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](#_Toc103292497)

[Installing the necessary packages on your PC 4](#_Toc103292498)

[TCP/IP SOCKET Communication with a PC 5](#_Toc103292499)

[Network Settings on the UR 5](#_Toc103292500)

[Network Settings on the PC 5](#_Toc103292501)

[Creating a URP script 7](#_Toc103292502)

[Preparing for the experiment 7](#_Toc103292503)

[Extracting data from the UR robot 8](#_Toc103292504)

# Introduction

Vister utilizes Universal Robots to automate the process of conducting various experiment. Not only does it limit the hassle of manually performing the experiment it also increases the accuracy and precision for conducting the experiment under same conditions.

With that said it can however be time consuming getting to understand the Universal Robots API and different way to send information back and forth from the robot itself and the PC.

With Vister this process is made easy as it utilizes the PolyScope software, already on the Universal Robot, to create the behavior and movement of the Robot, which is called a .URP file. A few small requirements has to be fulfilled concerning the .URP in order to send information to the PC and back again through TCP communication.

This document can be seen as a guide helping you to set everything up correctly. It is therefor highly recommended that this guide is read before hand as common issues and mistakes also will be covered.

Note: It is assumed that you already have a machine with a Vister 1.0 installment on it. Else read: “Vister Installment Guide” for more information

# Installing the necessary packages on your PC

Make sure that you have a PC running Linux. We recommend either Linux Mint or Ubuntu as they have been tested to work fine with VIster 1.0. You need at least a version 3 of Python. More information on installing Python can be found on the following webpage: <https://www.python.org/>

For communication with the UR Robot, Vister utilizes its Real-Time-Data-Exhange (RTDE) in combination with the UR\_RTDE interface developed at SDU. Hence both packages need to be installed on the device running Vister.

**Complete installation of RTDE combined with UR\_RTDE on Linux:**

*Run the following 3 commands in a Terminal Window (press CTRL + ALT + T to open new terminal window)*

sudo add-apt-repository ppa:sdurobotics/ur-rtde

sudo apt-get update

sudo apt install librtde librtde-dev

**Python installation of RTDE combined with UR\_RTDE on Linux:**

*Run the following command in a Terminal Window*

pip3 install --user ur\_rtde

Note: Vister 1.0 is a Python/ROS based application and therefor is the Python installment suitable.

UR\_RTDE’s documentation can be found on the following link:  
<https://sdurobotics.gitlab.io/ur_rtde/installation/installation.html>

# TCP/IP SOCKET Communication with a PC

Vister uses a TCP/IP SOCKET connection to communicate between the PC and the Universal Robot. Hence since section shows how this is setup.

First, connect the UR robot with the PC through an ethernet cable.

## Network Settings on the UR

Go to the Network Settings tab and find the IP-address of the robot. Example of an IP-address could look like this:

Write it down as you will need it later. Other than that, you must make sure that there are no UR caps or custom name settings activated as it may cause issue.

Note: Once it has been setup on the PC end, under the Network tab it will display a checkmark when it has established the connection.

## Network Settings on the PC

In order to establish the connection, you have to manually enter a wired connection under the Network Tab. This procedure looks a bit different depending on the type of Linux that has been installed, nevertheless the end goal will be the same.

**Linux Mint:**

Go to the Network tab, by tapping the network icon below in the corner. Go to the wired connection tab and click on the add a network.

You can name the network what ever you like, if you attend to connect to multiple different UR robots try using a naming convention that fit your use case.

After naming, go to the IPv4 tab. Here you must enter the same IP-address as the one found on the UR robot with a slight modification. Make sure you go a number up as it otherwise will not work.

So if you take the same example as before, and assume that the IP on the UR is the following:

On the PC you must change the last number from 128 to 129, so it looks like this:

Now the changes can be saved and with the robot connected with the Ethernet cable the pc should prompt you with a dialog a wired connection has been establish. IF you go to the Network Tab on the UR you should also see it states the connection is established.

**Ubuntu:**

Go to the Network tab, by tapping the network icon below in the corner. Go to the wired connection tab and click on the add a network.

You can name the network whatever you like, if you attend to connect to multiple different UR robots try using a naming convention that fit your use case.

After naming, go to the IPv4 tab. Here you must enter the same IP-address as the one found on the UR robot with a slight modification. Make sure you go a number up as it otherwise will not work.

So if you take the same example as before, and assume that the IP on the UR is the following:

On the PC you must change the last number from 128 to 129, so it looks like this:

Now the changes can be saved and with the robot connected with the Ethernet cable the pc should prompt you with a dialog a wired connection has been establish. IF you go to the Network Tab on the UR you should also see it states the connection is established.

# Creating a URP script

With the PC and Robot now able to communicate with each other it is now time to create a URP script that contains the motion path of the robot through one loop. It must be design in such a fashion that it has an initial position in which it is in top of the frame and therefor not overlapping when the CV algorithm checks for bounding boxes. Another condition to ensure great performance the last waypoint must be the same as the initial position. After this waypoint, the script must send a message to the PC that the following lap is done, by setting one of the output ports HIGH. This makes sure that the PC knows when to check whenever a malfunction has occurred.

Make sure the robot is in Local Mode in order to design the URP script

These are the 3 requirements for the URP script. So, to sum it up:

* The initial position must be in the top of the frame
* The last waypoint must be the same as the initial position
* Set one of the output ports HIGH at this point

With the script created you can save and store it directly on the UR robot. Next we are ready to prepare for performing the experiment using Vister.

Note: When creating the .URP file the robot has to be in “Local mode” once saved and the robot is in its initial position. Switch it to “Remote mode” to make it able to be controlled by the PC.

# Preparing for the experiment

At the moment it is only the Activation Test, which utilizes communication with the UR robot. When running this testing type, at one point you will be asked to give the location of the .URP file. If saved correctly on the robot, the location is simply the name of the file itself.

Note: Make sure the robot is in its initial position before running the experiment as it otherwise will cause the program to fail due to a safety restriction on the UR side

With the robot in its initial position and set to “Robot Mode”. You are now ready to open Vister.

# Extracting data from the UR robot

At the moment Vister allows you to store the following data from the UR robot:

* Timestamp
* Robot State
* TCP\_Force
* TCP\_Position
* TCP\_Speed

They will be stored at a given frequency with the default value being 500 hz. The UR API allows for even more data to be extracted, however they have be implemented manually by adjusting the source code slightly. In the file “*UR\_record.py*” and in the function “Data\_to\_be\_extracted()” you can add new available data by adding it to the list containing the mention data above.

Make sure to check the following PDF on which data the UR API allows to be saved. The naming must match the spelling of corresponding attribute in the PDF. Otherwise the API will not understand it.

Note: Before making any changes to the code, you should store a copy of the original file