TAČR: Advanced Robotic Architectures for Industrial Inspection (ADRA-2I)

Project Number: TF02000041

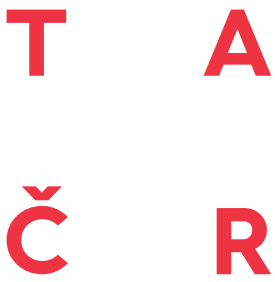
Advanced methods for image processing, data acquisition and NDT inspection (Czech pilot)

D7. (NTIS) Library for SW modules of ADRA conceptual architecture (SW)

(Software)

Tomas Cechura (ZCU), Martin Svejda (ZCU)

20 November 2017



Company ID:
ZCU - University of West Bohemia, SM - SmartMotion s.r.o., UJV - ÚJV Řež, a. s

Contents

1 Introduction

This document serves as a documentation for software library *VisioNDTlib*. This library was developed and implemented with focus on use of advanced computer vision algorithms in robotic applications and NDT (non-destructive testing) inspections. Computer vision is widely used in recent applications, but most of them are in connection with static robot workplaces and a precisely defined workspaces. Mobile robots must first identify the workspace and afterwards can use standard trajectory planners with fixed coordinate system. All developed algorithms are intended for use with single camera.

1.1 Software tools involved

We have made use of the open-source computer vision library **OpenCV** [1] for the development of advanced vision and identification algorithms. The OpenCV provides a complete infrastructure for digital image processing. In particular, we used functions to obtain images from connected vision sensors, a conversion between color spaces and display options used to display the results to the user. It is possible to use OpenCV port for C++ and Python programming language. The *VisioNDTlib* was implemented in Python.

Advanced vision algorithms were integrated into the **REX Control System** [2]. The REX is an industrial control system for automation projects. It is used in all fields of automation, robotics, measurement and feedback control. The final control system of robotic prototype will be most likely implemented using REX software tools. Therefore it is very beneficial to directly interface the VisioNDTlib library directly with REX Control System.

2 Marker definition and generation

As we focused on using single camera approach, we needed to use some specified object located in the visible scene. It is necesarry to know precise dimensions of the object because we use it as 3D space reference in 2D image. For this purpose, we chose the ArUco markers [3]. ArUco defines the approach on using synthetic binary square fiducial markers for computer vision applications.

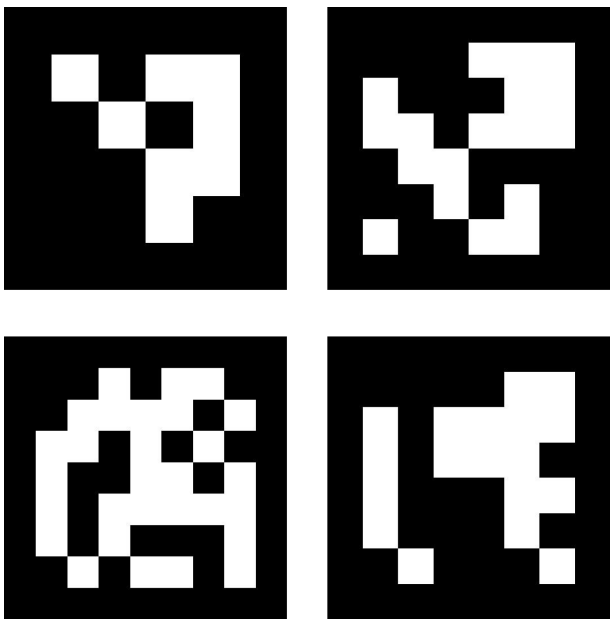


Figure 1: Example of four ArUco markers from different dictionaries

Wide black border of the marker ensures fast and reliable detection in the image. Inner binary matrix encodes marker unique identifier. Marker size determines the range of binary identifier. For marker size of 6 x 6 the identifier is composed by 36 bits.

2.1 Function - Export marker dictionary to Jpeg

First essential function implemented is responsible for definition and export of marker dictionary images. These images can be afterwards printed out and used for identification of specific workplaces with unique identifier. It is possible to generate predefined dictionaries or make custom ones.

Inputs:	
`Marker size`	Number of bits in (rows x columns)
`Dictionary size`	Number of markers in dictionary
Outputs:	
`Jpeg Files`	Unique markers saved in specified folder

Note: It is advantageous to use just as large dictionary as we need distinguishable markers, not more. The dictionary is generated to maximize inter-marker distance and therefore the identification will be more robust. The smaller the dictionary, the higher the robustness.

3 Camera parameters identification

Every camera has its own parameters. The manufacturer doesn't usually specify them so it is necessary to make an identification experiment. This identification is necessary only once for each setup. Computer vision usually uses the so-called pinhole camera model as point in 3D space is transformed into 2D image. Therefore we need to identify intrinsic camera parameters (focal lenghts and optical center coordinates). Camera lenses are usually affected by radial and tangential distortion. The pinhole camera model can be extended with distortion coefficients, which should be also identified.

Chessboards are often used for camera calibration in computer vision. The corners of chessboard can be refined with subpixel precision and due to known chessboard dimension we are able to identify all the important camera parameters by processing images of chessboard from several viewpoints. The main disadvantage of ordinary chessboard approach is the duty of complete chessboard visibility. ChArUco board introduces combination of chessboard and ArUco markers - it benefits from both approaches.

3.1 Function - Export ChArUco board to Jpeg

In order to identify the camera parameters we need to create a ChArUco board. This function implements the definition and the export of ChArUco board.

Inputs:	
`Marker dictionary`	Specify previously generated marker dictionary
`Number of chessboard squares`	Number of rows and columns of chessboard
`Length of squares`	Length of chessboard squares
`Length of markers`	Length of ArUco squares
`Output size`	Size of output image in pixels
Outputs:	
`Jpeg File`	ChArUco board jpeg file saved in specified folder

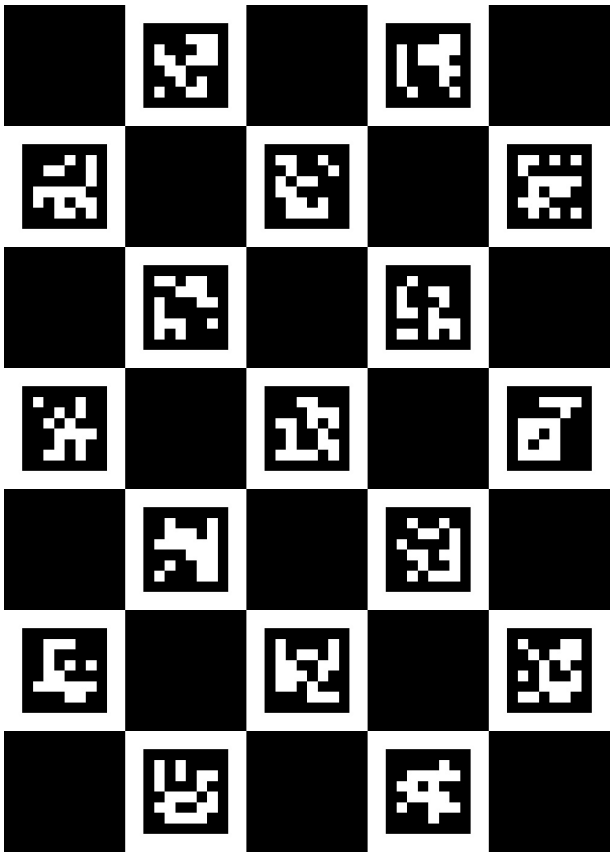


Figure 2: Example of ChArUco calibration board

3.2 Function - Camera calibration

This function implements identification of camera intrinsic parameters and distortion coefficients from multiple camera images.

Inputs:	
`Marker dictionary`	Specify previously generated marker dictionary
`ChArUco board`	Previously generated ChArUco board
`Number of images`	Number of images to be taken by camera during the measurement
Outputs:	
`Camera matrix`	Camera intrinsic parameters
`Camera distortion coefficients`	Distortion coefficients k_1, k_2, p_1, p_2, k_3 (radial coefficients, tangential coefficients)

Executing *Camera calibration* function will take specified number of images. The more various will be the board position and rotation in the image across all the images the more accurate will be the result of parameter identification. We provide, for example, identified camera parameters of *Creative Live! Cam Sync HD 720p* webcam used for testing:

$$\texttt{Camera matrix} = \begin{bmatrix} 613.57 & 0 & 323.63 \\ 0 & 612.72 & 236.86 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\texttt{Distortion coefficients} = \begin{pmatrix} 2.20 \cdot 10^{-1} & -1.59 & -1.16 \cdot 10^{-3} & -1.87 \cdot 10^{-3} & 3.75 \end{pmatrix}$$

4 Identification of camera location on the robot arm

Our goal - workplace coordinate system identification - can be achieved by knowing the exact location of the camera on the robot. It can be measured manually however our experiences shows that this approach often introduces significant errors. Therefore, it is desirable to first identify the location of the camera.

4.1 Function - Camera location identification

For this calibration experiment it is necessary to place the marker in the defined position and rotation with respect to robot base coordinate system and take several images while moving with robot end effector. It simultaneously communicates with robot control system by means of REST API. It waits for trigger to start the identification and afterwards reads robot space coordinates, which are essential for the computation.

This calibration function can be performed once and the results will be valid until the camera's location with respect to robot changes.

Inputs:	
`Camera parameters`	Camera matrix and distortion coefficients
`Marker dictionary`	Specify previously generated marker dictionary
`Marker length`	Dimension of marker square in desired units
`Number of images`	Number of images to be taken by camera during the measurement
`Robot IP address`	Robot control system IP address
Outputs:	
`Camera location`	Transformation matrix between robot end effector frame and camera location frame

The transformations under consideration are depicted in figure 3 in the meaning of the following homogeneous matrices.

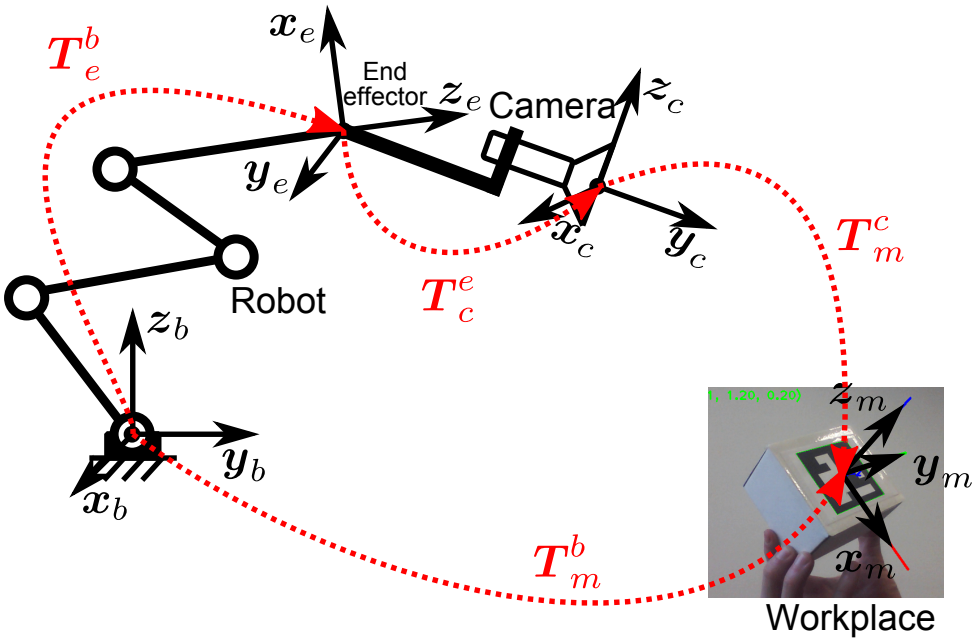


Figure 3: Transformations between the frames

Transformation between the coordinate frames F_k and F_j is given by the homogeneous matrix:

$$\boldsymbol{T}_k^j = \left[\begin{array}{ccc:c} & \boldsymbol{R}_k^j & & \boldsymbol{r}_{\boldsymbol{j,k}}^{\boldsymbol{j}} \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right], \quad \boldsymbol{R}_k^j = \begin{bmatrix} \boldsymbol{x}_k^j & \boldsymbol{y}_k^j & \boldsymbol{z}_k^j \end{bmatrix} \tag{1}$$

where $\boldsymbol{R}_k^j$ is the rotation matrix represents the coordinate axes of F_k with respect to F_j and $\boldsymbol{r}_{k,j}^k$ is the translation vector from the origin of F_k to the origin of F_j with respect to F_k.

- $\boldsymbol{T}_e^b$ The known transformation (translation and orientation) between the robot base frame F_b and the end effector frame F_e which is obtained from the robot control system
- $\boldsymbol{T}_c^e$ The transformation between the robot end effector frame F_e and the camera location frame F_c
- $\boldsymbol{T}_m^c$ The transformation between the camera location frame F_c and the workplace (maker) frame F_m which is obtained as output of camera data processing based on proposed ArUco markers approach
- $\boldsymbol{T}_m^b$ The transformation between the robot base frame F_b and the workplace (marker) frame F_m.

The transformation representing the camera location ${}^i\boldsymbol{T}_c^e$ is given for i-th recorded image as:

$${}^i\boldsymbol{T}_c^e = \left({}^i\boldsymbol{T}_e^b\right)^{-1} \cdot \boldsymbol{T}_m^b \cdot \left({}^i\boldsymbol{T}_m^c\right)^{-1} \tag{2}$$

where ${}^i\boldsymbol{T}_e^b$ is obtained from the robot control system (reading robot space coordinates), ${}^i\boldsymbol{T}_m^c$ is obtained from data processing based on proposed ArUco markers and $\boldsymbol{T}_m^b$ is position of the marker in the defined (constant calibrated) position.

The resulting camera positions ${}^i\boldsymbol{T}_c^e$ are not exactly the same because of introduced errors (robot kinematics, camera calibration, etc.) and the averaging of computed positions is necessary. The averaging of the translation measurement data ${}^i\boldsymbol{r}_{c,e}^c$ can be simple done as follows:

$${}^{\text{avg}}\boldsymbol{r}_{c,e}^c = \sum_{i=1}^{N} \frac{1}{N} \cdot {}^i\boldsymbol{r}_{c,e}^c \tag{3}$$

Unfortunately, averaging of the rotation matrices is not straightforward. It is clear that rotation matrices ${}^i\boldsymbol{R}_c^e$ can be recomputed to corresponding quaternions ${}^i\boldsymbol{Q}_c^e$ and there exists algorithm [4] which returns the resulting (average) quaternion ${}^{\text{avg}}\boldsymbol{Q}_c^e$ which fulfil the following condition:

$${}^{\text{avg}}\boldsymbol{Q}_c^e = \underset{\boldsymbol{Q}_c^e \in \mathbb{Q}}{\operatorname{argmin}} \sum_{i=1}^{N} \|\boldsymbol{R}_c^e(\boldsymbol{Q}_c^e) - {}^i\boldsymbol{R}_c^e\|_F^2, \quad \Rightarrow \quad {}^{\text{avg}}\boldsymbol{R}_c^e \tag{4}$$

where $\| \star \|_F$ is a matrix Frobenius norm (sum of the absolute squares of matrix elements).

The resulting transformation matrix ${}^{\text{avg}}\boldsymbol{T}_c^e$ of the camera location is given:

$${}^{\text{avg}}\boldsymbol{T}_c^e = \left[\begin{array}{ccc:c} & {}^{\text{avg}}\boldsymbol{R}_c^e & & {}^{\text{avg}}\boldsymbol{r}_{c,e}^c \\ \hdashline 0 & 0 & 0 & 1 \end{array}\right] \tag{5}$$

5 Identification of workplace coordinate system

This section is similar to Section 4. The difference is mainly in known and calculated parameters. For workplace coordinate system identification the goal is to estimate marker coordinate system with respect to robot base coordinate system. All other transformations are known - robot end effector with respect to robot base, camera location with respect to robot end effector and finally the marker pose with respect to camera.

This function is also synchronized with robot control system and is reading robot end effector position and orientation with every image taken.

5.1 Function - Camera pose estimation

Inputs:	
`Camera parameters`	Camera matrix and distortion coefficients
`Marker dictionary`	Specify previously generated marker dictionary
`Marker length`	Dimension of marker square in desired units
`Number of images`	Number of images to be taken by camera during the measurement
`Robot IP address`	Robot control system IP address
Outputs:	
`Workplace coordinate system`	Transformation matrix between robot base and marker (workplace)

The transformation between the robot base frame and the workplace frame is given for i-th recorded image directly as:

$$ {}^{i}\boldsymbol{T}_{m}^{b} = {}^{i}\boldsymbol{T}_{e}^{b} \cdot \boldsymbol{T}_{c}^{e} \cdot {}^{i}\boldsymbol{T}_{m}^{c} \tag{6} $$

where ${}^{i}\boldsymbol{T}_{e}^{b}$ is obtained from the robot control system (reading robot space coordinates), ${}^{i}\boldsymbol{T}_{m}^{c}$ is obtained from data processing based on proposed ArUco markers and $\boldsymbol{T}_{c}^{e}$ is known (calibrated) location of the camera frame with respect to end effector frame (from Camera location calibration, see Section 4).

The resulting transformation matrix ${}^{\text{avg}}\boldsymbol{T}_{m}^{b}$ between the robot base frame and marker (workplace) frame is obtained after averaging process (in analogy to Section 4) as:

$$ {}^{i}\boldsymbol{T}_{m}^{b} \quad \Rightarrow \text{ (averaging) } \Rightarrow \quad {}^{\text{avg}}\boldsymbol{T}_{m}^{b} \tag{7} $$

Following figures show how the markers are recognized in the image. Values shown in the figures are translation and rotation of marker with respect to camera coordinate system.

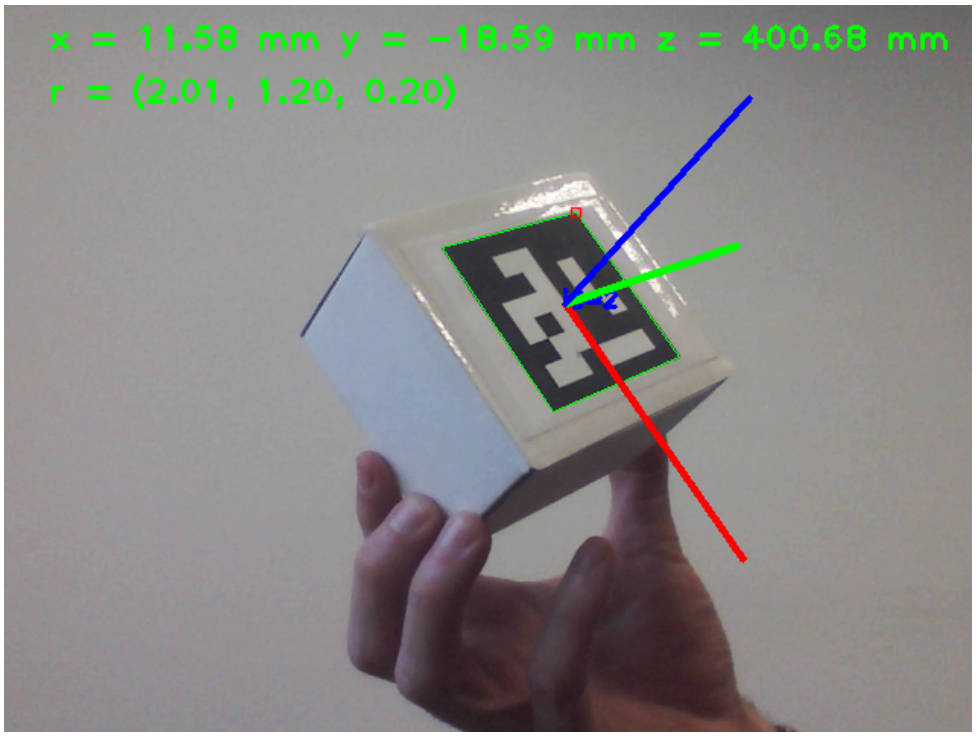


Figure 4: Camera pose estimation

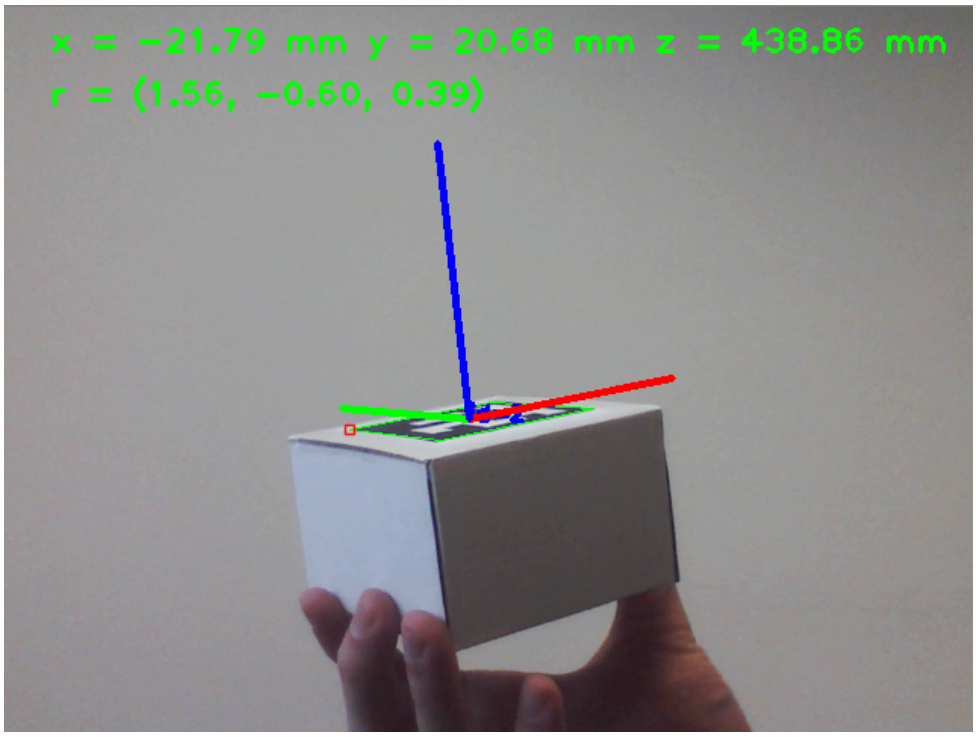


Figure 5: Camera pose estimation - limits

6 Conclusion

Implemented library gives us the possibility to identify workplace coordinate system by using only single calibrated camera and specific marker. This approach is beneficial mainly in com-

bination with mobile robots as they don't have any fixed workplace and therefore they require the precise knowledge of workplace coordinate system.

The library contains all necessary features for the entire robot calibration process at a new workplace. Complete infrastructure includes marker definition and library generation, camera calibration, identification of camera location on robot arm and camera pose estimation.

All the algorithms are platform independent and can be easily ported to desired platform running Python environment.

Proposed software is available on Department of cybernetics at the University of West Bohemia (e-mail: `tomek89@ntis.zcu.cz`).

Acknowledgement

This research was supported by project No. TF02000041 of the Technology Agency of the Czech Republic.

References

[1] "OpenCV library," https://opencv.org/.

[2] REX Controls s.r.o., "REX Control System," https://www.rexcontrols.com/rex.

[3] Aplicaciones de la Visin Artificial, Universidad de Crdoba, "ArUco: a minimal library for Augmented Reality applications based on OpenCV," https://www.uco.es/investiga/grupos/ava/node/26.

[4] L. Markley, Y. Cheng, J. Crassidis, Y. Oshman,"Averaging Quaternions", Journal of Guidance, Control and Dynamics 30(4): 1193-1196, DOI10.2514/1.28949, July 2007.