CALIBRACIÓN

Calibración de cámaras (ej8.cpp)

Almacenamiento de datos de calibración YAML

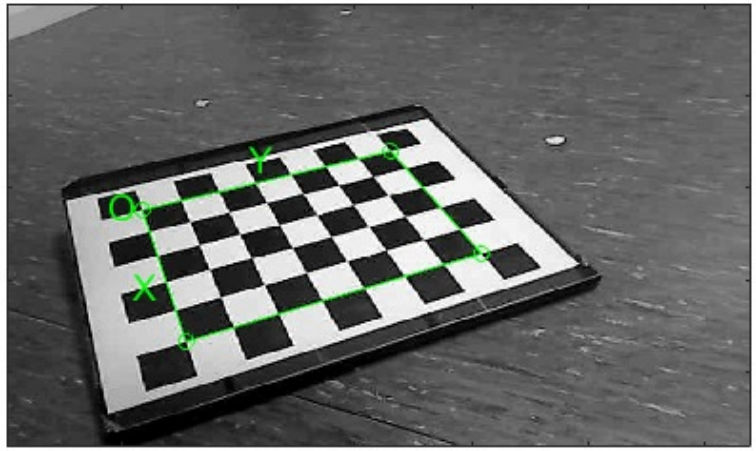
https://docs.opencv.org/4.5.5/d9/d0c/group__calib3d.html

https://docs.opencv.org/4.5.5/d4/d94/tutorial_camera_calibration.html

Calibración de Cámaras

- Ejemplo (**ej8.cpp**): partiremos del código **ej3.cpp** para la captura de imágenes de calibración

- Configurar datos patrón:




```
…….
// Calibration pattern inner corners(cols Y - axis, rows X - axis)
cv::Size patternSize(8, 6);
float squareSize = 27.0;      // square pattern side in mm

// 3D object points coordinates(x, y, z)
vector<cv::Point3f> objp3D;
for (int x = 0; x < patternSize.height; x++)
        for (int y = 0; y < patternSize.width; y++)
                objp3D.push_back(cv::Point3f(x*squareSize, y*squareSize, 0));


vector<vector<cv::Point3f>> objpoints;      // 3D point in real world space
vector<vector<cv::Point2f>> imgpoints;      // 2D points in image plane
int num_patterns = 0;                       // number of detected patterns. We need at lest 3
cv::Size imageSize;
    ……
```

Calibración de Cámaras

- Ejemplo (**ej8.cpp**):
- Cargar ficheros de imagen de un directorio:
  - void cv::utils::fs::**glob_relative** (cv::String &**directory**, cv::String &**pattern**, vector<cv::String> &**result**, bool **recursive** = false, bool **includeDirectories** = false)

```
#include <opencv2/core/utils/filesystem.hpp>
…….
// reading directory files
string  prefix = "Image";
vector<string> valid_extension = { "jpg", "jpeg", "png", "tiff", "tif", "bmp", "pgm" };
vector<string> directoryList;
vector<string> files;

cv::utils::fs::glob_relative(".", "*", files);

// filter image files
for(int idx=0; idx <files.size(); idx++)
{
        // Test for prefix
        if (files[idx].find(prefix) != 0) continue;

        // Test for valid extension
        bool valid = false;
        for (int i = 0; i < valid_extension.size(); i++)
                if ( valid_extension[i].size() < files[idx].size() &&
                        std::equal(valid_extension[i].rbegin(), valid_extension[i].rend(), files[idx].rbegin()) )
                { valid = true; break;  }

        if(valid) directoryList.push_back(files[idx]);
}
```

Calibración de Cámaras

- Ejemplo (**ej8.cpp**):

- Detectar esquinas patrón calibración:
  - cv::**findChessboardCorners**(OutputArray **image**, cv::Size &**patternSize**, OutputArray **corners** [, flags]) → bool

    retval: (bool) patternWasFound
    corners: vector<cv::Point2f> ó cv::Mat con una fila por esquina (x, y)
    patternSize: cv::Size(**cols** Y-axis, **rows** X-axis)
    flags: def: cv::CALIB_CB_ADAPTIVE_THRESH + cv::CALIB_CB_NORMALIZE_IMAGE
    cv::CALIB_CB_FAST_CHECK , cv::CALIB_CB_FILTER_QUADS

  - cv::**cornerSubPix**(**image**, InputOutputArray **corners**, cv::Size **winSize**, cv::Size **zeroZone**, **criteria**)

    criteria: cv::TermCriteria(cv::TermCriteria::EPS + cv::TermCriteria::MAX_ITER, 30, 0.011)
    winSize: cv::Size(11,11) Half of the side length of the search window
    zeroZone: cv::Size(-1,-1)

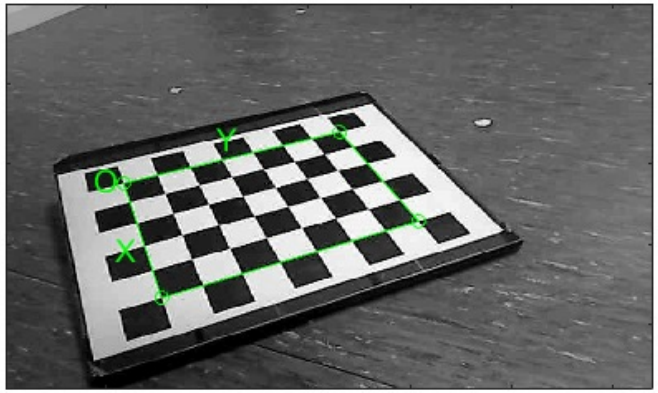
  - cv::**drawChessboardCorners**(**image**, cv::Size **patternSize**, InputArray **corners**, bool **patternWasFound**)

    patternSize: cv::Size(**cols** Y-axis, **rows** X-axis,)
    patternWasFound: (bool). Devuelto por cv::findChessboardCorners

Calibración de Cámaras

- Ejemplo (**ej8.cpp**): Detectar esquinas patrón calibración:

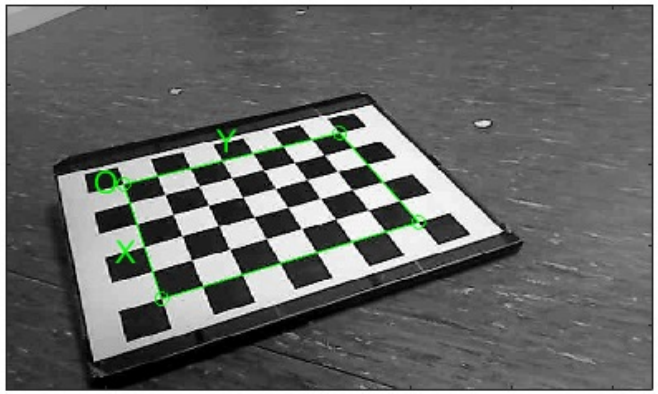
```
// Process image files
for (int i = 0; i < directoryList.size(); i++)
{
        cv::Mat image, gray_image;
        vector <cv::Point2f> corners;
        cout << "Processing image file: " << directoryList[i] << endl;

        image = cv::imread(directoryList[i]);

        cv::cvtColor(image, gray_image, cv::COLOR_BGR2GRAY);  // transforms to gray level
        imageSize = gray_image.size();

        bool patternWasFound = cv::findChessboardCorners(gray_image, patternSize, corners);

    .............
```



Calibración de Cámaras

- Ejemplo (**ej8.cpp**): Detectar esquinas patrón calibración:

```
    ……
    if (patternWasFound)
    {
        num_patterns++;
        cv::cornerSubPix(gray_image, corners, cv::Size(11, 11), cv::Size(-1, -1),
                        cv::TermCriteria(cv::TermCriteria::EPS + cv::TermCriteria::MAX_ITER, 30, 0.001));

        // add points to image / 3Dobjet lists
        objpoints.push_back(objp3D);
        imgpoints.push_back(corners);

        //  Draw and display the corners
        cv::drawChessboardCorners(image, patternSize, corners, patternWasFound);

        cv::imshow(WINDOW_CAMERA1, image);     // Display the resulting frame
        int key = cv::waitKey(1000);          // update image and wait 1 second
    }
}
……————————
```



Calibración de Cámaras

- Ejemplo (**ej8.cpp**): Calibración
  - cv.**calibrateCamera**(**objectPoints**, **imagePoints**, **imageSize**, **cameraMatrix**, **distCoeffs**, **rvecs** , **tvecs** [, **flags**[, **criteria**]]) → retval
    - **retval**: (double) overall RMS reprojection error
    - **objectPoints**: vector<vector<cv::Point3f>> (x,y,z)
    - **imagePoints**: vector<vector<cv::Point2f>> (x,y)
    - **imageSize**: tuple (**cols, rows**)
    - **cameraMatrix, distCoeffs**: (cv::Mat)
    - **rvecs**, **rvecs**: vector<cv::Mat>
    - **flags**: cv::CALIB_USE_INTRINSIC_GUESS, cv::CALIB_ZERO_TANGENT_DIST → (p1,p2) =0
      cv::CALIB_FIX_XXX (parámetro fijo)
    - **criteria**: cv::TermCriteria(cv::TermCriteria::EPS + cv::TermCriteria::MAX_ITER, 30, 0.001))
  - cv::**calibrateCamera**(**objectPoints**, **imagePoints**, **imageSize**, **cameraMatrix**, **distCoeffs**, **rvecs**, **tvecs**, stdDeviationsIntrinsics, stdDeviationsExtrinsics, perViewErrors [, **flags**[, **criteria**]]) → retval

$$camera\ matrix = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

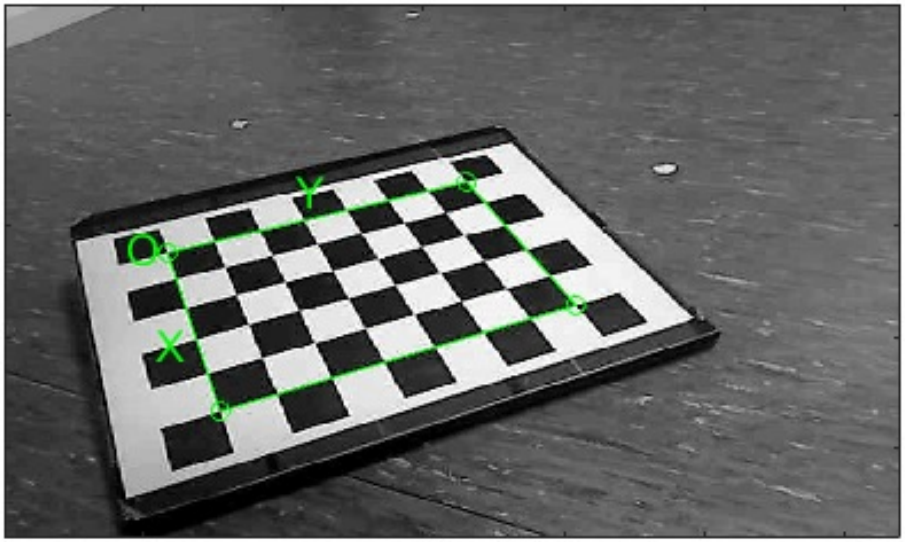
$$Distortion\ coefficients = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

$$(k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6[, s_1, s_2, s_3, s_4[, \tau_x, \tau_y]]]])$$

Calibración de Cámaras

- Ejemplo (**ej8.cpp**): Calibración

```
if (num_patterns >= 3)
{
        cv::Mat cameraMatrix, distCoeffs;
        vector<cv::Mat> rvecs, tvecs;

        double rms = cv::calibrateCamera(objpoints, imgpoints, imageSize, cameraMatrix, distCoeffs,
                                         rvecs, tvecs);

        cout << "RMS reprojection error: " << rms << endl;
    ............
```



Calibración de Cámaras: persistencia JSON

- Lectura/escritura de datos de calibración en ficheros YAML:

- Escribir fichero YAML:

```
….
// store calibration data in a JSON file
cv::FileStorage fs;
fs.open("camera_calib.yaml", cv::FileStorage::WRITE);
if (fs.isOpened())
{
        fs << "nx" << imageSize.width;
        fs << "ny" << imageSize.height;
        fs << "camera_matrix" << cameraMatrix;
        fs << "distortion_coefficients" << distCoeffs;
        fs << "avg_reprojection_error" << rms;

        fs.release();
}
```

Calibración de Cámaras: persistencia JSON

- Lectura/escritura de datos de calibración en ficheros YAML:

- Leer fichero YAML:

```
…..
        // Reading calibration data from file
        fs.open("camera_calib.yaml", cv::FileStorage::READ);
        if (fs.isOpened())
        {
                cv::Mat a, dist;
                fs["camera_matrix"] >> a;
                fs["distortion_coefficients"] >> dist;
                fs.release();      // close file

                cout << "Camera matrix:\n" << a << endl;
                cout << "distortion_coefficients:\n" << dist << endl;
        }
}
```

DETECCIÓN DE MARCAS

Librería ARUCO (Grupo AVA Univ. Córdoba) (ej9.cpp)

Instalación independiente: *compilar librería* ***aruco***
También disponible en **opencv-contrib** (implementación limitada)

Código fuente: http://sourceforge.net/projects/aruco/

https://www.uco.es/investiga/grupos/ava/node/26

Detección de Marcas: ARUCO

- Instalación de la librería **ARUCO** (Grupo AVA Univ. Córdoba)
  - Tutoriales: http://umh1782.edu.umh.es/opencv/

  - Windows ***OpenCV4.5.5-VC15 (VS2017)***: Descargar la librería ya compilada
    - http://umh1782.edu.umh.es/opencv/#Software
    - ***aruco3112-opencv455-vc15.zip***

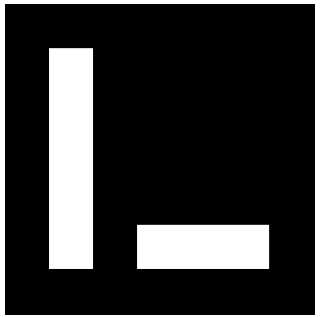
    - Descomprimir en la carpeta **c:\opencv**
      - Utilidades: Carpeta librerías: ***c:\opencv\bin***
      - Carpeta librerías: ***c:\opencv\lib***

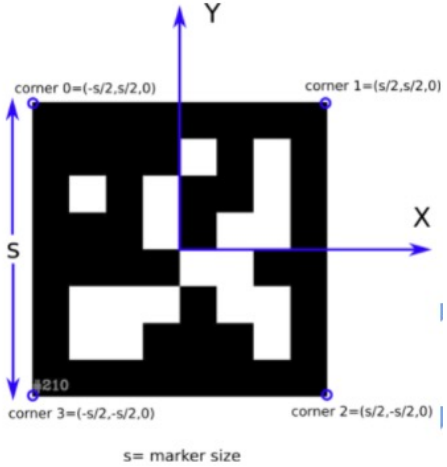
    - Añadir la librería a las hojas de propiedades:
      - ***Debug: aruco3112d.lib***
      - ***Release: aruco3112.lib***
    - Disponibles las hojas de propiedades ya configuradas en:
      - http://umh1782.edu.umh.es/opencv

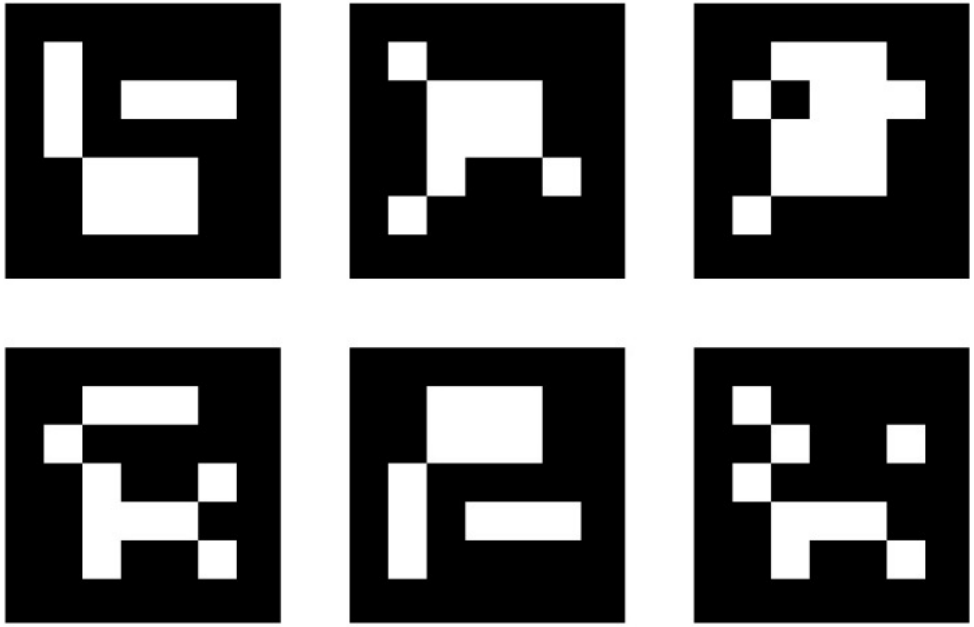
  - Otras plataformas: descargar código fuente y compilar con ***CMake***
    - http://sourceforge.net/projects/aruco/

Detección de Marcas: ARUCO

- Ejemplo (**ej9.cpp**)
  - Detectar y localizar marcadores codificados en la imagen
  - Cámara no calibrada
  - http://umh1782.edu.umh.es/opencv/
  - Fichero de tabla con marcadores (MarkerMap): ***markersBoard.pdf, marker_id1.pdf***







Detección de Marcas: ARUCO

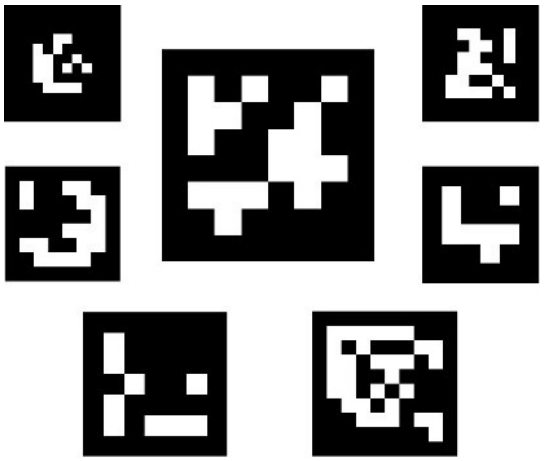
- Diccionarios de marcas disponibles:
  - Clase aruco::Dictionary
  - Funciones (static):
    - aruco::Dictionary::**loadPredefined** (DICT_TYPES type) → dict (aruco::Dictionary)

```
aruco::Dictionary dict = aruco::Dictionary::loadPredefined(aruco::Dictionary::ARUCO);
```

aruco::Dictionary::**ARUCO** standard ArUco Markers. 1024 markers, 5x5 bits, 0 minimum distance
aruco::Dictionary::**ARUCO_MIP_36h12** (más robusto) 250 markers 5x5 bits, hamming distance 12
aruco::Dictionary::**ARUCO_MIP_25h7** 100 markers, 4x4 bits, hamming distance 7
aruco::Dictionary::**ARUCO_MIP_16h3** 250 markers, 3x3 bits, hamming distance 3

// APRILTAGS
aruco::Dictionary::**TAG16h5** 4x4 bits, hamming distance 5, 30 codes
aruco::Dictionary::**TAG25h9** 5x5 bits, hamming distance 9, 35 codes
aruco::Dictionary::**TAG36h10** 6x6 bits hamming distance 10, 2320 codes
aruco::Dictionary::**TAG36h11** 6x6 bits, hamming distance 11, 587 codes

aruco::Dictionary::**CHILITAGS**
aruco::Dictionary::**ARTAG**
aruco::Dictionary::**ARTOOLKITPLUS**
aruco::Dictionary::**ARTOOLKITPLUSBCH**



Detección de Marcas: ARUCO

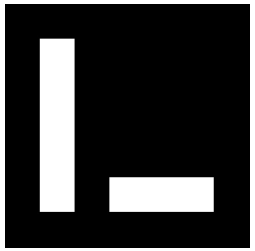
- Clase aruco::Dictionary / aruco::MarkerMap
  - aruco::Dictionary::**createMarkerMap**(cv::Size **gridSize**, int **MarkerSize**, int **MarkerDistance**, std::vector<int>& **Ids**, bool **chess_board** = false) → board (**aruco::MarkerMap**)

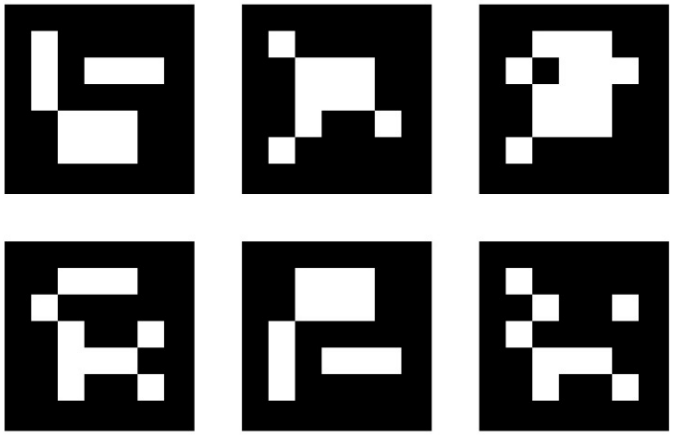
    MarkerSize: tamaño marcador (pixels)

    MarkerDistance : separación entre marcas (pixels)

- Crear imagen marcas:
  - aruco::Dictionary::**getMarkerImage_id**(int **id**, int **bit_size**, bool **addWaterMark** = true, bool **enclosed_corners** = false, bool **printExternalWhiteBorder**=false, bool **centralCircle**=false) → cv::Mat
  - aruco::MarkerMap::**getImage**(float METER2PIX=0) → cv::Mat

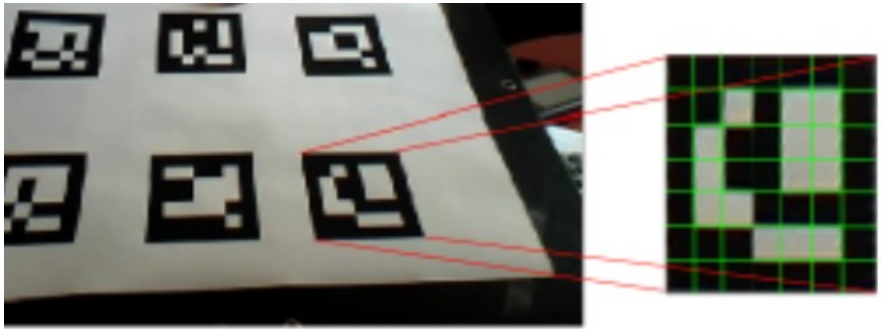
```
cv::Mat marker = dict.getMarkerImage_id( 1, 10, true, false, true, false);
vector<int> ids = {79, 248, 892, 814, 964, 142};
cv::Mat board = dict. createMarkerMap(cv::Size(3,2), 50,15,ids).getImage();
```



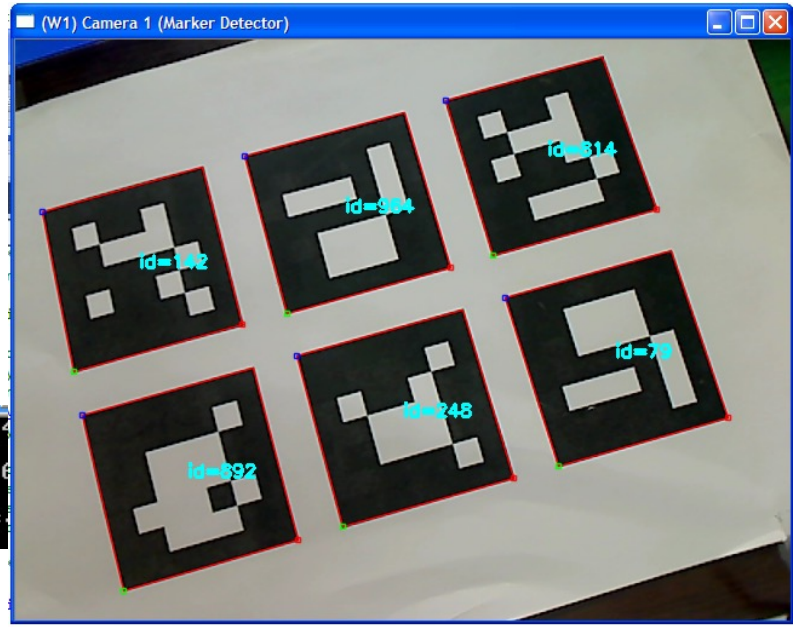


Detección de Marcas: ARUCO

- Ejemplo (**ej9.cpp**) Detección de marcas
- Clases:
  - aruco::**MarkerDetector**
    - aruco::MarkerDetector::**setDictionary**(int **dict_type**, float **error_correction_rate** = 0)
    - aruco::MarkerDetector::**detect** (cv::Mat &input, vector<aruco::Marker> &detectedMarkers)
    - aruco::MarkerDetector::**detect** (cv::Mat &input) → detectedMarkers vector<aruco::Marker>
  - aruco::**Marker**
    - aruco::Marker::**draw**(cv::Mat &**in**, cv::Scalar **color**=cv::Scalar(0,0,255), int **lineWidth**=-1, bool **writeId**=true, bool **writeInfo**=false)



```
Marker found id=142  Corners:[203.199, 241.962][62.3554, 279.877][36.8018, 14
54][170.196, 109.235]
Marker found id=248  Corners:[430.853, 373.421][286.081, 413.286][248.633, 26
21][387.444, 230.51]
Marker found id=814  Corners:[550.907, 147.541][413.062, 185.569][373.05, 54.
][504.638, 19.194]
```



Detección de Marcas: ARUCO

- Ejemplo (**ej9.cpp**) partiremos del código **ej1.cpp**
  - Detectar y visualizar marcadores codificados en la imagen
- Inicialización:

```
// ARUCO Library (Markers)
#include <aruco/aruco.h>
```

```
cv::Mat gray_image;
bool markerWasFound = false;
aruco::MarkerDetector markerDetector;  // handler for marker detector
vector<aruco::Marker> markers;         // storage for detected markers

// ARUCO,   original aruco dictionary. By default
markerDetector.setDictionary(aruco::Dictionary::ARUCO);
```

- Detección:

```
cv::cvtColor(capture, gray_image, cv.COLOR_BGR2GRAY)  # transforms to gray level

markers  = markerDetector.detect(gray_image);        // detects markers on image

markerWasFound = (markers.size()>0);          // checks if some marker was found
```

Detección de Marcas: ARUCO

- Ejemplo (**ej9.cpp**)
  - Detectar y visualizar marcadores codificados en la imagen
  - Acceder a las esquinas: sobrecarga operator[] en clase aruco:.Marker
  - Variable interna:
    - vector<cv::Point> **contourPoints**;

- Visualización:

```
// for each marker, draw info and its boundaries in the image
for (unsigned int i=0; i<markers.size(); i++)
{
        markers[ i ].draw(capture, cv::Scalar(0, 0, 255), 1);

        // display corner order
        for (int j = 0; j < 4; j++)
                cv::putText(capture, std::to_string(j), markers[ i ][ j ], cv::FONT_HERSHEY_DUPLEX, 0.4,
                        cv::Scalar(255, 0, 0), 1, cv::LINE_AA);
}

cv.imshow(WINDOW_CAMERA1, capture)     # Display the resulting frame
```

DETECCIÓN DE MARCAS

Librería ARUCO (Grupo AVA Univ. Córdoba) (ej9b.cpp)

- Leer datos calibración (YAML)
- Calcular la POSE 3D de cada marcador

Detección de Marcas: ARUCO

- Ejemplo (**ej9b.cpp**)
  - Detectar y localizar marcadores codificados en la imagen
  - Calcular la POSE y dibujar un cubo en 3D proyectado en la imagen
  - Cámara Calibrada
  - http://umh1782.edu.umh.es/opencv/
  - Fichero de calibración de la cámara: **calib_LogitechC300.yaml**
  - Fichero de configuración de marcas: **ARUCO_board.yaml**

```
%YAML:1.0
aruco_bc_nmarkers: 6
aruco_bc_mInfoType: 0
aruco_bc_markers:
  - { id:79, corners:[ [ -350., -225., 0. ], [ -150., -225., 0. ], [
      -150., -25., 0. ], [ -350., -25., 0. ] ] }
  - { id:248, corners:[ [ -100., -225., 0. ], [ 100., -225., 0. ], [
      100., -25., 0. ], [ -100., -25., 0. ] ] }
  - { id:892, corners:[ [ 150., -225., 0. ], [ 350., -225., 0. ], [
      350., -25., 0. ], [ 150., -25., 0. ] ] }
  - { id:814, corners:[ [ -350., 25., 0. ], [ -150., 25., 0. ], [ -150.,
      225., 0. ], [ -350., 225., 0. ] ] }
  - { id:964, corners:[ [ -100., 25., 0. ], [ 100., 25., 0. ], [ 100.,
      225., 0. ], [ -100., 225., 0. ] ] }
  - { id:142, corners:[ [ 150., 25., 0. ], [ 350., 25., 0. ], [ 350.,
      225., 0. ], [ 150., 225., 0. ] ] }
```

```
%YAML:1.0
# default calibration data

calibration_date: "Friday Feb 10 14:09:29 2012\n"

camera_matrix: !!opencv-matrix
 rows: 3
 cols: 3
 dt: d
 data: [ 523.18685, 0., 308.07685, 0., 521.88861, 194.65737, 0., 0., 1. ]

distortion_coefficients: !!opencv-matrix
 rows: 4
 cols: 1
 dt: d
 data: [ 0.05141, -0.18643, 0.00977, -0.00059 ]
```

Detección de Marcas: ARUCO

- Ejemplo (**ej9b.cpp**)
- Clases:
  - aruco::**CameraParameters**
    - void aruco::CameraParameters::**setParams**(cv::Mat cameraMatrix, cv::Mat distorsionCoeff, cv::Size size)
  - aruco::**Marker**
    - aruco::Marker::**calculateExtrinsics**(float **markerSize**, cv::Mat **CameraMatrix**, cv::Mat **Distorsion** = cv::Mat(), bool **setYPerpendicular**=true)
    - aruco::Marker::**calculateExtrinsics**(float **markerSize**, aruco::**CameraParameters** &CP, bool **setYPerpendicular**=true)
    - **setYPerpendicular**: **false**: Z perpendicular al plano / **true**: Y perpendicular al plano
    - Componentes de aruco::**Marker**: cv::Mat **Rvec**,**Tvec**;
- Funciones dibujo:
  - aruco::CvDrawingUtils::**draw3dCube** (cv::Mat &**Image**, aruco::Marker &**m**, aruco::**CameraParameters** &CP, bool **setYperpendicular**=false)
  - aruco::CvDrawingUtils::**draw3dAxis** (cv::Mat &**Image**, aruco::Marker &**m**, aruco::**CameraParameters** &CP)

Detección de Marcas: ARUCO

- Ejemplo (**ej9b.cpp**) partiremos de ej9.cpp
  - Declaración adicional de objetos

```
cv::Size camSize(640, 480);            // camera resolution
float markerSize = (float)76;          // size of ARUCO marker (mm)
aruco::CameraParameters cameraParameters; //ARUCO class for camera parameters

// Load calibration data from file
cv::Mat cameraMatrix, distCoeffs;

cv::FileStorage fs ("camera.yaml", cv::FileStorage::READ);
if (fs.isOpened())
{
        fs["camera_matrix"] >> cameraMatrix;
        fs["distortion_coefficients"] >> distCoeffs;

        cout << "Camera Calibration Matrix A: " << endl <<  cameraMatrix << endl;
        cout << "Distortion Coefs: " << distCoeffs << endl;

        // configure internal ARUCO cameraParameters object
        cameraParameters.setParams(cameraMatrix, distCoeffs, camSize);
}
```

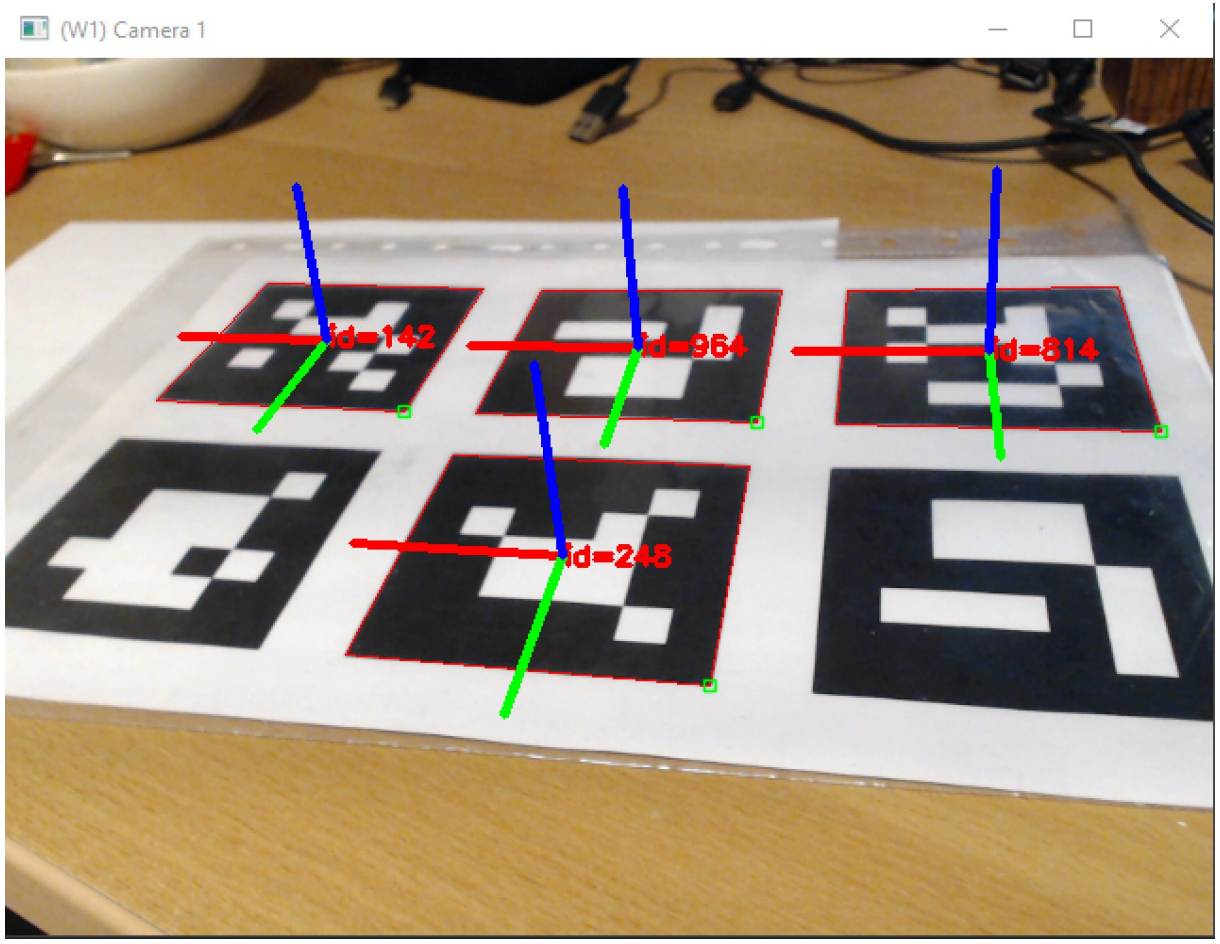
Detección de Marcas: ARUCO

- Ejemplo (**ej9b.cpp**)
  - Procesamiento:

```
//for each marker, draw info and its boundaries in the image and calculate POSE
for (unsigned int i=0; i<markers.size(); i++)
{
        // Calculate POSE
        // last parameter: setYPerpendicular If set the Y axis will be perpendicular to the Surface
        // otherwise, it will be the Z axis
        markers[ i ].calculateExtrinsics(markerSize, cameraParameters, false);

        cout << "Marker id:" << markers[i].id << "  Rvec: " << markers[i].Rvec.t() ;
        cout << "  Tvec: ” << markers[i].Tvec.t() << endl;

        // Draw axis for each marker
        markers[i].draw(capture, cv::Scalar(0, 0, 255), 1);
        aruco::CvDrawingUtils::draw3dAxis(capture, markers[i], cameraParameters);
        aruco::CvDrawingUtils::draw3dCube(capture, markers[i], cameraParameters);

        // display corner order
        for (int j = 0; j < 4; j++)
                cv::putText(capture, std::to_string(j), markers[ i ][ j ], cv::FONT_HERSHEY_DUPLEX, 0.4,
                        cv::Scalar(255, 0, 0), 1, cv::LINE_AA);
}
```

Detección de Marcas: ARUCO

- Ejemplo (**ej9b.cpp**).



DETECCIÓN DE MARCAS

Librería ARUCO (Grupo AVA Univ. Córdoba) (ej9c.cpp)

- Leer datos calibración (YAML)
- Calcular la POSE 3D de de un conjunto de marcadores (MarkerMap)

Detección de Marcas: ARUCO

- Ejemplo (**ej9c.cpp**) cálculo de POSE con múltiples marcas (Board)
- Métodos:
  - aruco::**MarkerMap**
    - aruco::MarkerMap::**readFromFile**(std::string **sfile)**
    - **std::pair**<cv::Mat, cv::Mat> aruco::MarkerMap::**calculateExtrinsics**(
      vector<aruco::Marker>& **markers**, float **markerSize**,
      cv::Mat **CameraMatrix**, cv::Mat **Distorsion**) → <rvec, tvec>
    - **std::pair**: <rvec, tvec>. Se accede mediante los miembros ***first, second***
- Funciones dibujo:
  - aruco::CvDrawingUtils::**draw3dAxis** (cv::Mat& **Image**, **CameraParameters**& CP,
    cv::Mat& **Rvec**, const cv::Mat& **Tvec**, float **axis_size**)
  - cv::**drawFrameAxes**(cv::Mat& **image**, cv::Mat& **cameraMatrix**, cv::Mat& **distCoeffs**,
    cv::Mat& **rvec**, cv::Mat& **tvec**, float **length**, int **thickness**=3)
    OX is drawn in red, **OY** in green and **OZ** in blue.
  - cv::**projectPoints** (InputArray **objectPoints**, InputArray **rvec**, InputArray **tvec**,
    InputArray **cameraMatrix**, InputArray **distCoeffs**, OutputArray **imagePoints**)
    objectPoints: vector<cv::Poin3f>
    imagePoints: vector<cv::Point2f>

Detección de Marcas: ARUCO

- Ejemplo (**ej9c.cpp**) partiremos del código ej9b.cpp
  - Inicializar objetos:

```
// Load ARUCO board info
aruco::MarkerMap boardInfo;

boardInfo.readFromFile("ARUCO_board.yaml");
```

  - Procesamiento:

```
// calculate extrinsics for the MarkerBoard
if (markerWasFound)
{
        std::pair<cv::Mat, cv::Mat> rvec_tvec;
        rvec_tvec  = boardInfo.calculateExtrinsics(markers, markerSize, cameraMatrix, distCoeffs);

        cv::Mat rvec = rvec_tvec.first;
        cv::Mat tvec = rvec_tvec.second;

        cv::drawFrameAxes(capture, cameraMatrix, distCoeffs, rvec, tvec, 80.0, 2);
        //aruco::CvDrawingUtils::draw3dAxis(capture, cameraParameters, rvec, rvec, 80.0);

        cout << "Board:  Rvec: " << rvec.t() << "  Tvec: " << tvec.t() << endl;
}
```

Detección de Marcas: ARUCO

- Ejemplo (**ej9c.cpp**) cálculo de POSE con múltiples marcas (Board)