

How to get a robot position from video images

Christoph Kendel

May 2, 2026

1 Info

My Robot-Arm does not have homing switches. I use WebCams to get the position of each joint. This way I can get the initial position without movement as well as continuously check the position while working under load.

I work with Aruco markers. With budget low resolution WebCams I can get reliable positions (when using two cams). But the angular recognition of the markers turned out not to be reliable. Thus I have to calculate the angle of each element from the relative positions of the markers. Here I explain how I these calculations are done.

2 Angles

The linear x Position is easy to get: the markers 201, 204, 200, 198, 229, 243 have all a fixed relative x position. Thus the average can be taken, which is saved as x_{242} the x -position of the hand-rotation-joint.

All the other values are the angles y , z , a , b , c and e .

2.1 Biceps y

To find the angle of the biceps (upper arm) there are different options, depending on the angle-of-view and which ArUcos are visible.

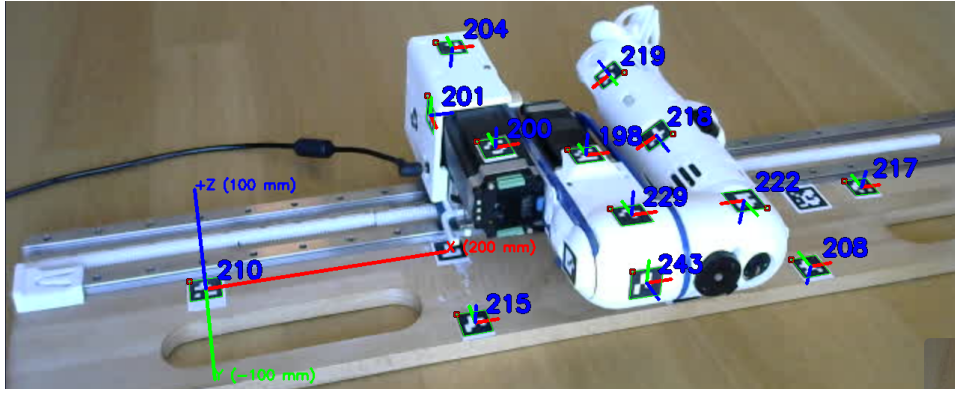


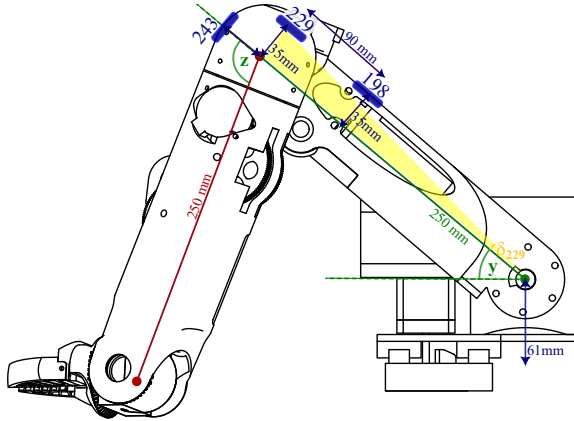
Figure 1: Video Capture with recognized Aruco markers

The calculation is based on two approaches: Position of the Markers 243 etc on the one hand, and relative position $198 \leftrightarrow 229$ on the other hand.

Several ways to calculate y :

- $229 \leftrightarrow 198 \tan(y) = \frac{\Delta z}{\Delta y}$
- From y_{Axis} and the positions of 243, 229, 198 (each position on its own) we can calculate $\tan(y + \delta)$ with a known δ from the geometry.

Thus we get y



Each available (if marker is visible) approach to calculate y is done, and the mean is calculated. We can check if they deviate too much, and give warnings.

2.2 Ellbow – Rotation

At the ellbow there is a motor that turns the forearm arround. Depending on which Arucus are visible, the position \mathbf{a} of the motor can be calculated:

- x of 223 or ... (only the x Position, in the graphics in green). It is independent of all other angles.

the y and z position of 223 etc can't be used, as it depends on the angle of the forearm and the angle of the biceps. Thus is it less reliable. (Although it would be a nice thing to use as a double-check). With the x_{223} compared to the x_{226} i can calculate the ellbow turning \mathbf{a} value:

$$\sin(\mathbf{a} + \alpha) = \frac{\Delta x}{35 \text{ mm}} = \frac{x_{223} - x_{226}}{35 \text{ mm}}$$

as I know the angle α (yellow triangle) from the printed geometry of the arm. The x_{226} is calculated from all visible markers of the sled and biceps, as their markers are fixed in x position. This α is calculated for all markers of the forearm.

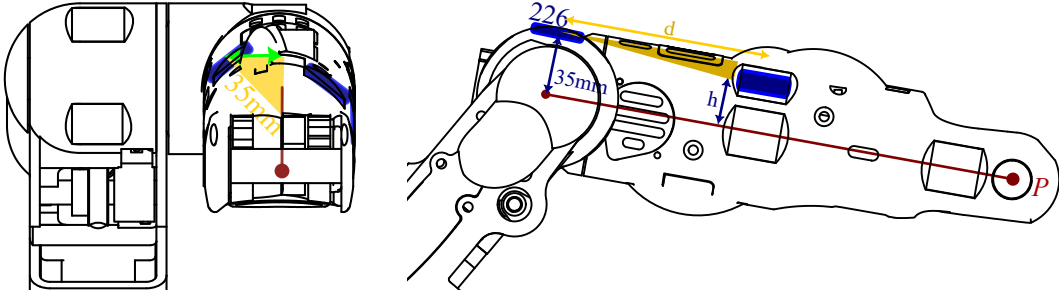


Figure 2: Front view and sideview of the forearm with calculation variables.

As the rotation \mathbf{a} may change for different markers, the median is taken.

2.3 Forearm

The position of the forearm motor (controlling motor in the biceps) can be calculated from the difference between the biceps position y and the angle of the forearm. Thus we need the forearm-angle. That can be calculated in two ways

- relative position of two markers on the forearm, that are on one line.
- relative y, z position of one marker on the forearm and one marker on the ellbow, e.g. 242 or 222.

Again: As I might get different results, I have to find a median value.

⇒ Hand-Joint P is known.

2.4 Hand

To get the hand position, the easiest way I see is to calculate the middle position between both finger-markers¹ (F_1 , F_2 here shown in green). The direction $P \rightarrow M$ in combination with the forearm-direction results in the angle b .

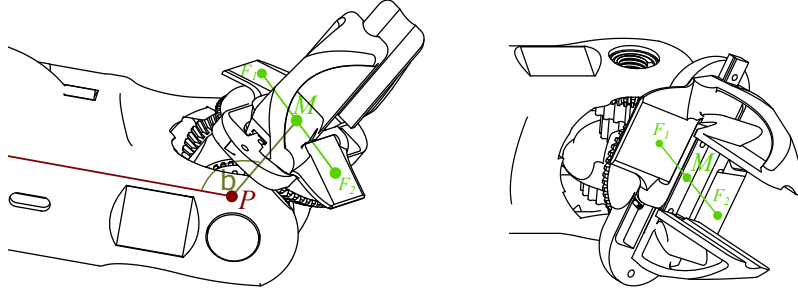


Figure 3: Hand with points to calculate the values for b , c , e .

The Distance between the hand markers is a result of the opening of the fingers e . So I can calculate e from that distance.

Furthermore the direction of the distance between the grippers gives me the rotation of the hand c .

Limitations with the Hand: The results depend heavily on the accuracy of P and M . Both are never measured directly. So have to double check if they make sense. Only if the distance \overline{PM} checks up to be the mechanical correct, that can be used to evaluate the correctness of P and M . And only if the point M lies in the plane of the rotation a around the forearm.

3 Program

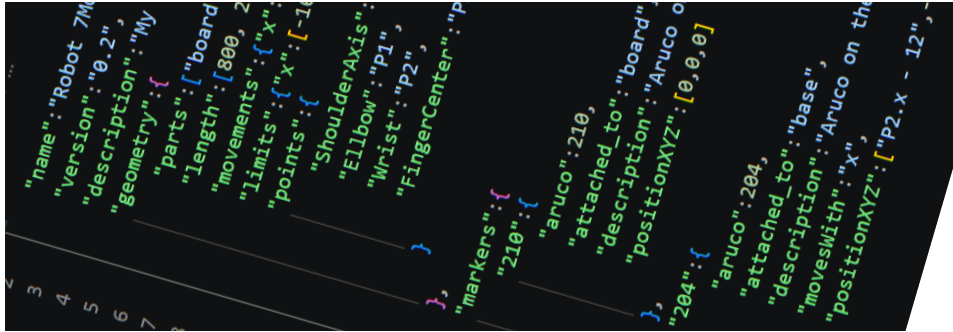
The NodeJS Program calculates the angles and checks, if they are the ones at the moment represented by the system.

First try is to just hard-code all the relevant ArucoPoints. That seems to be ok, but it gets impossible to maintain. After a week I don't recognize my own code. It's hard to make improvements, when you first have to re-think all those small steps, when you have constantly to look up all positions of the markers.

It seems to be a solution to save the geometry and all markers in one file. That can be edited, that can be improved. And here I can save the

¹The other way to calculate the hand position without the markers orientation would be: calculate the middle-hand-plane from the forearm rotation, with the distance of the finger-point to that plane, you'd also get the finger-opening and the projected position of the finger-marker to that plane, you'd get the angle.

relation between x,y, and the position of the Aruco Markers. Of course, at the moment the geometry is still proitty much hard-coded. But i have some flexibility. And mainly: I can loop over *all Arucos on the Board* to find the coordinate system. Or I can loop over all Arucos that attache to the “arm1”.



```
2  "name": "Robot 7M",
3  "version": "0.2",
4  "description": "My",
5  "geometry": {
6    "parts": {
7      "length": [800, 200],
8      "movements": {
9        "limits": {
10         "x": [-100, 100],
11         "y": [-100, 100]
12       },
13       "points": {
14         "ShoulderAxis": {
15           "x": 0, "y": 0, "z": 0
16         },
17         "Elbow": {
18           "x": 100, "y": 0, "z": 0
19         },
20         "Wrist": {
21           "x": 200, "y": 0, "z": 0
22         },
23         "FingerCenter": {
24           "x": 200, "y": 0, "z": 0
25         }
26       }
27     },
28     "markers": {
29       "210": {
30         "aruco": 210,
31         "attached_to": "board",
32         "description": "Aruco on the board",
33         "positionXYZ": [0, 0, 0]
34       },
35       "204": {
36         "aruco": 204,
37         "attached_to": "base",
38         "description": "Aruco on the base",
39         "movesWith": "x",
40         "positionXYZ": ["P2.x - 12", "P2.y - 12", "P2.z - 12"]
41       }
42     }
43   }
44 }
```

Figure 4: Robot Geometry in a .json File