Realidad Aumentada

Objetivos.

- Aprender a calibrar una cámara usando OpenCV.
- Aprender a utilizar la calibración de la cámara para realizar Realidad Aumentada proyectando un modelo 3D sobre la imagen.
- Aprender a leer un flujo de video.
- Aprender a utilizar OpenCV para almacenar/recuperar parámetros de un fichero.

Descripción básica.

Crea el programa “aug_real” que utiliza los parámetros de calibración de una cámara para superponer información virtual 3d sobre la imagen capturada de la cámara o el vídeo proporcionado.

En la versión básica, los recursos a utilizar son los proporcionados con la práctica y se dibujarán los ejes coordenados del mundo (Figura 1 (a)).

El programa realizará la secuencia siguiente:

Para cada imagen del flujo de vídeo de entrada (utiliza cv::VideoCapture para leer de un flujo de vídeo), el programa realiza los siguientes pasos:

1. Detectar el tablero de calibración usando cv::findChessboardCorners, y refinar las esquinas detectadas con cv::cornerSubPix.
2. Estimar la orientación de la cámara respecto al patrón (su “pose”) usando cv::solvePnP (ya tenemos los parámetros intrínsecos por lo que no será necesario utilizar cv::calibrateCamara().
3. Proyectar sobre la imagen la información 3D a dibujar. Como mínimo se deberían dibujar los ejes XYZ en colores Rojo, Verde y Azul sobre el punto origen del sistemas de referencia 3D (el patrón). El tamaño de los ejes debería ser el mismo que el tamaño de los cuadrados del tablero. Utilizar cv::projectPoints para calcular las coordenadas de imagen al proyectar puntos 3D y cv::line para dibujar sobre la imagen las líneas que forman los ejes.

Se deberán implementar las siguientes funciones:

La CLI del programa debería ser:

```
aug_real [-c] rows cols size intrinsics.yml <input video-file|cam-idx>
```

donde `rows`, `cols` and `size` dan la geometría del tablero de calibración, `intrinsics.yml` corresponde al fichero con los parámetros de calibración generado con la herramienta de OpenCV. Usa el parámetro "`-c`" para indicar que el flujo de entrada será la cámara con índice entero dado. Si no se usa este parámetro, se asume que el flujo de entrada será un fichero de vídeo. Para leer los parámetros intrínsecos del fichero intrinsics.yml utiliza la clase cv::FileStorage.
Se deberán implementar y usar estas funciones:

```
/**
 * @brief Load the intrinsic data to a file.
 * use 'error' for calibration error, 'image-height' for image's height,
 * 'image-width' for image's width, 'camera-matrix' for
 * the camera's matrix and 'distortion-coefficients' for
 * the distortion coefficients.
 * @arg[in] filename of the file where the data is loaded.
 * @arg[out] error is the reprojection error returned by calibration.
 * @arg[out] height is the image's height.
 * @arg[out] width is the image's width.
 * @arg[out] M is the camera matrix.
 * @arg[out] dist_coeffs are the distortion coefficients.
 * @return True if success.
 */
bool fsiv_load_intrinsic_data(const std::string& filename,
                              float & error,
                              int & height,
                              int & width,
                              cv::Mat & M,
                              cv::Mat & dist_coeffs);
/**
 * @brief Compute the 3D world coordinates of the inner board corners.
 * @warning The first corner will have coordinates (size, size, 0.0).
 * @arg[in] rows are the number of board's rows.
 * @arg[in] cols are the number of board's columns.
 * @arg[in] size is the side's length of a board square.
 * @return a vector of with (rows-1)*(cols-1) 3d points following a row,
cols travelling.
 */
std::vector<cv::Point3f> fsiv_compute_object_points( const int rows,
                                                     const int cols,
                                                     const float size);

/**
 * @brief Draw the world's coordinate axes.
 * @warning the use of cv::drawAxes() is not allowed.
 * Use color blue for axe X, green for axe Y and red for axe Z.
 * @arg[in] M is the camera matrix.
```

```
 * @arg[in] dist_coeffs are the distortion coefficients.
 * @arg[in] rvec is the rotation vector.
 * @arg[in] tvec is the translation vector.
 * @arg[in] size is the length of the axis.
 * @arg[in|out] img is the image where the axes are drawn.
 * @pre img.type()=CV_8UC3.
 */
void fsiv_draw_axes(const cv::Mat& M, const cv::Mat& dist_coeffs,
                    const cv::Mat& rvec, const cv::Mat& tvec,
                    const float size,
                    cv::Mat img);
```

Descripción opcional.

- Dibuja un modelo 3D más complejo (ver Figura 1.(b)).

Para ello añade la opción ‘-m’ a la CLI e implementa la interfaz:

```
/**
 * @brief Draw a 3D model using the world's coordinate system.
 * @arg[in] M is the camera matrix.
 * @arg[in] dist_coeffs are the distortion coefficients.
 * @arg[in] rvec is the rotation vector.
* @arg[in] tvec is the translation vector.
 * @arg[in] size is the length of the axis.
 * @arg[in|out] img is the image where the axes are drawn.
 * @pre img.type()=CV_8UC3.
 */
void fsiv_draw_3d_model(const cv::Mat& M, const cv::Mat& dist_coeffs,
                const cv::Mat& rvec, const cv::Mat& tvec,
                const float size,
                cv::Mat img);
```

- Añade un nuevo programa que permita calibrar una cámara con la CLI:

```
calibrate_camera output-file.yml rows cols size input-image_1
input-image_2 ... input-image_n
```

Utiliza cv::FileStorage para escribir los resultados de la calibración implementando la función:

```
/**
 * @brief Write the intrinsic data to a file.
 * use 'error' for calibration error, 'image-height' for image's height,
```

```
 * 'image-width' for image's width, 'camera-matrix' for
 * the camera's matrix and 'distortion-coefficients' for
 * the distortion coefficients.
 * @arg[in] filename of the file where the data is loaded.
 * @arg[in] error is the reprojection error returned by calibration.
 * @arg[in] height is the image's height.
 * @arg[in] width is the image's width.
 * @arg[in] M is the camera matrix.
 * @arg[in] dist_coeffs are the distortion coefficients.
 * @return True if success.
 */
bool fsiv_write_intrinsic_data(const std:string& filename,
                               const float error,
                               const int height,
                               const int width,
                               const cv::Mat& M,
                               const cv::Mat& dist_coeffs);
```

- Muestra una imagen “virtual” sobre el tablero visto en el flujo de vídeo como si el tablero fuera esta imagen (ver Figura 1(c)).

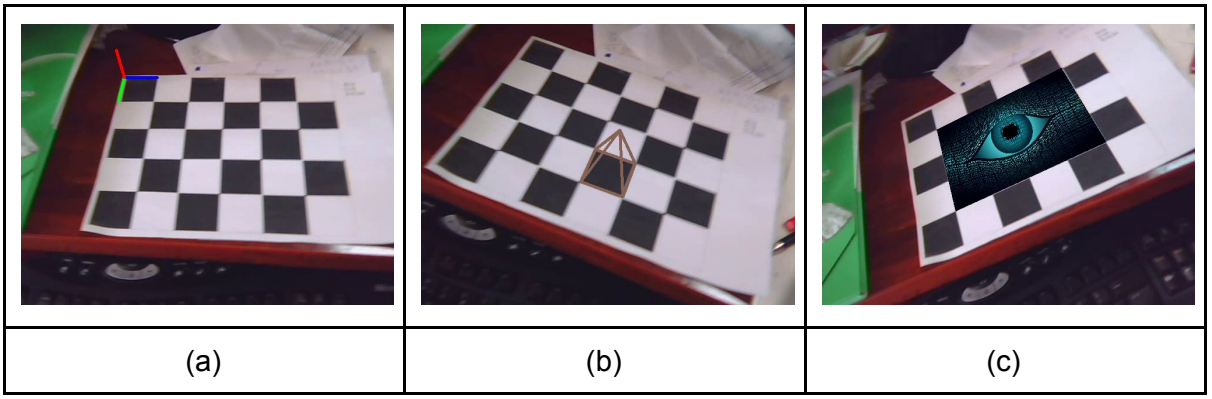
Para ello debes calcular la transformación en perspectiva necesaria para proyectar el rectángulo externo en coordenadas de imagen que define el tablero al proyectarse sobre la imagen y las esquinas de la imagen virtual a dibujar, esto se calcula con cv::getPerspectiveTransform y después debes dibujar esta imagen “virtual” sobre la imagen del tablero con cv::warpPerspective.
Para activar esta función usa el parámetro CLI `[-i=<image>]`

Para implementar esta funcionalidad se debe implementar la interfaz:

```
/**
 * @brief Project input image on the output.
 * @arg[in] input is the image to be projected.
 * @arg[in] image_points define the projected board corners on the
output image where the input image must be projected.
 * @arg[in|out] is the output image.
 * @pre input.type()==CV_8UC3.
 * @pre output.type()==CV_8UC3.
 */
void fsiv_project_image(const cv::Mat& input, const
std::vector<cv::Point2f>& image_points, cv::Mat& output);
```

- Renderizar un vídeo sobre el tablero detectado.

Para activar esta función usa el parámetro CLI `[-v=<video>]` e implementa la función indicada en el punto anterior (ver Figura 1(c)).



(a) (b) (c)

Fig. 1. Ejemplos de las distintas formas de realidad aumentada. (a) Mostrando los ejes del mundo (Z está invertido). (b) Un modelo 3d (una pirámide). (c) Una imagen (puede ser fija o los frames de un video).

Evaluación.

El código está bien escrito, estructurado y compilado.	10%
El código realiza el funcionamiento básico usando la calibración y video proporcionados y dibujando los ejes coordenados.	40%
Se dibuja un modelo 3d complejo.	5%
Se implementa un programa para calibrar la cámara y se usa un vídeo grabado con la cámara del alumno para realizar la realidad aumentada.	30%
Se proyecta una imagen sobre el tablero.	10%
Se proyecta un vídeo sobre el tablero.	5%

(* La entrega fuera de plazo supondrá una penalización en la nota obtenida).