Tracking Insects on a Raspberry Pi

1. Short introduction to tracking and how this script works:

This tracking approach is divided into 5 Steps: filtering, segmentation, detection, tracking, stitching and filtering of tracks.
First an image is recorded and converted to gray scale. Then the image is filtered. In this case a band pass filter is applied to remove both very small objects and very large objects. Small objects can be small particles, or in this experiment little cracks in the flour surface, that occur when the bugs move. Large object can be for example the difference in illumination between one side of the dish and the other.
After filtering the bugs should appear as smooth spots with high intensity values. In the next step, the segmentation, the script tries to find all pixels belonging to bugs. This is done by choosing any pixel above a certain intensity value. This value, called a threshold, is determined by analyzing the intensity distribution of all pixels. Most pixels belong to the background and form a large gaussian shaped peak. Therefore finding background pixels many standard deviations away from the mean is unlikely. Any value above the threshold t=m + sd * f (m: mean pixel intensity, sd: standard deviation of the pixel intensity, f: some additional factor) can then be classified as “bug”. Next the script finds the center of the individual bugs. This step is called detection. An additional filter is applied for this step. Since bugs have a fixed size, all areas below a threshold size are excluded. This deals manly with fragmentation's of single bugs. Detections are then calculated for each isolated area as their center of mass. Next detections in one frame have to be connected to the detections in another frame to perform tracking. This script uses a nearest neighbor approach, were the closest detections are iteratively connected until no more pairs below a maximum distance can be found. If a detection in the new frame is not connected, it becomes the start of a new track. This tracking method allows only connection between detections from one frame to the very next frame. If one connection is missed, the track is interrupted. To address this problem the script uses stitching after all images are recorded and analyzed. In stitching the ends and starts of tracks are connected if they are close enough in time and space. Currently stitching excludes any tracks that are to far apart in time, and then uses the same nearest neighbor approach as was used for tracking to connect the tracks closest in space. Stitching is also allowed for tracks that overlap in time.
Finally one can filter the track in different ways. Common things to do are to remove very short tracks, or to remove none moving tracks. These things are easily implemented, so feel free to add them at one point. This script includes another filtering step: the removal of parallel tracks. Sometimes a bug is given to detections. This can for example happen if he is covered with flour in the middle. In this case a second track can run parallel and very close to the actual track for some time. Such a track is identified by looking at both the average distance of its detections as well as the distance of its start and end to other tracks.

2. Setting up the camera on a raspberry pi:

Plug in the camera on the raspberry main bord. The camera plug has two small handles, that you need to pull up. Then insert the camera wire. Make sure the pins are on the correct side. Now you have to pull the handles down again to fix the connection. See
https://youtu.be/lAbpDRy-gc0?t=53
for video instructions.
Now we have to activate the camera. Start the pi, open a terminal and run

```
sudo apt-get update
sudo apt-get upgrade
```

(if you don’t do this camera option might not appear in the config menu)
Enter the conifg menu with the command

```
sudo raspi-config
```

Enable camera, either on the main page of the config menu, or go to interface options and enable the camera there.
Reboot the pi with

```
sudo reboot
```

your camera should now work. Check this by taking a single image. Use the command

```
raspistill -t 1 -o test.jpeg
```

Here -t tells the camera how long to wait before taking an image. Default values is 5000 ms. -o specifies an output file with either relative or absolute path.
You might have to repeat these steps if you ever start your pi without the camera.

3. Installing all necessary python packages and other software

Note: python3 comes already installed on the raspberry pi.

The required packages are numpy, scipy ,cv2 , tqdm, imageio, matplolib, picamera.
Generally you can install packages with the comand “sudo apt-get install python3-”package name””. Then you should be able to import them in a python3 environment with “import “package name””.
Please try installing the packages in the given order below. If any problem should arise fell free to contact me for further instructions. Note that some packages might take several minutes to install.
Use the following commands to install the packages:

```
sudo apt-get install python3-numpy
sudo apt-get install python3-scipy
sudo apt-get install python3-tqdm
sudo apt-get install python-opencv #should correctly install cv2 to python3
sudo apt-get install python3-imageio
sudo apt-get install python3-matplotlib
sudo apt-get install python3-picamera
```

Best check after each installation step if the packages are installed correctly by opening a python window with the command

```
python3
```

Then try importing the package with one of the following commands

```
import numpy
import scipy
import cv2 #installed as opencv
import tqdm
import imageio
import matplotlib
import picamera
```

In order to view the output videos on the raspberry pi you can used imagemagick.
installed it by typing

```
sudo apt-get install imagemagick
```

Then you should be able to display gifs, avis and maybe other videos with the command

```
annimate video_file.gif # animate has been installed as part of imagemagick
```

from the console.

4. Getting ready to start a tracking experiment

To start an experiment you need to do three things:
1) Focus the camera and make sure the dish is in the field of few of the camera.
2) Mark the area of the dish.
3) Choose appropriate parameters for tracking. Especially "sd_threshold", "max_dist", "s_max" should be considered.

4.1 Focusing the camera

The camera can be focused using the white plastic tool provided with the camera. Place the tool on the camera lens and carefully scre2 left or right to zoom in or out. This is easiest when opening a live stream. (Note let me know if this doesn’t work for some reason). Note that you don’t have to be to exact when adjusting the focus, as the image is somewhat blurred during processing anyway.

Go into the python3 environment with the command

```
python3
```

Now copy and paste the following lines in the terminal

```
from picamera import PiCamera
import time

camera=PiCamera() #establishes a connection to the camera
camera.resolution = (1400, 1000) # sets the resolution for the camera. Note that setting the
#resolution for the picamera also changes the field of view. Thus make sure this is the same value as
#used for tracking
camera.start_preview() #starts preview
time.sleep(20)  # preview will last for 20 seconds, Increase if necessary
camera.stop_preview() # stops preview
```

Now you can push the dish in the field of view and adjust the focus of the camera.
With this the experiment is physically set up. Now we have to select the dish area and choose appropriate parameters

4.2 Selecting a dish area - The gui_segmentation.py script

First we need to take a single example image. Use the command

```
raspistill -t 1 -o -w 1400 -h 1000 -o test.jpeg
```

Here w and h give the resolution of the image in pixels width and height. Make sure to use the same values as will be used in the experiment.
Now start the script gui_segmentation.py by opening a terminal, navigating to the folder of the script and typing

```
python3 gui_segmentation.py
```

Be aware:
1) the script is somewhat buggy, closing and reopening the script is always a good idea.
2) if you started the script via the terminal, use

```
pkill python3
```

in another terminal to close it. This is not necessary if you start it from some other programming environment.
3) The script runs rather slow on the raspberry. Also opening several images in a row might slow it down further. Best close and reopen it if you feel it gets to slow.

Selecting a mask for the dish area

open an image by pressing "open image". Draw the outline of the dish by clicking on the image and drawing a circle. Press save changes. The mask will be saved as the file mask.py in the folder of the script. A message will be printed to the console showing the output path.

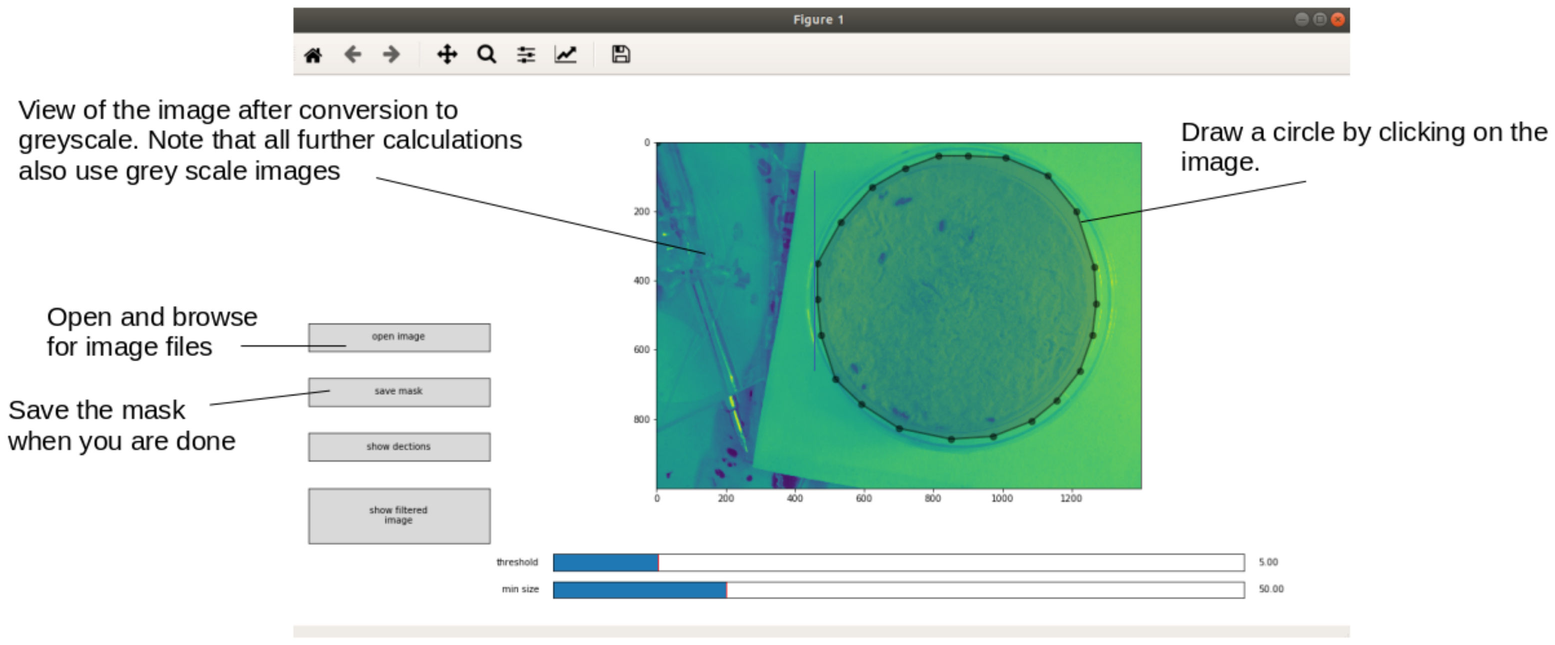


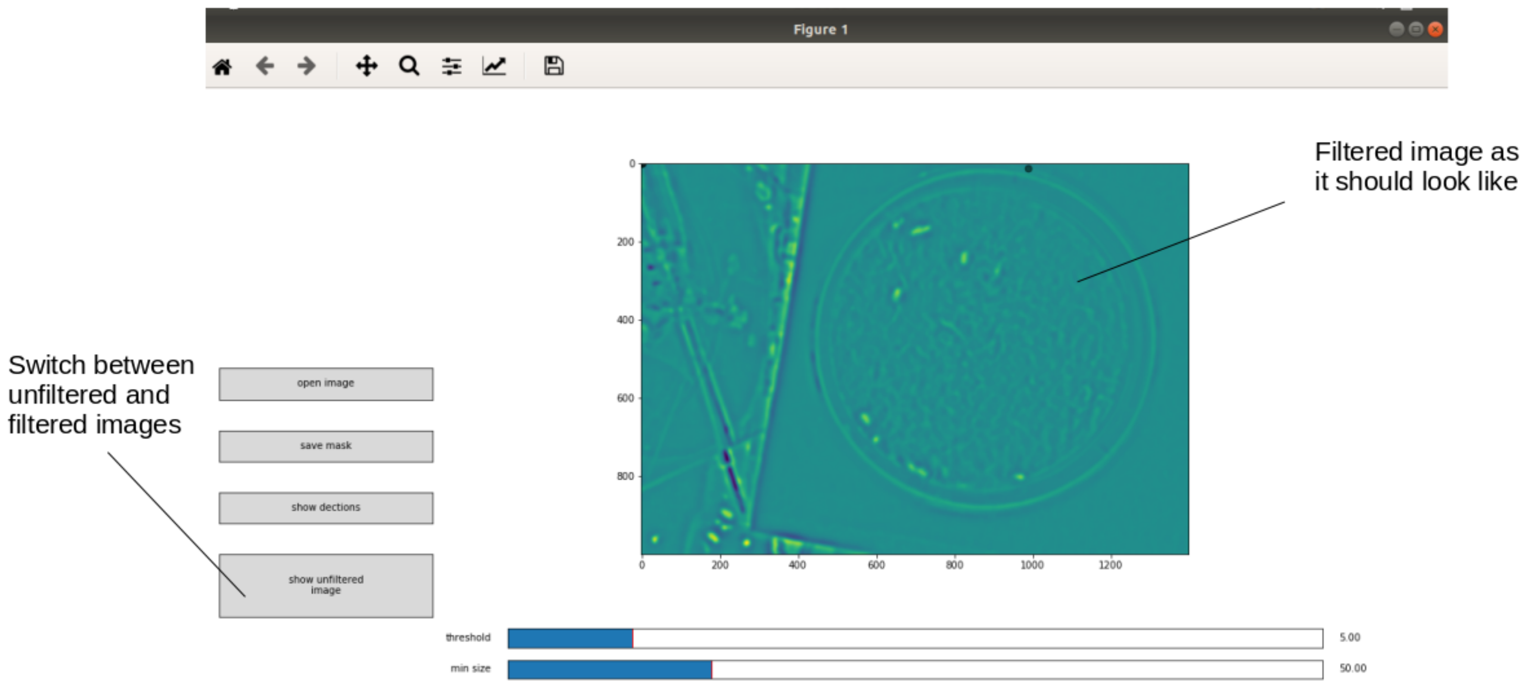
Figure 1
View of the image after conversion to greyscale. Note that all further calculations also use grey scale images
Draw a circle by clicking on the image.
Open and browse for image files
Save the mask when you are done
open image
save mask
show dections
show filtered image
threshold
5.00
min size
50.00

Using the Graphical User Interface to select a mask

While drawing the mask it is also possible to reposition individual vertices and do other things. A right mouse click will delete your previous mask.
Refer to https://matplotlib.org/3.1.1/api/widgets_api.html#matplotlib.widgets.PolygonSelector for some more information.

4.3 Choosing the correct parameters

(see Appendix for an exhaustive list of parameters, their meaning and recommended values)
The gui_segmentation.py can also help you to select some parameters for successful tracking. Open an image by pressing the open image button. First choose show filtered image. Check if the bugs a clearly visible blobs (like in the figure below). Also check if other objects,such as scratches (furchen) from the bugs in the flower are blurred or filtered out completely. If this is not the case change the parameters s1 and s2. (You will have to edit gui_segmentation.py to do this.)



View of the filtered image. In this example the bugs are clearly distinguishable form the background, and other objects are blurred. If this is not the case you need to tune the parameters s1 and s2.

Then check for the sd_threshold. Click on the slider to display the segmentation as a red mask. Our goal is to have the bugs covered with this mask. The absolute value of the threshold is also displayed on the right over the slider. It refers to the filtered image. You can use this to choose a good threshold. Just view the filtered image, hover over a bug and check what intensity value is displayed in the lower left corner. Note that no value might be displayed if you have drawn a mask in this image or have displayed the segmentation. You can delete any displayed mask with a right mouse click (only while hovering over the image) then the value of pixel intensity will be displayed again in the lower left corner. A sd_threhold of 5 is normally a good starting value.
Press show detections to check out the coordinates that the program predict for a bug. If you see points on mask regions that are very small, you can try the min_size parameter to exclude them. Click on the “min size” slider (and then click on “show detections” again) and see what happens.

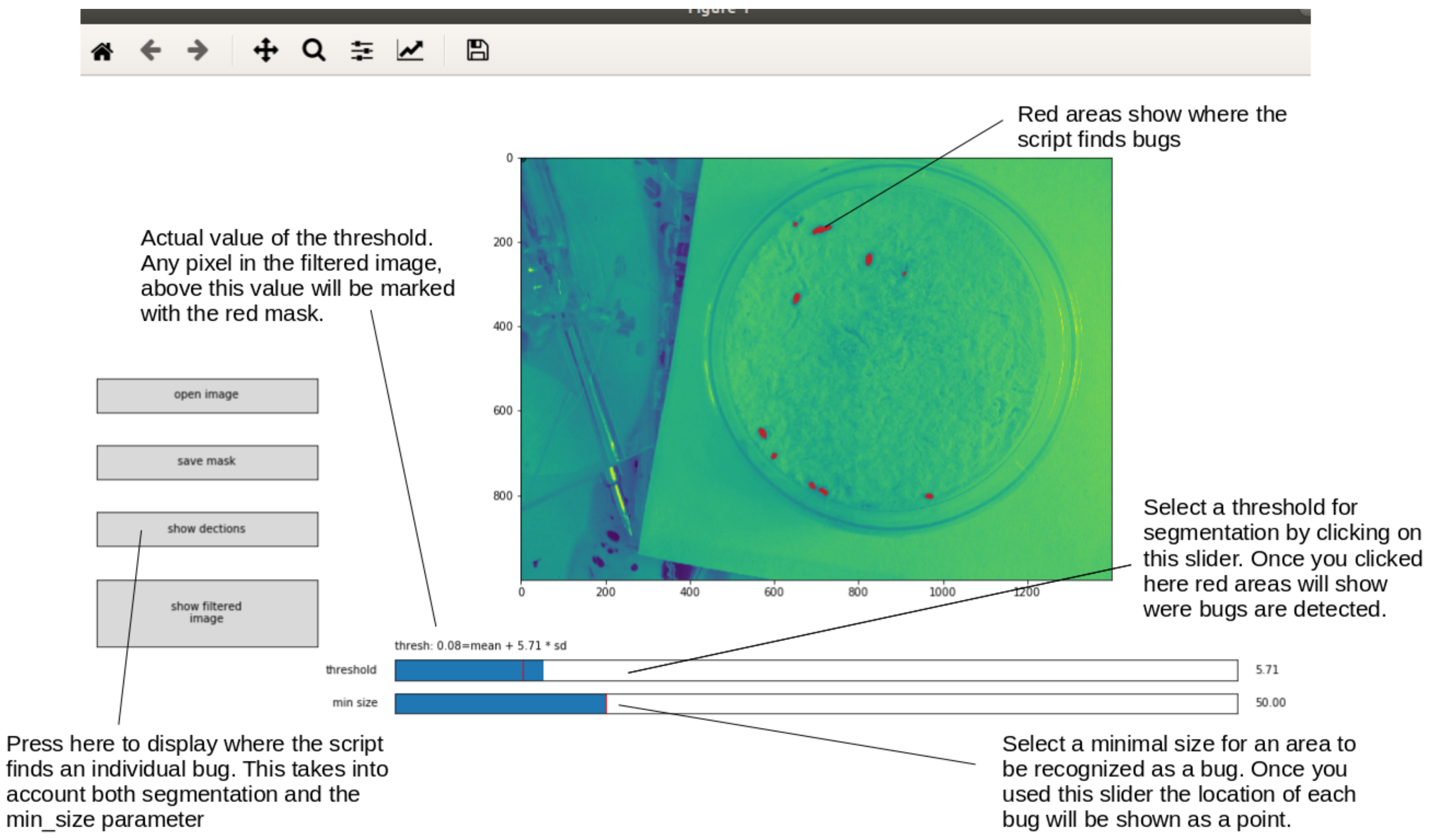


Figure 1
open image
save mask
show dections
show filtered image
thresh: 0.08=mean + 5.71 * sd
threshold
5.71
min size
50.00
Red areas show where the script finds bugs
Actual value of the threshold. Any pixel in the filtered image, above this value will be marked with the red mask.
Select a threshold for segmentation by clicking on this slider. Once you clicked here red areas will show were bugs are detected.
Press here to display where the script finds an individual bug. This takes into account both segmentation and the min_size parameter
Select a minimal size for an area to be recognized as a bug. Once you used this slider the location of each bug will be shown as a point.

Using gui_segmentation.py to choose some tracking paramters.

For the other two parameters you need to consider how fast the bugs are moving. If you run the program at the highest frame rate, it could analyses one frame every 5 seconds.
Thus you have to choose max_dist slightly large then the maximal distance a bug can travel in 5 seconds. You could also try to be a bit smaller. Note the bugs run faster if you shook the dish shortly before, so probably its best to wait until they calm down. I'd recommend to choose s_max to be 1.25*max_dist or smaller.

5. Starting a Tracking experiment

Open the script tracking_experiment.py. Check all parameters at the beginning of the file and change their values if appropriate. You need to provide 3 names to out put files as well for the images (if you want to save them) and a folder where everything is stored.. The first output file is a text file for the raw tracks, the second one a text file for the stitched and filtered tracks, then a .pickle file for the time points at which an image is taken. The files will be generated if they don't exist, or overwritten if they do. However you need to make sure that the folder you located them in exists.
The tracking_eperiment.py script will run for the given number of frames, then perform stitching and filter tracks and the make a video of the tracking. Its however recommended to make videos separately after the script has run through

6. Output and post processing

The script will produce an text file where each row stands for a frame and each column for a track. The first Column contains the frame numbers. Track columns have no separated header. Each cell of this frame-track-matrix contains two entries, representing the x and y position of a bug. If these entries are -1 then no detection at this frame was found. In the script vizualization_and_analysis.py you will find functions to read the output files into dictionaries. Each dictionary assigns a track id to a list of 3 values: x and y position of a detection and the number of the frame it occurred in. For some calculations it might be useful to transform this dictionary to an array. Use the return_track_array function for this. Also you will find the function to produce videos here. Among other things you can choose which subset of frames to display, how many tailing detections are displayed and whether to use actual recorded images in the background. Note that this part might(?) requires high ram usage if many frames are to be displayed. You should probably not go over 2000 frame when making these videos on the pi. An easy way to monitor ram usage is to go to a terminal and open a task manager like program by typing "top"

Appendix

Table 1: Overview of parameters

Name	Function	How to Choose	Recommended Values	Things to look out for

Segmentation and tracking				
s1	Parameter in Filtering. Sets the size of small objects to be removed. Blurrs out e.g. shadows in the flower. If the surface gets uneven.	Check the filterd image with gui_segmentation.py. Change this in the gui_segmetation script itself.	5 in this setup	Depends on the resolution
s2	Parameter in Filtering. Sets the size of small objects to be removed. Blurrs out e.g. diffrences in illumination in areas of the dish	See above	12 in this setup	Can easily be set higher, but this will slightly increase calculation time
sd_threshold	Calculation of a threshold for segmentation according to threshold=mean+standard_deviation*sd_treshold	Check the segmentation with gui_segmentation.py	5	Gives false results if no bugs are present or the mask is drawn incorrectly
min_treshohld	Optional minimal threshold. If no objects are present the formula might set the threshold to low.		None	Since there are always bugs in the dish we should not need this.
min_size	Filtering all objects below this size.	Check the detection with gui_segmentation.py	30 (pixel)	Should be much lower then the size of a bug in pixels
max_dist	Maximal distances allowed to connect two detections while tracking	Estimate how far a bug can move in Frame (in pixels)	150 pixel	Bugs might move around faster if you shake the dish
Post processing				
s_max=150	Stitching: Maximal distance allowed for stitching two tracks	You probably should try different values here. Make sure there are no wrong connection introduced by stitching.	150 pixel	This parameter is rather hard to choose
f_max=10	Stitching: Maximum difference of frames between end of one track and start of another track allowed for stitching.		10 frames	Depends on the frame rate. Don’t make this to high.
f_min=-2	Minimum difference of frames for stitching. Negative Values means that overlapping tracks can be merged.		-2	Should be smaller or equal to zero, but don’t make it to small
mean_dist	Removal of parallel Tracks: Maximal average distance of a track to another track, that allows removal.	Check if there are incorrect Tracks e.g. due to detecting a bug twice. The choose these parameters.	30	Incorrect parallel Tracks might not occur very often or at all in your

				setup. You can also try leaving this step out.
end_dist	Removal of parallel Tracks: Maximal distance of a track end and start to another track, that allows removal.	See above	30	
Imaging parmameters				
n_frames	Number of frames to be recorded/analyzed. This will set the length of the experiment.			
spf	(Seconds per Frame) Time between two frames in seconds	You should go as low as possible. The goal should be to have always the same length of time steps. Confirm this by looking at the time printed to the console each frame	5 seconds	
keep_images	Set weather images are saved	Always save your images if you have enough space. SD cards offer essential limitless read and write circles	True	At these settings you can record for roughly 10 hours at 2 seconds per frame.

Common problems

The camera and the dish must remain at the same place during the experiment. It might be beneficial to fix the camera with tape. Additionally somehow marking the exact position of the dish could be useful -

The format of the output files is not suit for a large number of tracks. If you expect more then a ~4000 tracks (also considering that the program is likely produce multiple tracks per insect) you need to save the output in a different format.

Other useful things

Finding the IP address of your pi:
Open a terminal and enter the command
```
hostname -I
```

Note that the IP address changes depending on which network you connect it to.