

Vister

How to use Vister

The following document is a part of the Vister software developed in collaboration with SDU and LEGO and seek to help the user in setting up the system correctly.

Table Of Content

[Introduction 4](#_Toc103354323)

[Setting everything up 4](#_Toc103354324)

[How to run Vister 4](#_Toc103354325)

[Manually running Vister 5](#_Toc103354326)

[The Activation Test 6](#_Toc103354327)

[How it works 6](#_Toc103354328)

[How to set it up 7](#_Toc103354329)

[The Motion Tracker Test 8](#_Toc103354330)

[How to set it up 8](#_Toc103354331)

[Key Notes 8](#_Toc103354332)

# Introduction

Vister 1.0 includes two different testing types. One that utilizes communication with a Universal Robot to conduct malfunction testing. This test is called “Activation Test”. The other one helps tracking a moving object to count the laps and how long a lap took, which is called “Motion Tracker Test”. Vister was designed and develop in such a fashion that it can be extended with more testing types given that the developer has some knowledge about ROS and Python programming.

In this guide you will learn more about these two different testing types, what they are cable of and when to use which.

Note: Vister 1.0 only has a Terminal Interface

# Setting everything up

It is assumed that you have a PC running Linux with a Vister 1.0 installment, a Pylon Camera, and a Universal Robot e-series robot.

If you have not installed Vister yet, please read the following guide: “”. If you are experiencing any issues connecting to the Universal Robot, please read this guide for more information: “”.

Make sure that you have the Pylon Camera connected before opening the Vister application as you might run into issues otherwise. If you tend to use the Activation Test, you should make sure you are connected, and have a. URP script beforehand.

# How to run Vister

Vister is a ROS based application and therefor requires you to run terminal commands to get the software up running. It can be an in tedious process if you have no prior knowledge of ROS.

However, this section covers how you can either run it manually by inserting these commands in the Terminal or automatically by clicking the Vister icon.

## Manually running Vister

“Why run it manually if I can just run it by clicking the Vister icon?”. We recommend that at least the first time you run it manually as it gives an understanding of what happens in startup, and it will help you to determine what causes failure the times when it do not startup.

Note: You can open a new Terminal by pressing: CTRL + ALT + T

The first step is to get ROS Core up and running as it is needed to run any ROS command. Open a new Terminal Window and run the following command:

Roscore

This will start up the roscore. This terminal window has to be open all the time while using Vister as it will otherwise fail to run the other ROS commands.

Note: If you at any point tries to run of the other command and it states it cannot find the file. It’s a good change that you haven’t startup up roscore.

The next step is to open our Pylon Camera. For this you need to open a new terminal window and run the following command:

Roslaunch pylon\_camera pylon\_camera\_node

Note: It is important that you use “roslaunch” and not “rosrun”

This will activate the camera. You will see some warning messages, which you do not have to worry about. If any of the text shows up in red there is a good chance that the issue is that your camera is not connected properly.

At the bottom you also must look for the message “Camera is calibrated”. If you do not see this message or it rather states it is not calibrated, you have to go back set the URL for the camera calibration file properly. Read more about this in the PDF: “How to setup Vister”.

Note: Vister 1.0 comes with a camera calibration file created for the Pylon Camera: acA1440-220um

You must leave the Terminal running the Pylon Camera open as well. The last step is to open the test setup node. Open a new terminal and run the following command.

Roslaunch computer\_vision computer\_vision\_start.launch

After this command you should be prompt with a message of choosing either the “Activation Test” or the “Motion Tracker Test”. Voila! You are now ready to run the experiment you want!

# The Activation Test

This testing type was designed with the spring test from LEGO in mind. It helps to run long experiment that uses the UR robot to activate each component. Hence if you have not setup the TCP communication between the PC and the robot it is time to do so. Read the following guide for more information: ““.

The CV algorithm in this test will at the end of each lap check whenever a malfunction has occurred or not. As it must be a generalized testing tool some few requirements have to be fulfilled for it to work.

* The object of interest must be mounted in a fixed position.
* The algorithm assumes that the initial position of each object is in perfect condition.
* The background is uniformly white, and the objects have good contrast.
* There is always the required light luminated on the test setup.

The two last requirements ensures that the camera can see the objects clearly and therefore limits the possibility of the CV algorithm to mistakenly think a malfunction has occurred.

The two first requirements must be fulfilled otherwise the test data will be unusable in determining the quality. It is easier to understand why when you understand how the algorithm operates.

## How it works

You have lined up all your objects and created a. URP scripts that enables that robot to activate all the objects in a given manner throughout the lap.

The CV algorithm then checks for malfunction by comparing the bounding box value of each object, which its initial bounding box value. If the values change great it is assumed that a malfunction has occurred as we then know that the object does not have the same shape as it initially had.

It will store the malfunction videos, if any had occurred, as well as the bounding box values in a CSV file. Besides that, the robot will also store data in another CSV file. These data can then be compared for then to determine the objects disabilities.

## How to set it up

With roscore running in one terminal and the pylon camera running in another and you have started up the test setup node, type ‘a’ to choose Activation Test.

A big window will open those shows what the camera sees. Now you must specify where each of the objects are located by drawing individual region of interest (ROI). You are prompt with the following 3 commands during this setup of ROI’s:

* New ROI
* Delete ROI
* Ready to run

Note: Always start with the upper left corner of your ROI and then the bottom right

Once you are ready run the ready to run command. Afterwards you will have to specify the location of which you want all the data and videos to be stored. We recommend that you create an empty folder for each of your experiment.

At some point it will ask for the IP-address of the robot. Here you type in the IP-address, which is found under the Network tab under the settings of the UR.

It will ask you to switch to “Remote Control” if you are not already in this mode.

Note: Make sure that the robot is in its initial position

It will ask you to enter the name/location of your .URP file. You should have the file already stored directly on the UR itself. If so, you just type in the name of the .URP file

Note: Make sure you type it correctly, otherwise the program will fail dure to safety restrictions on the UR. You in this case must start up the test setup node all over again

Once all the information is typed in experiment should startup by itself, all you must make sure is that the lightning does not change drastically during the experiment time span and that the background does not change either.

# The Motion Tracker Test

The Motion Tracker Test looks for a maker with color properties that clearly distinguish from any thing else in the scene. Hence this maker should be placed on the moving object itself.

You then specify the overall region of interest. This makes sure that the CV algorithm will only look inside the given region for this color, afterwards you specify the goal line by creating a smaller region of interest. For this test there are the following requirements in setting it up.

* The marker must be a complete other color than the object itself
* The angle of the camera must ensure that the maker can only be seen in the goal line ROI when it has completed one lap
* The lighting must not change drastically over the span of the experiment.

## How to set it up

With roscore running in one terminal and the pylon camera running in another and you have started up the test setup node, type ‘m’ to choose Motion Tracker Test.

A big window will open which shows what the camera sees. Now you must use the trackbars below to isolate the color of the marker. This enables you to use all punch of colors for the marker as long they are easily distinguishable.

With the interval set you will then be prompt to draw the region of where the CV algorithm should look for the marker and afterwards you must specify the goal line by drawing the smaller ROI.

Note: Make sure that ROI of the goal line does not cover where the marker may be at another state on the track as it otherwise falsely will believe it has performed a lap

This is essentially all the major things you must take into consideration when setting up your experiment. All the other stuff is primarily number of laps, location to store data and etc.

# Key Notes

* Vister 1.0 is an early version primarily made to test that all the functionality works as it should while still being useful.
* Though it is not intended but you could use the Activation Test to simply extract data from the UR for experiment that is not fit for the actual Activation Test