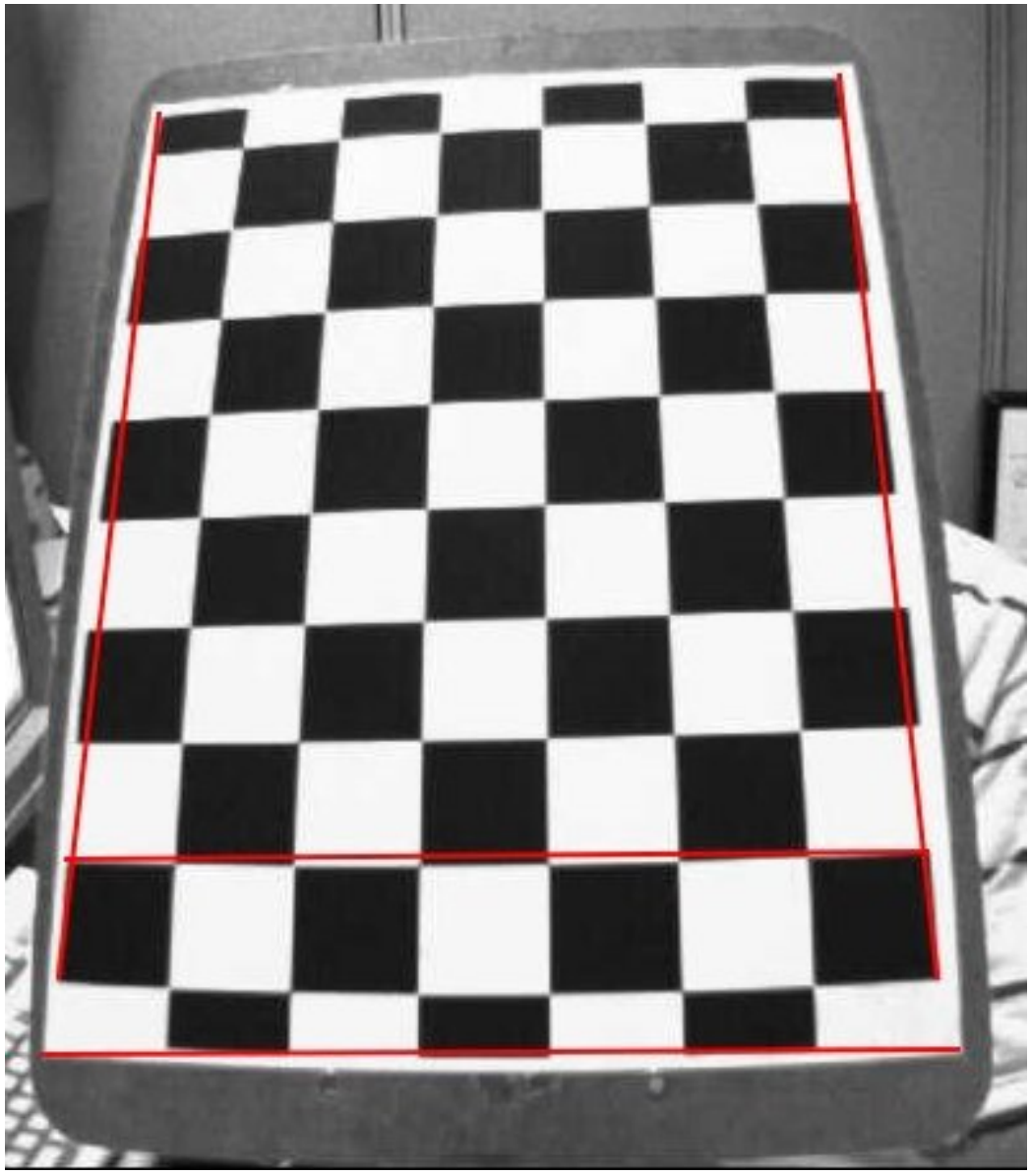
Calibración de cámaras

Araguás, Gastón Redolfi, Javier

17 de Abril del 2019

Ejemplo de distorsión

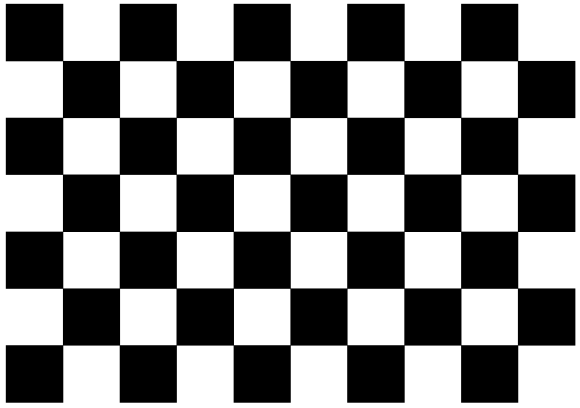
Ejemplo de distorsión



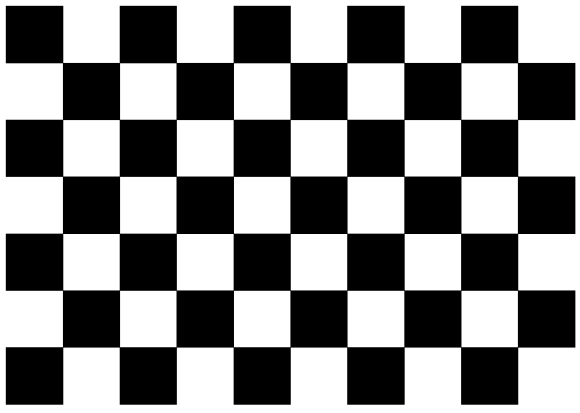
Patrón de calibración

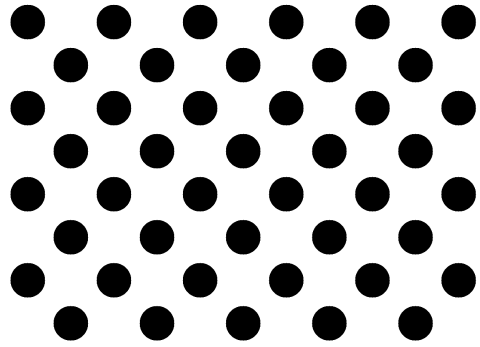
Diferentes tipos de patrones

Diferentes tipos de patrones

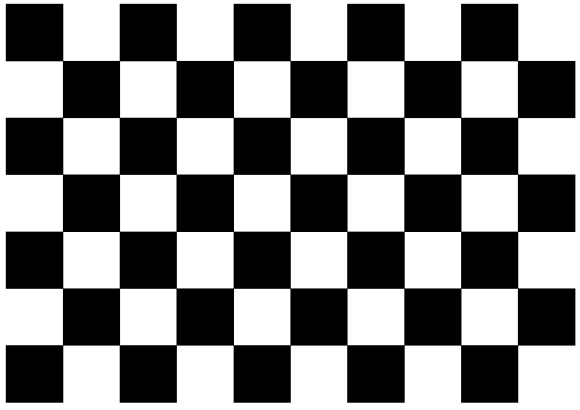


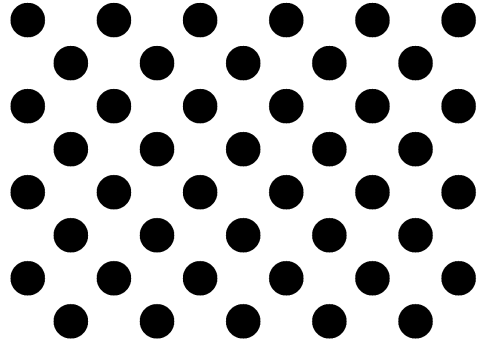
Diferentes tipos de patrones

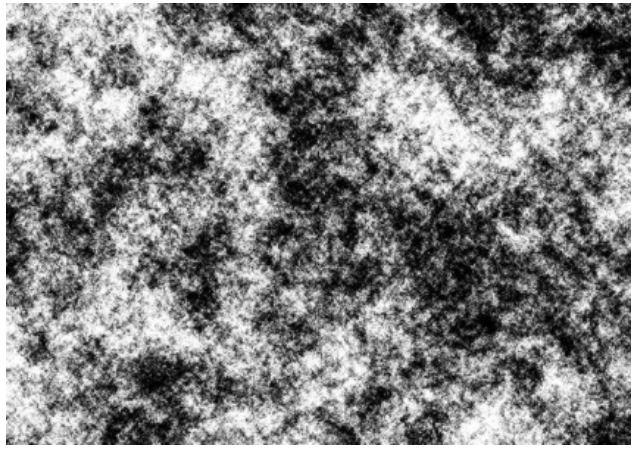




Diferentes tipos de patrones







Pasos para calibrar una cámara

- Capturamos imágenes del patrón
- Si detectamos el patrón
- Creamos correspondencias entre puntos en el patrón y puntos en la imagen
- Calculamos la matriz de calibración y los coeficientes de distorsión

Detección del patrón

Detección del patrón

Usamos la función

```
retval, corners = cv2.findChessboardCorners(image, patternSize)
```

- **`image`** imagen en donde buscaremos
- **`patternSize`** es una tupla que indica el número de esquinas internas
- **`retval`** True si encontró el patrón, False en caso contrario
- **`corners`** ubicaciones de las esquinas en la imagen

Refinamiento de las esquinas

Las coordenadas de las esquinas son aproximadas, por lo tanto para encontrarlas en forma más precisa usamos:

```
corners2 = cv2.cornerSubPix(image, corners, winSize, zeroZone,
                            criteria)
```

- **`image`** imagen en donde encontramos las esquinas
- **`corners`** ubicaciones iniciales de las esquinas en la imagen
- **`winSize`** tamaño de la ventana de refinamiento
- **`zeroZone`** ?
- **`criteria`** criterio de terminación
- **`corners2`** ubicaciones refinadas de las esquinas en la imagen

Marcar puntos en el patrón

```
image2 = cv2.drawChessboardCorners(image, patternSize, corners,
                                   patternWasFound)
```

- **`image`** imagen en donde dibujaremos las esquinas
- **`patternSize`** es una tupla que indica el número de esquinas internas
- **`corners`** ubicaciones de las esquinas en la imagen
- **`patternWasFound`** indica si se encontró todo el patrón

Código para obtener imágenes del patrón

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

import os
import cv2

criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)

dir = 'tmp'
if(not os.path.exists(dir)):
    os.mkdir(dir)
output_file = '{}/{:05}.png'

cap = cv2.VideoCapture(0)

counter = 0

while(True):
    ret, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    ret, corners = cv2.findChessboardCorners(gray, (7, 5), None)

    if ret is True:
        corners2 = cv2.cornerSubPix(gray, corners, (11, 11), (-1, -1), criteria)
        img = cv2.drawChessboardCorners(img, (7, 5), corners2, ret)

    cv2.imshow('img', img)

    key = cv2.waitKey(200) & 0xFF
    if(ret is True) and (key == ord('s')):
        cv2.imwrite(output_file.format(dir, counter), gray)
        counter += 1
    if(key == ord('q')):
        break

cv2.destroyAllWindows()
```

Cálculo de los parámetros de distorsión

Obtención de los parámetros distorsión

Usamos la función

```
retval, cameraMatrix, distCoeffs, rvecs, tvecs =
cv2.calibrateCamera(objectPoints, imagePoints, imageSize)
```

- **`objectPoints`** puntos correspondientes en el espacio de coordenadas del patrón (3D)
- **`imagePoints`** puntos correspondientes en el espacio de coordenadas de la imagen (2D)
- **`imageSize`** tamaño de la imagen usado para inicializar la matriz intrínseca de la cámara
- **`retval`** ?
- **`cameraMatrix`** matriz de cámara
- **`distCoeffs`** vector de salida con los coeficientes de distorsión k_1, k_2, p_1, p_2
- **`rvecs`** vectores de rotación para cada vista del patrón
- **`tvecs`** vectores de translación para cada vista del patrón

Código para calcular los parámetros de distorsión

```
#! /usr/bin/env python
# -*- coding: utf-8 -*-

import numpy as np
import cv2
import glob

criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 30, 0.001)

objp = np.zeros((5 * 7, 3), np.float32)
objp[:, :2] = np.mgrid[0:7, 0:5].T.reshape(-1, 2)

objpoints = []
imgpoints = []

images = glob.glob('tmp/*.png')

for fname in images:
    img = cv2.imread(fname)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    ret, corners = cv2.findChessboardCorners(gray, (7, 5), None)

    if ret is True:
        objpoints.append(objp)

        corners2 = cv2.cornerSubPix(gray, corners, (11, 11), (-1, -1),
                                    criteria)
        imgpoints.append(corners2)

        img = cv2.drawChessboardCorners(img, (7, 5), corners2, ret)
        cv2.imshow('img', img)
        cv2.waitKey(300)

cv2.destroyAllWindows()

ret, mtx, dist, rvecs, tvecs = cv2.calibrateCamera(objpoints, imgpoints, gray.shape[::-1],
                                                   None, None)
```

Práctica 0 - Calibración de una cámara

Práctica 0 - Calibración de una cámara

Calibración de una cámara

- Utilizar los códigos de arriba para calibrar una cámara.
- Usar pickle para guardar la matriz de calibración y los coeficientes de distorsión.
- Para corroborar la correcta calibración usar el resultado de calibración obtenido para eliminar la distorsión de una imagen.

Práctica 0 - Calibración de una cámara

Calibración de una cámara

- Utilizar los códigos de arriba para calibrar una cámara.
- Usar pickle para guardar la matriz de calibración y los coeficientes de distorsión.
- Para corroborar la correcta calibración usar el resultado de calibración obtenido para eliminar la distorsión de una imagen.

Ayuda

- **`cv2.undistort`**
- Como la distorsión en la parte externa de la imagen es mayor, capturar una imagen en donde se vea una línea sobre alguno de los márgenes exteriores de la imagen para que la distorsión sea apreciable.