Projeto Demonstrativo 2 - Calibração de câmeras

Samuel Venzi Lima Monteiro de Oliveira
14/0162241
samuel.venzi@me.com

23-04-2016

1 Objetivos

O objetivo deste processo é realizar um estudo sobre câmera e sua calibração. A partir desse estudo, desenvolver um método para calculo de dimensões de objetos dada a distância entre este e a câmera.

2 Introdução

O uso de câmeras no âmbito da visão computacional é uma competência essencial devido às inúmeras aplicações práticas existentes. Tais aplicações vão desde câmeras de vigilância até carros autônomos, portanto é necessário saber sobre o funcionamento de câmeras e entender seus detalhes para que seja bem aplicada. Os principais pontos abordados neste estudo são a calibração de câmeras e, a partir disso, a avaliação do tamanho de objetos no frame da imagem capturada.

A estrutura básica das câmeras atuais segue a estrutura consagrada pelos primeiros estudos feitos sobre capturas de imagens. O que essa estrutura permite é a entrada de luz por um orifício ou conjunto de lentes e sua projeção em um filme ou sensor que captura a imagem. As câmeras modernas utilizam lentes e sensores, e é importante entender como essas duas partes influenciam na maneira que deve se lidar com a manipulação de imagens dessas câmeras.

Com a introdução de lentes nas câmeras, foi possível a captura de imagens de muito melhor qualidade pois elas permitiam maior entrada de luz sem o desfoque da imagem, o que acontecia com o aumento do orifício. Porém, a depender da qualidade de fabricação da lente, com elas ocorre o fenômeno da distorção que pode ser um grande problema para certas aplicações. No caso do uso de sensores, certas caracteríscas são importantes serem conhecidas, como seu tamanho e o tamanho de seus píxels.

A calibração da câmera tem como função anular certas distorções da imagem que possam ocorrer por sua parte física. Portanto, alinhando hardware e software pode-se ter a representação mais fiel possível e a partir dessa representação retirar informações de interesse.

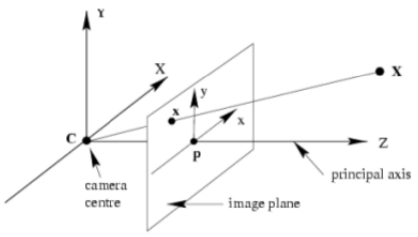

Figura 2.1

Na figura 2.1 pode-se verificar algumas das variáveis de calibração. O sistema de referência é o plano da imagem, chamado também de frame e as coordenadas dos pixels no frame são dados por um par ordenado (x,y).

A partir de estudos, são extraídas informções sobre a câmera, como seus parâmetros intrínsecos e extrínsicos. Onde os parâmetros intrínsecos caracterizam propriedades óticas, geométricas e digitais da câmera, além de sua distorção.

3 Materiais e Metodologia

3.1 Materiais

- Computador com ambiente Linux (Ubuntu)
- Câmera própria do computador
- OpenCV
- Bola com 6cm de diâmetro

3.2 Metodologia

Antes de mexer com a câmera em si, foi criada uma aplicação que captura o clique do mouse em dois pontos arbitrários da janela, desenha um linha e calcula a distância em pixels entre eles. O programa primeiramente cria uma imagem genérica, foi escolhida uma completamente preta, e com o uso de funções do OpenCV, captura e desenha uma linha ente dois pontos. Após isso ele calcula a distância Euclidiana Bidimensional.

```
#include "opencv2/highgui/highgui.hpp"
#include <iostream>

using namespace std;
using namespace cv;

void CallBackFunc(int event, int x, int y, int flags, void* ptr){
if (event == EVENT_LBUTTONDOWN){
vector<Point>*p = (vector<Point>*)ptr;
    p->push_back(Point(x,y));
}

}

void GetDistance(void* points){
float distance;
vector<Point>*p = (vector<Point>*)points;
Point diff = p->at(0) - p->at(1);
distance = sqrt(diff.x*diff.x + diff.y*diff.y);
cout<<"Distância entre os pontos é: "<<distance<<endl;
}


int main(int argc, char const *argv[])
{
vector<Point> points;
int height = 500, width = 500;
int i, j;
```

```
bool click = false;

Mat img(height, width, CV_8U);
namedWindow("My Window", 1);

while(!click){
imshow("My Window", img);
waitKey(100);
setMouseCallback("My Window", CallBackFunc, (void*)&points);
if (points.size() == 2)
{
line(img, points.at(0), points.at(1), Scalar(255, 255, 255));
destroyWindow("My Window");
click = true;
}
}
waitKey(100);
GetDistance((void*)&points);
imshow("Line", img);
waitKey(0);
return 0;
}
```