

# User Manual

## EvoOeko-3D-Tracking

## Preliminary remarks

This manual serves as a User Guide for a half-automatic 3D movement analysis setup.

It lists the recommended hardware components as well as the software components of the 3D movement analysis system, explains the calibration setup and the video analysis in details.

The important MATLAB commands are characterized by grey boxes whereas important links are written in bold.

The graphical figures provide a better overview.

The associated MATLAB files can be downloaded via

[www.github.com/HorstmannM/3DTracking](https://www.github.com/HorstmannM/3DTracking)

---

If there are any questions, please contact:

Martin Horstmann (Martin.Horstmann@rub.de),

Sina M. Becker (Sina.Becker@rub.de),

Linda C. Weiss (Linda.Weiss@rub.de)

## Table of Content

Preliminary remarks .....	2
Table of Content.....	3
1. 3D Movement Analysis Setup .....	4
1.1. Hardware .....	4
1.2. Software .....	5
1.3. Network and slot assignment.....	6
2. Getting started .....	7
2.1. Installing the support package for Raspberry Pi .....	7
2.2. Installing the Computer Vision System Toolbox™ .....	7
2.3. Calibrating the system.....	8
2.4. Running the 3D Tracking file .....	18
3. GUI.....	19
4. Information Field .....	20
5. Camera Field.....	20
6. Video Field .....	21
6.1. Video.....	21
6.1.1. Video settings .....	21
6.1.2. Video recording .....	22
6.2. Position detection .....	23
6.2.1. Single video pair .....	24
6.2.2. Multiple video pairs.....	25
6.3. 2D .....	25
6.3.1. Synchronisation .....	25
6.3.2. 2D tracking.....	26
6.4. 3D .....	29
6.4.1. 3D calculation .....	29
6.4.2. 3D plot .....	29
6.5. Results .....	30
6.5.1. Write results and View results .....	30
7. Option to use a second computer for analyses.....	31
8. Suggestions for debugging and FAQ.....	32

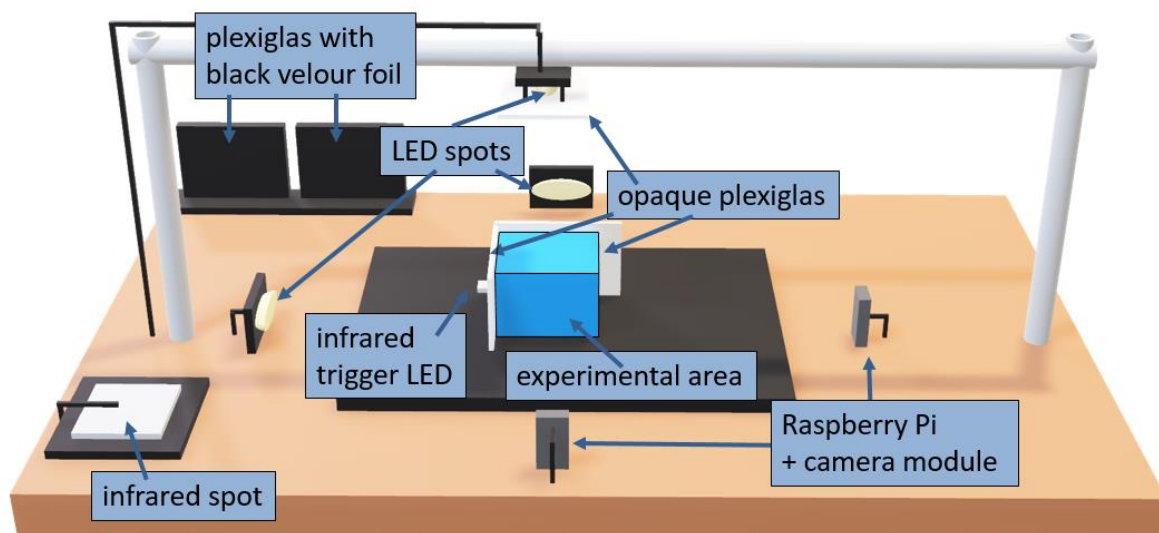
# 1. 3D Movement Analysis Setup

## 1.1. Hardware

The enclosed MATLAB file was designed to enable half-automatic 3D movement analysis, of e.g. five small aquatic organisms, with Raspberry Pi cameras. Therefore, we recommend the following hardware components. Of course, the user can individually adapt the setup.

### Hardware components

- Computer system requirements:
  - Windows (Windows 7 Service Pack 1 or higher), Mac (MacOS El Capitan or higher) or Linux (e.g. Ubuntu 14.04 LTS or higher)
  - Intel or AMD x86-64 processor with four logical cores
  - 4-6 GB of HDD space for the software installation; if possible SSD
  - 8 GB RAM
  - 2 displays
  - speakers
- 2x Raspberry Pi 3 Model B V 1.2 + 2x Raspberry Pi Camera module NoIR V2
- 2x 4GB SD card
- 1x commercial router
- 3x commercial network cables
- 1x low reflection table
- Light source:
  - 1x customized infrared LED spot, which can be placed on top of the experimental area
  - 1x infrared trigger LED to synchronise the videos
  - 3x white LED spots (10 W)
- 2x opaque plexiglas (white background)
- 2x opaque plexiglas with black velour foil (black background)
- 1x experimental area, e.g. an aquarium



**Figure 1: Recommended setup.** The recommended setup consists of two Raspberry Pi 3 equipped with the Raspberry Pi camera module NoIR V2, three white LED spots, one infrared LED spot as well as one infrared trigger LED, backgrounds with opaque plexiglas and black velour. The experimental area is an aquarium. The whole system stands on a low reflection table and is connected with a computer system via a router.

## 1.2. Software

You need the following MATLAB files to work with the EvoOeko-3D-Movement analysis setup.

1. **Tracking3D\_v2.m**
2. **CalibrationOfCameraSetUp.m**

The files can be downloaded from:

[www.github.com/HorstmannM/3DTracking](https://github.com/HorstmannM/3DTracking)

The enclosed MATLAB files run under **MATLAB R2018a**. We recommend installing all available components. The software can be downloaded from e.g. the official mathworks website:

[\*\*https://de.mathworks.com/downloads/\*\*](https://de.mathworks.com/downloads/)

Additionally, you have to install **FFmpeg** and Microsoft Office Excel on your computer system:

<https://www.ffmpeg.org/>

<https://www.office.com/>

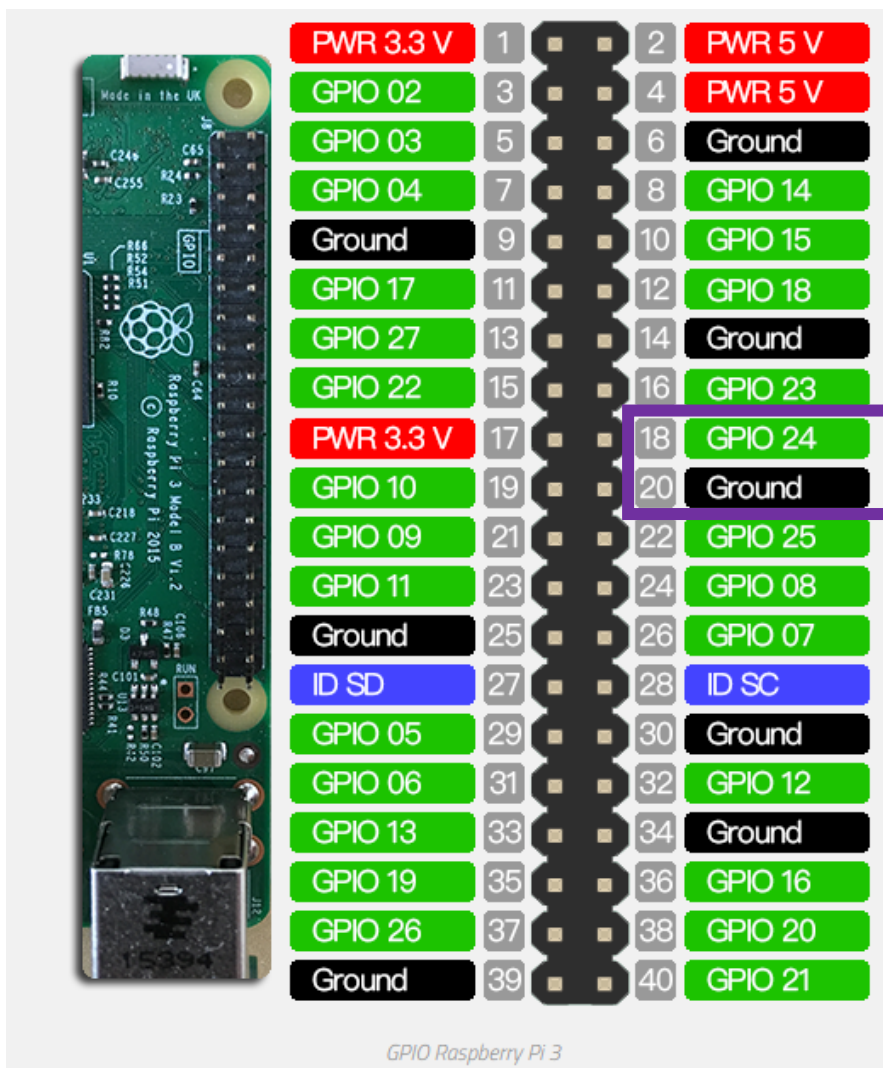
### 1.3. Network and slot assignment

Please, connect your computer system and the two Raspberry Pi 3 Model B V1.2 to the router. Additionally, the infrared trigger LED must be connected with one Raspberry Pi via the GPIO connector. It is needed to synchronize the videos during the analysis later on. We connected the infrared trigger LED with our “first” Raspberry Pi (Pi red/Rpi1). According to the default settings in the **Tracking3D\_v2.m** file, we recommend the following slot assignment. Of course, you can adjust it to your system.

```

394 %turn on and off the infrared LED to synchronize videos later on
395-     pause(1.5)
396-     writeDigitalPin(getappdata(h.main, 'rpil'), 24, 1);
397-     pause(0.5)
398-     writeDigitalPin(getappdata(h.main, 'rpil'), 24, 0);

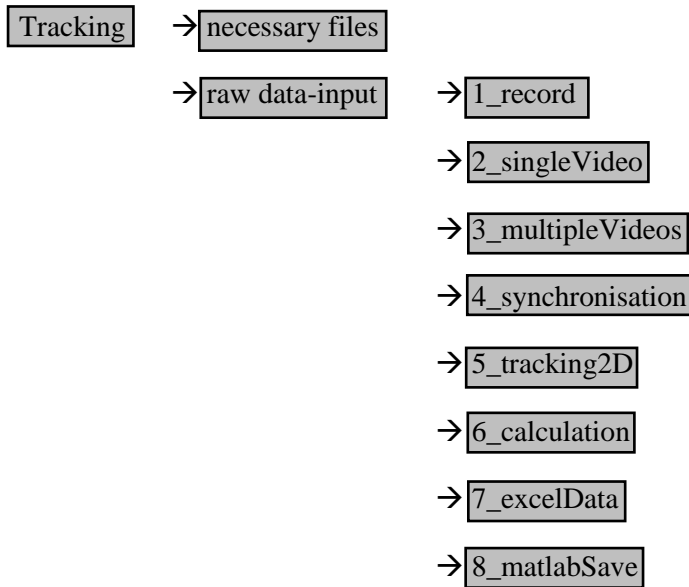
```



**Figure 2: GPIO Raspberry Pi 3 Model B V 1.2:** The infrared trigger LED must be connected with one Raspberry Pi 3 via the GPIO connector. It is needed to synchronize the videos during the analysis. We recommend the framed slot assignment. (<http://techgeeks.de/raspberry-pi-erste-schritte-und-uebersicht/>)

## 2. Getting started

Making sure that everything works smoothly, create the following folder structure:



Next, you have to set the path. Open MATLAB R2018a, click on “Home” and then on “Set path”. Click on “Add with Subfolders” and choose the “Tracking” folder.

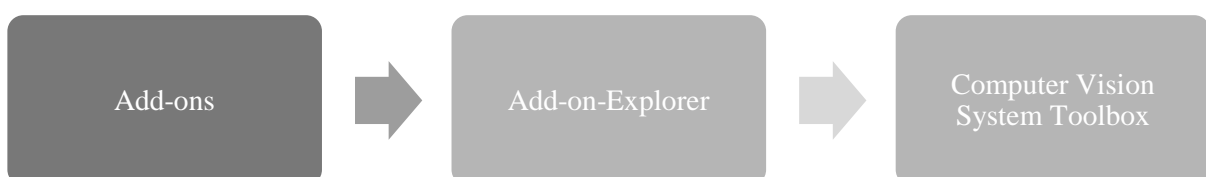
### 2.1. Installing the support package for Raspberry Pi

The Raspberry Pi package can be launched using the toolstrip. Click on “Add-ons” and then “Get Hardware Support Packages”. Navigate to the required package “MATLAB Support Package for Raspberry Pi Hardware” and follow the installation setup.



### 2.2. Installing the Computer Vision System Toolbox™

The Computer Vision Toolbox package can be launched using the Add-On-Explorer. Click on “Add-ons” and navigate to the required package “Computer Vision System Toolbox” and follow the installation setup.



## 2.3. Calibrating the system

Before you can start with the video analysis, you have to calibrate your system. A precise calibration is very important to avoid huge errors during the analysis of 3D data. We recommend using the checkerboard procedure.



The calibration can be conducted with the enclosed **CalibrationOfCameraSetUp\_v2.m** file and a MATLAB application, the **Camera Calibrator**. Please, install the **Computer Vision System Toolbox** first (see 2.2). The following steps will explain the calibration in detail.

### (1) The checkerboard procedure via the Camera Calibrator



#### (a) Printing the checkerboard pattern

The checkerboard pattern can be printed via MATLAB. Please, use the following command.

```
open checkerboardPattern.pdf
```

#### (b) Taking images for the calibration

You can execute the associated part (lines 1-56) of the **CalibrationOfCameraSetUp.m** file to conduct the calibration. Please, clear everything before the start of the camera calibration.

```

1  %CALIBRATION OF CAMERA SET UP
2  %clear everything before start
3- clear rpi1           %just clear raspi 1
4- clear rpi2           %just clear raspi 2
5- clear                %clear all
6- clear cam1           %just clear raspi-cam 1
7- clear cam2           %just clear raspi-cam 2
8- clc                  %clear command window
  
```

Notice, to enhance clarity, one Raspberry Pi and its related components are coded with “1” and the other one with “2”.

First Raspberry Pi = Pi red = cam1 = rpi1      ➔ Image1 or originX1, etc

Second Raspberry Pi = Pi green = cam2 = rpi2      ➔ Image2 or originX2, etc



Next, you have to check the IP address of the two Raspberry Pi. Using headphones, you can listen to the current IP address of each Raspberry Pi right after the booting. Enter them in lines 11 and 12 of the **CalibrationOfCameraSetUp.m** file.

```
10 %set up connection
11- rpi1 = raspi('192.168.178.22', 'pi', 'raspberrypi');
12- rpi2 = raspi('192.168.178.23', 'pi', 'raspberrypi');
```

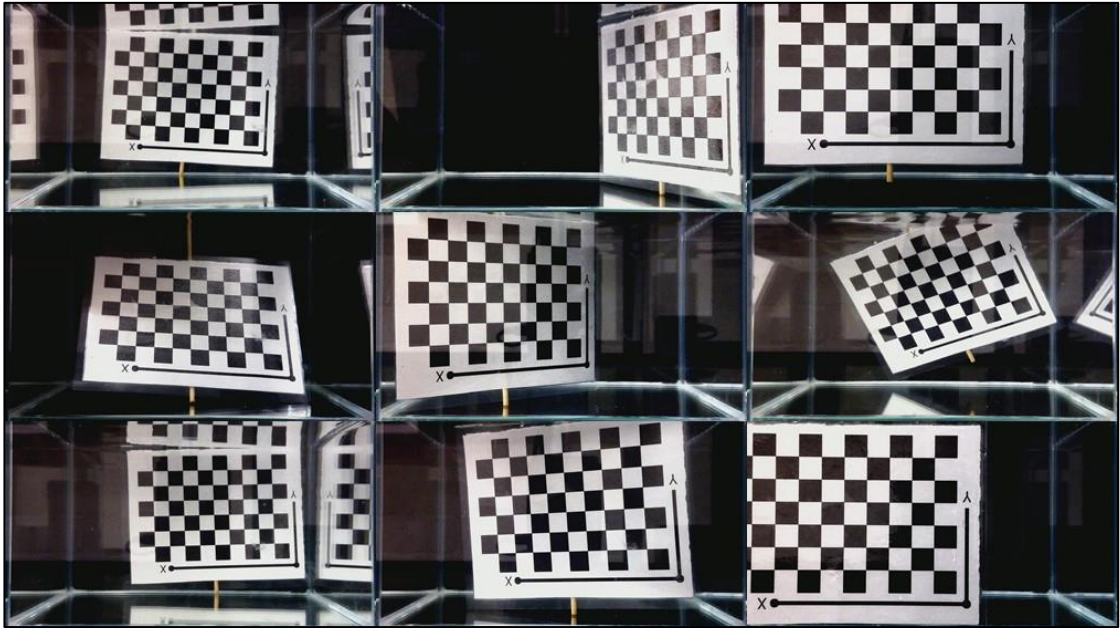
Afterwards, you can add the Raspberry Pis to your MATLAB session. The connection can be set up using the following commands. Of course, you can adjust the settings.

```
14 %add new raspi
15 % targetupdater
16
17- cam1 = cameraboard(rpi1, 'Resolution', '1920x1080', 'FrameRate', 30,
    'Quality', 100, 'Rotation', 180, 'Brightness', 50, 'Contrast', 50,
    'ExposureMode', 'backlight', 'Sharpness', 100);
18- cam2 = cameraboard(rpi2, 'Resolution', '1920x1080', 'FrameRate', 30,
    'Quality', 100, 'Rotation', 180, 'Brightness', 50, 'Contrast', 50,
    'ExposureMode', 'backlight', 'Sharpness', 100);
```

You can check your cameras using the livestream function. The duration can be adjusted by setting alternative values instead of 10 (duration in frames) in line 23 or 31.

```
21 %checking cameras
22 %Livestream for checking cam 1
23- for i = 1:10
24-     img1 = snapshot(cam1);
25-     imagesc(img1);
26-     drawnow;
27- end
28 axis equal
29- %Livestream for checking cam 2
30- figure;
31- for i = 1:10
32-     img2 = snapshot(cam2);
33-     imagesc(img2);
34-     drawnow;
35- end
36- axis equal
```

In the following the experimental area will be referred to as aquarium. You must take multiple views of the checkerboard inside the aquarium and one to three images of the checkerboard outside the aquarium from each camera to determine the camera positions and the common origin (see (4)). Make sure, that for the outside pictures the checkerboard pattern is perfectly aligned with the margin of the aquarium in the lower right (cam1) respectively lower left (cam2) corner. Please, only use **png.files** for the calibration setup. Notice that the first images are likely to show older snapshots and that it will take a few moments until the current image will be saved in the folder. We recommend taking at least 80 images per camera, including the images of the checkerboard outside the aquarium. Make sure that the X-Y-axes are visible in the images.



**Figure 3: Checkerboard procedure:** Taking multiple views of a checkerboard enables an exact calibration of the camera setup.

Please, take the snapshots using the following commands.

```

39      % CALIBRATION
40      % taking snapshots for the calibration (~80 per camera), including images
41      % of the checkerboard on the outside of the aquaria (for the calibration of
42      % the origin)
43-     figure;
44-     img_CAM1=snapshot(cam1);
45-     imagesc(img_CAM1);
46-     timecode1=datetime('now', 'Format', 'd_MM_y_HH_mm_ss');           %capture the
47-     %time at start of recording
48-     Filename_cal1=sprintf('calib1 %s.png', timecode1);
49-     imwrite(img_CAM1,Filename_cal1);   %grab image for calibration
50-     axis equal
51-
52-     img_CAM2=snapshot(cam2);
53-     imagesc(img_CAM2);
54-     timecode1=datetime('now', 'Format', 'd_MM_y_HH_mm_ss');           %capture the
55-     %time at start of recording
56-     Filename_cal2=sprintf('calib2 %s.png', timecode1);
57-     imwrite(img_CAM2, Filename_cal2);   %grab image for calibration
58-     axis equal

```

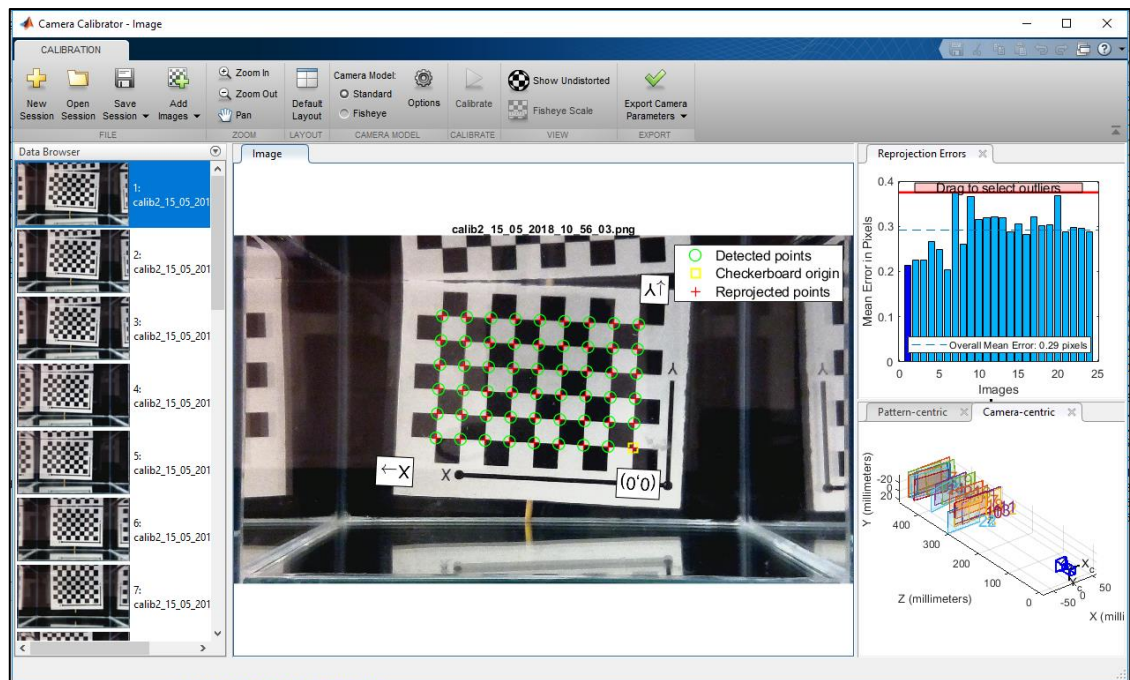
### (c) Using the Camera Calibrator application

The next part is conducted with the **Camera Calibrator** application. MATLAB uses the **Computer Vision System Toolbox™** calibration algorithm which is based on the pinhole camera model. As this model does not account for lens distortion, the algorithm also includes the radial and tangential lens distortion. The calibration algorithm calculates the camera matrix using the extrinsic and intrinsic parameters. This needs to be conducted separately for both perspectives. For further information see:

**<https://de.mathworks.com/help/vision/ug/single-camera-calibrator-app.html>**

Add the images and determine the size of the checkerboard square. The calibrator analyses the images to ensure they meet the calibrator requirements and then detects points. Please, click Calibrate for further analysis. The calibration results can then be improved by selecting outliers and right clicking on the selected images removes those and recalibrate. We recommend including at least 20 pictures per camera and to adjust a very low mean error and overall mean error. We chose a mean error of about 0.43 px and an overall mean error of 0.33 px. Make sure that the pictures of the checkerboard outside the aquarium are still among the selected ones. Please, save the camera parameters afterwards. We recommend using the following file names:

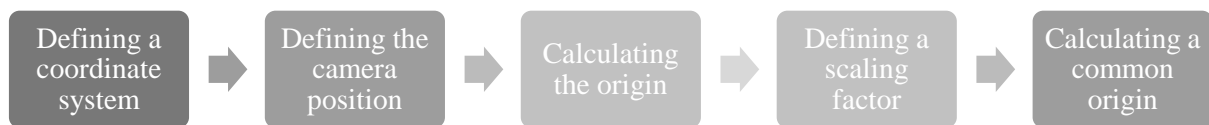
**Camera red:**     „cameraParams\_Date\_red“  
**Camera green:**  „cameraParams\_Date\_green“



**Figure 4: Camera Calibrator:** The Camera Calibrator application is used to calibrate the camera setup.

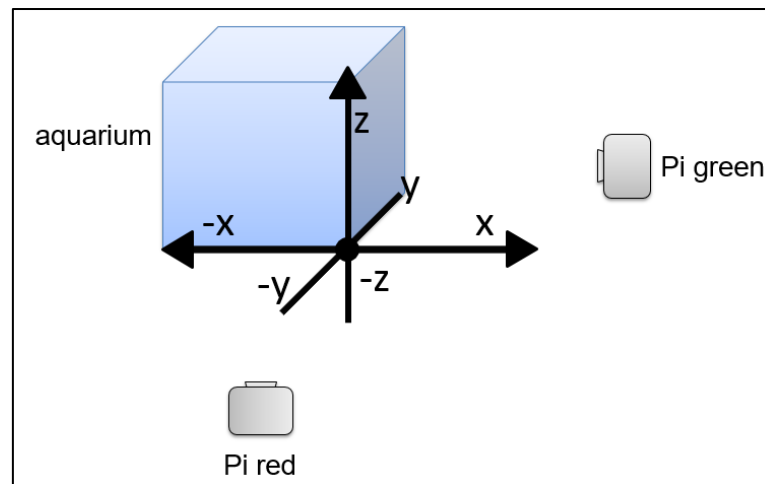
## (2) Defining the camera setup

The definition of the exact camera positions is conducted via the associated part (lines 63-146) of the **CalibrationOfCameraSetup.m** file. The exact procedure of the steps (a)-(d) will be explained for one camera (lines 63-106) and has to be repeated for the second camera (lines 110-146).



### (a) Defining a coordinate system

First, the coordinate system for the 3D movement analysis system must be determined. In Figure 5 you can see an example of our coordinate system.



**Figure 5: Defining the coordinate system.** The basis of the three-dimensional analysis lies on the defined coordinate system. It is needed to determine the exact position of the two cameras. The figure shows an example for a coordinate system.

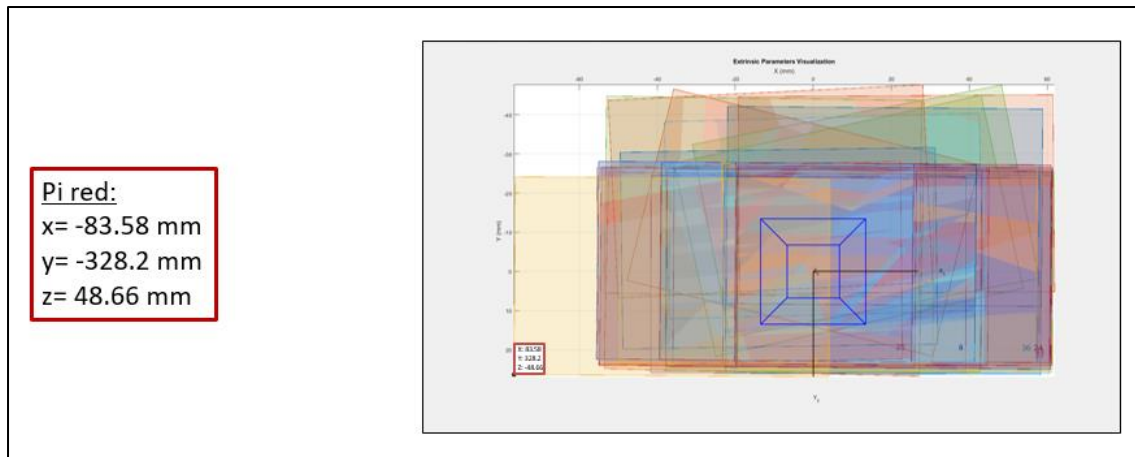
#### (b) Defining the camera position

You can define the exact camera position via the coordinate system and the 3D plot. Please, copy and paste the file name of the camera parameters in line 70.

```

63 % Prior calibrations for Pi red
64
65 %load the camera parameters exported from the 'Camera Calibrator App' for
66 %both cameras. In the following they are called 'cameraParams_red' and
67 %'cameraParams_green'
68
69 % define the camera position (from the 3D plot of the calibration)
70- figure; showExtrinsics(cameraParams_150518_red, 'cameraCentric');
```

The command loads a figure window with the 3D plot of the calibration. Choose the right view and mark the origin with the magnetic data cursor to get the coordinates. Therefore click in the lower corner where the checkerboard was perfectly aligned with the aquariums margins. The magnetic data cursor is available in the toolbar of the figure window.



**Figure 6: Defining the camera positions.** The exact camera positions can be defined via the 3D plot. The calculated coordinates have to be transferred into the MATLAB file.

Afterwards, the calculated coordinates must be transferred into the **CalibrationOfCameraSetUp.m** file. Marking lines 74-76 and pressing F9 saves the settings. Notice that the camera coordinates may be swapped so that you have to change the signs.

```

72 % attention, coordinates may be swapped! (read numbers from the data
   cursor
73 % dialogue and fill them in here, adjusted for the camera position)
74- cam1x=-83.58;
75- cam1y=-328.2;
76- cam1z=48.66;

```

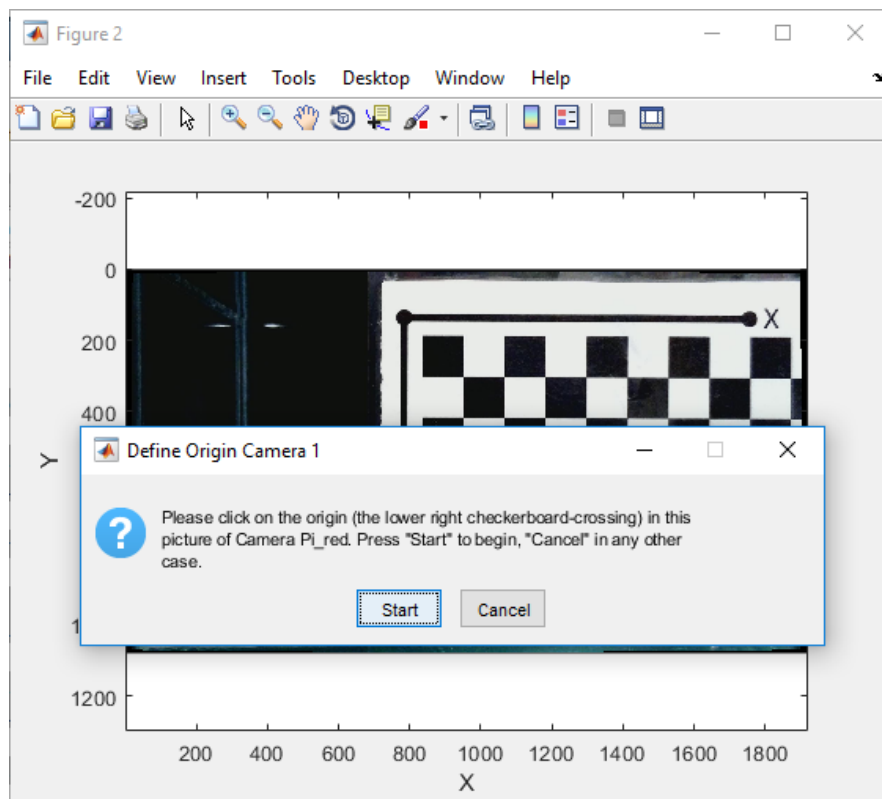
## (c) Calculating the origin

Each camera gets its own origin that must be defined via crosshairs. Please, execute the associated part (lines 78-94) of the **CalibrationOfCameraSetUp.m** file. Copy the name of one image showing the checkerboard outside the aquarium and paste it in line 79. MATLAB will open a figure. Notice that we work with undistorted images. If you do not want to use undistorted images, just enter Image1 instead of J1 in parentheses in line 84. Please, follow the commands.

```

78 % import image with chess board in front from Camera red into Matlab to
   % calibrate lengths etc.
79- Image1=imread('calib1_15_05_2018_11_03_35.png');
80 % undistort image according to camera parameters
81- [J1,newOrigin] = undistortImage(Image1,cameraParams_150518_red);
82 % plot the undistorted image
83- figure;
84- image(J1)
85- axis equal
86- xlabel('X')
87- ylabel('Y')
88
89 % define the origin (just once for a steadily positioned camera)
90
91- message = sprintf('Please click on the origin (the lower right
   % checkerboard-crossing) in this picture of Camera Pi_red. Press "Start" to
   % begin, "Cancel" in any other case. ');
92- reply = questdlg(message, 'Define Origin Camera 1', 'Start', 'Cancel',
   % 'Start');
93
94- [originX1,originY1]=ginput(1);

```



**Figure 7: Defining the origin:** Each camera has got its own origin. It can be defined using crosshairs. The origins are needed to calculate the common origin of the camera setup.

## (d) Defining a scaling factor

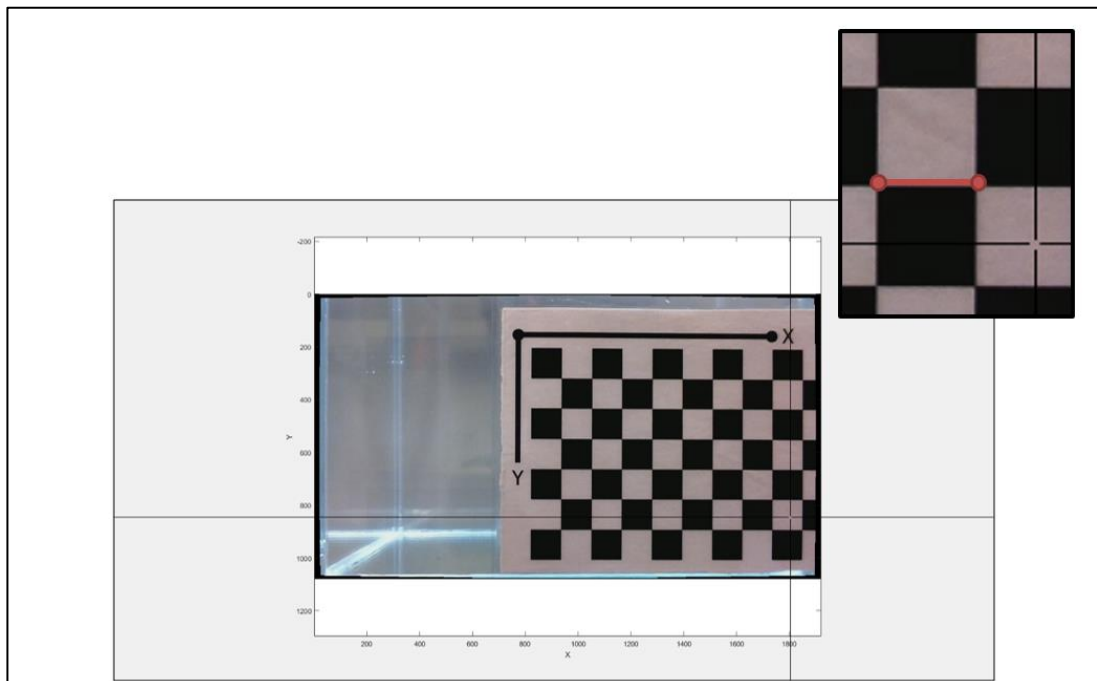
For the further analysis, you have to define the scaling factor. The scaling factor can be defined via crosshairs. It is needed to convert pixel into mm.

Please, execute the associated part (lines 96-106) of the **CalibrationOfCameraSetUp.m** file and capture two x-y-coordinates from each image. Please, transfer the data into the MATLAB file.

```

96      % capture two x-y-coordinates from the image with the help of
      % crosshairs to
97      % determine the scaling factor
98
99-     message = sprintf('Define a length of 1.01 cm by two clicks in
      the neighbouring edges of a checkerboard square in the picture
      of Camera Pi_green. Press "Start" to begin, "Cancel" in any
      other case. ');
100-     reply = questdlg(message, 'Define Length (1.01 cm/Checkerboard-
      Square) Camera 2', 'Start', 'Cancel', 'Start');
101
102-     [X2,Y2]=ginput(2);
103
104      % calculate the length of a mm in pixels
105-     d_2 = ((X2(1) - X2(2)) ^ 2 + (Y2(1) - Y2(2)) ^ 2) ^ 0.5;
106-     d_mm_2=d_2/10.1;

```



**Figure 7: Defining the scaling factor.** The scaling factor can be defined via crosshairs. Capture two x-y-coordinates from each image and transfer the data into the MATLAB file.



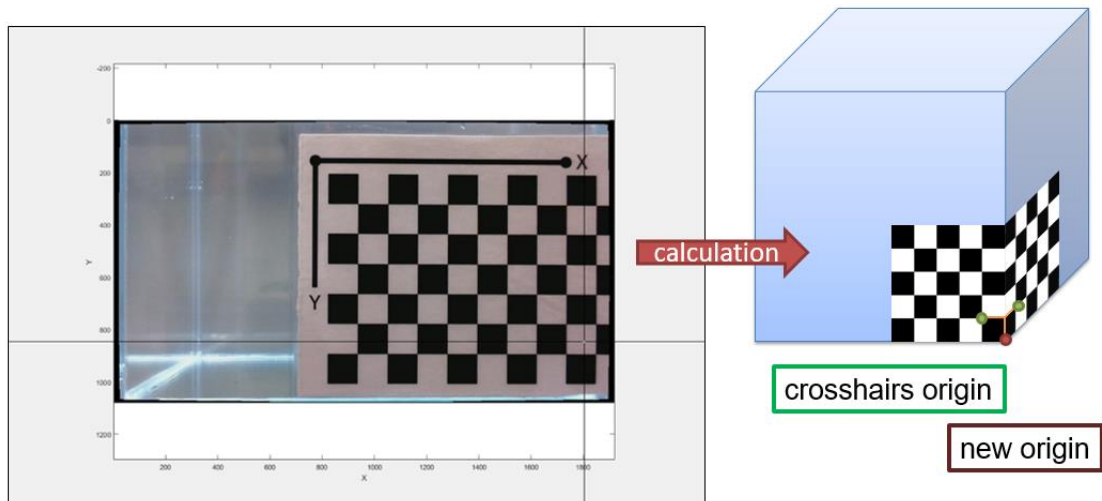
## (e) Defining the common origin

The new common origin results from the measured coordinates (Figure 8) and can be calculated via the lines 149-153 of the **CalibrationOfCameraSetUp.m** file. Notice, the default settings refer to our aquarium as well as the used checkerboard and have to be measured and adjusted individually. We had a glass thickness of 4 mm and a distance of 19.5 mm respectively 29.5 mm of the checkerboard crossing to the upper respectively lower edge.

```

149 %correct the clicked origins for the real origin on the outer lower right
    corner of the aquarium, seen from Pi_red
150- originX1=originX1+4*d_mm_1;
151- originY1=originY1+19.5*d_mm_1;
152- originX2=originX2-4*d_mm_2;
153- originY2=originY2+29.5*d_mm_2;

```



**Figure 8: Defining the common origin.** The common origin is calculated from both origins, which have been defined via crosshairs.

### (3) Saving the calibration settings

Notice that we recognized overexposure while using the white background due to the infrared cameras. Therefore, we always skip the first 30 frames of the videos (line 157). Please, rename your camera parameters (lines 159-161) and delete all necessary variables (lines 164-170). After the calibration the calibration workspace should be saved (line 173). The **CalibrationOfCameraSetUp.m** file will later be loaded in the **Tracking3D\_v2.m** file.

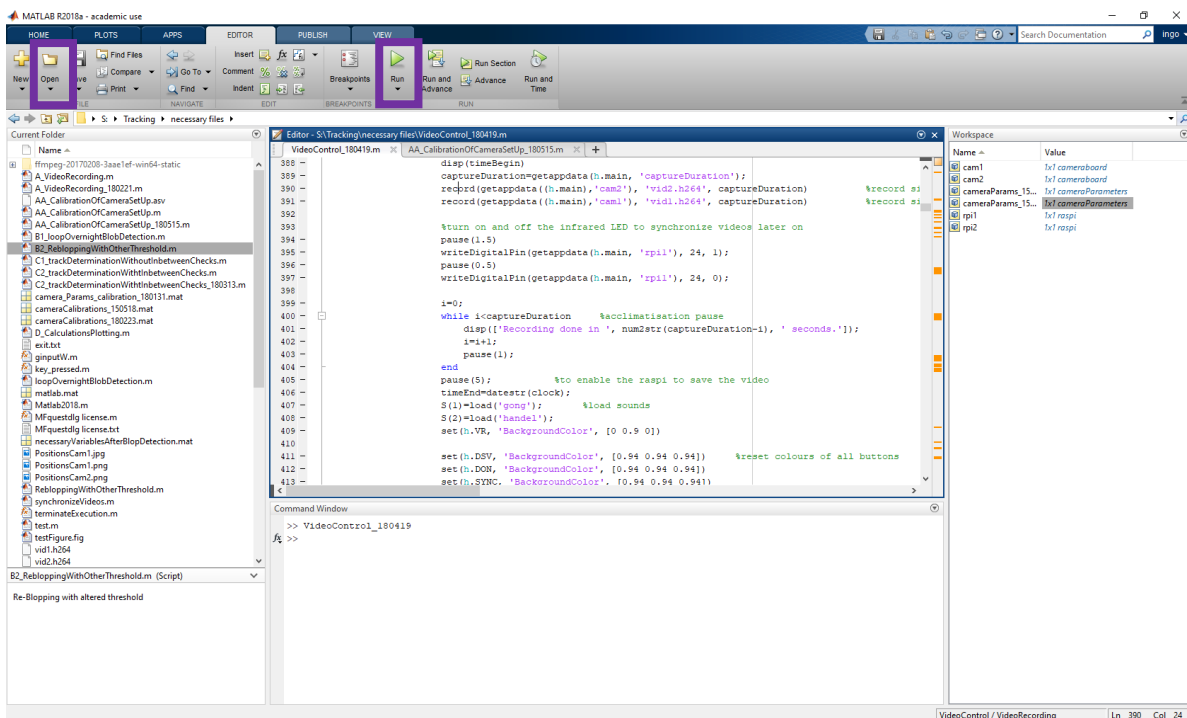
```

155 %the first second of each video is often too bright, so skip the first
156 %30 frames later on
157- firstFrame=30;
158
159 %rename camera parameters (adjust the name of your saved camera-parameters)
160- cameraParams_red=cameraParams_150518_red;
161- cameraParams_green=cameraParams_150518_green;
162
163
164 %delete unnecessary variables
165- clear rpi1
166- clear rpi2
167- clear cam1
168- clear cam2
169- clear cameraParams_150518_red
170- clear cameraParams_150518_green
171
172 %Save the workspace with all calibrations
173- save('cameraCalibrations_150518.mat')

```

## 2.4. Running the 3D Tracking file

To run the 3D Tracking file, you must start MATLAB and open the **Tracking3D\_v2.m** file. Pressing the “Run” icon and “change folder” (if requested) will open the graphical user interface.

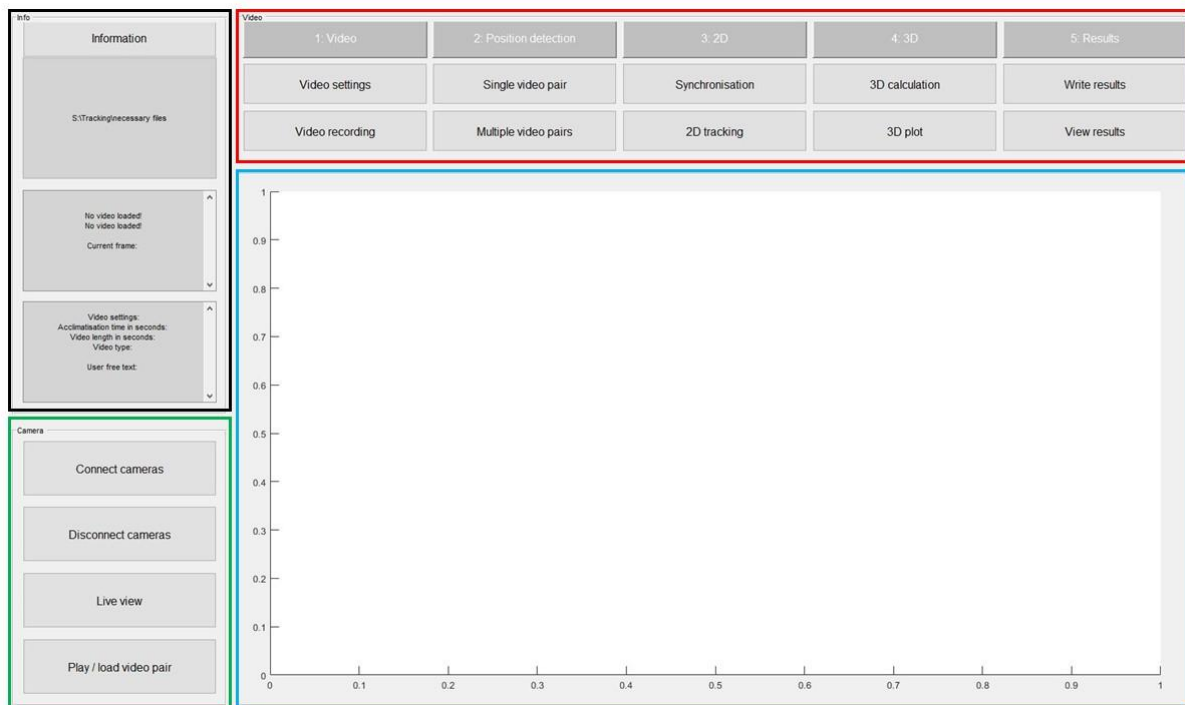


**Figure 9: Running the 3D Tracking file:** The **Tracking3D\_v2.m** file has to be started in Matlab. The „Open“ icon loads the **Tracking3D\_v2.m** and the “Run“ icon opens the GUI.

### 3. GUI

To simplify the analysis of the recorded videos, we developed a graphical user interface<sup>1</sup> that can be seen in Figure 10. It is divided into 4 parts.

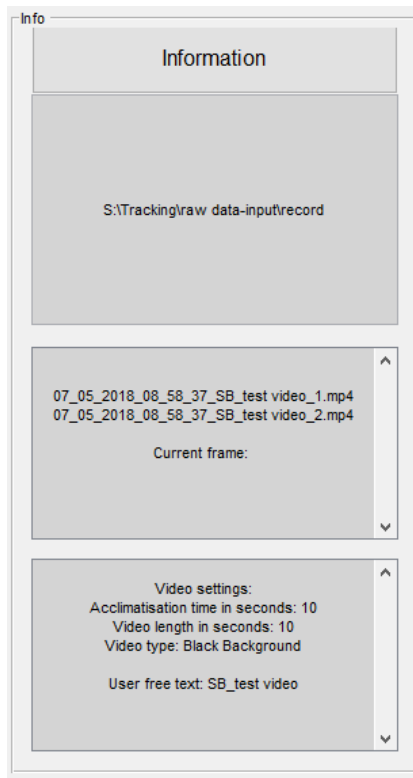
- (1) The **information field** (black frame) where you get all the important information about the current folder, the loaded videos and the video settings.
- (2) The **camera field** (green frame) where you can connect your camera, load a live image and reset or stop your analysis.
- (3) On top is the **video field** (red frame) where you find all the important analysis steps.
- (4) The last part is the **figure field** (blue frame) where you can display different figures such as the live images or 3D plots. For further information read the following chapters.



**Figure 10: Graphical user interface.** The **Tracking3D\_v2.m** file is operated via a GUI. The GUI consists of four different parts. The information field, the camera field, the video field and the figure field.

<sup>1</sup> The code for the graphical user interface was written by Winfried Junke (<https://homepage.ruhr-uni-bochum.de/winfried.junke/>)

## 4. Information Field



The general information field is divided into three parts:

### Information about current folder

The information field **Current folder** shows the activated folder in which your files are saved.

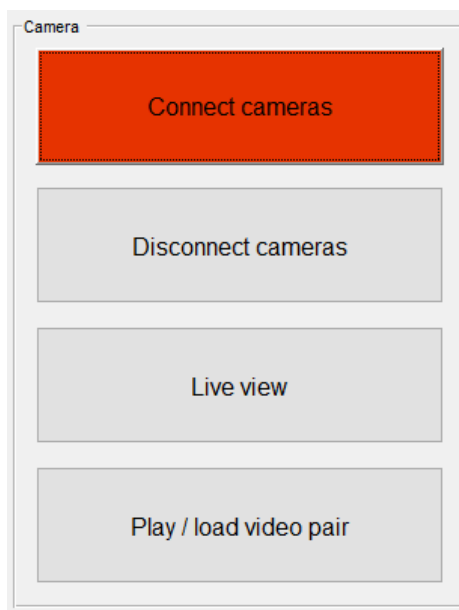
### Information about current video pair

The information field **Video information** informs about the currently loaded video pair. Additionally, the current frame is displayed during 2D tracking adjustments.

### Information about video settings

The information field **Video settings** displays the important information of the adjusted settings. Here, you can check your acclimatisation time, the video length, the video type as well as your entered user free text.

## 5. Camera Field



The camera field is divided into four parts:

The button **Connect cameras** connects your Raspberry cameras with the system. Please, make sure that you have connected the cameras before you start the recording. A red background colour indicates that the cameras are connected.

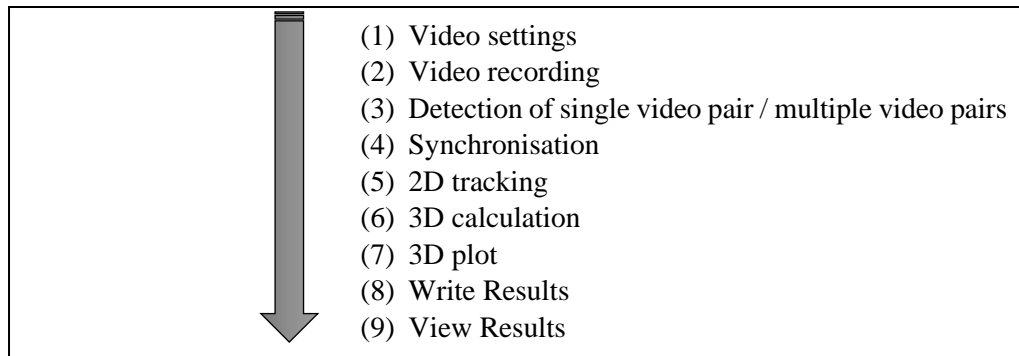
The button **Disconnect cameras** disconnects your cameras from the system. The background colour of the button “Connect cameras” switches to grey.

The button **Live view** starts the live view (infrared trigger LED will flash 1x). You can define the duration of the live view in seconds. During the live view the background colour of the button is red. It changes to grey when the live view is over.

The button **Play / load video pair** gives you the opportunity to play the current video pair or to load another one. Notice that the background colour of the button is red while a video is played. The option “Load another video pair” will close the current figure and open the file select dialogue.

## 6. Video Field

The analysis of the videos can be conducted following a fixed order. Every step is individually saved so that you can edit your analysis or stop it whenever you want. Pressing ESC<sup>2</sup> stops the currently running process immediately.

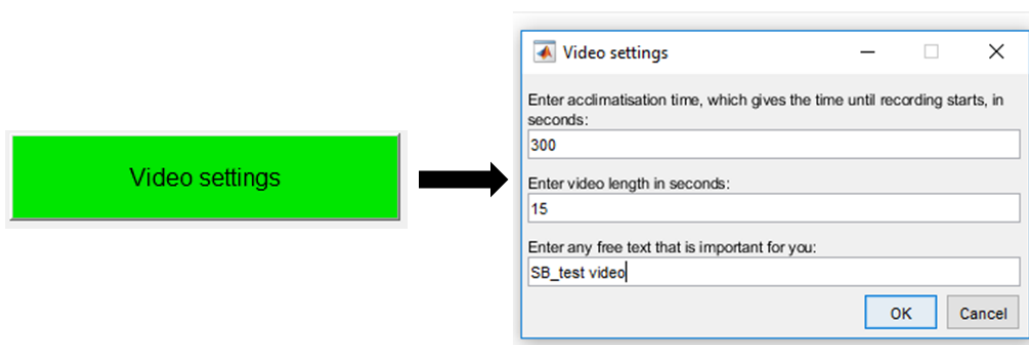


A red background colour signalizes running processes whereas a green background colour indicates already processed analysis steps. Please, notice repeatedly pressing the **Video recording** button will set all backgrounds to grey, as for the new current video pair none of the analyses has been conducted. All progresses are nonetheless saved in the respective folders under the original video file name so that you can continue with your analysis from each point. Notice that you have to wait until the video pair is completely loaded. The following paragraphs explain the functions step-by-step.

### 6.1. Video

#### 6.1.1. Video settings

Before you start the recording, you should adjust your video settings. There is a default acclimatisation time and video length which can be adjusted in the script. You can edit the video length as well as the acclimatisation time in seconds manually. Additionally, you can enter any free text that is important for you. The background colour of the button switches to green after clicking ok.



**Figure 11 : Adjusting the video settings:** Before the recording, the acclimatisation time, the video length and free text can be edited.

<sup>2</sup> The code of the terminateExecution function was written by “Pursuit” ([https://stackoverflow.com/questions/10033078/matlab-implementing-what-ctrlc-does-but-in-the-code?utm\\_medium=organic&utm\\_source=google\\_rich\\_qa&utm\\_campaign=google\\_rich\\_qa](https://stackoverflow.com/questions/10033078/matlab-implementing-what-ctrlc-does-but-in-the-code?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa)).

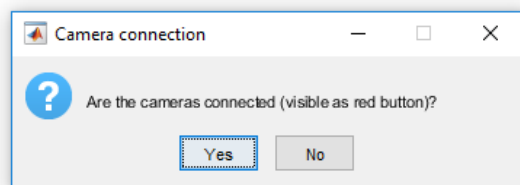
### 6.1.2. Video recording

Before starting video recording be sure that the infrared LED is visible in both views. This can be checked easily with the live view mode. The button **Video recording** starts the recording of the videos. In a first step you will be reminded to connect the cameras beforehand. If you have not connected the cameras yet, please do so. Afterwards you will be asked which background you are using.

We have preadjusted three different interchangeable settings in order to make our 3D Tracking setup as flexible as possible. You can distinguish between (a) the black background, (b) the white background or (c) the infrared background.

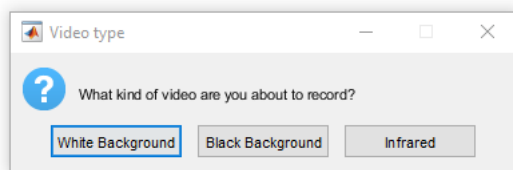
Each background has its own advantages. E.g. the infrared setup enables the analysis of nocturnal behaviours. Just check which background is best for your analysis. Notice that you cannot track more than five individuals simultaneously.

(a)



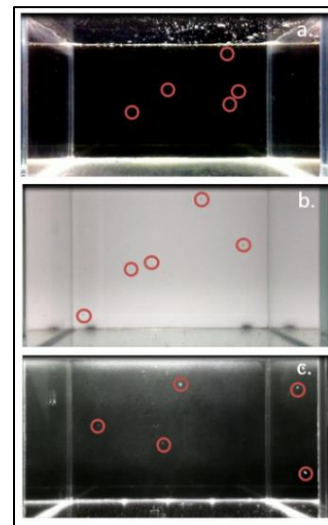
**Figure 12: Camera connection:** Make sure that your cameras are connected.

(b)



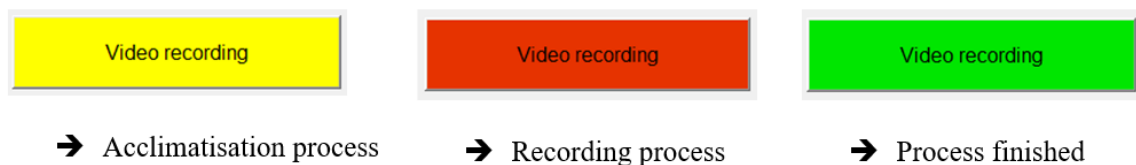
**Figure 13: Background:** You can distinguish between three different backgrounds.

(c)



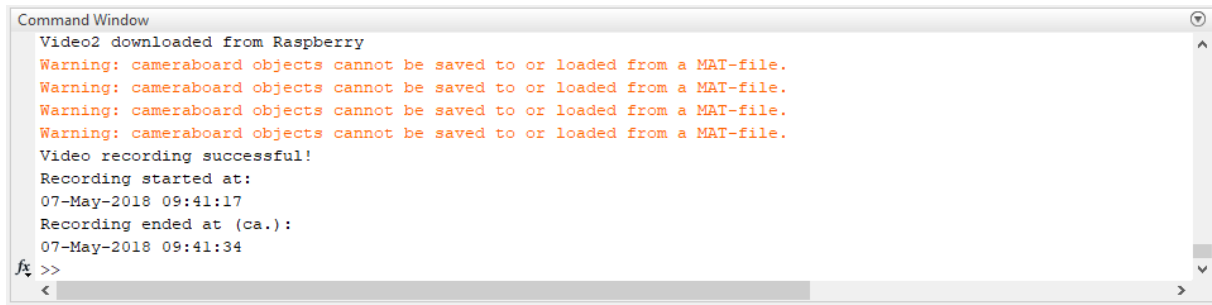
**Figure 14: Different backgrounds:** The setup consists of three interchangeable backgrounds.

During the acclimatisation time the background colour of the **Video recording** button is yellow. It switches to red during the recording and finally to green when the process is finished.



**Figure 15: Colour change of Video recording button:** During the acclimatisation time it is yellow, during the recording red and finally green when the process is finished.

Additionally, the MATLAB command window displays the acclimatisation time as well as the video time. You can check whether the recording process is finished or not. The end of the recording is also indicated by a sound. After the recording, MATLAB saves the two videos and one MATLAB file in the folder **1\_record**. Notice that you can ignore the warnings.



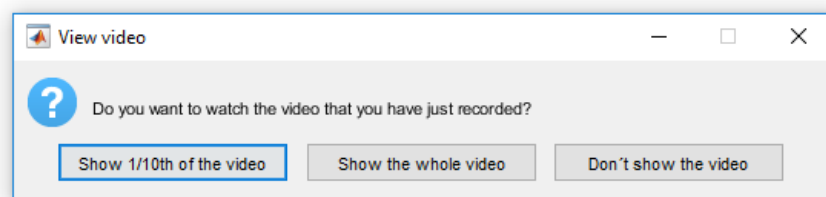
```

Command Window
Video2 downloaded from Raspberry
Warning: cameraboard objects cannot be saved to or loaded from a MAT-file.
Warning: cameraboard objects cannot be saved to or loaded from a MAT-file.
Warning: cameraboard objects cannot be saved to or loaded from a MAT-file.
Warning: cameraboard objects cannot be saved to or loaded from a MAT-file.
Video recording successful!
Recording started at:
07-May-2018 09:41:17
Recording ended at (ca.):
07-May-2018 09:41:34
fx >>

```

**Figure 16: Command Window:** The warnings after the video recording can be ignored.

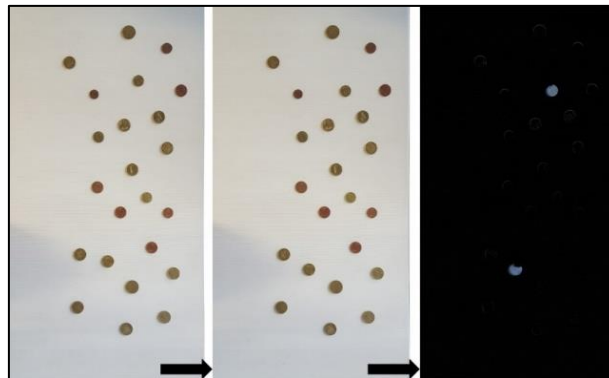
Afterwards, you can decide whether you would like to watch the whole videos, parts of the videos or nothing. The **Play / load video pair** button will be red while the video is playing and grey when the video is over.



**Figure 17: View video:** After the recording the user can decide whether he or she would like to watch the videos.

## 6.2. Position detection

The position detection<sup>3</sup> is done by subtracting the single images from each other. The result is that the evolved image shows the difference between two images as can be seen in Figure 18. Here, you have got two options **Single video pair** and **Multiple video pairs**. Please notice, it is necessary that both video files and the record figure are in the **1\_record** folder.



**Figure 18: Subtracting images:** Two images are subtracted from each other to display the difference between them.

<sup>3</sup> The position detection is partly based on the code written by "Image Analyst" (<https://de.mathworks.com/matlabcentral/answers/237938-image-segmentation-of-connected-shapes-problem>).

This process can be conducted with distorted and undistorted images. As we are using an aquarium as experimental area, we work with distorted images. Of course, you can adjust this setting in the **Tracking3D\_v2.m** file. Just uncomment line 604 instead of line 603.

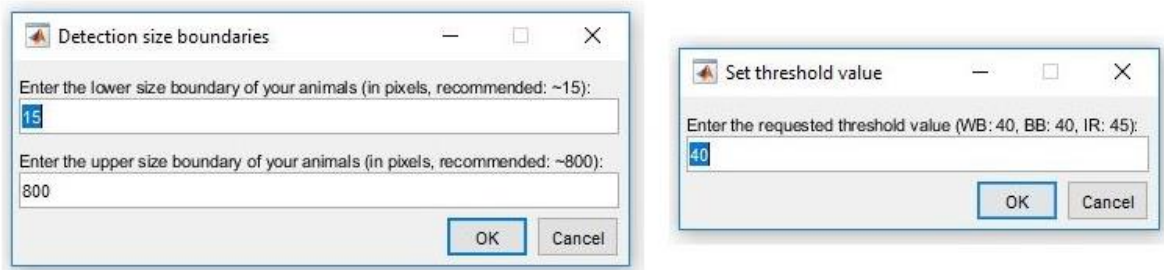
```

597 % blob detection for Camera/Raspi Green
598-     for k2 = calib.firstFrame:1:(vid2.NumberOfFrames-10)
599-         img11 = read(vid2, k2);
600-         [img11u,~] = undistortImage(img11,calib.cameraParams_green);
601-         img12 = read(vid2, k2+10);
602-         [img12u,~] = undistortImage(img12,calib.cameraParams_green);
603-         originalImage=imsubtract(img11u, img12u);           %use the images
                    coorrected by the calibartion parameters
604         %originalImage=imsubtract(img11, img12);           %use the images
                    uncorrected by the calibartion parameters

```

### 6.2.1. Single video pair

Single videos can be easily edited via the function **Single video pair**. Please notice, it is necessary that both video files are in the record folder. Otherwise the position detection is not possible. At the start you will be asked to enter the lower and upper size boundary of your animals. The default for the lower size boundary is 5 whereas the default for the upper boundary is 800. Next, you have to enter a threshold value. Just check which threshold value fits best for your videos. Default is 35



**Figure 19: Detection size boundaries and treshold value:** The lower and upper boundary size of the animals as well as the threshold value for the used background can be defined at the beginning of the single video pair process.

MATLAB displays the process in the command window. You can check whether the detecting process is finished or not. The results will be saved in the folder **2\_singleVideo**.



## 6.2.2. Multiple video pairs

We recommend using the function **Multiple video pairs** if you have more than one video of one background type. E.g. you can run the process overnight. At the start you will be asked to enter a threshold value. Just check which threshold value fits best for your videos.

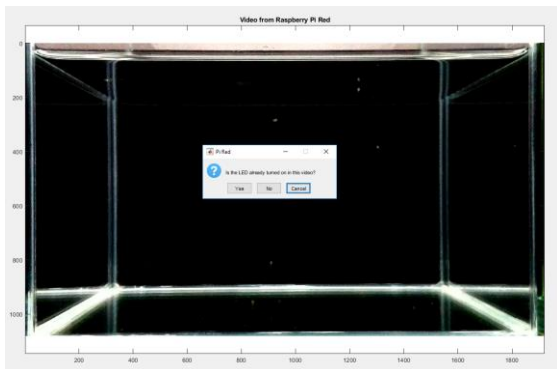
MATLAB displays the process in the command window. You can check whether the detecting process is finished or not. Please notice, it is necessary that all video files are in the record folder. Otherwise the position detection is not possible. The results will be saved in the folder **3\_multipleVideos**. The computer system can be shut down after the process. Just uncomment line 855 in the **Tracking3D\_v2.m** file.

```
851 %If you want your computer to shut down after it has finished the blop
852 %detection, uncomment the following line and execute it with the whole
853 %script, it will shut down the whole system, unless there are MS Office
854 %programs running
855 %system (shutdown -s);
```

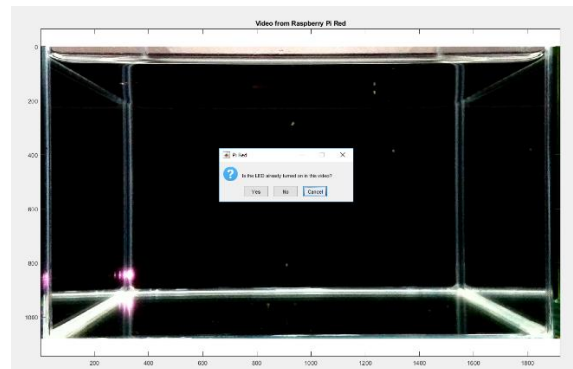
## 6.3. 2D

### 6.3.1. Synchronisation

Before the analysis, the two videos have to be synchronised so that we have got one common starting point. Please, check whether the infrared trigger LED is already on or not. MATLAB saves a MATLAB file in the folder **2\_synchronisation**.



**Figure 20: Infra-red trigger LED is off:** The infrared LED is still off. Press **No** to continue, **Cancel** to quit.

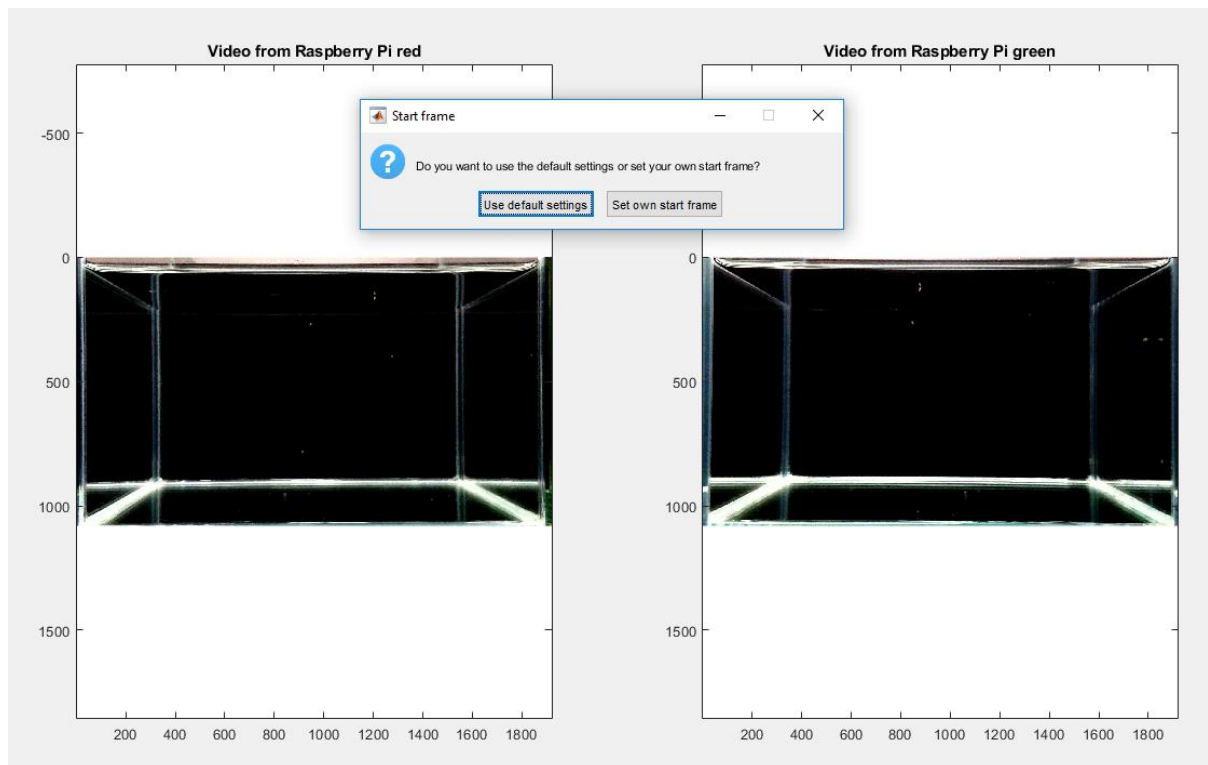


**Figure 21: Infrared trigger LED is on:** The infrared LED is on. Press **Yes** to continue, **Cancel** to quit.

### 6.3.2. 2D tracking

The 2D Tracking creates the single tracks of each animal in each video. All important information will be saved in Video 1, which is separately saved (ending **TR1**). This allows you starting tracking of Video 2 independently. The complete tracking will be saved in another file (ending **TR**).

At the beginning, you can decide whether you would like to watch the videos beforehand or not. You can even decide how much you would like to see. Next you have to define the start frame. Here, you can use the default settings or set your own start frame by scrolling through the videos. The latter option might be advantageous if you cannot see all animals in the first start frame.



**Figure 22: Setting the start frame:** The user can decide whether he or she would like to use the default settings or set her or his own start frame. In some cases it could be necessary to define an own start frame because not all animals could be detected in the default start frame.

Then you have to mark the starting position of each animal. We expect that you observe the tracks of five animals. If you have less than 5 animals in your experimental area just mark your animals first and then place the superfluous starting positions in the margins of the figures. The single animals can be marked with crosshairs. You have to confirm how many animals you marked. The colour of the crosshairs can be adjusted in the **Tracking3D\_v2.m** file (line 3976)<sup>4</sup>. Just change the argument **'BackgroundColor'**. You can choose any RGB colour.

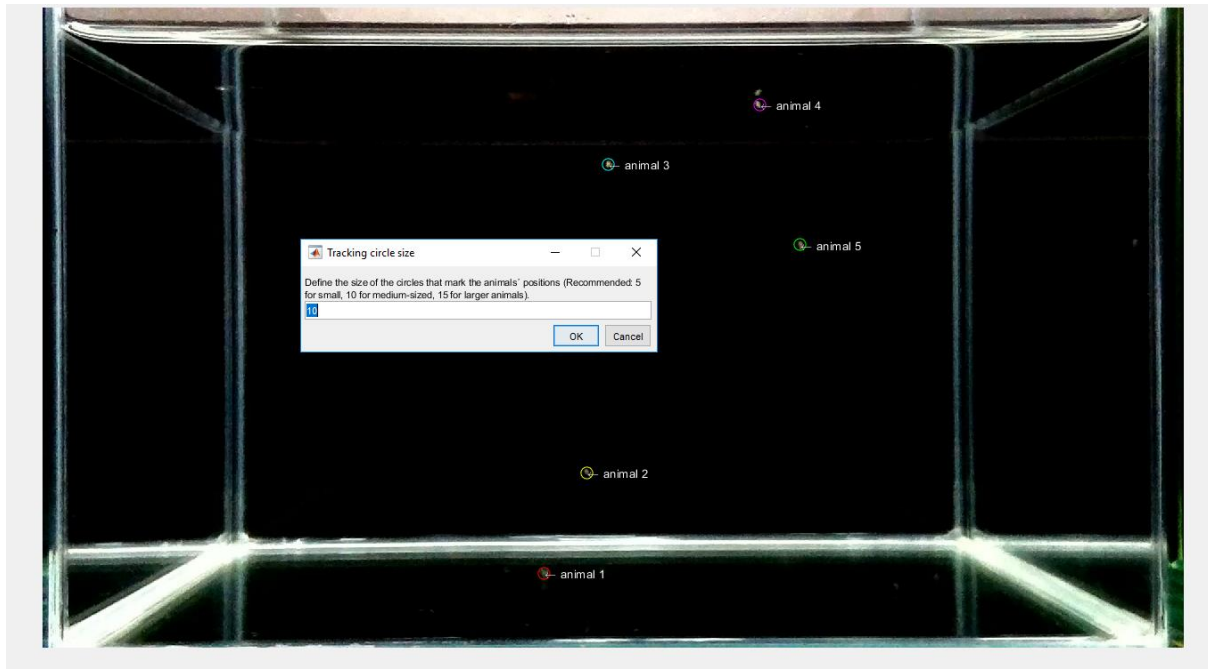
```

3972 function crossHair = createCrossHair(fig)
3973 % Create thin uicontrols with black backgrounds to simulate fullcrosshair
    pointer.
3974 % 1: horizontal left, 2: horizontal right, 3: vertical bottom, 4: vertical top
3975- for k = 1:4
3976-     crossHair(k) = uicontrol(fig, 'Style', 'text', 'Visible', 'off', 'Units',
        'pixels', 'BackgroundColor', [1 1 1], 'HandleVisibility', 'off', 'HitTest',
        'off'); %#ok<AGROW>
3977- end
3978- end

```

Please, make sure that you choose the same order in each video. You can check your marking with the saved image files in the folder **5\_tracking2D**.

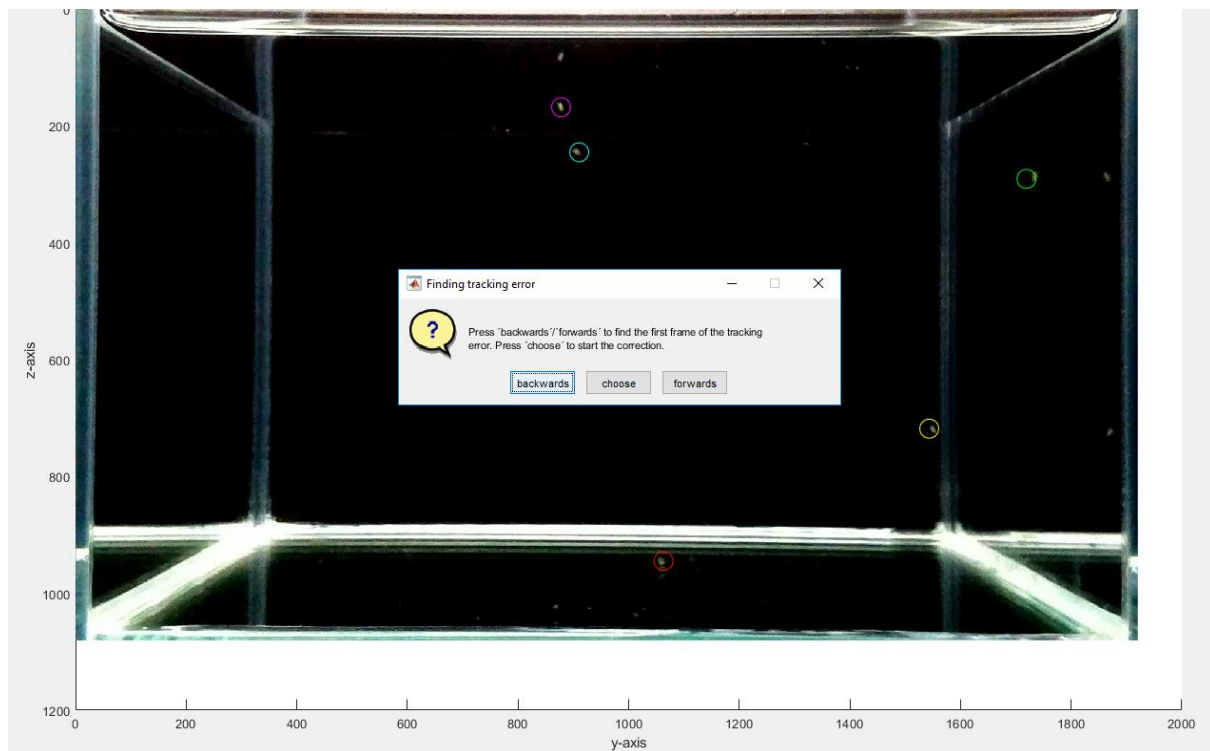
The size of the tracking circle is another important aspect. Please make sure that your animals are completely surrounded by the circles but note that the circles should not be too big. The default setting is 10. We recommend 5 for small animals (up to 900 $\mu$ m), 10 for medium-sized (up to 1300  $\mu$ m) and 15 for larger animals (up to 2000 $\mu$ m). Notice, that the size of the circles has no influence on the technical tracking accuracy.



**Figure 23: Tracking circle size:** The tracking circle size can be defined before the 2D tracking. The animals should be surrounded by the circles.

<sup>4</sup> We modified an original MATLAB code (Ginput function) to change the colour of the crosshairs.

The tracking starts automatically after you have edited the settings. Pressing **space** interrupts the tracking process and gives you the opportunity to correct it<sup>5</sup>. You can scroll backwards or forwards to find the first frame of the tracking error. The option **choose** directs you to the correction modus. Here, each track can be individually corrected.



**Figure 24: Finding tracking error:** Tracking errors can be corrected pressing **space** and following the instructions.

If you have difficulties in following your animals, you can adjust the speed of the videos. Just adjust the delay in line 949 of the **Tracking3D\_v2.m** file.

```

935- function Tracking(~,~)
936-
937-     global interruption
938-     tracking1=getappdata(h.main, 'tracking1');
939-     if tracking1 ==0
940-         set(h.TRACK, 'BackgroundColor', [0.9 0.2 0])
941-         StartFrame_Red=getappdata(h.main, 'StartFrame_Red');
942-         StartFrame_Green=getappdata(h.main, 'StartFrame_Green');
943-         vid1=getappdata(h.main, 'vid1');
944-         vid2=getappdata(h.main, 'vid2');
945-         blobM=getappdata(h.main, 'blobM');
946-         blobM2=getappdata(h.main, 'blobM2');
947-         timecode2=getappdata(h.main, 'timecode2');
948-         freeText=getappdata(h.main, 'freeText');
949-         delay=0;
950-         figPos_1=getappdata(h.main, 'figPos_1');
952-
953-         interruption=0;

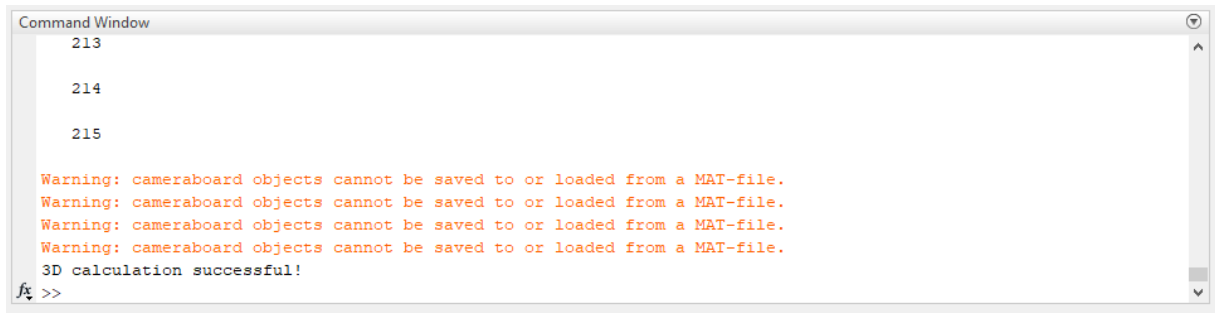
```

<sup>5</sup> To avoid covering parts of the frame, we used the MFquestdlg function which is adjustable in position. The code was written by “Saeid” <https://de.mathworks.com/matlabcentral/fileexchange/31044-specifying-questdlg-position?focused=5186677&tab=function>.

## 6.4. 3D

### 6.4.1. 3D calculation

The button **3D calculation** converts the 2D data into 3D data. The MATLAB command window displays the process.



```

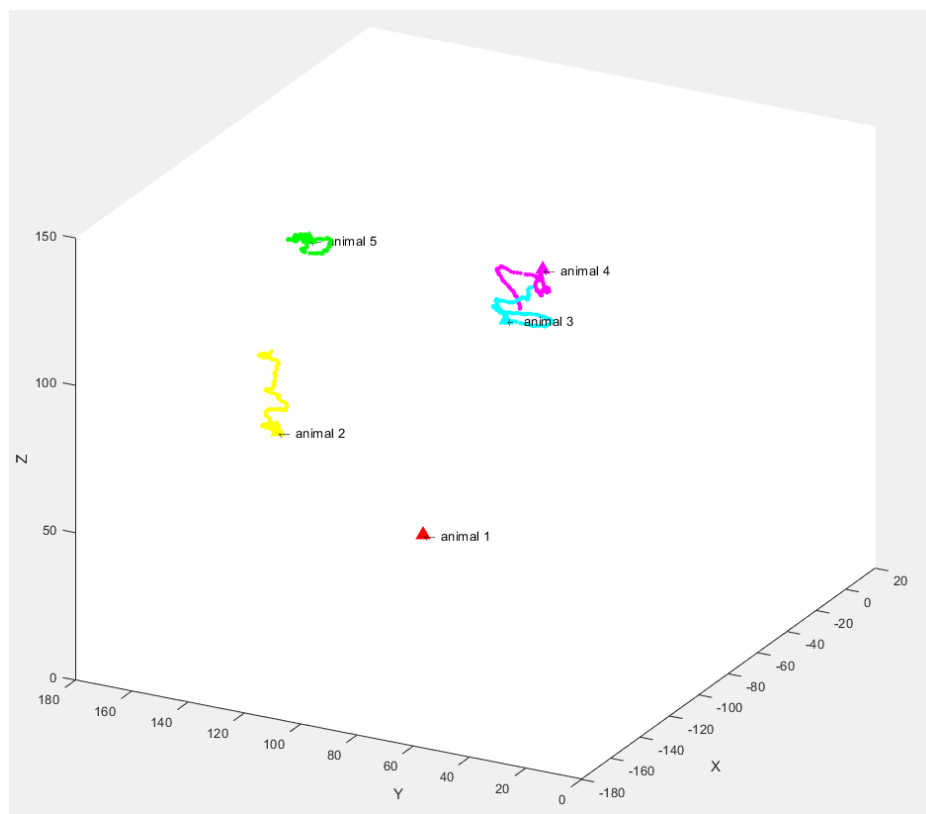
Command Window
213
214
215

Warning: cameraboard objects cannot be saved to or loaded from a MAT-file.
Warning: cameraboard objects cannot be saved to or loaded from a MAT-file.
Warning: cameraboard objects cannot be saved to or loaded from a MAT-file.
Warning: cameraboard objects cannot be saved to or loaded from a MAT-file.
3D calculation successful!
fx >>
  
```

**Figure 25: 3D calculation:** The command window displays the process of the 3D calculation. The warnings can be ignored.

### 6.4.2. 3D plot

You can plot your tracks via the button **3D plot**. You can decide whether you would like to plot all tracks or single tracks. We also integrated a third option that allows you to observe the developing of tracks. The triangle marks the starting position. There are a lot of possibilities how you can visualise your tracks. E.g. you can display the tracks from different views or rotate them. Just use the integrated toolbar.



**Figure 26: Visualisation of 3D plot:** The analysed tracks can be shown in a 3D plot.

## 6.5. Results

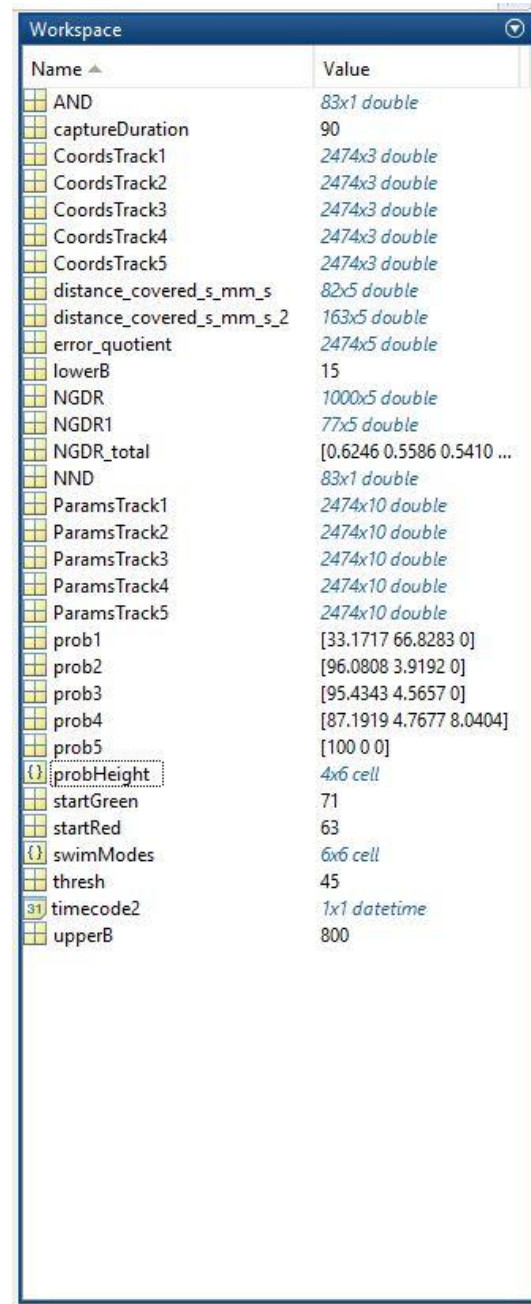
### 6.5.1. Write results and View results

**Write results** saves all your results in an excel sheet and as a MATLAB workspace. Your results can also be displayed in the usual MATLAB workspace. Just use the **View results** button.

#### Write Results

<b>captureDuration</b>	Video length in seconds
<b>CoordsTrackX</b>	Calculated X, Y and Z coordinates in mm
<b>ParamsTrackX</b>	X, Y and Z coordinates for calculation*
<b>error_quotient</b>	Error quotient of the intersect in mm
<b>velocityPerSec</b>	Velocity per second in mm/s
<b>velocityPer0.5Sec</b>	Velocity per half second in mm/s
<b>NGDRevery2.5Sec</b>	NGDR every 2.5 seconds
<b>NGDRwholeVideo</b>	Mean NGDR
<b>Probability swim modes</b>	Distinction between 5 swimming types
<b>NNDPerSec</b>	Nearest neighbour distance
<b>ANDPerSec</b>	Average neighbour distance
<b>Probabilities heigth</b>	Distinction between 3 heigth
<b>Thresh</b>	Thresh value
<b>upperB</b>	Upper boundary size
<b>lowerB</b>	Lower boundary size
<b>circSize</b>	Tracking circle size
<b>startRed</b>	Start frame of camera Pi red
<b>startGreen</b>	Start frame of camera Pi green
<b>errFrames</b>	Errors during 2D tracking

#### View Results



Name	Value
AND	83x1 double
captureDuration	90
CoordsTrack1	2474x3 double
CoordsTrack2	2474x3 double
CoordsTrack3	2474x3 double
CoordsTrack4	2474x3 double
CoordsTrack5	2474x3 double
distance_covered_s_mm_s	82x5 double
distance_covered_s_mm_s_2	163x5 double
error_quotient	2474x5 double
lowerB	15
NGDR	1000x5 double
NGDR1	77x5 double
NGDR_total	[0.6246 0.5586 0.5410 ...]
NND	83x1 double
ParamsTrack1	2474x10 double
ParamsTrack2	2474x10 double
ParamsTrack3	2474x10 double
ParamsTrack4	2474x10 double
ParamsTrack5	2474x10 double
prob1	[33.1717 66.8283 0]
prob2	[96.0808 3.9192 0]
prob3	[95.4343 4.5657 0]
prob4	[87.1919 4.7677 8.0404]
prob5	[100 0 0]
probHeight	4x6 cell
startGreen	71
startRed	63
swimModes	6x6 cell
thresh	45
timecode2	1x1 datetime
upperB	800

\*Given are first the X, Y, Z coordinates from the line of Pi red, then of Pi green, then the resulting position and finally the shortest distance between both lines

## 7. Option to use a second computer for the analysis

Alongside the MATLAB script for video capture and analysis of paths on the tracking-computer, we release the tracking@home-script. With this script, all steps except for the video-recording, for which external hardware is necessary, can be conducted on any computer running Matlab R2016b or higher.

### What you have to do to make the tracking program work on another computer:

1. In addition to at least Matlab R2016b, Microsoft Office Excel needs to be installed. Furthermore, the folder structure described in chapter 2 of this manual needs to be identical on the second computer/an external drive.
2. Within the folder “necessary files”, the tracking@home-script needs to be saved with the name Tracking3D\_v2 (or whatever name you have called your tracking script with which you record).
3. Within MATLAB add all subfolders of the “tracking” folder structure to the path of MATLAB.
4. Copy the video files and the \*\_RC.fig-file into the 1\_record folder of the additional computer. If you want to continue a video analysis started on your computer connected to the cameras, the videos must be transferred to the 1\_record-folder and the last step of analysis, e.g. \*\_SY.fig for synchronisation, copied into the respective folder of your additional computer.
5. Load the \*.fig-file via double click or open it with play/load video pair.
6. Click on “Disconnect cameras” to use the added functionality to set the path to the tracking folder. After that, all subsequent steps will run as usual.
- 7.

The tracking@home-script is adjusted for a single screen. If you have two screens available, the windows can manually be moved or placed with the `set(gcf, 'position', [])`-lines.

If you use multiple computers, we recommend to use the one with the best CPU, RAM and graphics card equipment for the single video/multiple video detection, as these steps can be very time consuming and work fastest under these prerequisites.

## 8. Suggestions for debugging and FAQ

### Setup

- (1) What are the differences between the background types? Are there any advantages?**  
(A) You can use the black and the white background to simulate daytime scenarios and the infrared background to simulate nocturnal scenarios. Depending on the animal size, the black or the white background provides a better contrast.
- (2) Why do you use black velour foil and opaque plexiglas?**  
(A) We use opaque plexiglas to guarantee a diffuse light source. Additionally, the black velour foil ensures a constant lighting and provides higher contrast.
- (3) What is the function of the infrared trigger LED?**  
(A) We figured out a time difference of 3-5 frames between the two cameras. The infrared trigger LED is needed to synchronize the two videos.
- (4) Do I need to use the Raspberry Pi 3 and the recommended cameras?**  
(A) Of course, you can customize our recommended system. In this case, make sure that you also have to adjust the MATLAB files.

### Video recording

- (1) I cannot record videos. What can I do?**  
(A) Did you connect the cameras? If so, please check all the other connections. It may be necessary to restart the whole system.
- (2) I do not want to acclimate my animals. Is there a possibility to directly start the recording process?**  
(A) Yes, just enter 0 for the acclimatisation time.
- (3) Can I observe more than five individuals simultaneously?**  
(A) No, this is not possible.
- (4) I would like to observe less than five individuals simultaneously. What aspects do I need to watch out for?**  
(A) In this case, we recommend to place the other starting positions outside your experimental area or outside the expected movement range.

### Position detection

- (1) The position detection does not work. What can I do?**  
(A) Please check whether all video files are in the “record” folder.
- (2) Why do you recommend certain threshold values?**  
(A) Depending on animal size, animal velocities and contrast of the animal to the background, adjusting the threshold values may be necessary. Low threshold values lead to the detection of noise and therefore can aggravate the movement analysis and decelerate the position detection. On the other hand, high threshold values may miss small movements, while position detection executes faster. Testing to find a compromise is recommended here.



**(3) Why do you recommend certain boundary sizes?**

(A) If boundary ranges are set too wide, noise may be recognized as movement. On the other hand, a too narrow range will lead to “holes” in the track, if the animal covers more or less area (in pixel) than set with the boundaries. Both prevent a continuous movement analysis flow. Always keep in mind, that depending on the alignment to the camera, its distance to the camera and its movement velocity, the covered area may change severely!

**(4) Why do you work with undistorted images?**

(A) Undistorted images are corrected images. Image correction might be necessary as straight lines may do not appear to be straight lines because of phase transition (air → water) or barrel distortion.

**2D Tracking**

**(1) I cannot see all animals. What should I do?**

(A) Set your own start frame. You can skip the video frames until you can see all animals.

**(2) I cannot distinguish between the single animals. How can I figure out which track belongs to which animal?**

(A) If the animals move overlapping and distinguishing between both animals is impossible, go on with the movement analysis for a few more frames and then set the positions via movement analysis interruption again. It may be necessary to search for the scene in question in the original video files and compare both views.

**(3) I accidentally choose the wrong order in the second video. Do I have to repeat the position marking in both videos?**

(A) Yes, you have to repeat the position marking in both videos.

**(4) Why are the sizes of the tracking circle important?**

(A) Too big circles might lead to tracking errors as the user is no longer able to decide whether the tracking is still correct or not. We recommend to work with almost equal animals and to choose a tracking circle that only just covers the whole animal.

**3D plot**

**(1) The 3D calculation does not work. Do I have to start from the beginning?**

(A) Please, check whether you have conducted the calibration correctly and all files are in the correct folders.

**(2) The three-dimensional plot looks strange. What have I done wrong?**

(A) Please, check your 2D tracking again.

## **Results**

### **(1) Which results will be calculated?**

(A) The programme automatically calculates the position, movement speed, NGDR and derivatives of that.

### **(A) How can I save my results?**

(B) Just use the “Save results“ button. Your results will be automatically saved as Excel and MATLAB files.