

三维重建作业

1 三维重建方法

双目立体视觉的三维重建过程分为几个步骤：双目相机标定、立体矫正、立体匹配、获取深度图。由于网站<https://vision.middlebury.edu/stereo/data/>上给出的数据已经完成立体矫正，因此主要部分是双目的立体匹配。常见的双目立体匹配算法有三种：BM、SGBM [1]、GC [2]，经典的几种算法的区别如下：

- (1) BM：块匹配（Block Matching）算法，由于是局部匹配速度很快，但精度较低，得到的视差图效果较差；
- (2) SGBM：半全局的块匹配（Semi-Global Block Matching）算法，在对每个像素使用块匹配之后，利用 Graph Cut 算法进行全局优化，牺牲了一定的速度，但提高了匹配精度；
- (3) GC：Graph Cut 算法，全局优化算法，速度很慢，但精度很高。

可以发现，立体匹配需要在速度和精度之间取舍，因此不同的算法适用于不同的场景

2 实验流程

2.1 实验数据选取

从数据中选取同一场景的两个图片，如图1，图片已经经过立体矫正，因此分别采取 BM、SGBM、GC 算法进行立体匹配，从而获取视差图。



图 1：实验选取场景：artroom, 2021 Mobile stereo datasets

2.2 实验结果

实验首先采用 OpenCV 库中的 BM、SGBM 算法分别进行立体匹配。BM 和 SGBM 算法利用的是 OpenCV-4.6.0 版本的 `cv::StereoBM` 和 `cv::StereoSGBM` 类。由于涉及的参数较多，采用滑动条方式手动调节参数，如图2(a)。

经过调节，效果较好的两组参数如表1和2。为了更好对比两种算法的好坏，大部分参数调节为相同。

表 1: BM 相关参数设置

BM 参数	设定值	参数含义
preFilterCap	32	预处理滤波器的判断阈值
blockSize ¹	15	SAD 窗口大小
minDisparity	0	最小视差
numDisparities	240	最大视差值与最小视差值之差，必须是 16 的整数倍
textureThreshold	1000	低纹理区域判断阈值，保证有足够的纹理以克服噪声
uniquenessRatio	1	视差唯一性百分比
speckleWindowSize	100	检查视差连通区域变化度的窗口大小，值为 0 时取消 speckle 检查
speckleRange	32	视差变化阈值，当窗口内视差变化大于阈值时，该窗口内的视差清零
disp12MaxDiff	1	左视差图（直接计算得出）和右视差图（通过 cvValidateDisparity 计算得出），之间的最大容许差异，默认为-1

¹ 加粗表示影响较大的参数，改变会对效果产生较大的影响，下同。

表 2: SGBM 相关参数设置

BM 参数	设定值	参数含义
preFilterCap	32	预处理滤波器的判断阈值
blockSize	5	SAD 窗口大小
minDisparity	0	最小视差
numDisparities	240	最大视差值与最小视差值之差
P1	600	相邻像素视差增/减 1 时的惩罚系数
P2	2400	相邻像素视差变化值大于 1 时的惩罚系数
uniquenessRatio	10	视差唯一性百分比
speckleWindowSize	100	检查视差连通区域变化度的窗口大小
speckleRange	32	视差变化阈值
disp12MaxDiff	1	左右视差图之间的最大容许差异

BM 算法和 SGBM 算法立体匹配的结果如图2(b)和2(c)。对比可知，由于仅考虑了局部区域，BM 算法生成的视差图噪点较多，效果相对较差；SGBM 算法考虑了全局信息，生成的视差图较

为平滑，但细节略差。

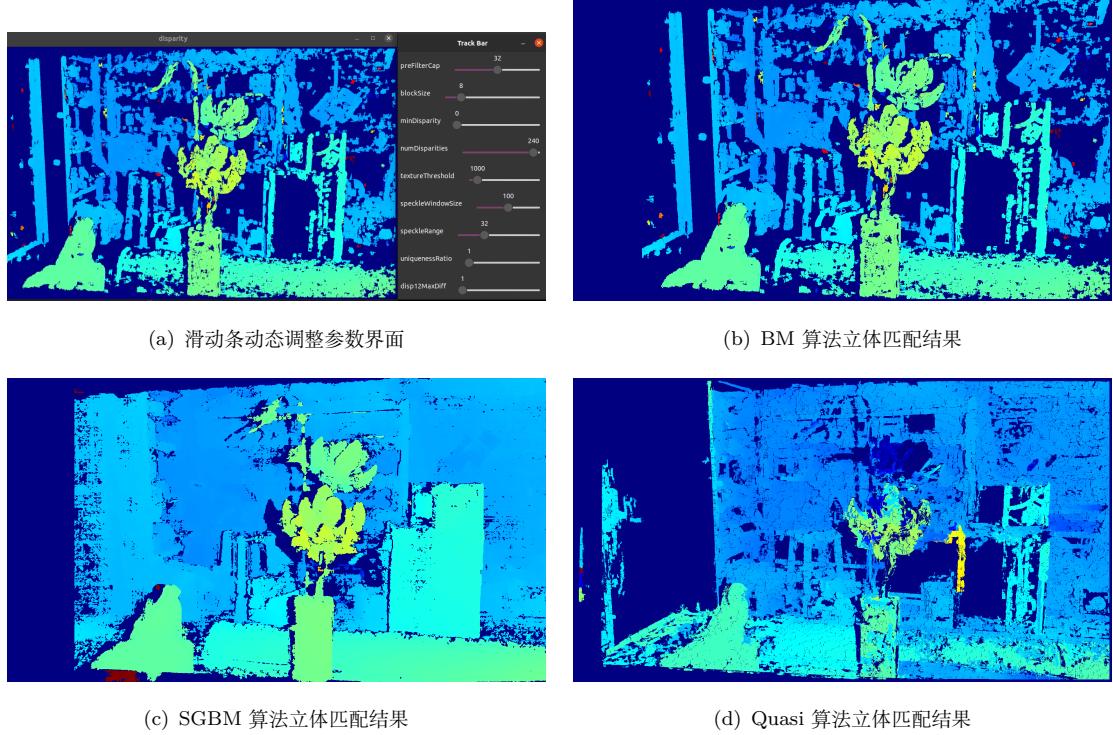


图 2: 立体匹配实验结果

由于 OpenCV 在 3.x 之后移除了 GC 算法，因此在 OpenCV-4.6.0 的 stereo 库中选取了 QuasiDenseStereo 类，尝试用于立体匹配。该算法由 Stoyanov 等人 [3] 提出，旨在机器人辅助手术过程通过三维重建恢复组织结构和形态，该算法围绕一组候选特征匹配传播视差信息，尽可能避免镜面高光、仪器遮挡和视图依赖照明偏差等问题。

实验结果如图2(d)，可以发现该算法在识别细小物体方面效果较好，但整体效果不如 SGBM 算法。

参考文献

- [1] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007.
- [2] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 1, pages 105–112. IEEE, 2001.
- [3] Danail Stoyanov, Marco Visentini Scarzanella, Philip Pratt, and Guang-Zhong Yang. Real-time stereo reconstruction in robotically assisted minimally invasive surgery. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 275–282. Springer, 2010.

附录

BM 算法代码如下 (C++):

```
1 #include <iostream>
2 #include <algorithm>
3 // 核心库，包含一些cv常量
4 #include <opencv2/core.hpp>
5 // 标定重建工具箱
6 #include <opencv2/calib3d.hpp>
7 // 图像处理库
8 #include <opencv2/imgproc.hpp>
9 // 读写图片相关库
10 #include <opencv2/imgcodecs.hpp>
11 // 图像GUI
12 #include <opencv2/highgui.hpp>
13
14 int windowSize = 8;
15 int preFilterCap = 32;
16 // int blockSize = 2 * windowSize - 1;
17 int minDisparity = 0; // 最小视差，默认值为0
18 int numDisparities = 240; // 最大视差值与最小视差值之差，必须是16的整数倍
19 int textureThreshold = 1000; // 低纹理区域判断阈值，保证有足够的纹理以克服噪声
20 int uniquenessRatio = 1; // 视差唯一性百分比
21 int speckleWindowSize = 100; // 检查视差连通区域变化度的窗口大小，值为0时取消
    speckle 检查
22 int speckleRange = 32; // 视差变化阈值，当窗口内视差变化大于阈值时，该窗口内的视差
    清零
23 int disp12MaxDiff = 1;
24
25 void getDisparity(const cv::Mat& imageLeft, const cv::Mat& imageRight, cv::Mat&
    disparity, cv::Mat& colorMap) {
26     cv::Ptr<cv::StereoBM> bm = cv::StereoBM::create();
27     bm->setPreFilterCap(preFilterCap); // 预处理滤波器的判断阈值
28     bm->setBlockSize(2 * windowSize - 1); // SAD窗口大小
29     // bm->setROI1();
30     bm->setMinDisparity(minDisparity); // 最小视差，默认值为0
31     bm->setNumDisparities(numDisparities); // 最大视差值与最小视差值之差，必须是16
        的整数倍
32     bm->setTextureThreshold(textureThreshold); // 低纹理区域判断阈值，保证有足够的
        纹理以克服噪声
33     bm->setUniquenessRatio(uniquenessRatio); // 视差唯一性百分比
34     bm->setSpeckleWindowSize(speckleWindowSize); // 检查视差连通区域变化度的窗口大
        小，值为0时取消 speckle 检查
35     bm->setSpeckleRange(speckleRange); // 视差变化阈值，当窗口内视差变化大于阈值
        时，该窗口内的视差清零
36     bm->setDisp12MaxDiff(disp12MaxDiff); // 左视差图（直接计算得出）和右视差图（通过
        cvValidateDisparity计算得出），之间的最大容许差异，默认为-1
37
38     bm->compute(imageLeft, imageRight, disparity);
```

```
39     cv::Mat disparityGray, disparityRGB;
40     // 转换图像的存储格式（深度）除以16转化为真实视差
41     disparity.convertTo(disparityGray, CV_32F, 1.0 / 16);
42     // 转化为8U类型
43     cv::Mat disp8U = cv::Mat(disparity.rows, disparity.cols, CV_8UC1);
44     // 归一化
45     cv::normalize(disparity, disp8U, 0, 255, cv::NORM_MINMAX, CV_8UC1);
46     // 伪彩色图
47     // cv::Mat colorMap;
48     cv::applyColorMap(disp8U, colorMap, cv::COLORMAP_JET);
49 }
50
51 static void onChange(int pos, void* data) {
52     std::vector<cv::Mat> images = *((std::vector<cv::Mat*>*)data);
53     cv::Mat disparity, colorMap;
54     getDisparity(images[0], images[1], disparity, colorMap);
55     cv::imshow("disparity", colorMap);
56 }
57
58 int main(int argc, char const *argv[])
59 {
60     // 采用灰度图模式读取原始图
61     int colorMode = cv::IMREAD_GRAYSCALE;
62     cv::Mat imageLeft = cv::imread("../images/im0.png", colorMode);
63     cv::Mat imageRight = cv::imread("../images/im1.png", colorMode);
64     // std::cout << imageLeft.type() << std::endl;
65     cv::Mat disparity, colorMap;
66
67     // 根据初始参数获取视差图并保存
68     getDisparity(imageLeft, imageRight, disparity, colorMap);
69     // std::cout << disparity << std::endl;
70     cv::imwrite("../images/output/BM.png", colorMap);
71     cv::namedWindow("disparity", cv::WINDOW_NORMAL);
72     cv::resizeWindow("disparity", 960, 540);
73     cv::imshow("disparity", colorMap);
74
75     // 利用滑动条修改参数
76     std::vector<cv::Mat> images;
77     images.push_back(imageLeft);
78     images.push_back(imageRight);
79     cv::namedWindow("Track Bar");
80     cv::createTrackbar("preFilterCap", "Track Bar", &preFilterCap, 63, onChange, &
81     images);
82     cv::createTrackbar("blockSize", "Track Bar", &windowSize, 50, onChange, &images
83     );
84     // cv::createTrackbar("minDisparity", "Track Bar", &minDisparity, 100, onChange
85     , &images);
86     // cv::createTrackbar("numDisparities", "Track Bar", &numDisparities, 100,
87     onChange, &images);
```

```
84     cv::createTrackbar("textureThreshold", "Track Bar", &textureThreshold, 10000,
85         onChange, &images);
86     cv::createTrackbar("speckleWindowSize", "Track Bar", &speckleWindowSize, 200,
87         onChange, &images);
88     cv::createTrackbar("speckleRange", "Track Bar", &speckleRange, 100, onChange, &
89         images);
90     cv::createTrackbar("uniquenessRatio", "Track Bar", &uniquenessRatio, 100,
91         onChange, &images);
92     // cv::createTrackbar("disp12MaxDiff", "Track Bar", &disp12MaxDiff, 1000,
93         onChange, &images);
94     // setTrackBar(imageLeft, imageRight);
95     cv::waitKey(0);
96     cv::destroyAllWindows();
97     return 0;
98 }
```

SGBM 算法代码如下：

```
1 #include <iostream>
2 #include <algorithm>
3 // 核心库，包含一些cv常量
4 #include <opencv2/core.hpp>
5 // 标定重建工具箱
6 #include <opencv2/calib3d.hpp>
7 // 图像处理库
8 #include <opencv2/imgproc.hpp>
9 // 读写图片相关库
10 #include <opencv2/imgcodecs.hpp>
11 // 图像GUI
12 #include <opencv2/highgui.hpp>
13
14 void SGBM(const cv::Mat& imageL, const cv::Mat& imageR, cv::Mat& disp) {
15     cv::Mat grayImageL, grayImageR;
16     cv::cvtColor(imageL, grayImageL, cv::COLOR_BGR2GRAY);
17     cv::cvtColor(imageR, grayImageR, cv::COLOR_BGR2GRAY);
18     // cv::namedWindow("ImageL", cv::WINDOW_NORMAL);
19     // cv::namedWindow("ImageR", cv::WINDOW_NORMAL);
20     // cv::imshow("ImageL", grayImageL);
21     // cv::imshow("ImageR", grayImageR);
22     cv::Ptr<cv::StereoSGBM> sgbm = cv::StereoSGBM::create(0, 16, 3);
23     int numberofDisparities = ((imageL.size().width / 8) + 15)&(-16);
24     // std::cout << numberofDisparities << std::endl;
25     int SADWindowSize = 5;
26     int sgbmWinSize = SADWindowSize > 0 ? SADWindowSize : 3;
27     int cn = imageL.channels();
28
29     sgbm->setPreFilterCap(32);
30     sgbm->setBlockSize(sgbmWinSize);
31     sgbm->setP1(8 * cn * sgbmWinSize * sgbmWinSize);
32     sgbm->setP2(32 * cn * sgbmWinSize * sgbmWinSize);
```

```
33     sgbm->setMinDisparity(0);
34     sgbm->setNumDisparities(numberOfDisparities);
35     sgbm->setUniquenessRatio(10);
36     sgbm->setSpeckleWindowSize(100);
37     sgbm->setSpeckleRange(32);
38     sgbm->setDisp12MaxDiff(1);
39
40     sgbm->compute(grayImageL, grayImageR, disp);
41 }
42
43 int main(int argc, char const *argv[])
44 {
45     cv::Mat rgbImageL = cv::imread("../images/im0.png", cv::IMREAD_COLOR);
46     cv::Mat rgbImageR = cv::imread("../images/im1.png", cv::IMREAD_COLOR);
47
48     cv::Mat disp;
49     // SGBM 算法
50     SGBM(rgbImageL, rgbImageR, disp);
51
52     // 除以 16 得到真实的视差
53     disp.convertTo(disp, CV_32F, 1.0 / 16);
54     // 转化为 8U 类型
55     cv::Mat disp8U = cv::Mat(disp.rows, disp.cols, CV_8UC1);
56     // 归一化
57     cv::normalize(disp, disp8U, 0, 255, cv::NORM_MINMAX, CV_8UC1);
58     // 伪彩色图
59     cv::Mat colorMap;
60     cv::applyColorMap(disp8U, colorMap, cv::COLORMAP_JET);
61     cv::imwrite("../images/output/SGBM.png", colorMap);
62     // cv::namedWindow("Disparity", cv::WINDOW_NORMAL);
63     // cv::imshow("Disparity", colorMap);
64
65     cv::waitKey();
66     cv::destroyAllWindows();
67     return 0;
68 }
```

Quasi 算法代码如下：

```
1 #include <iostream>
2 #include <algorithm>
3 // 核心库，包含一些cv常量
4 #include <opencv2/core.hpp>
5 // 标定重建工具箱
6 #include <opencv2/calib3d.hpp>
7 // 图像处理库
8 #include <opencv2/imgproc.hpp>
9 // 读写图片相关库
10 #include <opencv2/imgcodecs.hpp>
11 // 图像GUI
```

```
12 #include <opencv2/highgui.hpp>
13
14 #include <opencv2/stereo/quasi_dense_stereo.hpp>
15
16 void quasi(const cv::Mat& imageL, const cv::Mat& imageR, cv::Mat& disp) {
17     cv::Mat grayImageL, grayImageR;
18     cv::cvtColor(imageL, grayImageL, cv::COLOR_BGR2GRAY);
19     cv::cvtColor(imageR, grayImageR, cv::COLOR_BGR2GRAY);
20
21     // cv::namedWindow("ImageL", cv::WINDOW_NORMAL);
22     // cv::namedWindow("ImageR", cv::WINDOW_NORMAL);
23     // cv::imshow("ImageL", grayImageL);
24     // cv::imshow("ImageR", grayImageR);
25     cv::Ptr<cv::stereo::QuasiDenseStereo> quasi = cv::stereo::QuasiDenseStereo::
26         create(grayImageL.size());
27     // 处理图片
28     quasi->process(grayImageL, grayImageR);
29     // // 得到视差图
30     // uint8_t displvl = 80;
31     // cv::Mat disp;
32     disp = quasi->getDisparity();
33 }
34
35 int main(int argc, char const *argv[])
36 {
37     cv::Mat rgbImageL = cv::imread("../images/im0.png", cv::IMREAD_COLOR);
38     cv::Mat rgbImageR = cv::imread("../images/im1.png", cv::IMREAD_COLOR);
39
40     cv::Mat disp;
41     // SGBM算法
42     // SGBM(rgbImageL, rgbImageR, disp);
43     quasi(rgbImageL, rgbImageR, disp);
44
45     // 转化为8U类型
46     disp.convertTo(disp, CV_8U);
47
48     // 伪彩色图
49     cv::Mat colorMap;
50     cv::applyColorMap(disp, colorMap, cv::COLORMAP_JET);
51     cv::imwrite("../images/output/Quasi.png", colorMap);
52     // cv::namedWindow("Disparity", cv::WINDOW_NORMAL);
53     // cv::imshow("Disparity", colorMap);
54
55     cv::waitKey();
56     cv::destroyAllWindows();
57 }
```