Davide Zanin 1035601

# ELABORAZIONE DI DATI TRIDIMENSIONALI – HOMEWORK 1

## Procedura utilizzata

### Stereo camera calibration

Per la calibrazione della fotocamera (sinistra, destra e poi stereo) sono state usate le foto indicate nel file “input\_photo.txt” e affinché il programma le carichi correttamente devono trovarsi nella stessa directory del file eseguibile nella cartella “calibration\_set”.

Le foto vengono caricate una ad una e tramite la funzione **findChessboardCorners** vengono individuati i corner. Questa funzione vuole come input l'immagine in cui ricercare i corner, un oggetto di tipo Size che specifica la dimensione delle patter dei corner da trovare (nel nostro caso 7x5), un vector di Point2f in cui verranno inseriti i corner trovati e un flag. I flag utilizzati sono i seguenti:

* CV\_CALIB\_CB\_ADAPTIVE\_THRESH use adaptive thresholding to convert the image to black and white, rather than a fixed threshold level (computed from the average image brightness).
* CV\_CALIB\_CB\_NORMALIZE\_IMAGE normalize the image gamma with EqualizeHist before applying fixed or adaptive thresholding.
* CV\_CALIB\_CB\_FILTER\_QUADS use additional criteria (like contour area, perimeter, square-like shape) to filter out false quads that are extracted at the contour retrieval stage.
* CALIB\_CB\_FAST\_CHECK Runs a fast check on the image that looks for chessboard corners, and shortcuts the call if none are found. This can drastically speed up the call in the degenerate condition when no chessboard is observed.

La funzione **findChessboardCorners** ritorna inoltre un booleano che specifica se i corner sono stati individuati correttamente.

Per migliorare la precisione nel posizionamento dei corners viene usata la funzione cornerSubPix usando come parametri l'immagine usata nela fuzione precendente, però convertita in scala di grigi, il vettore di corner trovati i parametri di default consiglia nella documentazione di OpenCV, cioè

* Size(11,11),
* Size(-1,-1),
* TermCriteria(CV\_TERMCRIT\_ITER+CV\_TERMCRIT\_EPS, 30, 0.01));
* winSize – Half of the side length of the search window. For example, if winSize=Size(5,5) , then a search window would be used
* zeroZone – Half of the size of the dead region in the middle of the search zone over which the summation in the formula below is not done. It is used sometimes to avoid possible singularities of the autocorrelation matrix. The value of (-1,-1) indicates that there is no such size
* criteria – Criteria for termination of the iterative process of corner refinement. That is, the process of corner position refinement stops either after a certain number of iterations or when a required accuracy is achieved. The criteria may specify either of or both the maximum number of iteration and the required accuracy

Una volta che sono state analizzate tutte le foto viene utilizzata la funzione **calibrateCamera** che restituirà il matrice con i parametri intrinseci della fotocamera e il vettore con i coefficienti di distorsioni. Questa funzione viene chiamata due volte, una volta per la fotocamerea sinistra e una volta per la fotocamera destra, quindi alla fine si avranno due matrici con i parametri intrinseci (M0 per la fotocamera sinistra e M1 per la fotocamera sinistra) e due vettori con i coefficienti di distorsione (D0 per la fotocamera sinistra e D1 per la fotocamera destra). I volori ottenuti si trovano nei file......

A questo punto viene eseguita la calibrazione stereo usando la funzione stereoCalibrate a cui vengono passati i seguenti parametri:

* **objectPoints**
* **imagePoints[0] e imagePoints[1]**
* **M[0], D[0], M[1] e D[1]**
* img.size(),
* RR,
* T,
* E,
* F,
* cvTermCriteria(CV\_TERMCRIT\_ITER+CV\_TERMCRIT\_EPS, 100, 1e-5),
* CV\_CALIB\_FIX\_INTRINSIC

Questa funzione ritorna anche un valore double che indica l'errore di proiezione. Successivamente con la funzione **stereoRectify**  vengono inizializzate la matrice di rotazione tra il sistema di coordinate della fotocamera sinistra e quella destra (RR), le matrici di rotazione per le due fotocamere (R0 e R1), le matrici di proiezione nel sistema di coordinate rectfy (P0 e P1) e la matrice disparity-to-depth (Q). Il flag è stato impostato al valore CV\_CALIB\_ZERO\_DISPARITY, in questo modo le immagini la funzione fa in modo che, nella vista rettificata, i punti principali delle immagini delle due fotocamere abbiano le stesse coordinate in pixel. Infine il valore di alpha è stato impostato a 0 in questo modo le immagini rettificate vengono zoomate e shiftate in modo da contenere solo pixel validi.

Come ultima operazione è stata chiamata due volte al funzione initUndistortRectifyMap in modo da inizializzare le matrici di undistortion e rectification per entrambe le fotocamere (mx0, my0, mx1, my1). Come tipo per le matrici mx0 e mx1 è stato scelto CV\_16SC2. Con quest’ultima operazione si conclude la fase di calibrazione.

### Image undistort e rectify

Per rettificare un immagine è sufficiente usare la funzione remap passando come parametri l’immagine da rettificare, l’immagine di output e le matrici mx e my della fotocamera da cui è stata scattata la foto e un flag che specifica il metodo di interpolazione da utilizzare, che è stato impostato a INTER\_LINEAR in modo da usare l’iterpolazione bilineare.

### Disparity map calculation

Il calcolo dell’immagine di disparità viene calcolata a partire da una foto scattata dalla fotocamera sinistra e una dalla fotocamera destra, entrambe prima devono essere rettificate. Una volta fatto ciò viene usata la classe StereoSGBM che usa l’algoritmo semi-global block matching per il calcolo delle corrispondenze stereo nelle due immagini. I parametri della classe sono stati impostati come segue:

int numberOfDisparities = ((img.size().width/8) + 15) & -16;

sgbm.preFilterCap = 4;

sgbm.SADWindowSize = 3;

int cn = img.channels();

sgbm.P1 = 8\*cn\*sgbm.SADWindowSize\*sgbm.SADWindowSize;

sgbm.P2 = 32\*cn\*sgbm.SADWindowSize\*sgbm.SADWindowSize;

sgbm.minDisparity = 16;

sgbm.numberOfDisparities = numberOfDisparities;

sgbm.uniquenessRatio = 5;

sgbm.speckleWindowSize = 300;

sgbm.speckleRange = 64;

sgbm.disp12MaxDiff = -1;

sgbm.fullDP = 1;

(spiegazione)

A questo punto è sufficiente passare all’oggetto di tipo StereiSGBM le due immagini rettificate e un immagine di output per ottenere la disparity map.

### Range image calculation

Calcolare la range image è molto semplice, è sufficientre scalare i colori della disparity map in un intervallo più ampio, in modo che risultino più visibili. Questo viene fatto con la seguente funzione:

disparity.convertTo(range, CV\_8U, 255/((sgbm.numberOfDisparities)\*16.));

### Generation of the point could from the disparity image

Per la generazione delle coordinate (x, y, z) dei punti della disparity map, è stata usata la funzione reprojectImageTo3D che data la disparity map e la matrice Q fornisce una matrice che contiene le coordinate 3D di ogni punto della disparity map. Inoltre è stato specificato alla funzione che gestisca i pixel con disparity minima impostando il valore di z ad un numero molto alto.

Per colorare i punti della cloud è stata anche caricata l’immagine sinistra da cui è stata calcolata la disparity map e, una volta che è stata rettificata, a ciascun punto (x, y, z) che veniva creato a partire dalla matrice generata con la funzione reprojectImageTo3D veniva assegnato il colore della foto rettificata del pixel (x, y).

Tutti i punti che presentavano coordinate maggiori in modulo di 80 non vengono inseriti nella cloud.

### Visualization with PCL

Per visualizzare la point cloud è stato sufficiente creare un oggetto viewer di tipo PCLVisualizer che visualizzerà la nuovola di punti e un oggetto rgb di tipo PointCloudHandlerRGBField che gestisce la colorazione dei punti della point cloud. A questo punto sono stati aggiunti la point cloud e rgb a viewer e ne è stata avviata la visualizzazione con il metodo spin.

## Parametri ottenuti per la calibrazione della telecamera stereo

## Immagini ottenute