3 Axis Robot following a Laserpointer by using Stereovision and Templatematching

Johannes Wüstner, Nicolai Schwartze

February 6, 2020

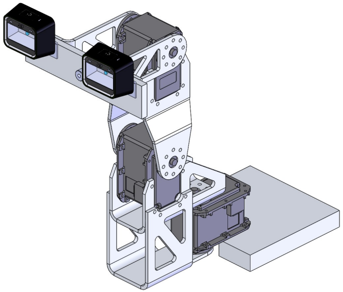
Project Description

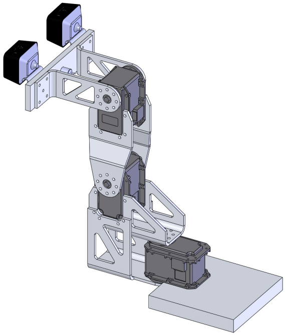
- construct a 3-axis robot with camera/laser tool
- calculate the backwards kinematic
- program robot functions in C-DLL
- calibrate 3D camera system
  - camera projection matrix
  - hand-eye calibration
- detect laser point on white flip-chart
- computer vision using OpenCV and Python
- drive with red robot laser to green user laser

Task

- design a 3-axis robot in SolidWorks
- assemble the parts
- using Dynamixel MX-64AT motors
- attach camera/laser tool

Result





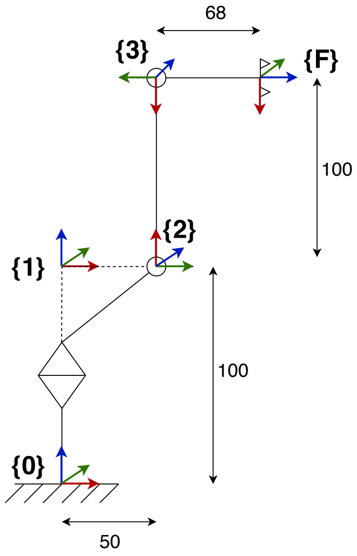
Task

- generate robot-backwards kinematic
- develop equation for setting angle of motor
- establish transformation matrix for hand-eye coordination

Problem

- backwards kinematic
  - establish DH parameter $\rightarrow$ transformation matrix TCP to robot base
- setting motor angle
  - only need two DOF $\rightarrow$ overdetermined
- hand-eye coordination
  - estimate the right distance camera to TCP

Solution I



i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	100	θ_0
2	-90	50	0	$\theta_1 - 90$
3	0	100	0	$\theta_2 + 180$
F	90	0	68	0

Solution II

$$\theta_0 = arctan(\frac{Y_0}{X_0}) \tag{1}$$

$$\theta_1 = 0.0 \tag{2}$$

$$\theta_2 = -1 \cdot arctan\left(\frac{Z_0 - 100 - 100cos(\theta_1)}{\sqrt{X_0^2 + Y_0^2} - 50 - 100sin(\theta_1)}\right) \tag{3}$$

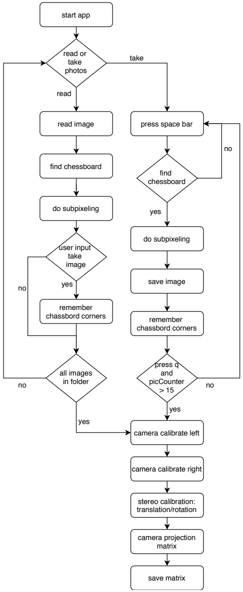
Task

- calculate camera matrix for both single cameras
- estimate translation and rotation between cameras

Problem

- unstable against repetition
- chessboard must not have the same hight and width
- three quality control measures:
    - plausibility: translation vector between cameras
    - measure actual 3D points
    - comparison with matlab

Solution



start app
read or take photos
take
read
read image
find chessboard
do subpixeling
user input take image
yes
no
remember chassbord corners
all images in folder
no
yes
press space bar
find chessboard
no
yes
do subpixeling
save image
remember chassbord corners
press q and picCounter > 15
no
yes
camera calibrate left
camera calibrate right
stereo calibration: translation/rotation
camera projection matrix
save matrix

Task

- detect green laser point
- detect red laser point (not used; would be helpful for controller)

Problem

- first idea: use threshold for colour of laser
- detect contour
- calculate centre of mass
- problem: unstable against other lighting conditions

Solution

- use template matching
- created template calibration app before every run
- more stable against lighting conditions
- but can only work on specified background (like a flip-chart)

using the CV_TM_CCOEFF_NORMED

$$R(x,y) = \frac{\sum_{x',y'}(T'(x',y') \cdot I'(x+x',y+y'))}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}} \tag{4}$$

also works for coloured images, this equation is applied for every channel

Summary, Further Work

- works (sort of) but slow convergence
- overshooting
- resting offset
- not using θ_1
- 3 USB ports needed - opencv does not find camera on hub
- red laser uses loads of batteries $\frac{220mAh}{45mA} = 4.8h$

- implement controller
- correct laser offset to align with axis θ_2
- use undistort before calculateing 3D point
- power supply for laser