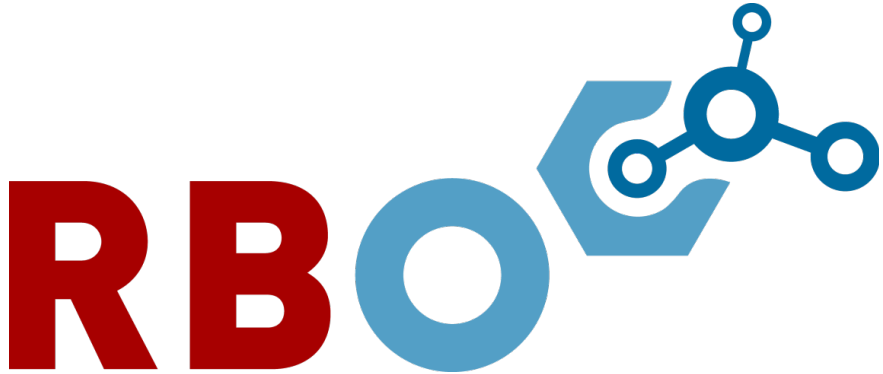
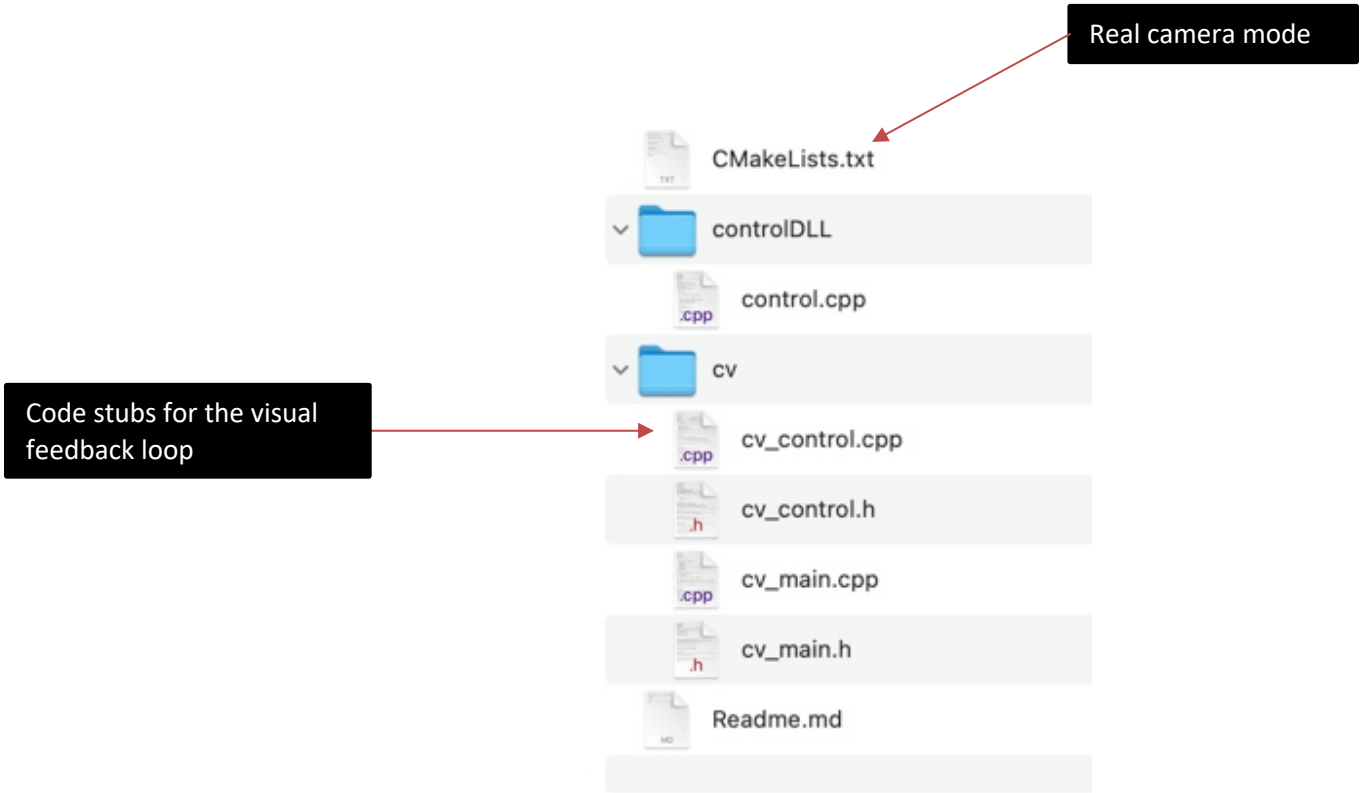


Visual Servoing: Coding

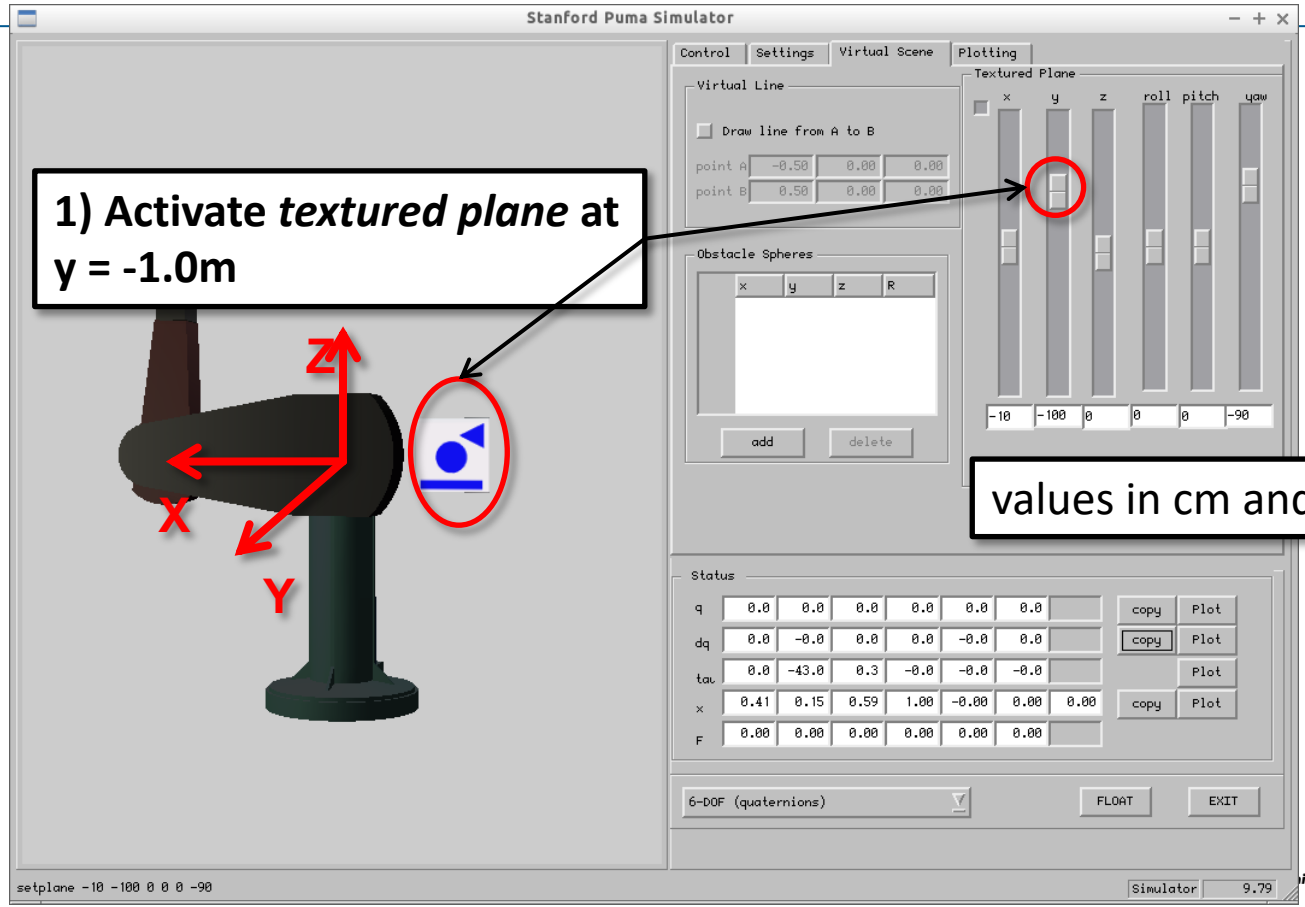
Aditya Bhatt

Technische Universität Berlin

The Extended Puma Simulator

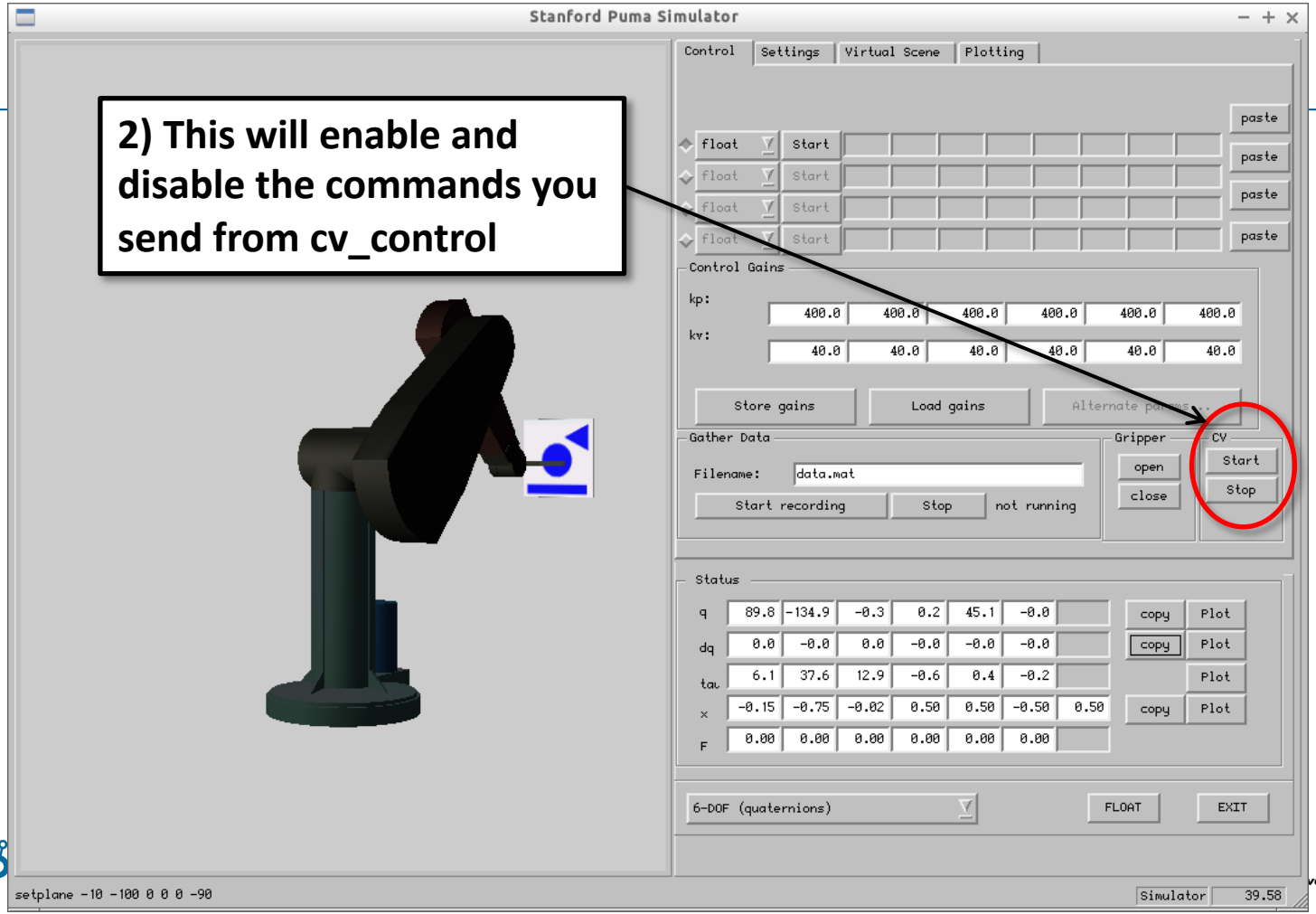


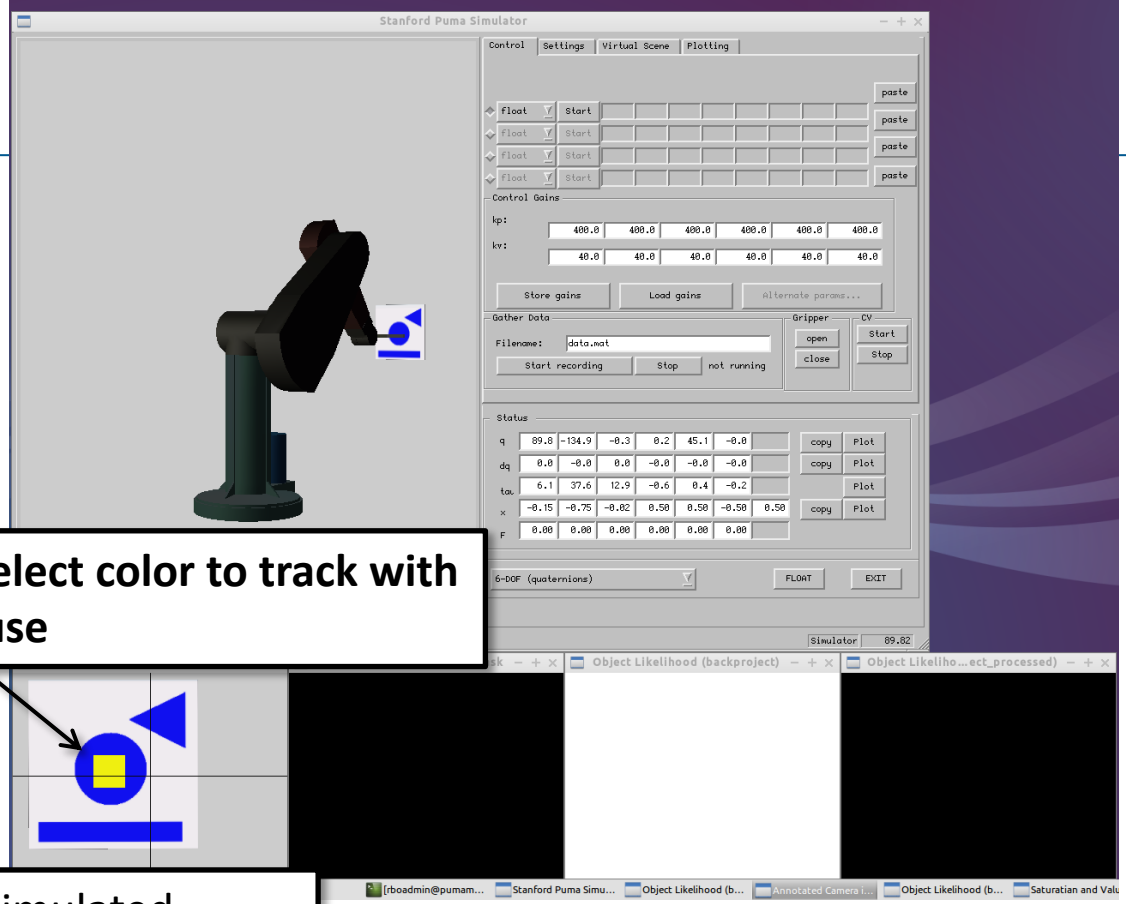
Simulator: Virtual Scene



RBO

Technische Universität Berlin

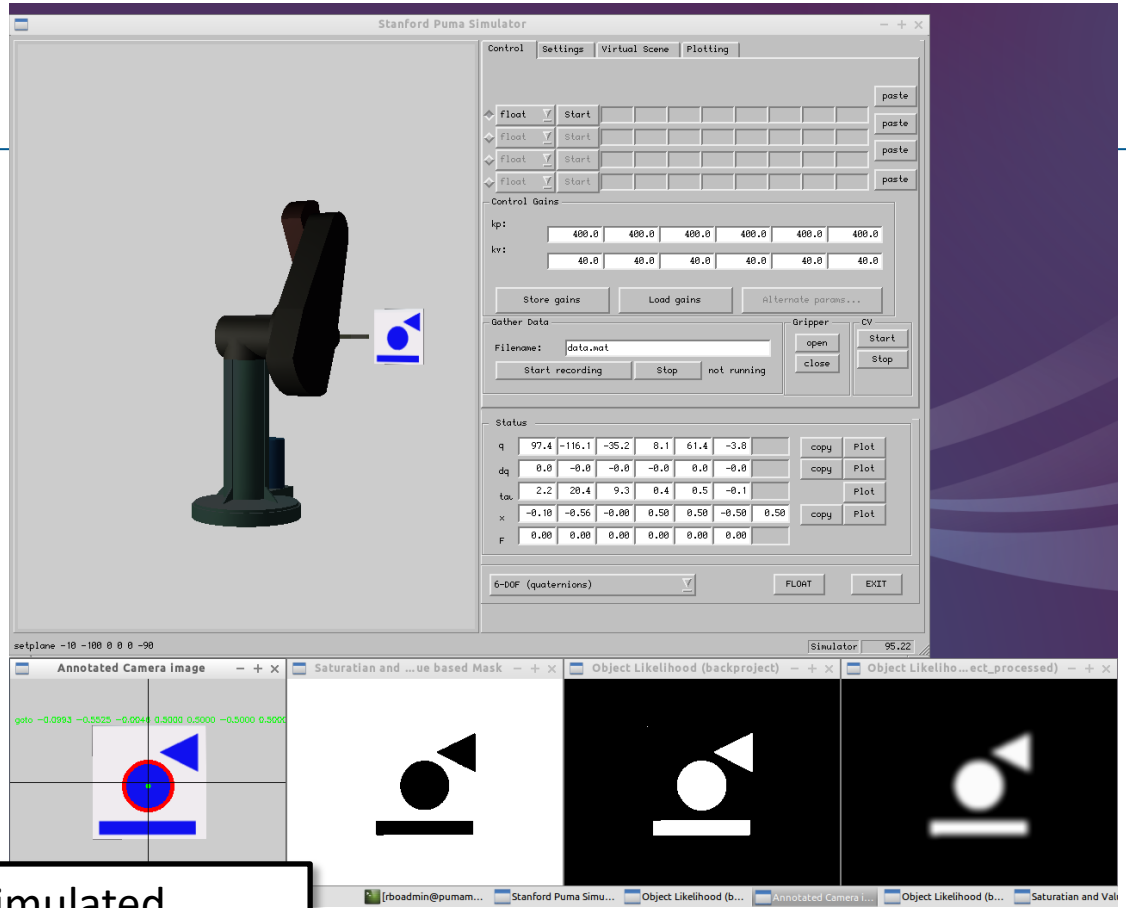

Stanford Puma Simulator
2) This will enable and disable the commands you send from cv_control
Control
Settings
Virtual Scene
Plotting
paste
float
Start
Control Gains
kp:
kv:
400.0
40.0
Store gains
Load gains
Gather Data
Filename:
data.mat
Start recording
Stop
not running
Gripper
open
close
CV
Start
Stop
Status
q
dq
tau
x
F
copy
Plot
6-DOF (quaternions)
FLOAT
EXIT
setplane -10 -100 0 0 0 -90
Simulator
39.58



3) Select color to track with mouse

View of the simulated camera which is mounted on the end-effector

Result of cvCalcBackProject and some image processing



View of the simulated camera which is mounted on the end-effector

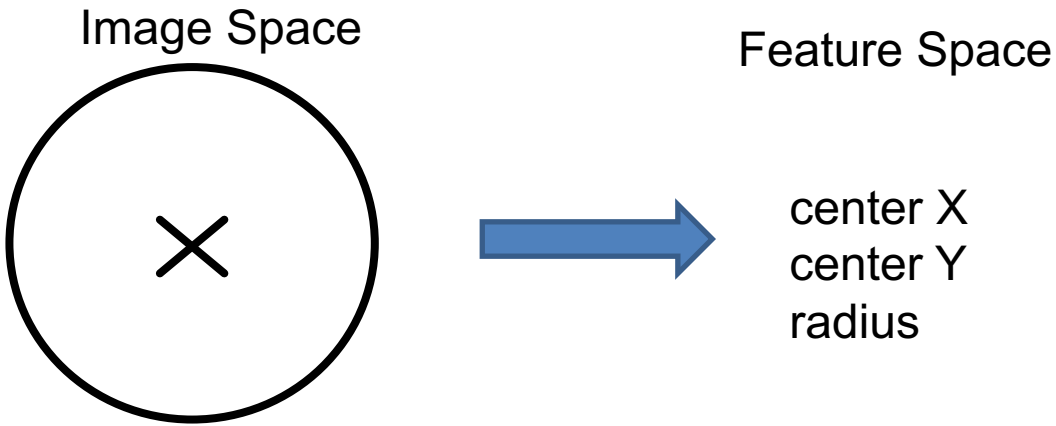
Result of cvCalcBackProject and some image processing

OpenCV

- (**Open** Source **C**omputer **V**ision) is a library of programming functions for real time computer vision

- Documentation: http://docs.opencv.org/

Hough Circle Transform



Virtual Scene

- The diameter of the circle can be found in the variable `diameter_real`
  In the simulation it is 0.107m and in reality 0.15m.
- The position of the plane is relative to the robot base frame

Methods to implement: `cv_control.cpp`

`bool` **`findCircleFeature`**`(cv::Mat& image, Mat& backproject, Circle &crcl)`

This function detects the circle in backproject and returns true if the circle was found. The circle parameters should be stored in `crcl`. It is only called after you have selected a color in the “Visual Servo” window.

`void` **`getImageJacobian`**`(PrMatrix3 &Jv, float u, float v, float z, float f, float diameter)`

In this function you must assign your image Jacobian to `Jv`. The parameters `f` and `diameter` are constant and pre-calculated for you.

`float` **`estimateCircleDepth`**`(float f, float diameter, Circle &crcl)`

Here you have to implement the calculation of the depth of the circle.

`void` **`controlRobot`**`(Circle &crcl, PrVector &x, Mat& img, char *cmdbuf)`

You get this from us. Implements the main CV loop. It uses your `getImageJacobian` and `estimateCircleDepth` and the other functions you implement. The function is only called when `findCircleFeature` returns true. The parameter `crcl` contains your result of `findCircleFeature` , `cmdbuf` is an array of char that should contain the new robot command (this has to switch from command “goto” to “track”).

RBO

versität Berlin

Methods to implement: `cv_control.cpp (2)`

`void transformFromOpenCVFToFF(PrVector3 vector_opencvf, PrVector3& vector_ff)`

Transform a feature vector from openCV frame (origin in upper left corner of the image) to feature frame (origin at the center of the image)

`void transformVelocityFromCFToEEF(PrVector3 vector_cf, PrVector3& vector_eef)`

Transform the desired velocity vector from camera frame to end-effector frame

You can hard code this transformation according to the fixed transformation between the camera and the end effector (see the sketch in your assignment)

`void transformVelocityFromEEFToBF(PrVector x_current_bf, PrVector3 vector_eef, PrVector3& vector_bf)`

Transform the desired velocity vector from end-effector frame to base frame

You cannot hard code this transformation because it depends of the current orientation of the end-effector wrt the base

Make use of the current state of the robot x (the pose of the end-effector in base frame coordinates)

Here you have to implement the calculation of the depth of the circle.

Testing with a real camera

- Consult CMakeLists.txt to see how to build for a real camera (any USB webcam should work)

Everybody has to test her/his feature detection algorithm with a real camera **before** the final presentation!

Processing Pipeline