水平多目立体匹配

何琪辰　06122477
计算机工程与科学学院

概述

立体视觉是指用摄像机获取同一场景的不同图像以获得深度信息的方法。完成这项工作的方法有很多种。其中水平多目图像采集进行立体匹配是较为常见的一种做法。

由于现实生活中，没有理想状态的图像采集设备，不能将一场景的投影完全拍摄下来。在获取图像的过程中不可避免的丢失了场景原有的信息。如何恢复这些信息是立体匹配中所要完成的事情。

在双目匹配中，如果获得图像的视差，和已经获得的摄像机焦距以及基线，就可

首先，摄像机本与大气环境的影响。摄像机设备的图像感应器或多或少会产生杂点，对最终结果产生影响。

其次，现实生活中的世界里，不少纹理会有周期性现象。比如书架上堆放的书本。对于计算机来说，如何克服周期性场景的误匹配也是十分重要的。

另外，计算机处理的数字图像，其亮度值是离散的，只能取某几个值，必定会造成一对多的情况。

为了解决这个情况，通常情况下，双目摄像机是不够的，往往是三目，多目获多目正交获取图像。

实施过程

为了减少周期性误匹配和图像噪音带来的干扰，水平多目立体匹配采用计算图像中连续匹配窗通过图像间的平方差的和来寻找对应点。而对于匹配窗的宽度是十分重要的，其选择的大小会对结果产生重要的影响。

如果窗口过大，会导致信息量过大，而无法满足找到确定匹配位置。极端情况是窗口宽度和图像宽度同样大小，那么匹配点的选取就没有意义。

如果窗口过小，会导致信息量过少，容易产程误匹配，极端情况是，窗口只有 1 个像素宽度，这样导致与水平双目立体匹配无差异，仅仅是多幅图像进行校验而已，容易产生周期性误匹配。

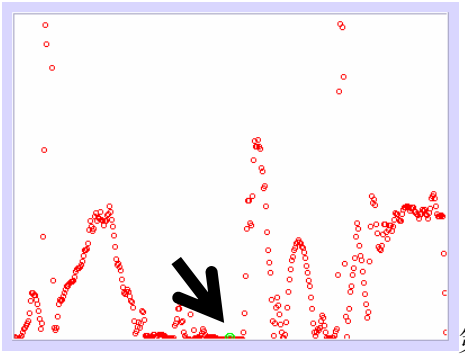
现取以下五幅图片进行立体匹配。

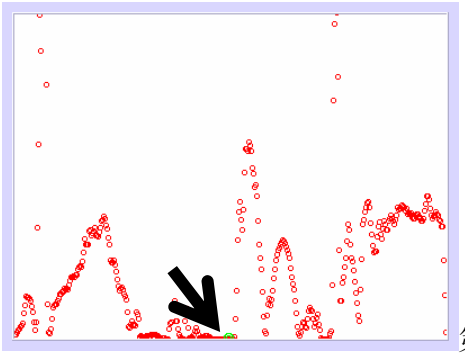
必要假设

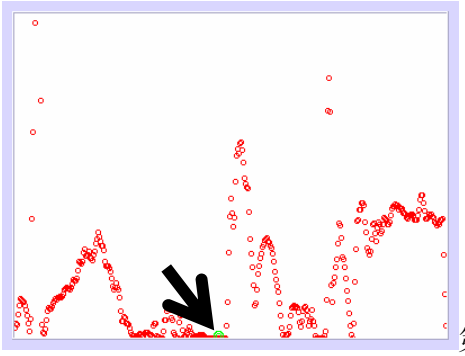
为了简化问题，进行如下假设：

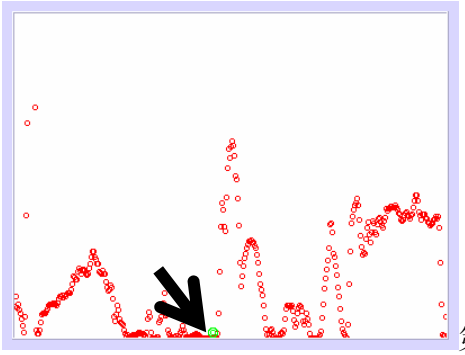
1. 该摄像设备完美无缺，能够清晰地记录场景的细节，不会有杂点。
2. 假设 5 个摄像头之间的距离相同，即向像机的基线为 0、B、2B、3B、4B。

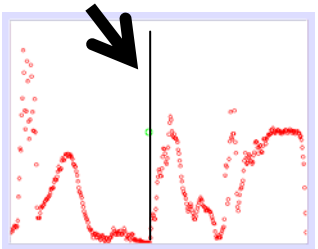
以第一张图为基准，再另外 4 张图中寻找匹配点。以 x:200, y:100 为例。


第二幅图像的匹配点出现位置


第三幅图像的匹配点出现位置


第四幅图像的匹配点出现位置


第四幅图像的匹配点出现位置


由于不知道摄像机的基线和焦距，并且，使用到距离计算离散的数据容易产生误差，这里没有采用到距离的方式寻找匹配点。此图为 4 个数据的 SSSD 假设

fB 为 1。

最终对整幅图像的深度恢复浅色区域为靠前位置，深色区域为远离摄像机位置，获得深度图如下：

图 1 窗宽=3

观察此图可以发现，原图像上较光滑的区域产生了明显的误匹配，书架上放纸的地方也产生了误匹配。

测试其他宽度结果如下：

图 2 窗宽=1

图 3 窗宽=6

图 4 窗宽=15

由比较得知，窗口宽度为 3 时是准确率较高的情况，这是由于场景本身的性质有关。

总结

水平多目方法是对原来图像上一个窗口宽度图像与比较图像上一个随机窗口的期望进行比较，找到最小值，也就是最符合原来窗口中图像的图像，从而找到匹配点。但这带来一个问题。一是精度问题，由于窗口的存在，无法找到具体的对应点，只能说明在这一片区域中是最符合原来亮度分布情况的。二是光滑区域的匹配问题，周期性误匹配能够通过选择合适的窗口来避免，但是光滑区域由于光线的变化不足，导致无法找到确切的匹配点的问题，也是不能得到很好的解决。三是对于镜面反射表面的物体，由于光源与场景之间存在较大距离，摄像机的移动必定会导致物体上反光点的移动，从而造成误匹配。

[参考文献]

1. Okutomi M, Kanade T. 1993. A multiple-baseline stereo. IEEE-PAMI, 15(4):353-363
2. Takeo Kanade. 1995. Development of a Video-Rate Stereo Machine. Proceedings of International Robotics and System Conference August 5-9
3. 宋毅，崔远平，居鹤华. 2006. 一种图像匹配中 SSD 和 NCC 算法的改进. 计算机应用 2005.02:42-44
4. 詹总谦，张祖勋等. 2004. 基于多基线立体匹配技术的三维重建. 地理空间信息 2004 年 12 月(2):17-19

附录：

主要程序清单

```
void window::stereo(void)
{
	int x, y;
	int d;
	const int NW = 6;
	int j;
	int e;
	long long s;
	int mine, maxe;
	mine = 1000;
	int minId;
	maxe = 0;
	int dr;
	int mind, maxd;
	mind = 100000;
	maxd = 0;
	int negCnt = 0;
	int i;

	for (y=0; y<HEIGHT; ++y)
	{
		for (x=0; x<WIDTH; ++x)
		{
			mine = 10000;
			for (d=0; d<WIDTH; ++d)
			{
				e = 0;
				for (i=1; i<5; ++i)
				{
					for (j=0; j<NW; ++j)
					{
						int i0;
						int i1;
						i0 = m_mat[0][y*WIDTH+x+j];
						i1 = m_mat[i][y*WIDTH+d+j];
						e = e + ((i0 - i1) * (i0 - i1));
					}
				}
				if (e < mine)
				{
					minId = d;
					mine = e;
				}
			}
			dr = x - minId;
			if (dr > maxd)
			{
				maxd = dr;
			}
			if (dr < mind)
			{
				mind = dr;
			}
			if (dr < 0)
			{
				dr = 0;
			}
			dr /= 2;
			if (dr > 255)
			{
				dr = 255;
			}
			m_resultPicbox->setPoint(x, y, dr, dr, dr);
		}
	}
}
```