**Tasks**

Figure 1: Bounding box of the person in frame 62 of the "walkstraight" sequence.

**Task 1 (50 points):** Write a Matlab function called find\_bounding\_box that takes in as argument the name of an image file from the "walkstraight" sequence, and computes the bounding box of the person. The function should RETURN the bounding box, as a matrix of four numbers: [top row, bottom row, left column, right column]. Furthermore, as a side effect, the function should display a figure that shows the original image, with a yellow (color code: [255 255 0]) rectangle superimposed, representing the detected bounding box. Your function can use data from any frame of the sequence in order to determine the bounding box for the frame in question.

Your function should be named find\_bounding\_box, and should take a single argument, i.e., the filename specifying a frame of the sequence. For example:

> [t, b, l, r] = find\_bounding\_box('walkstraight/frame0052.tif');

Don't worry about how the function works when the person is not visible or is only partially visible.

**Task 2 (50 points):** Write a Matlab function called person\_present that can tell when no person is present. Don't worry about how your algorithm performs on borderline cases, like frames 5-32 when the person is not fully visible. However, your algorithm should be able to tell, for example, that there is no person at frame 3, and that there is a person at frame 62. The function should return 1 if the person is present, and 0 otherwise.

Your function should be named person\_present, and should take a single argument, i.e., the filename specifying a frame of the sequence. For example:

> person\_present('walkstraight/frame0052.tif');

**Task 3 (optional - extra credit - 20 points):** Write a Matlab function called person\_speed that returns the average speed of the person, in pixels per frame, between two frames. Your solution can be built on top of your find\_bounding\_box function: call find\_bounding\_box twice, to find the person in both frames, and calculate the speed based (somehow) on the results of the find\_bounding\_box function.

Your function should be named person\_speed, and should take two arguments, i.e., the filenames specifying two frames of the sequence. For example:

> person\_speed('walkstraight/frame0052.tif', 'walkstraight/frame0062.tif');

Again, don't worry about how the function works when the person is not visible, or is only partially visible.

**Task 4 (optional, for fun and no credit):** Design and implement a computer vision-based heuristic, that can tell us something about the pose of the person. One pose (let's call it Pose 1) is exemplified at frames 48, 67, and 84 (among others), where one leg is extended forward and another leg is extended backwards. Another pose (let's call it Pose 2) is exemplified at frames 40, 58, and 75, where the legs are next to each other. You can incorporate your heuristic into the solution for task 1 (the function can print, as a side effect, POSE 1 or POSE 2). Again, don't worry about cases in between, like frames 56 or 78, in that case just let your program print its best guess.

Useful code and hints

* Almost the entire solution for Task 1 is included in the code we covered in the lecture. You just need to package it up nicely as a single function.
* Use the addpath function to let Matlab know where your Matlab code is located. Type help addpath to see how that works.
* In general, familiarize yourselves with the code we used in the lectures, you will find lots of Matlab tricks there that can be handy for this assignment.
* File draw\_rectangle.m (attached) implements a function that draws a rectangle.
* Files parse\_frame\_name.m and make\_frame\_name.m (attached) contain code that you should feel free to use, and that you may find useful if you want your code to automatically figure out the filename of the next frame, or previous frame, and so on. For example, try:
* [sequence\_name, frame] = parse\_frame\_name('walkstraight/walkstraight0062.tif');
* filename = make\_frame\_name(sequence\_name, frame+1);

Things to note

* The correct solutions should be functions, not scripts. See the read\_gray.m file to see an example of a function.
* There is no single unique answer for any of the tasks. Just make sure your solution behaves reasonably well.
* It goes without saying that the solutions should be based only on computer vision, not on tricks like using the frame numbers. For example, if your solution for task 2 simply checks if the frame numbers are too small or too large, that is not a computer vision-based solution.

How to submit

Assignment solutions should be submitted on Canvas. The submission should include a zip file containing your Matlab code and a README.txt file. The zip file should be named assignment1.zip. Your submission should contain all Matlab files that your solution uses, regardless of whether you wrote them or you downloaded them from the course website. The README file should contain the following:

* Name and TxState ID of the student.
* A description, in text, of the solution for Task 2, and how well it worked (examples where it worked, examples where it didn't work, if any).
* A description, in text, of the solution for Task 3, and how well it worked (examples where it worked, examples where it didn't work, if any).
* A description, in text, of the solution for Task 4 (if you tried it), and how well it worked (examples where it worked, examples where it didn't work, if any).

We try to automate the grading process as much as possible. Not complying precisely with the above instructions causes a significant waste of time during grading, and thus points will be taken off for failure to comply, and/or you may receive a request to resubmit.

Submission checklist

* Was the submitted zipped file called assignment1.zip?
* Did you include a README.txt file, as specified?
* Was the attachment in ZIP format? We will not accept .rar, .tar, .gz, or any other filetypes, and we will not accept submissions where multiple files are attached separately.
* Did you make sure that the attachment includes only code and text files, and NO IMAGE files?