Master in Informatics and Multimedia Engineering

Computer Vision and Mixed Reality

1st Project – Part B

Marker Based Augmented Reality

1. Goal

a. Develop a computer vision application which implements marker based augmented reality that allows the inclusion of virtual elements aligned with real fiducial markers.

b. Familiarization with the OpenCV (**Open**-Source **C**omputer **V**ision) library for real-time application programming and ArUco marker based library.

2. Development

a. Camera Calibration

Camera calibration is the process to compute the intrinsic parameters of the camera – the Perspective Transformation – and the distortion coefficients (optional) for correcting the radial and tangential distortions caused by the camera lens.

OpenCV Camera Calibration Tutorial [1] shows the method to implement this task with build-in OpenCV functions. However, there are alternative methods using other libraries as described, for example, in [2, 3].

Create an application with the method proposed in [1] (or chose another one) to calibrate offline your computer camera and save the intrinsic parameters for future use.

b. Detection and camera pose estimation with ArUco library

ArUco library for OpenCV [3, 4] is a marker based "minimal library for Augmented Reality applications based on OpenCV".

Create an application to detect the ArUco markers and estimate the camera pose for each marker – the View Transformation – to make possible the registration of virtual objects aligned with the markers. The ArUco markers can be created with the proper method but in [5] there is an ArUco marker generator. A python examples with ArUco library can be found in [6].

c. Registration of virtual objects

Add virtual objects registered with the world coordinate system associated with the detected markers (different object for each marker ID).

Bibliography

[1] - https://docs.opencv.org/master/dc/dbb/tutorial_py_calibration.html

[2] - https://medium.com/vacatronics/3-ways-to-calibrate-your-camera-using-opencv-and-python-395528a51615

[3] - https://www.uco.es/investiga/grupos/ava/node/26

[4] - https://docs.opencv.org/master/d9/d6d/tutorial_table_of_content_aruco.html

[5] - https://chev.me/arucogen/

[6] - https://mecaruco2.readthedocs.io/en/latest/notebooks_rst/Aruco/aruco_basics.html

Main *OpenCV* function for camera calibration and pose estimation:

Goal	OpenCV function(s)
Finds the positions of internal corners of the chessboard.	*findChessboardCorners()*
Renders the detected chessboard corners.	*drawChessboardCorners()*
Finds subpixel-accurate positions of the chessboard corners.	*cornerSubPix()*
Finds the camera intrinsic and extrinsic parameters from several views of a calibration pattern.	*calibrateCamera()*
Returns the new camera intrinsic matrix based on the free scaling parameter.	*getOptimalNewCameraMatrix()*
Transforms an image to compensate for lens distortion.	*undistort()*
Finds an object pose from 3D-2D point correspondences using the RANSAC scheme.	*solvePnPRansac()*
Project 3D points to an image plane.	*projectPoints()*
Converts a rotation matrix to a rotation vector or vice versa.	*Rodrigues()*

Main *OpenCV.ArUco* functions for marker generation, detection and pose estimation:

Goal	OpenCV function(s)
Returns one of the predefined ArUCo dictionaries.	*Dictionary_get()*
Draw a canonical marker image.	*drawMarker()*
Load the ArUCo parameters for the detectMarker process.	*DetectorParameters_create()*
Basic marker detection.	*detectMarkers()*
Draw detected markers in image.	*drawDetectedMarkers()*
Pose estimation for single markers.	*estimatePoseSingleMarkers()*
Draw coordinate system axis from pose estimation.	*drawAxis()*