AVADS User Manual

Updated December 17th, 2019
Alex Peebles, apeebles@vt.edu

Automated Video Analysis for Dynamic Systems (AVADS) is a MATLAB graphical user interface that can automatically track high contrast markers throughout a video and compute a variety of kinematic outcomes. While this program was designed to provide athletic trainers, clinicians, and physical therapists with an easy method to objectively measure human movement, AVADS can be useful for many applications.

Table of Contents

System requirements

This version of AVADS has been tested on MATLAB version r2018b – r2019a, and will likely work with any newer version. The program does require the MATLAB image processing toolbox and works best on desktop computers with at least 8 GB of installed RAM.

QuickStart instructions

To start, save the AVADS.m and AVADS.fig files in the same location, then open and run the AVADS.m file in MATLAB. Click the *Open New* button to open a new video. Points can be manually tracked using the *Manual Digitize* buttons, or automatically tracked using the *automatic tracking* button. In order to automatically track markers, a threshold first needs to be set to segment the desired marker from its background. Once markers are tracked, AVADS allows the user to compute multiple angles and kinematic based outcomes. Additionally, maker data and computed kinematics can be stored for offline analysis and reopened at a later date to continue data processing.

Detailed instructions

A: Open new file and Clear names

Clicking *Open New* will prompt you to navigate to and open a video file. While AVADS has been tested using .mp4 and .mov video files, it should work for any video file type which MATLAB's vid and read function work with. When you first open AVADS, this will open the current directory, however, if opening a second video without closing AVADS the program will open the directory of your previously opened video. Additionally, if you have previously processed and saved a video (See section M), all of your saved options, marker data, and computed variables will restore upon opening the video again as long as the "_processed.mat" file is saved in the same folder as the video file. Clicking *Clear names* will set all marker names, angle names, and output variable names back to their default setting.

B: Name each marker

In this program, you are able to track up to 8 markers. Here, you can give each marker a specific name which exports when you save your data (See section M). Note, the color next to each marker will correspond with the colored block overlaid the marker during the tracking process. Note: You will need to change the name of each marker you want to track before proceeding with digitizing and automatic tracking.

C: Set threshold

The ability to automatically track markers in AVADS relies on the ability to segment markers from the background. To do this, a threshold is set where pixels darker then the threshold are set to black and pixels lighter than the threshold are set to white (i.e. the image has been binarized). In AVADS, click *Test threshold* to change the image to black and white, and then use the slider bar to find a value (between 1 and 255) where the marker you wish to track turns either black or white, and the immediately surrounding pixels turn the opposite. As the automatic tracking functionality finds the centroid of every white blob of pixels, if you are tracking black markers you need to select *Invert Black/White* to make your markers white. Note that you can play or skip through your video (See sections O and T) to see if your threshold will work across time. Once you've identified an appropriate threshold, unclick *Test threshold* and proceed with manual and automated marker tracking (See sections D and E).

D: Manually digitize all markers

Before you can automatically track markers, you will need to first manually digitize each marker on at least one frame of your video. To digitize all markers at once, select *Manually digitize all markers*. You will first be prompted to zoom in on the first marker, then after pressing enter a red crosshair will appear and you will be prompted to click on the center of the first marker. After digitizing the first marker, you are allowed to zoom to the second maker and so on and so forth until you have digitized all the markers you have named (See section B). Note: If you do not change the name of any markers (See section B), you will not be able to digitize. Color blocks will be overlaid on the frame you've digitized, and then the program automatically advances to the next frame. If you wish to make these blocks bigger, change the marker block width (See section Q).

E: Automatically track markers

Once you have selected an appropriate threshold to segment your maker (See section C) and have digitized markers for at least one frame of data (See section D), you can begin automatic tracking by selecting *Automatic tracking*. If an error occurs and you wish to stop *automatic tracking*, simply unclick the button. The option to only *plot every XX frames* is intended to speed up data processing, markers are still tracked behind the scene for every frame and the video is just updated periodically so the user can identify if a tracking error has occurred. Also, please note that automatic marker tracking works best when markers have been identified

for at least two consecutive frames. If errors in tracking occurs after one frame, fix the tracking mistakes with *manual digitize one marker* on the second frame (See section G) and continue *automatic tracking*.

F: Clear this and future frames
If an error in tracking occurs and you wish to clear marker data, select *Clear this and future frames*.

G: Manually digitize one marker
If the marker you are tracking is occluded (blocked) and automatic tracking fails, navigate to the frame after the occlusion ends, select *Manually digitize one marker* and digitize your marker before resuming automatic tracking.

H: Compute segment and joint angles
Angles are computed between the intersection of two lines in this program. Here, you can use digitized marker data to form lines between two markers, and create two lines to form an angle (e.g. angle between thigh and shin segments). Also, *Ang1Name* and *Ang2Name* are editable text boxes, and can be changed to any desired name which will export when saving data (See section M). If you only want to compute one angle with respect to the image horizon, you can select *Horz* for both options in line 2.

I: Name and export user-defined output variables
While processing videos for various projects it may be advantageous to save variables (e.g. peak velocity). Here, you can name any output you would like and type in the value of that output. These variables will be saved and exported with the marker trajectories (See section N).

J: Compute max, min, range, or mean of timeseries data
Once digitizing markers and/or creating segment and joint angles is complete, AVADS can automatically compute standard statistics (max, min, range, or mean) of data either for the entire trial or between user defined frames. In the first list box, select which stat measure you wish to compute, then in the second text box select what signal you want to compute the variable on (can be a digitized marker or constructed angle). Next, select if you want to compute the outcome on the position (or angle), velocity, or acceleration of the selected signal. Finally, select the frame range you wish to compute the variable for (or leave it as beginning and end), and click *Compute*.

K: Frame rate
This box will report the frame rate of the collected video. This is not an editable box.

L: Plot data
Once markers have been digitized (manually or automatically) and/or angles been constructed, you can plot your data using the signal selection list box and *Plot data* button. You can also plot the velocity or acceleration of marker positions or constructed angles.

M: Cut and Fill missing gaps of data and Save all data
If an occlusion occurred during marker tracking (See section E), you can select the frame range of the occlusion and fill over the missing data by selecting *Cut & Fill*. Before selecting *Cut & Fill*, select the type of interpolation you wish to perform (spline or linear). After selecting *Cut & Fill*, zoom in on the occlusion in the plot, press enter, and then click on the fame to the left of and then to the right of the occluded section. Note, only the X component of the cursor crosshair is used to select the frame range of the occlusion.

To save digitized marker coordinates, computed angles, and saved output variables for further offline analysis, or to save data analysis progress to resume at a later date, click *Save data*. All variables and options will be saved in the same folder as the opened video and in a file named with the beginning of the video file name with an "*_processed.mat"* added on the end of the name.

N: Filter marker data
If desired, you can enter a filtering frequency here and press *Low pass filter*, and AVADS will use a fourth order recursive low-pass butterworth filter on XY marker coordinates. Clicking *Unfilter* will revert the data back to its original state before the *filter* button was pressed.

O: Play video, zoom, pan, and cursor
To play the video, click the *Play video* button. To stop the video, unclick *Play video*. To zoom, pan, or click on the video or graphed data, click the corresponding buttons. Note, cursor allows you to click on a point and the value of that point will be displayed.

P: Measure lengths and angles
Selecting measure length will allow you to click on any two points on the image and a line between the two points with its corresponding length will be drawn on the image. Selecting measure angle will let you click on three points on the screen and two lines with their corresponding angle will be drawn on the screen. Note: these lines and numbers are not saved in the videos image, and changing the frame will erase them.

Q: Marker block width
Colored blocks with a specified pixel width will be overlaid on the video frames where markers have been identified to ensure proper tracking. Depending on the camera's resolution, marker size, and/or distance from the camera, you can change the block width to make data visualization easier.

R: Calibrate video
Marker coordinates are saved during *digitization* and *automatic tracking* in pixel units (i.e. how many pixels they are away from the corner of the image). To convert from pixel distances to meters, measure the length of an object with a known distance in the video (See section Q), enter the measured length of the object in pixel units, then the true length in meters, and click *Calibrate video*. Note that for the calibration to be accurate, the object should be in the same plane as your markers (not closer/further), oriented parallel to the camera, and be much larger than the tracked markers.

S: Frame number
Here the currently displayed frame of the video and number of frames will be shown.

T: Control frame number
To change the frame of the video displayed, you can scroll through the image, or click the left and right arrow buttons to go forwards/backwards one frame. Clicking further or closer along this bar will skip 10 frames.

U: Video
This is where the video images and graphed data will be displayed.

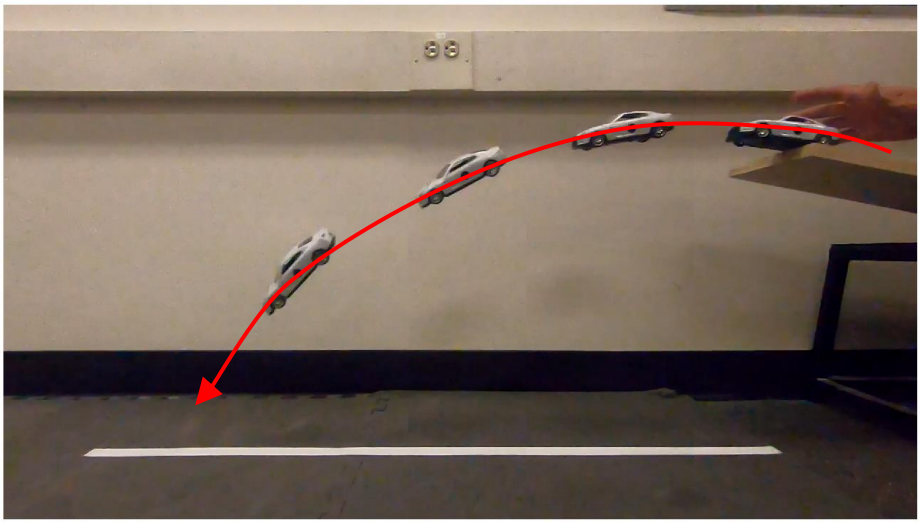
V: User message box
This box will display instructions and messages to the user throughout marker tracking and data analysis.

Guided tutorials

The following tutorials are intended to 1) demonstrate how AVADS can be used to assess kinematic outcomes, 2) suggest multiple labs or class activities where AVADS can be used to determine how well theoretical equations can predict motion, and 3) demonstrate how our group has used AVADS for biomechanics research aimed to help prevent injuries in athletics. **All videos for this tutorial can be downloaded below!!**

Projectile motion: Download video

A matchbox car was shot off a ramp (Figure), and your goal is to measure the cars velocity in the x and y direction as it leaves the ramp, the peak height of the car in the air, and distance it travels before contacting the ground. The car weight is 0.153 kg and the white strip of tape is 1 meter long.



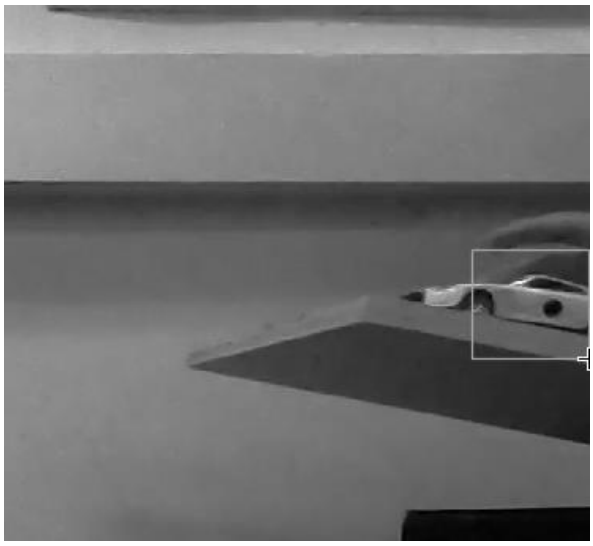
Step 1: Click *Open new*, then navigate to and open the file named "Car.mp4".

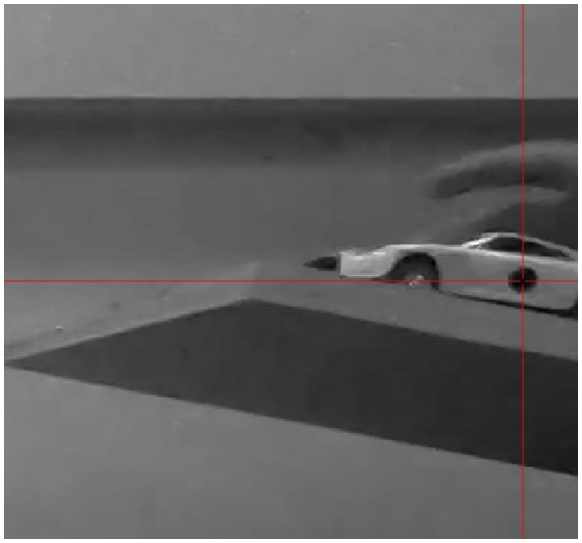
Step 2: For this video, we are going to track one marker placed nears the car's center of mass (along its long axis). Change *M1Name* to *Car*.

Step 3: The car comes into the field of view on frame 79, so drag the slide bar and/or click the advance frame button to navigate to that frame.

Step 4: Find an appropriate threshold to visualize the black dot against the white car. Click *Test threshold*, and slide the threshold bar to change the threshold. I found that setting the threshold to 50 worked well, you generally want to set a threshold that allows you to visualize as much of the marker as possible. As this is a black marker on a white background, click *Invert black/white*. Once you have selected a threshold you are happy with, unselect *Test threshold*.

Step 5: Change the *marker block width* to 7 so you can see the marker better during tracking. On frame 79, click *Manually digitize all markers*, zoom in, press enter, and then click on the center of the black circle (See figure below).





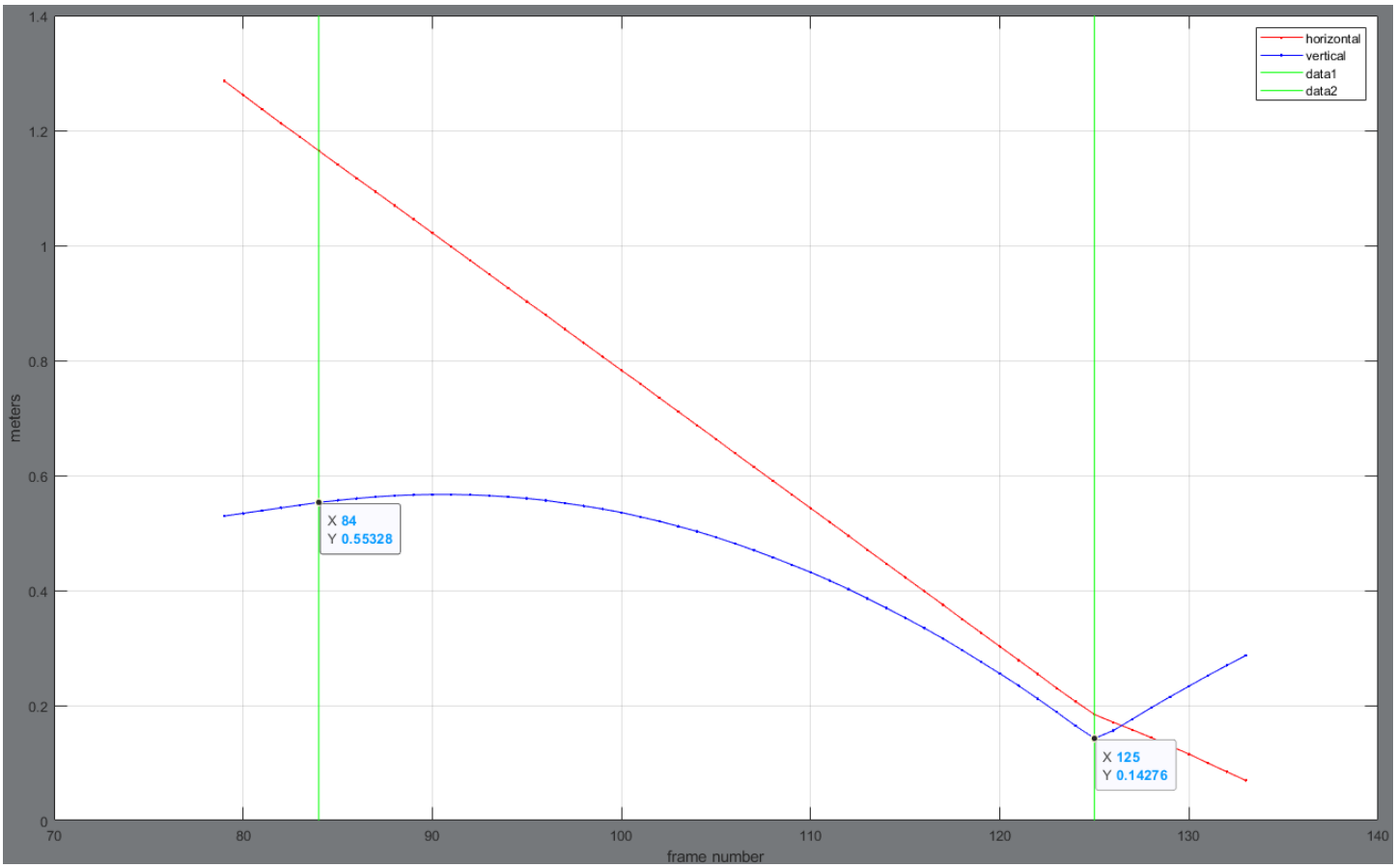
Step 6: Click *Automatic tracking*, and watch as the red color marker does not track the dot on the car. Unclick *Automatic tracking* and go back to frame 80. Select *Manually digitize one marker* and make sure M1 is selected in the drop-down list. Zoom in, and correct the markers position on frame 80. Then, on frame 81, click *Clear this and future frames* and try automatic tracking again.

Step 7: The program should correctly track the car in the air, however, with the rapid change of direction at ground contact it does loose the marker at frame 126. Unclick *automatic tracking*, navigate to frame 126, manually digitize M1, then clear from frame 127 and on. If desired, you can automatic track for a few more frames until the car leaves the field of view, however, we will only analyze the first flight phase.

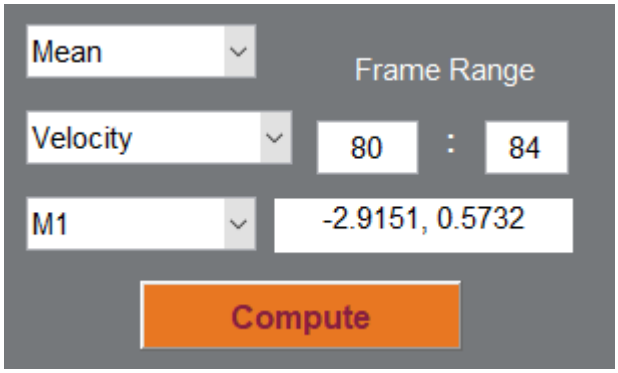
Step 8: Calibrate the image by first measuring the length of the white tape in pixels. Select *Measure Length*, and then click on the left and right sides of the tape (you should get something around 1464). Enter this distance into the left of the calibration equations, and enter 1 meter to the right. Finally, click *Calibrate video*.

Step 9: Visually identify which frames takeoff and ground contact occur, I chose frame 84 for takeoff and 125 for ground contact. Change *Var1Name* to takeoff, *Var1Value* to 84, *Var2Nam*e to Contact, and *Var2Value* to 125, and then select *Event* next to Var1 and Var2 values.

Step 10: Now, click *Plot Data* while M1 and Position are selected and you should see a plot that looks like this: While the position data is still plotted, click *Cursor*. Click on the vertical trajectory at takeoff, then while pressing shift click on the vertical trajectory at contact. You now should have a plot that looks like the plot blow. Take the difference between the two heights as the takeoff height (0.553-0.143=0.41 meters). Change *Var3Name* to TOHeight and *Var3Value* to 0.4099.

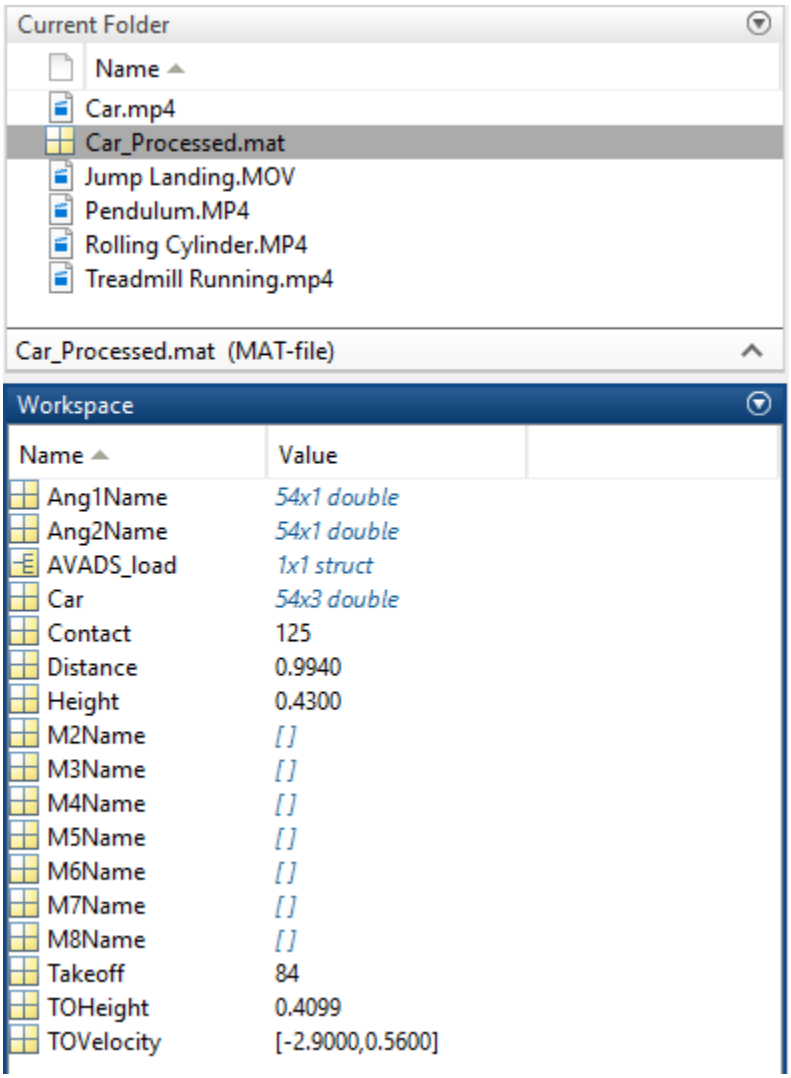


Step 11: Compute the mean velocity of marker 1 between frames 80 and 84 and click compute, to get V_o (See below). You should get V_o = -2.92 m/s in the x direction and 0.57 in the y direction. Change *Var4Name* to TOVelocity and *Var4Value* to -2.92, 0.57.



Step 12: Compute the range position of marker 1 between frames 84 and 125 to get travel height and distance. You should get distance = 0.994 meters and height = 0.430 meters. Change *Var5Name* to Distance, *Var5Value* to 0.994, *Var6Name* to Height, and *Var6Value* to 0.43.

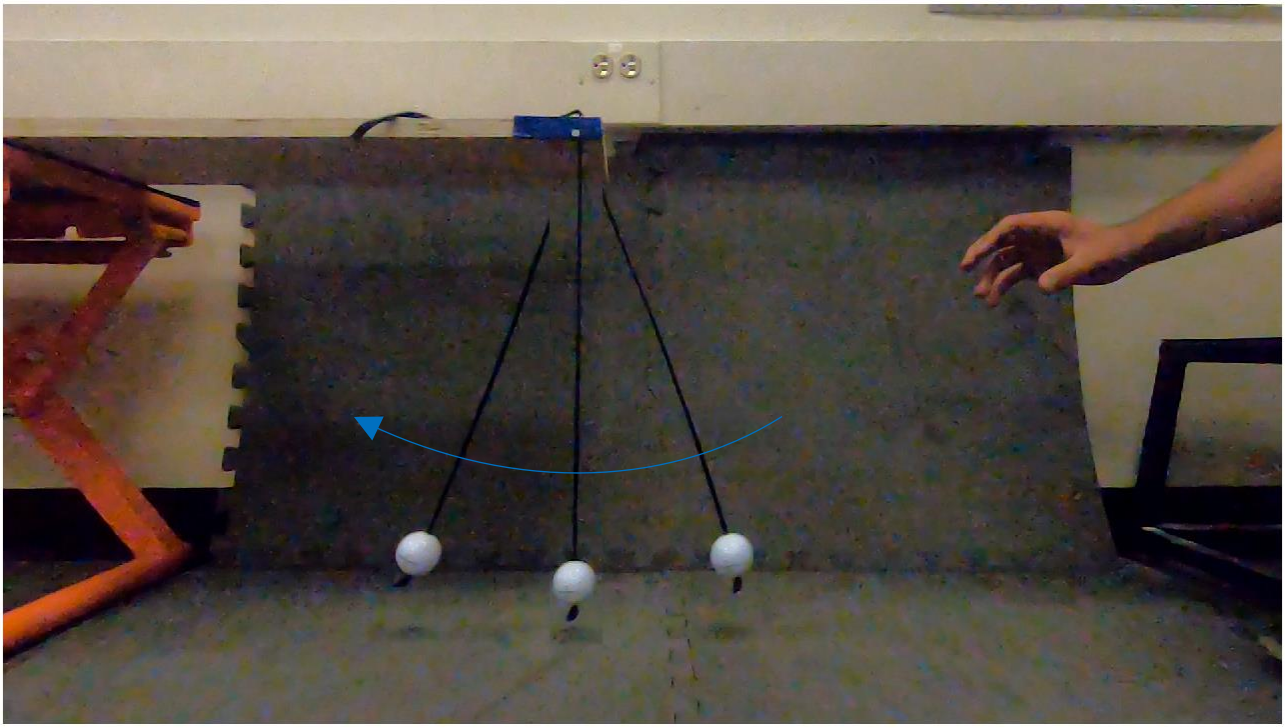
Step 13: Click *Save Data*, and then go back to MATLABs command window. In the file tree to the left, navigate to where Car.mp4 was stored. There should be a new file saved with the video named "Car_Processed.mat". Double-click on that file to open it. All of your tracked marker trajectories and saved output variables should be stored in this file.



Suggested class activity: Build a similar experiment, and test the effect of car speed, weight, and initial velocity on height and distance. How well do kinematic equations predict these two outcomes? What assumptions are we making with these equations which may lead to prediction errors?

Pendulum: Download here

In this example we have a 0.05 kg golf ball oscillating about a pivot point 0.435 meters away and filmed at 120 frames per second. Your goal is to measure the period of oscillation and energy loss between the first and fifth oscillation.



Step 1: Open "Pendulum.mp4".

Step2: In this video we are going to track the position of the pivot point and the golf ball. Change *M1Name* to Pivot and *M2Name* to Mass.

Step 3: Click *Test threshold* and as we are tracking white markers make sure "Invert Black/White" is unclicked. Try changing the threshold around so that you can visualize the pivot point and golf ball well, I found that 130 worked for segmenting both markers.

Step 4: Change *marker block width* to 7, click *Manually digitize all markers*, zoom in on the white square, press enter, click on the center of the white square (approximating pivot point), then zoom out and back in on the golf ball, hit enter, and then click on the center of the golf ball.

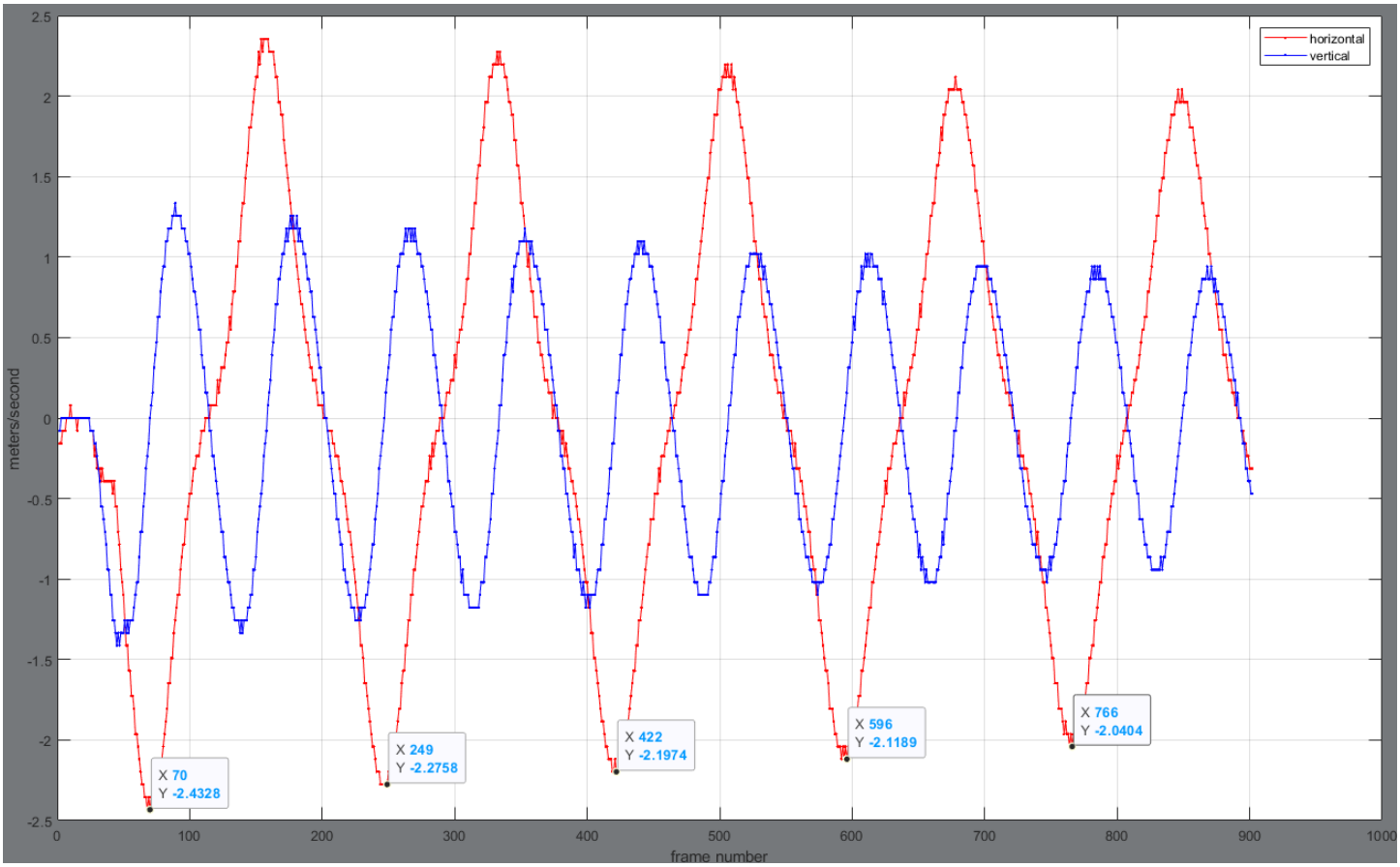
Step 5: Click *Automatic tracking* and watch the system track the marker for at least one full period. Then, unclick *Automatic tracking*, change "Plot every 2 frames" to "plot every 20 frames" and resume autotracking for the rest of the video.

Step 6: Change *Ang1Name* to PendAngle, select M1 and M2 for line 1, and Horz for both option under line 2. Plot *position* of *Ang1* and visually ensure that you have expected oscillations with no twitches in the signal (usually the sign of a brief tracking error).

Step 7: Click *Measure length*, and measure from the pivot point to the center of the golf ball, I got 664.5 pixels. Enter this number and the pendulum length (0.435 meters) into the calibration equation and click *calibrate*.

Step 8: Change the filter frequency to 10 Hz and click *Low pass filter*.

Step 9: Plot the velocity of marker 2, click *cursor*, and then click on the first 5 minimum peaks (left passes) which holding down control. Your plot should look like this:



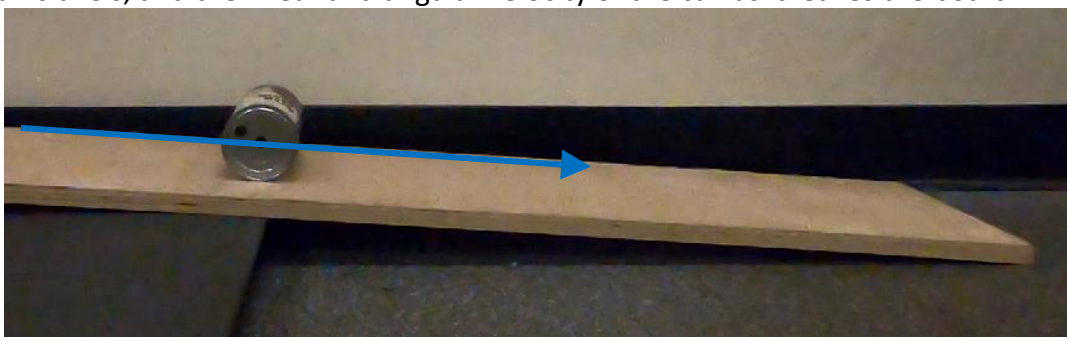
Step 10: Manually compute period for first 5 full oscillations using the 'X' component in the above plot, which corresponds to frame number. Manually compute the difference between each consecutive frame numbers and divide by 120 frames/s to get period in seconds. I got ~1.5 s. Save this as variable 1 and name it period.

Step 11: Manually compute kinetic energy for the first and fifth oscillation during the left pass ($KE=\frac{1}{2}mv^2$). The velocities (Y component in above plot) are 2.43 and 2.04 m/s, and the ball mass is 0.05 kg, corresponding to 0.148 and 0.104 Joules. Save these values as variables 2 and 3, then click *Save data*.

Suggested class activity: Test the effect of changing mass or length on oscillation frequency. How well do certain assumptions (i.e. sin(x)=x) hold true for small verses large angles?

Rolling without slipping: Download here

In this example, we have a can of delicious and nutritious black beans (0.459 kg; diameter 7.4 cm; length 11.2 cm) rolling down a 1.215 meter board. Compute the angle of the board with respect to the horizon, the distance the can travels, and the linear and angular velocity of the can as it leaves the board.



Step 1: Open "Rolling Cylinder.mp4".

Step 2: Change *M1Name* to Center and *M2Name* to Edge.

Step 3: Find a threshold that segments the black dots from the can, I found 20 works well. Make sure *Invert Black/White* is checked.

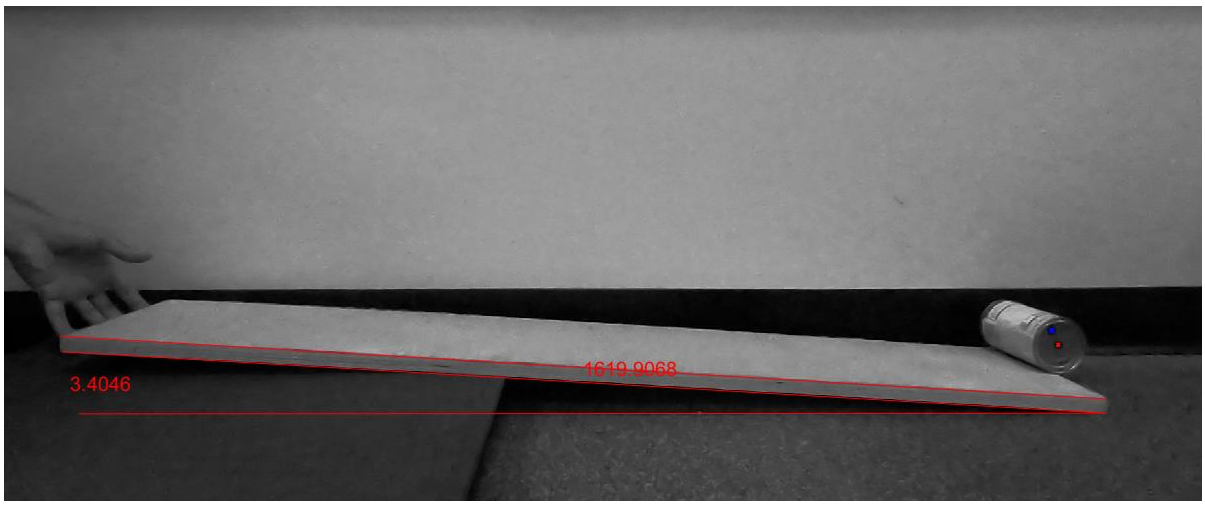
Step 4: Click *Manually digitize all markers* and click on the black dots (Center First).

Step 5: Select *Automatic tracking* and let AVADS track the markers throughout the entire video.

Step 6: Change *Ang1Name* to CanAng, and create an angle by selecting M1 and M2 for line 1 and Horz for both in line 2.

Step 7: Measure the length of the board (I got 1619) and calibrate the video (true length = 1.215 meters).

Step 8: Click *Measure angle* and click on the bottom left corner of the board, then bottom right, and then along the horizon from the bottom right of the board (See below, I got 3.4 degrees). Save this as *Var1Value* and name it *board angle*.

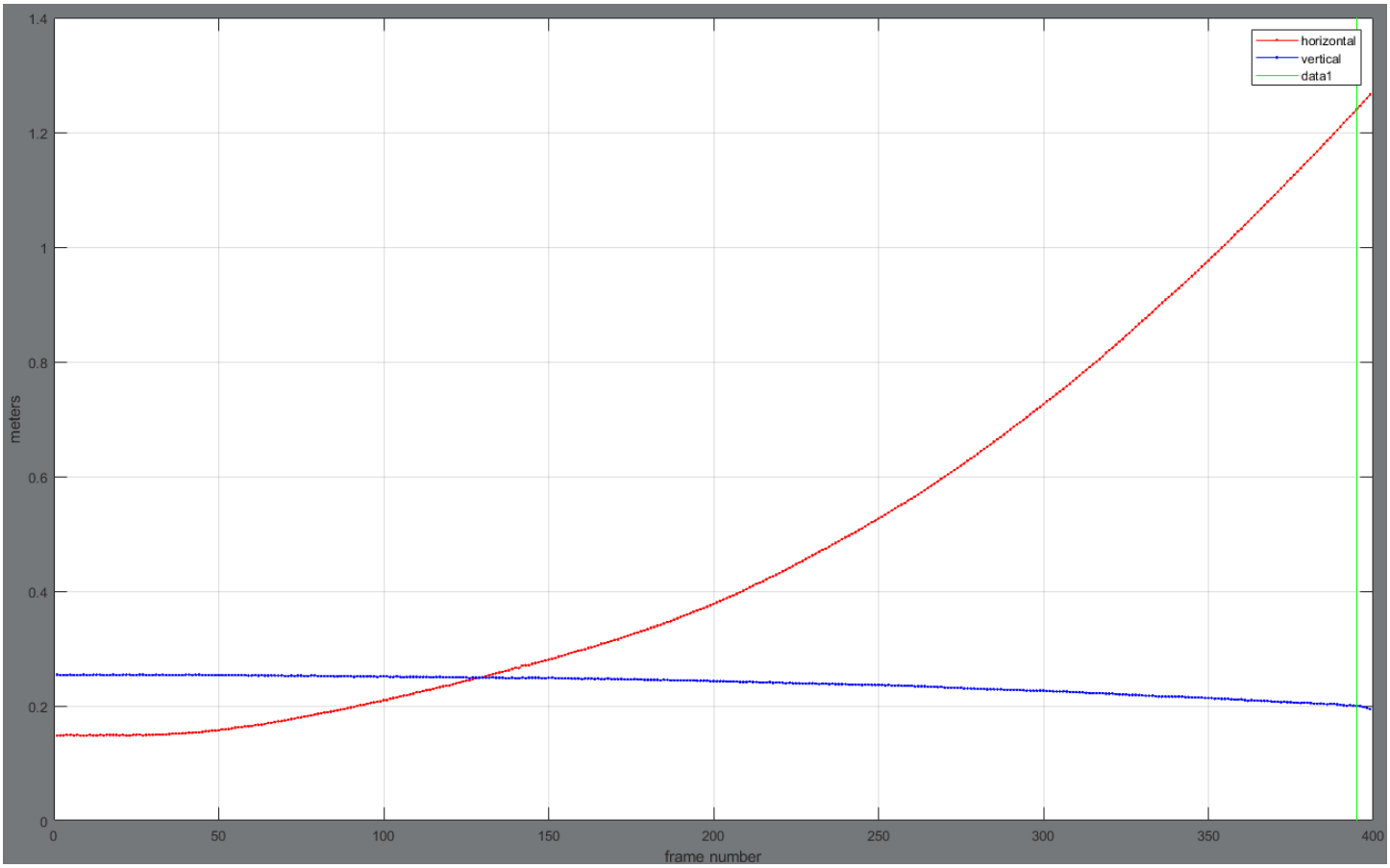


Step 9: Identify the frame in which the can leaves the board, I chose frame 395. Save this as variable 2, name it can off, and check that this is an event.

Step 10: Compute the range of Marker 1's position between frame 1 and 395 (I got 1.092 and 0.055 meters in the X and Y direction). Save this (1.092, 0.055) as variable 3 and name it distance traveled.

Step 11: Compute the average velocity of marker 1 between frames 391 and 395 (I got 0.720, -0.054 m/s in the X and Y). Save this as Variable 4.

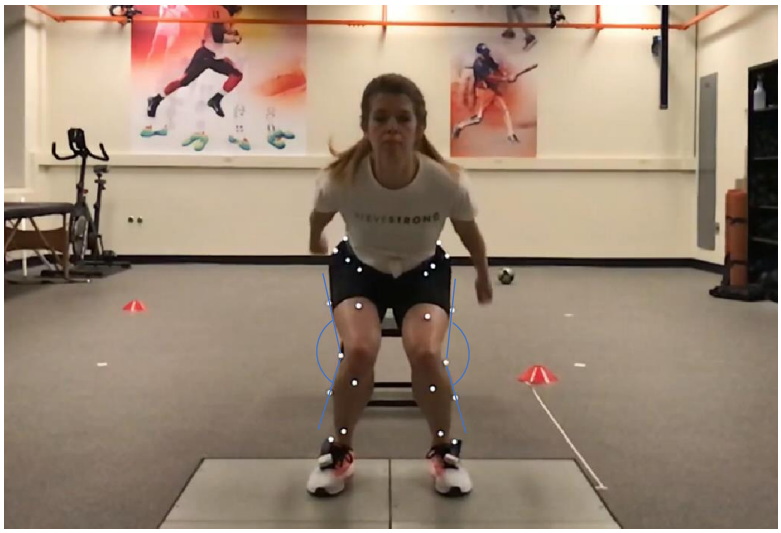
Step 12: Compute the average velocity of angle 1 between frames 391 and 395 (I got 1463.7 degrees/s). Save this as Variable 5 and then click *Save Data*



Suggested class activity: Determine the effect of slope, mass, and radius on the cylinder's velocity. How well do the derived equations of motion predict the measurements made in the video?

Knee valgus during a jump-landing: Download here

In this video you have an athlete completing a forward drop vertical jump. Frontal plane 2D knee valgus (shown below) is an important measure for injury risk screening. Your goal is to compute knee valgus angle at initial contact, peak knee valgus angle, and knee valgus range of motion during the first ground contact.



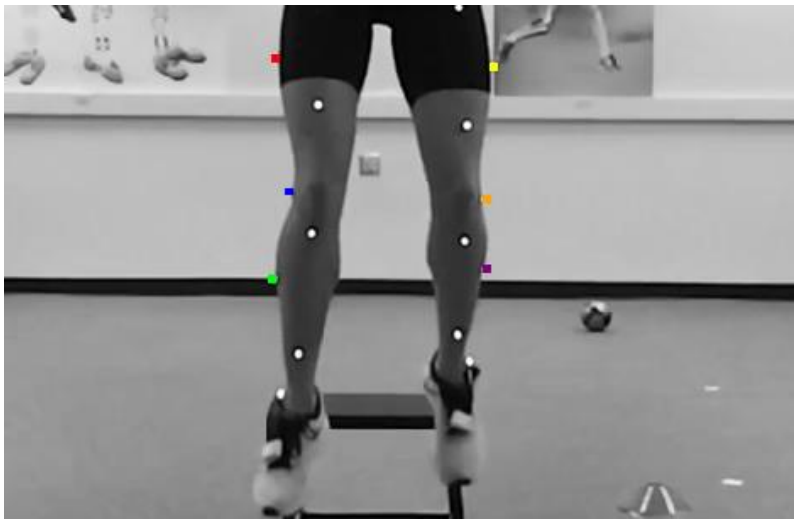
Step 1: Open "Jump Landing.MOV".

Step 2: Name Markers 1-6 RTh, RKn, RSh, LTh, LKn, and LSh.

Step 3: Scroll through the video to when the participant is in the air before landing. I chose frame 860.

Step 4: Set a threshold to segment the white markers from the dark background, I chose 230.

Step 5: Manually digitize the lateral thigh, knee, and shank markers on each leg in the order that you named them. The marked image should look like below.

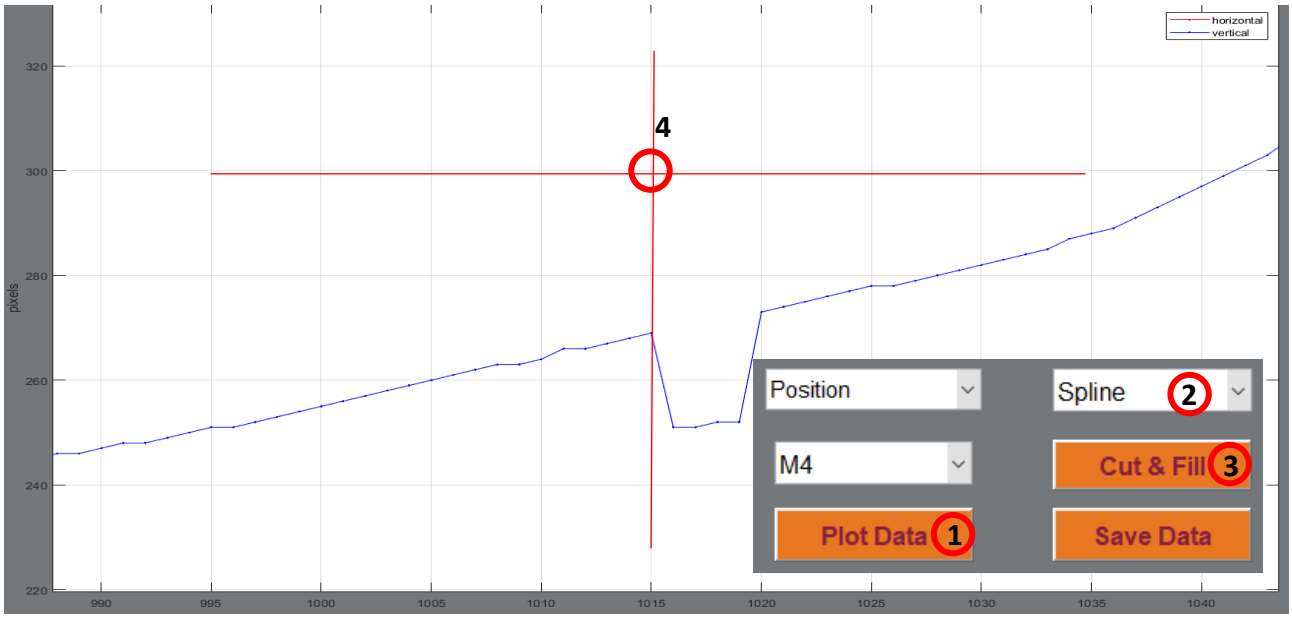


Step 6: Click *automatic tracking* and let the program track the six markers. Unclick *automatic tracking* when the left thigh marker becomes occluded (arm blocks the marker and tracking fails).

Step 7: Navigate to the first frame after the occlusion, when you can fully see the left lateral thigh marker again (frame 1020). Click *Manually digitize one marker* with M4 selected, zoom in, and click on the lateral thigh marker.

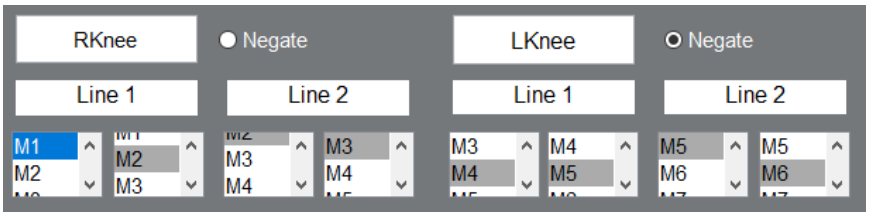
Step 8: On frame 1021, click *clear this and future frames*, then resume *automatic tracking* until the participant is off the plates.

Step 9: Correct the Left Thigh markers location during the occlusion. Plot the position of M4, then zoom in on the occlusion (Only need one of the traces, see below). Make sure that *Spline* is selected for the fill type and click *Cut & Fill*. Click on the screen with the vertical crosshair lined up with the frame to the left of the occlusion and then click to the right of the occlusion (see below). Only the horizontal location (vertical line) of this click is being used to identify the beginning and end of the occlusion. After doing so, the graph should refresh and the occlusion should be spline filled.

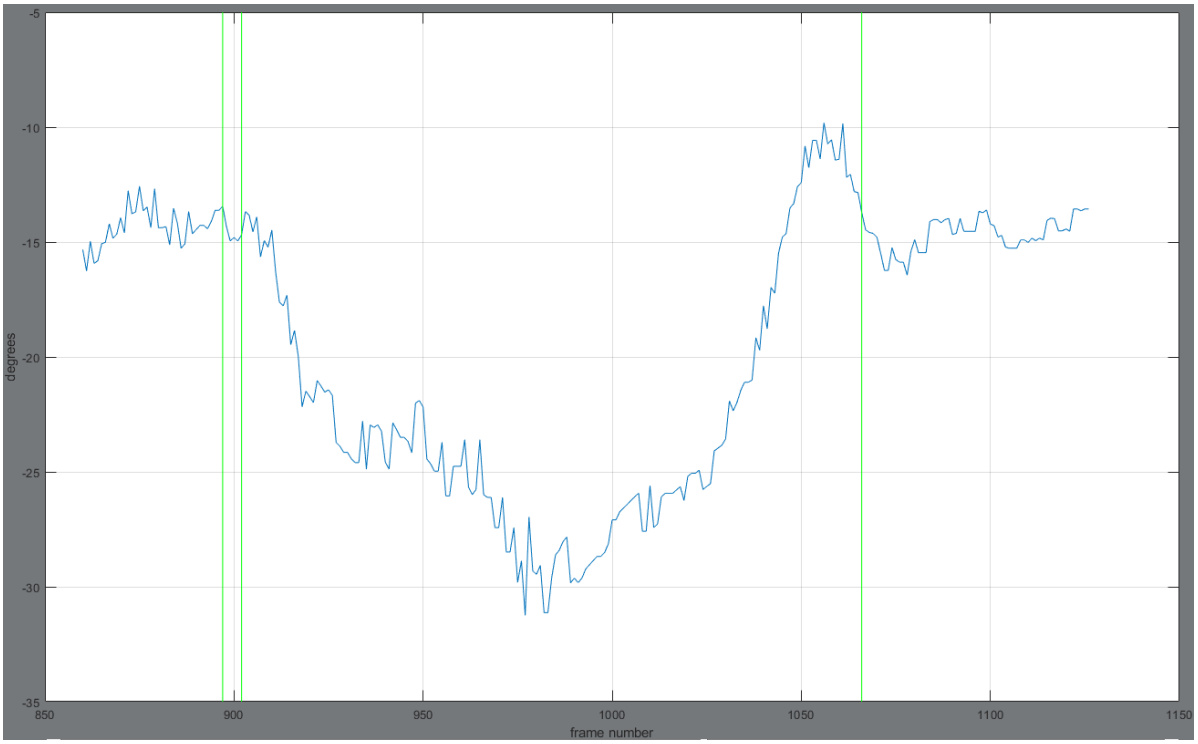


Step 10: Identify the time frames when initial contact occurs on the right and left foot, and then when toe off occurs, I chose frames 897, 901, and 1067, respectively. Name *Var1Name RIC*, *Var2Name LIC*, and *Var3Name TO*, and then change *Var1Value* to *897*, *Var2Value* to *901*, and *Var3Value* to *1067*, and select event next to all three variables.

Step 11: Change *Ang1Name* to *RKnee* and *Ang2Name* to *LKnee*. Knee angles for this project are created by forming a line between the thigh and knee and another between the knee and shank. Therefore, for RKnee angle line 1 select Marker 1 and 2, and for line 2 select marker 2 and 3. Do the same for the LKnee with markers 4-6 (see below). Click *Negate* for LKnee so that both signals will have the same sign for valgus and varus (negating LKnee will make valgus negative for both Left and Right Knee).

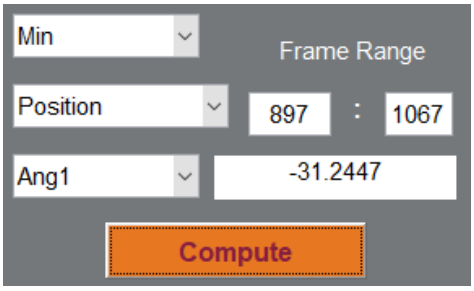


Step 12: Plot the position of Ang1, and you should get a graph that looks like the following. This corresponds with the first angle we created (RKnee), and the green lines on the plot show initial contacts (right then left) and toe off). Do the same for Ang2.



Step 13: Compute valgus angle at initial contact for the right and left knee angles. Do this directly in the plot by selecting cursor and clicking on the intersection of the signal and its corresponding initial contact line. The valgus angle is the Y-component of this intersection point. I got -14 for the right knee and -6 for the left knee. Change *Var4Name* to *IC Angle* and *Var4Value* to *[-14, -6*].

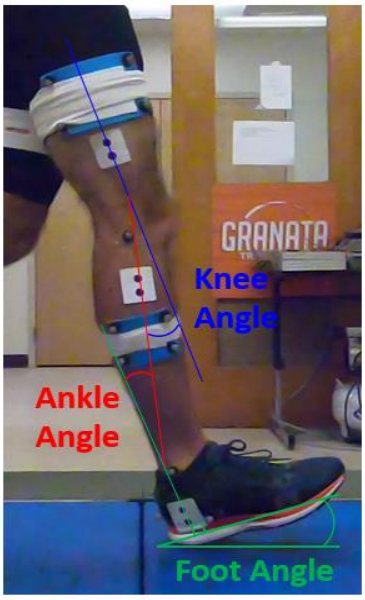
Step 14: Compute peak valgus angle during ground contact by changing the compute options to match below and clicking compute. Shown is the right knee (Ang1 and IC is frame 897). To calculate peak valgus for the left knee, change these to Ang2 and 902 for the (should get ~-23). Change *Var5Name* to *Peak Angle* and *Var4Value* to *[-31.2, -23.3].*



Step 15: Compute range of motion during ground contact the exact same as you did peak angle, except choose *Range* instead of *Min*. You should get 21.3 and 19.7 for the right and left knees, respectively. Save these values for Var6, then click *Save Data*.

Suggested class activity: Compare peak knee valgus angle between men and women athletes.

Sagittal plane running kinematics: Download here



In this example we have a participant running on a treadmill with markers on the lateral aspect of their thigh, shank, and foot. We want to track the two dots on each of these markers, use these dots to create segment angles, and then use the segment angles to compute knee and ankle angles. Your goal is to compute knee and ankle angles throughout the entire 10 second video, to calculate knee and ankle angle at the first heel strike, to calculate peak knee flexion during the first stance phase, and to calculate knee flexion range of motion for the first stance phase.

Step 1: Open "Treadmill Running.mp4".

Step 2: Change *M1Name – M6Name* to *Th1*, *Th2*, *Sh1*, *Sh2*, *Ft1*, and *Ft2*, respectively.

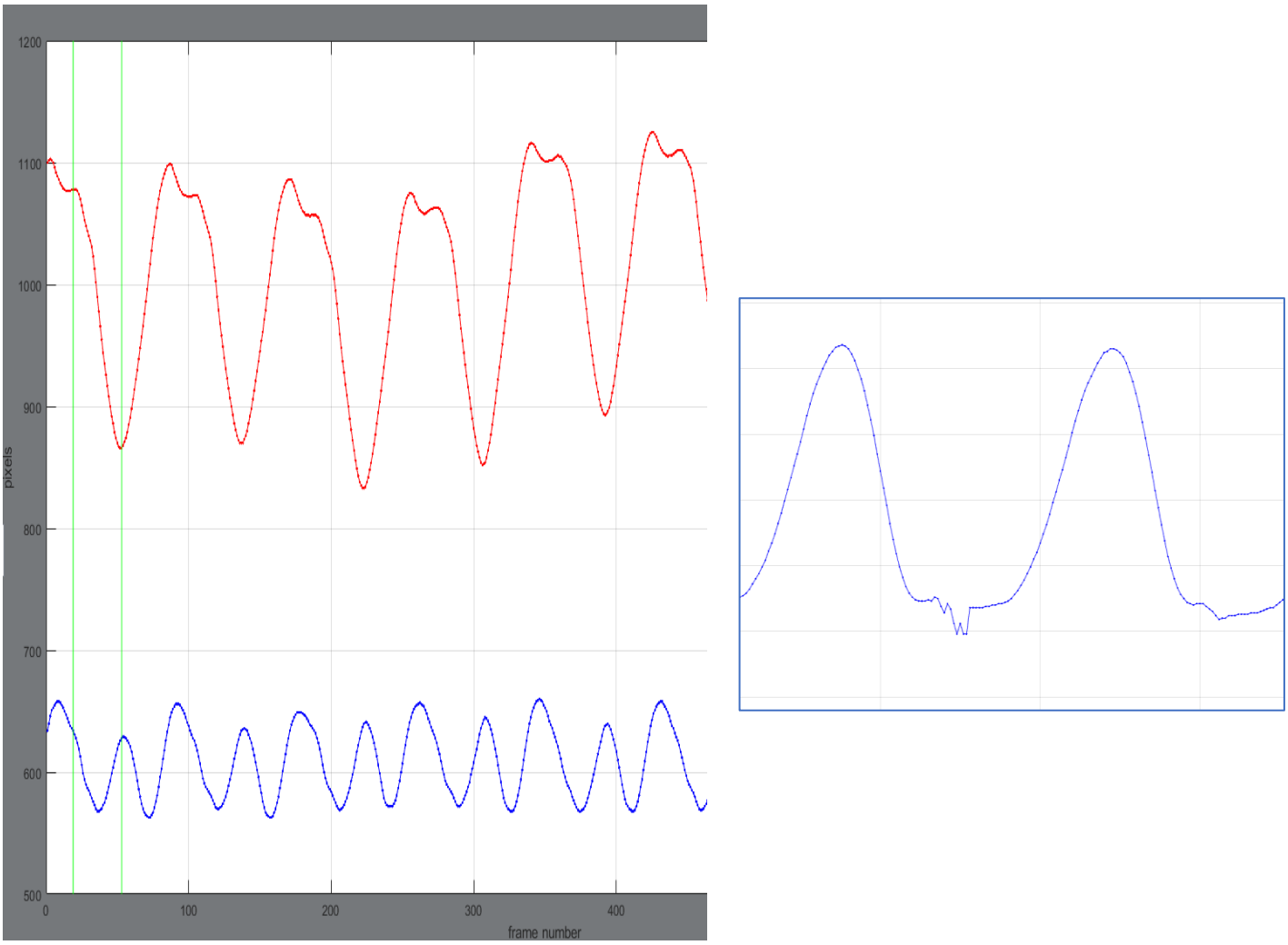
Step 3: Find a threshold to segment the black dots from the white sticker background. I found that 75 worked well. Also, make sure "Invert black/white" is selected as we are tracking black markers.

Step 4: Click *Manually digitize all markers* on frame 1 and then manually zoom in on and click on each marker in the order that you named them. Remember you are allowed to zoom in between selecting each marker, and you need to click enter once your done zooming and ready to digitize. Click on the top marker first for each segment (i.e. Th1 is the dot closest to the hip).

Step 5: Change *Marker block width* to 7 to improve data visualization.

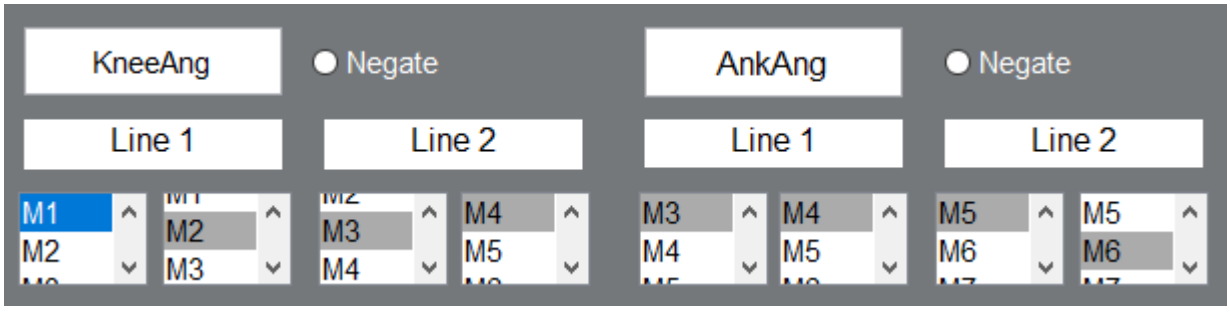
Step 6: Click *Automatic tracking*, however get ready to unclick *Automatic tracking* quickly as the feet markers will get lost on frame 2. Navigate back to frame 2 and then select *Manually digitize one marker* with M5 selected, and correct the location of the Ft1 marker. If M6 also needs to be corrected on frame 2 do so, then click *clear this and future frames* on frame 3 resume *Automatic tracking* for the rest of the video. If desired, you can change the *skip frames* option to ~20 to speed up tracking.

Step 7: Especially if you were skipping frames during *automatic tracking*, it is important here to plot every marker to make sure your data is continuous (below left) and does not have any irregular patches (below right). Irregular patches are usually the sign of the marker losing its track for a brief period of time and then fixing and reattaching itself. In the case below to the right, I had originally tracked with a higher threshold (80) and the lowest foot maker lost its track near heel strike on a few strides due to a shadow on the foot.

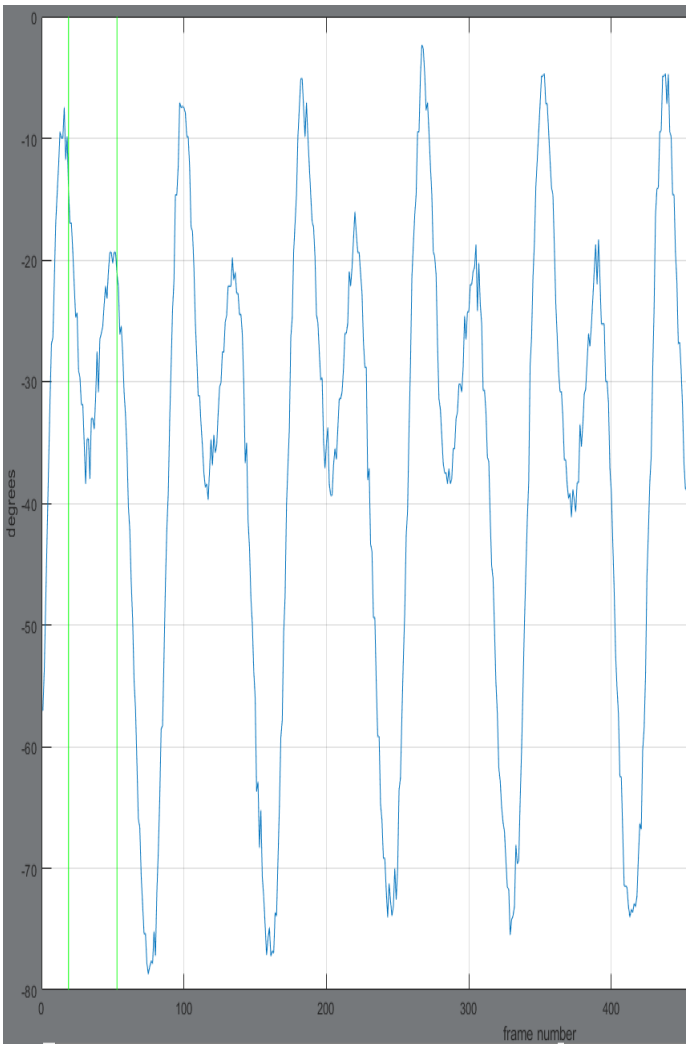


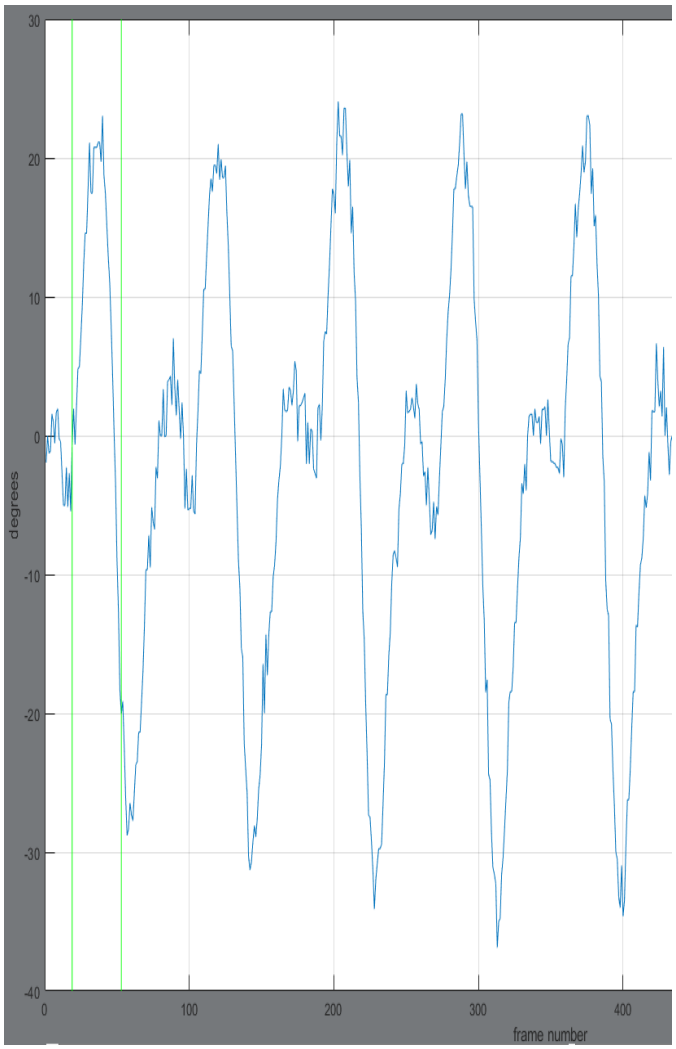
Step 8: Change the *filtering frequency* to 7 Hz and low pass filter your marker data.

Step 9: Create knee and ankle angles. To compute knee angle we are creating a thigh segment angle between M1 and M2 and a shank segment angle between M3 and M4, and then taking the difference between the two segment angles as the joint angle. Similarly, ankle angle is computed between the shank (M3 and M4) and foot (M5 and M6) segment angles. Your two angle setups should look like this:



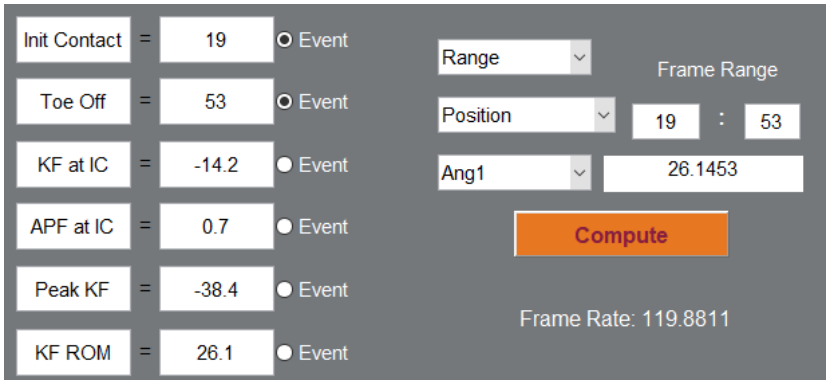
Step 10: Plot the knee (Ang1, below left) and ankle (Ang2, below right) angle time series to view joint angles throughout the trial.





Step 11: Identify heel strike and toe off for the first right leg stance phase, I chose frames 19 and 53. Set these frames in variables 1 and 2 and check off that they are events. Then, plot the knee and ankle angles again and use the cursor to measure knee and ankle angle at the first heel strike (I got -14.2 and -0.7 degrees, respectively).

Step 12: Compute the minimum knee angle between the first heel strike and toe off (I got -38.4 degrees) and knee angle range of motion (I got 26). Click "Save Data".



<u>Suggested class activity</u>: Determine the effect of increasing cadence by 10% on knee and ankle angles at initial contact