contain a scale bar, and uses them for an internally calculated EFA. Images must have a comparatively high contrast to work, all objects are detected automatically, the version of EFA used does not need a fixed starting point, and the analysis is run automatically.

SHAPE does a decent job if all you need is EFA, but it does not allow to perform any other forms of data extraction or any other form of outline analysis and the raw outline coordinates of objects are not easily exported for use in other programs. SHAPE is available here.

1.1 Extracting geometric morphometric data in R

Instead of using this software, one can also use the image capabilities of R. Nevertheless, so far only one package supports geometric morphometric data extraction, viz. 'geomorph'. Within 'geomorph', two main functions exist:

- 'digitize2d()': This function reads a set of images (.jpg file type only) and theoretically allows to manually digitize landmarks in them. However, in it's current version it plots the extracted landmarks in the wrong position on the image, making it impossible to check whether they are set correctly or not. It is therefore of limited use at the moment (this may be addressed in a future update).
- 'digit.fixed()', 'digit.curves()', and 'digitsurface()': These functions allow to extract 3D landmark and semi-landmark data from a shape3D or mesh3D object. Extracting 3D data is complicated and these functions are therefore very useful should you have access to 3D data (e.g. microCT scans or laser scans). In this course, we will stick to the 2D case, which is by far the dominant use-case, and so will not deal with these functions.

Besides the 'geomorph' package, an entire suite of outline (semi-landmark) and landmark extraction functions is available in my source code file 'MorphometricExtraction_Functions.r'. These source codes are also available and perpetually updated in my GitHUB repository. I am also planning to publish them as a morphometric utility package in the near future. The two main functions in this source file are:

- 'OutlineExtraction()': This function allows you to read a set of images (.ppm or .tif file types), manually chose a starting point, and let the program extract the outline as a defined number of semi-landmarks automatically. The function needs a fairly high contrast between object and image, as R only supports rather basic auto-thresholding; a black object on white background (e.g. created with the much more advanced thresholding function of FIJI) is recommended. For the same reason, at the moment the code does not support .jpg images, as the mediocre image compression of this format leads to artifacts that throw off the auto-detect algorithm. Results are exported as an .nts file.
- 'LMExtract()': This function allows you to read a set of images (.ppm, .tif, or .jpg file types) and to manually extract a defined number of landmarks in all images. Results are exported as either an .nts or a .tps file.

2 Setting up the R session

We first need to load the functions necessary to extract outlines and landmarks from images.

```
#Set working directory
setwd("C:/R_Data/Erlangen_Morphometrics/Session1_DataExtraction")

#Load functions
source("MorphometricExtraction_Functions.r")
source("MorphoFiles_Function.r")
```

3 Extracting outlines (semi-landmarks)

3.1 Image preparation

The first step in outline semi-landmark data extraction is to prepare the images accordingly. Ideally, you want an image with a black object on white background, as for instance created via thresholding in FIJI. At the very least, you need a high contrast between a dark object and a light background (a basic thresholding is performed in the code).

Secondly, you need to rotate the images in a way that they will work with the algorithm. That means all image need to be rotated such that the morphologically homologous starting point of the outline is situated on the left side of the object. The algorithm then works as follows:

1. You click inside the object, due right of the morphologically homologous starting point.
2. From there, the algorithm moves pixel by pixel left, until it finds a high contrast between two neighbouring pixels (rim of the object). This position is chosen as the starting point of the outline.
3. The algorithm then searches across the surrounding pixels for the position of highest contrast to find the next pixel along the border. Is does this pixel by pixel until it ends up back at the starting point.

The following images (Figs 1–5) give an expression of the process.

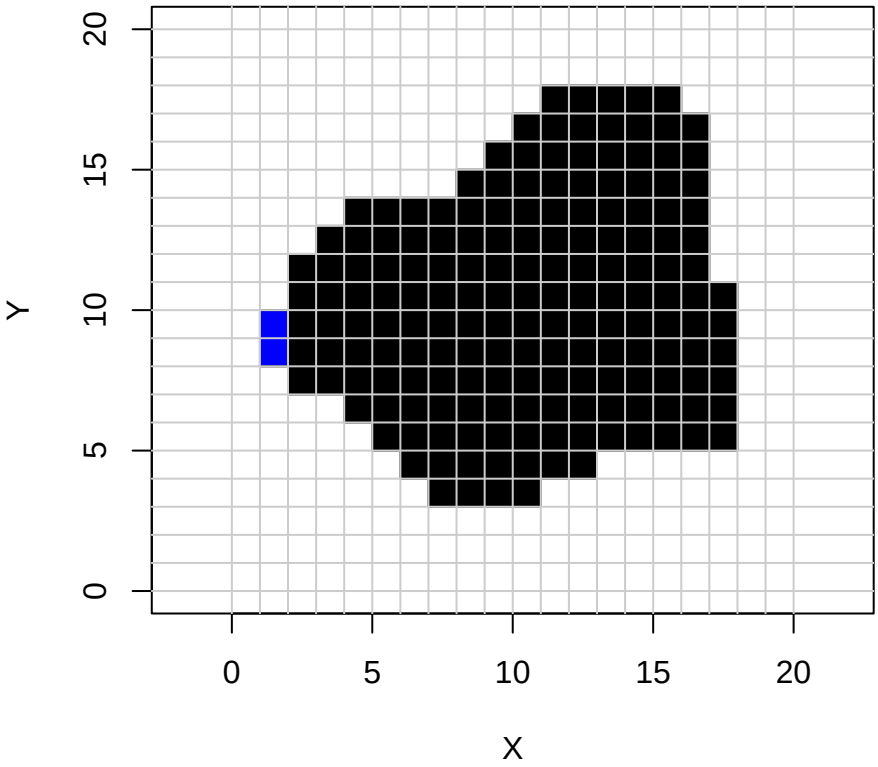

Figure 1: Object of interest, with grid overlay indicating pixels in the image. The chosen morphologically homologous starting point is indicated in blue.

The initial outcome of this procedure is a list of x- and y-coordinates for each pixel along the outline. To make the data usable for outline analysis procedures, these data are then homogenized to a defined number of equidistant outline semi-landmark coordinates by resampling.

3.2 Outline extraction

The actual image preparation process is too time consuming to be done here. I brought prepared images of belemnite micro-hooks (Fig. 6) to practice outline extraction in R. The images are from hooks of specimens from the species *Chondroteuthis wunnenbergi* from the lower Toarcian (Jurassic) of northern Germany.

EXERCISE 1: Study the documentation of the function 'OutlineExtraction' in 'MorphometricExtraction_Functions.r' and understand the parameters.

1. What would you think is the intended morphologically homologous starting point?
2. What do you think the parameter 'Scale' of the 'OutlineExtraction'-function should be set to?

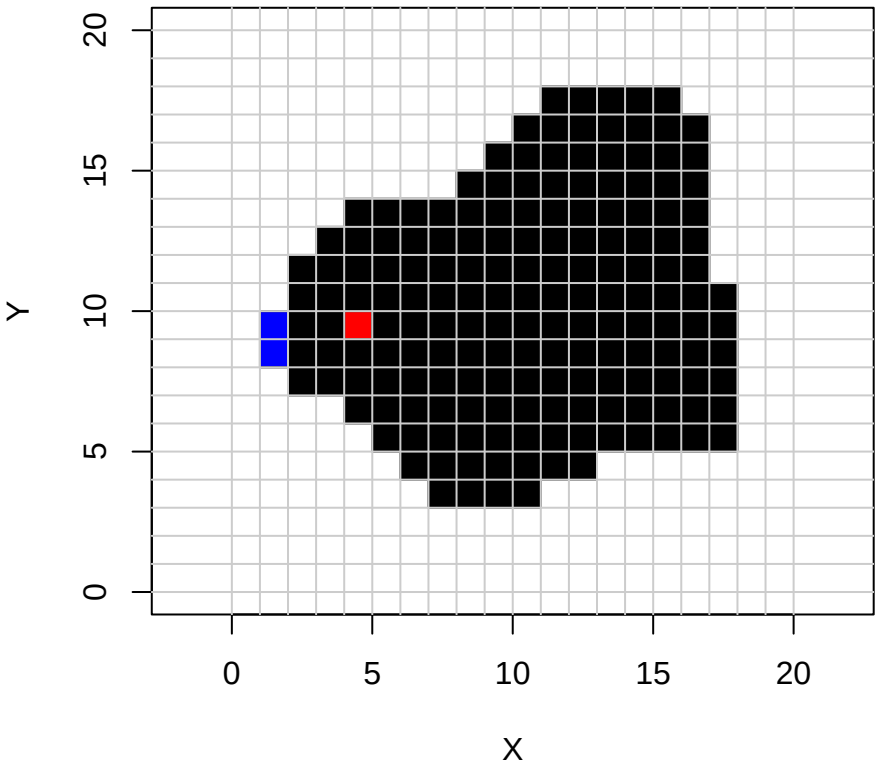

Figure 2: You click somewhere inside the object (red) due right of the morphologically homologous starting point (blue).

The 'OutlineExtraction'-function requires a vector that contains the names of the images from which you want to extract outlines. Instead of entering the image names manually, we can use R's capability to find files within a folder to compose this list automatically.

```
#Find all .tif files in the BelemniteHooks subfolder.
setwd("C:/R_Data/Erlangen_Morphometrics/Session1_DataExtraction")
Image.files<-list.files(path="BelemniteHooks", pattern=".tif")

#87 images is too many to work on in this exercise.
#We can randomly choose a subset of the image list.
#You can change the number in set.seed() to choose a different subset.
#You can change the 'size'-parameter in sample() to pick a different number of images.
set.seed(12345)
Image.work<-sample(Image.files, size=10)
Image.work
```

```
##  [1] "021.tif" "055.tif" "081.tif" "030.tif" "061.tif" "077.tif" "010.tif"
##  [8] "083.tif" "043.tif" "018.tif"
```

From this randomly chosen subset of images of belemnoid hooks, we can now extract the outlines.

EXERCISE 2: Use the function 'OutlineExtraction' in 'MorphometricExtraction_Functions.r' to extract outlines of some belemnoid hooks and save them as an .nts file. Pay attention to the following especially:

1. What is your choice of number of outline points? Why did you choose this number? Tip: You can just extract different numbers of outline points from one belemnoid hook and plot all outcomes to see, what is a good number.
2. What is your choice for the smoothing iterations? Smoothing reduces arbitrary surface irregularities,

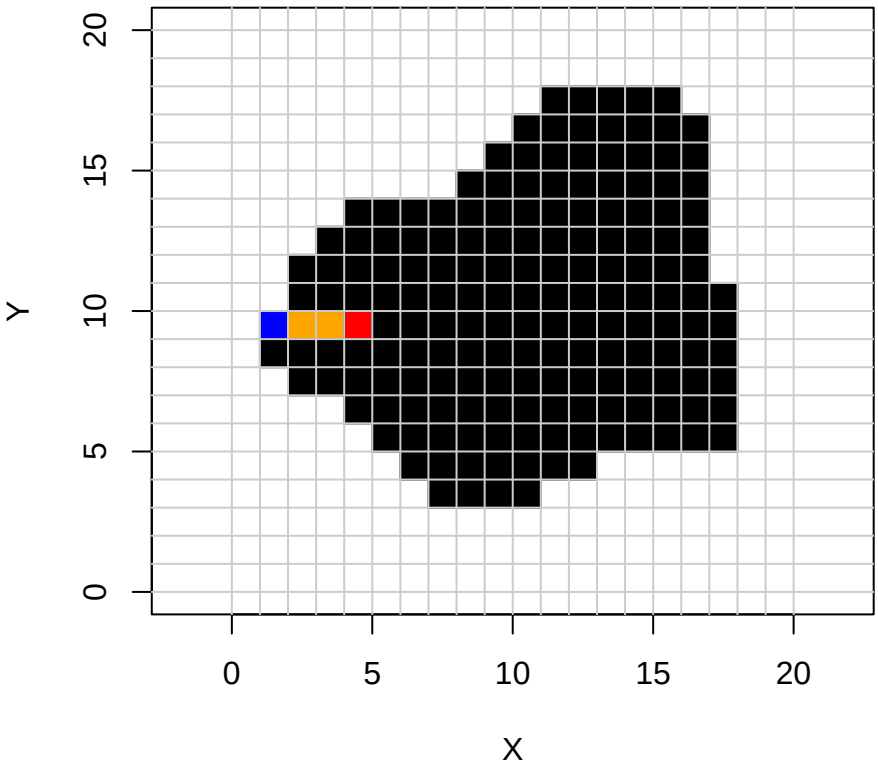

Figure 3: From the point you clicked (red) the algorithm moves left pixel by pixel (orange), until it finds the edge of the object and declares the corresponding pixel the morphologically homologous starting point of the outline (blue).

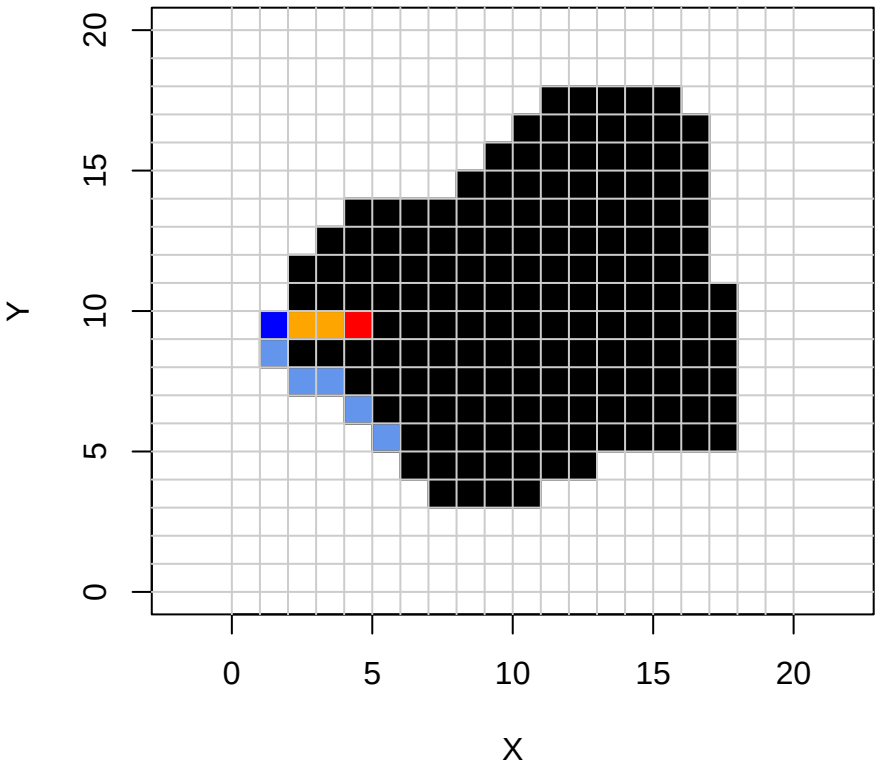

Figure 4: From the so-defined morphologically homologous starting point of the outline (blue), the algorithm moves counter-clockwise, finding the position of each pixel along the outline of the object based in the contrast to the background (light blue).

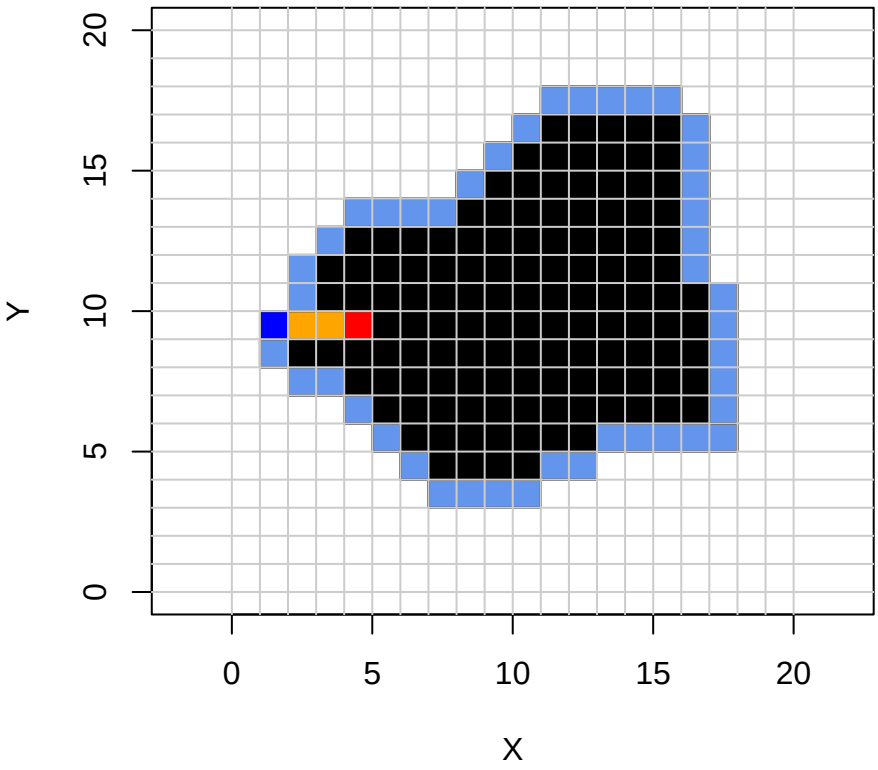

Figure 5: The outline extraction stops, when the outline (light blue) reaches back to the initial starting point (blue).

but too much smoothing deletes actual data. Tip: You can use different degrees of smoothing in one belemnoid hook and plot all outcomes to see, what is a good value.

```
#We can start by extracting the outline of just one of the selected images.
#We can repeat the extraction process for a number of reasonable values for
# OutlinePoints (e.g. 50, 100, 150, and 200) to find a good value.

setwd("C:/R_Data/Erlangen_Morphometrics/Session1_DataExtraction/BelemniteHooks")
OutlineExtraction(Images=Image.work[1], Output="PointNumberTest_50",
                  Specimen.Labels=NULL, OutlinePoints=50, Smoothing=1, Scale=TRUE,
                  RawVersion=TRUE)
OutlineExtraction(Images=Image.work[1], Output="PointNumberTest_100",
                  Specimen.Labels=NULL, OutlinePoints=100, Smoothing=1, Scale=TRUE,
                  RawVersion=TRUE)
OutlineExtraction(Images=Image.work[1], Output="PointNumberTest_150",
                  Specimen.Labels=NULL, OutlinePoints=150, Smoothing=1, Scale=TRUE,
                  RawVersion=TRUE)
OutlineExtraction(Images=Image.work[1], Output="PointNumberTest_200",
                  Specimen.Labels=NULL, OutlinePoints=200, Smoothing=1, Scale=TRUE,
                  RawVersion=TRUE)
```

After we extracted the outlines with different numbers of equidistant points, we can plot them (Fig. 7) and compare them to the original (Fig. 8). Conservatively, we aim at the lowest number of outline points that can still recreate the shape sufficiently.

```
#We can now plot all versions to visually check for a good number of outline points
```

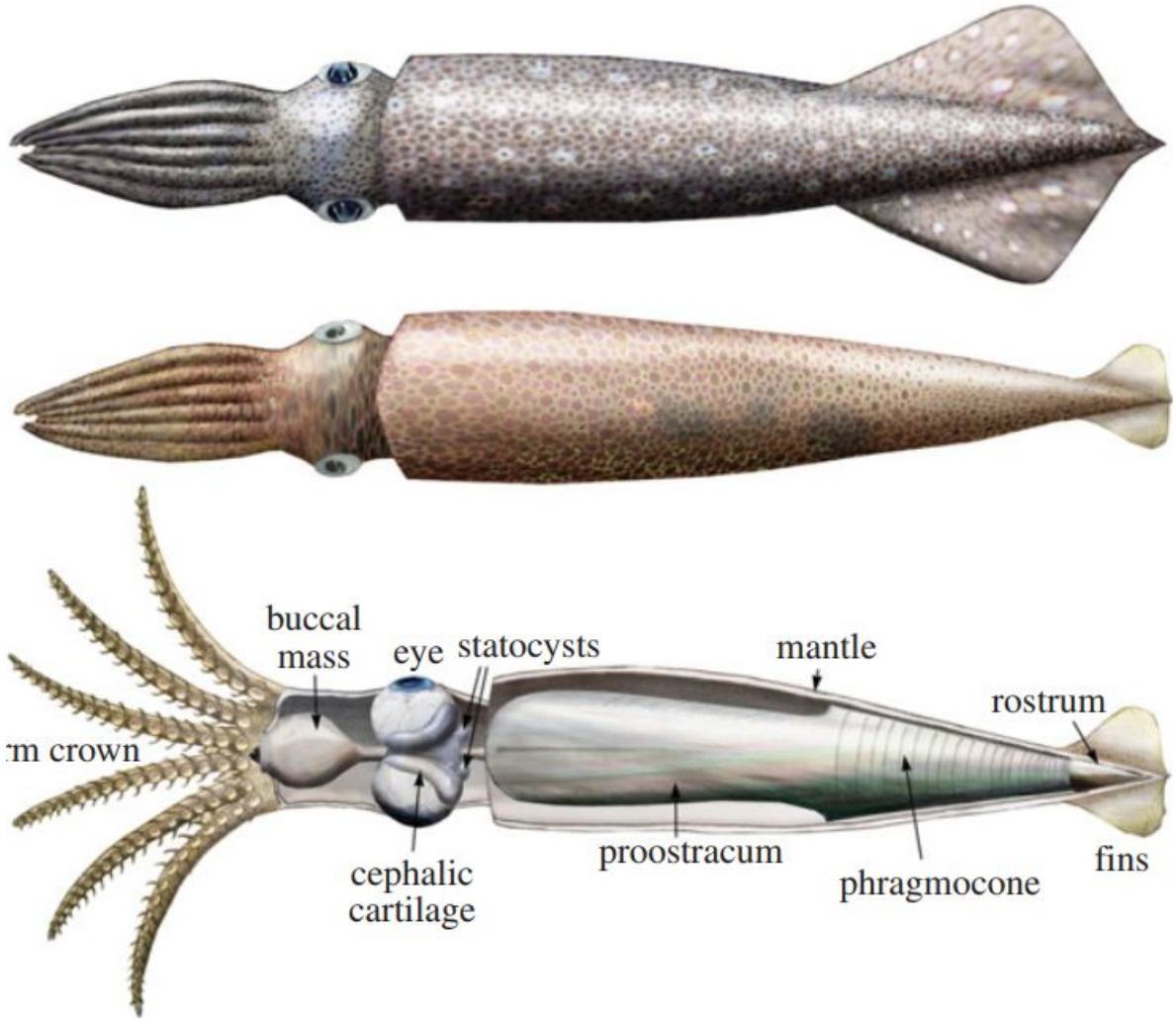

Figure 6: Depiction of a belemnite animal with armhooks visible in the lowermost picture. The armhooks in belemnites served the same purpose as suckers in modern coleoidea. From Klug et al. (2016) Adaptations to squid-style high-speed swimming in Jurassic belemnitids. *Biol. Lett.* 12: 20150877

```
setwd("C:/R_Data/Erlangen_Morphometrics/Session1_DataExtraction/BelemniteHooks")
Outline.50<-Read.NTS("PointNumberTest_50_Raw.nts")
Outline.100<-Read.NTS("PointNumberTest_100_Raw.nts")
Outline.150<-Read.NTS("PointNumberTest_150_Raw.nts")
Outline.200<-Read.NTS("PointNumberTest_200_Raw.nts")

layout(matrix(1:4, 2, 2, byrow=TRUE))
plot(Outline.50[,"x",1], Outline.50[,"y",1], type="p", xlab="x", ylab="y", asp=1,
     pch=16, cex=0.6, main="50 points")
lines(Outline.50[,"x",1], Outline.50[,"y",1], lwd=0.5, col="grey50")
plot(Outline.100[,"x",1], Outline.100[,"y",1], type="p", xlab="x", ylab="y", asp=1,
     pch=16, cex=0.6, main="100 points")
lines(Outline.100[,"x",1], Outline.100[,"y",1], lwd=0.5, col="grey50")
plot(Outline.150[,"x",1], Outline.150[,"y",1], type="p", xlab="x", ylab="y", asp=1,
     pch=16, cex=0.6, main="150 points")
lines(Outline.150[,"x",1], Outline.150[,"y",1], lwd=0.5, col="grey50")
plot(Outline.200[,"x",1], Outline.200[,"y",1], type="p", xlab="x", ylab="y", asp=1,
     pch=16, cex=0.6, main="200 points")
lines(Outline.200[,"x",1], Outline.200[,"y",1], lwd=0.5, col="grey50")
```

A comparison with the original image shows, that 100 outline points are likely enough to recreate the shape sufficiently for use in the remaining belemnite hooks.

Now that we settled on a particular number of outline points, we need to define a good value for the smoothing parameter. Like with the outline points, we try a few values and compare them visually.

```
#We can start by extracting the outline of just one of the selected images.
#We can repeat the extraction process for a number of reasonable values for
# Smoothing (e.g. 2, 3; 1 and 0 are already available from the first round) to find
# a good value.

setwd("C:/R_Data/Erlangen_Morphometrics/Session1_DataExtraction/BelemniteHooks")
OutlineExtraction(Images=Image.work[1], Output="SmoothTest_2",
                  Specimen.Labels=NULL, OutlinePoints=100, Smoothing=2, Scale=TRUE,
                  RawVersion=FALSE)
OutlineExtraction(Images=Image.work[1], Output="SmoothTest_3",
                  Specimen.Labels=NULL, OutlinePoints=100, Smoothing=3, Scale=TRUE,
                  RawVersion=FALSE)
```

Again, we can compare the different smoothing degrees visually by plotting and choose the one that eliminates arbitrary irregularities without deleting important image information (Fig. 9).

```
#We can now plot all versions to visually check for a good degree of smoothing

setwd("C:/R_Data/Erlangen_Morphometrics/Session1_DataExtraction/BelemniteHooks")
Outline.Raw<-Read.NTS("PointNumberTest_100_Raw.nts")
Outline.1<-Read.NTS("PointNumberTest_100.nts")
Outline.2<-Read.NTS("SmoothTest_2.nts")
Outline.3<-Read.NTS("SmoothTest_3.nts")

layout(matrix(1:4, 2, 2, byrow=TRUE))
plot(Outline.Raw[,"x",1], Outline.Raw[,"y",1], type="l", xlab="x", ylab="y", asp=1,
     main="No smoothing")
plot(Outline.1[,"x",1], Outline.1[,"y",1], type="l", xlab="x", ylab="y", asp=1,
     main="1 iteration")
plot(Outline.2[,"x",1], Outline.2[,"y",1], type="l", xlab="x", ylab="y", asp=1,
```

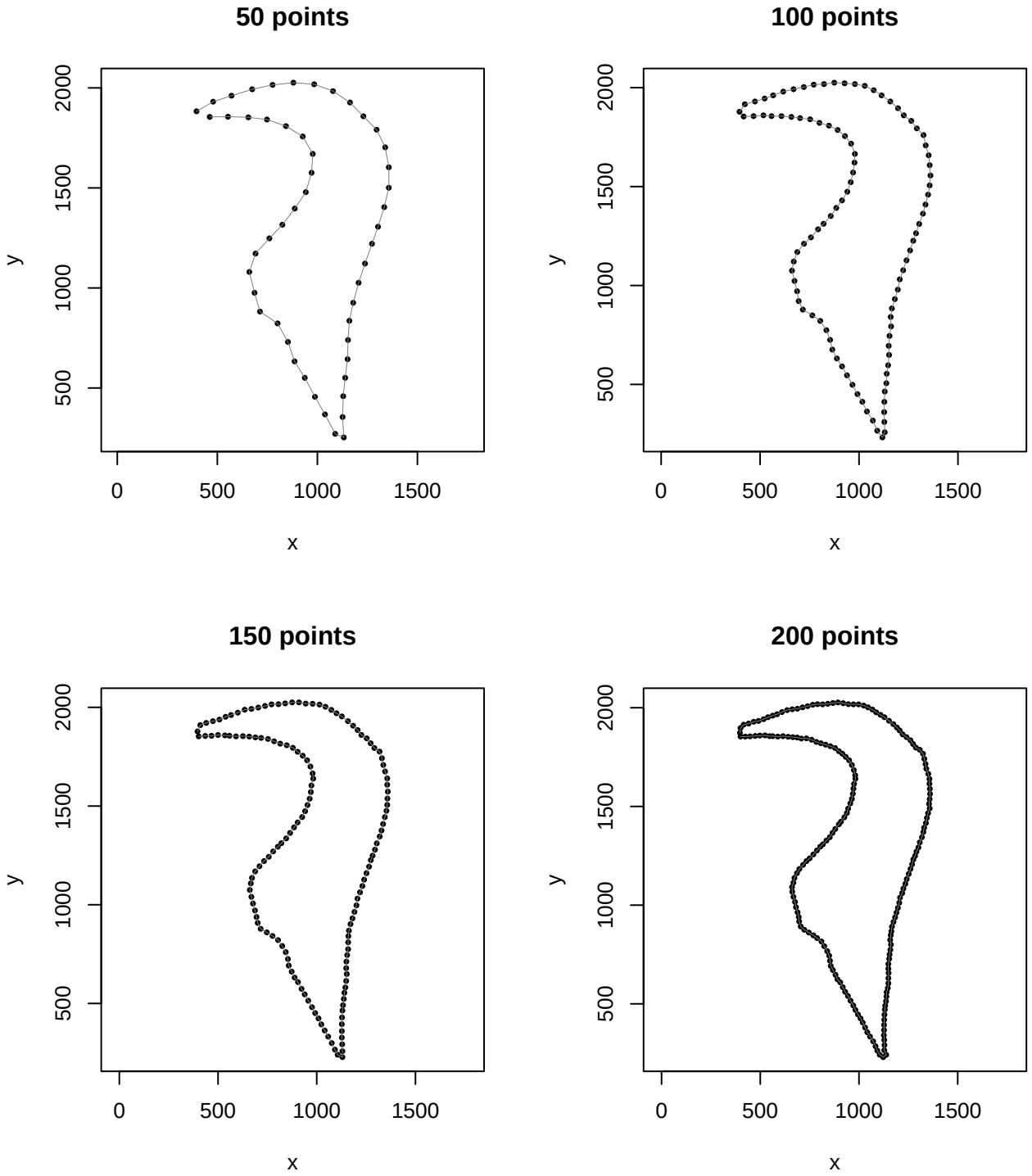

Figure 7: Comparison of different numbers of equidistant outline points for belemnite hooks.

Figure 8: Original image of this belemnite hook for comparison.

```
     main="2 iterations")
plot(Outline.3[,"x",1], Outline.3[,"y",1], type="l", xlab="x", ylab="y", asp=1,
     main="3 iterations")
```

Whichever values you choose for the number of outline points and the degree of smoothing, you could now proceed to extract outlines from more belemnoid hooks using the chosen parameters.

4 Extracting landmarks

4.1 Image preparation

The extraction of landmark data is much more straightforward, partly because at this time it must be done mainly by hand. There are no technical requirements for the images themselves, other than the format can be read by whatever software is used for the extraction.

A few preparations are helpful, however:

1. You want the images to be well contrasted, in order to be able to see the details well enough to easily identify the landmark features.
2. It is advisable that images are rotated such that all objects lie in approximately the same orientation in the image and are consistent in left–right alignment. This is both not necessary for the analyses, the superimposition processes applied to the data will eliminate all inconsistencies here, but it makes it much easier for you to extract landmarks without making mistakes.

As an exercise, we will here use skulls of some species of the family Mustelidae (Fig. 10) for landmark extraction.

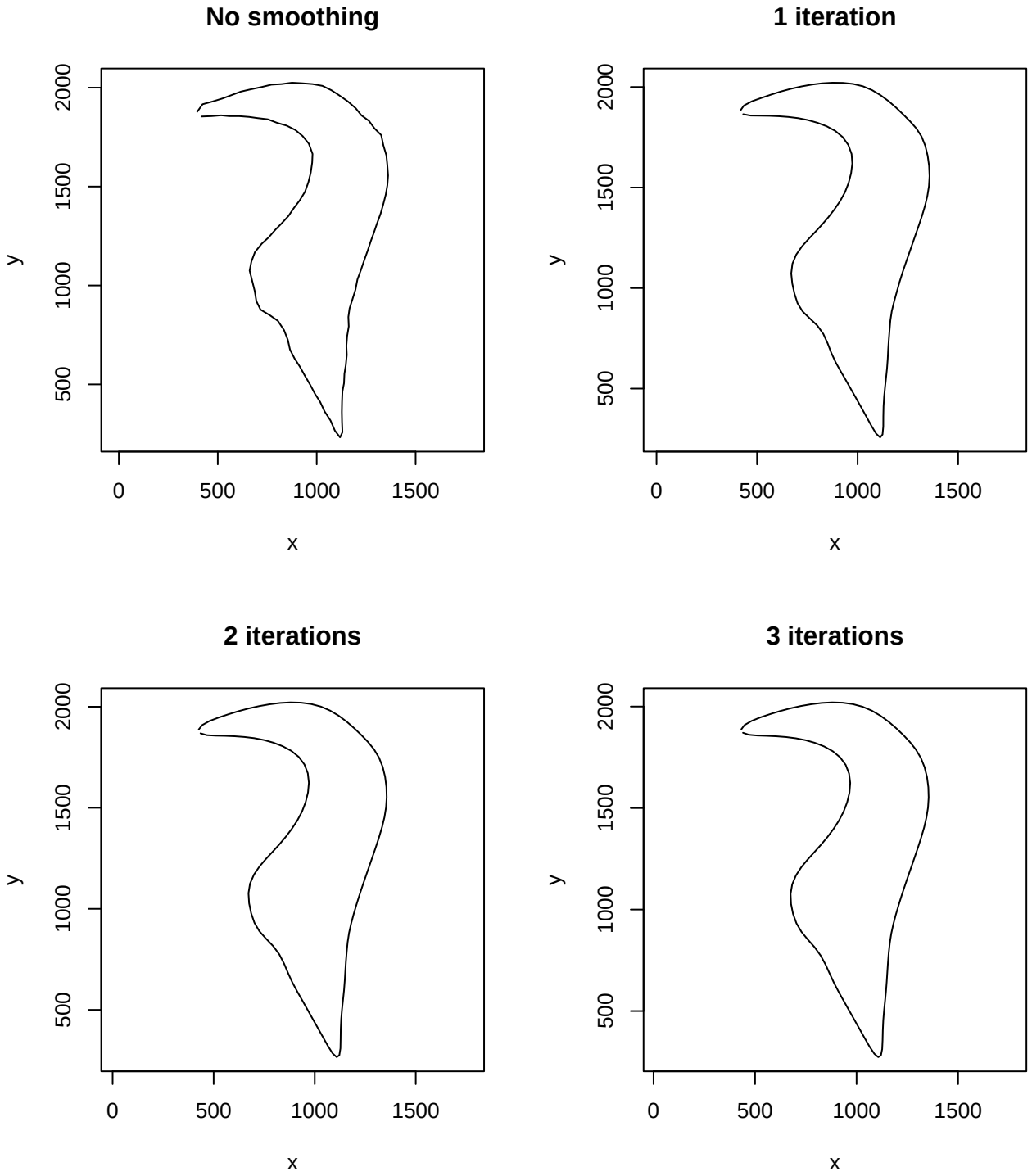

Figure 9: Comparison of different numbers of outline smoothing for belemnite hooks.

Figure 10: Examples of Mustelidae species. From https://sites.google.com/site/carnivoramustelidae10/.

4.2 Preparing a template for landmark positions

One of the most important steps in landmark data extraction is to develop a good template for the landmarks that should be extracted. As a reminder, different types of landmarks can be distinguished:

- **Type I landmarks/anatomical landmarks**: Well defined and biologically homologous points, for instance points where bones meet, nerve canal openings, tubercles.
- **Mathematical landmarks**: Defined on the basis of geometric properties
  - **Type II landmarks**: Points that are defined by a local property, such as maximum curvature of the shell
  - **Type III landmarks**: Landmarks at extremal points of a structure (e.g. the tip of the finger bone) or at constructed points (e.g. the centroid of the eye cavity)

When developing the template for landmark extraction, keep in mind what defines a good landmark of any type:

- It is well defined, i.e. easily and un-arbitrarily identifiable in every specimen.
- It is well delimited, i.e. it is small and sharp and therefore its position is unambiguous.
- It is omnipresent, i.e. you will be able to in all specimens/species involved in the study.

EXERCISE 3: Develop a suitable template for the extraction of landmark data from skulls of specimens of the family Mustelidae.

The image set contains three images each belonging to the species *Martes foina* (beech marten), *Meles meles* (European badger), *Mustela erminea* (stoat), *Mustela lutriola* (European mink), and *Mustela nivalis* (least weasel).

One possible landmark template could look like shown in Fig. 11 and Table 1.

Figure 11: Possible landmark template for skulls of Mustelidae (compare Table 1).

Table 1: Type and description of all landmarks in landmark template for skulls of Mustelidae (compare Fig. 11).

No.	Type	Description
1	I	Bottom of nuchal crest
2	I	Dorsal rim of occipital condyle
3	I	Border between occipital condyle and auditory bulla
4	I	Frontal rim of auditory bulla
5	I	Mandibular nerve canal
6	I	Mandibular nerve canal
7	I	Frontal suture between Nasale and Maxilare
8	II	Supra-orbital spike

4.3 Landmark extraction

EXERCISE 4: Study the documentation of the function 'LMExtract' in 'MorphometricExtraction_Functions.r' and understand the parameters.

The 'LMExtract'-function requires a vector that contains the names of the images from which you want to extract landmarks. Instead of entering the image names manually, we can use R's capability to find files within a folder to compose this list automatically.

```
#Find all .jpg files in the Mustelidae subfolder.
setwd("C:/R_Data/Erlangen_Morphometrics/Session1_DataExtraction")
Image.files<-list.files(path="Mustelidae", pattern=".JPG")
Image.files
```

```
##  [1] "MartesFoina_1_lateral.JPG"     "MartesFoina_2_lateral.JPG"
##  [3] "MartesFoina_3_lateral.JPG"     "MelesMeles_1_lateral.JPG"
##  [5] "MelesMeles_2_lateral.JPG"      "MelesMeles_3_lateral.JPG"
##  [7] "MustelaErminea_1_lateral.JPG"  "MustelaErminea_2_lateral.JPG"
##  [9] "MustelaErminea_3_lateral.JPG"  "MustelaLutreola_1_lateral.JPG"
## [11] "MustelaLutreola_2_lateral.JPG" "MustelaLutreola_3_lateral.JPG"
## [13] "MustelaNivalis_1_lateral.JPG"  "MustelaNivalis_2_lateral.JPG"
## [15] "MustelaNivalis_3_lateral.JPG"
```

From these images of Mustelidae skulls, we can now extract the landmarks according to the chosen template.

EXERCISE 5: Use the function 'LMExtract' in 'MorphometricExtraction_Functions.r' to extract outlines of Mustelidae skulls and save them as a .tps file.

```
setwd("C:/R_Data/Erlangen_Morphometrics/Session1_DataExtraction/Mustelidae")
LMExtract(Images=Image.files, Output="Mustelidae_Landmarks", Specimen.Labels="Names",
          Scale=TRUE, ScaleParam=NULL, N=8, Export="TPS")
```

5 Future developments

As noted before, I am trying to publish these functions in a morphometric utility package in the near future. I would be happy if you could take the time to anonymously answer this survey and tell me, what you think about the functions and what you would like to see included in future iterations.