文章编号:1007-2780(2013)03-0450-09

面向低纹理图像的快速立体匹配

张来刚[1,3],魏仲慧[2],何 昕[2],孙 群[1]

(1. 聊城大学 汽车与交通工程学院,山东 聊城 252059,E-mail:zhlgang@163.com;
2. 中国科学院 长春光学精密机械与物理研究所,吉林 长春 130033; 3. 中国科学院大学,北京 100049)

摘 要:立体匹配中,低纹理区域容易产生匹配多义性,导致失配,为了解决低纹理匹配问题,通常采用增大聚合窗口或全局优化算法(如:动态规划算法),但是此类算法会导致边缘处的视差模糊不清,因此,文章提出了一种基于边缘图像的快速立体匹配算法。首先,对立体图像对进行边缘检测和 Sobel 滤波;然后,基于 Sobel 滤波后的图像,先后计算水平聚合代价和垂直聚合代价;最后,利用 WTA(Winner-Take-All)优化算法得到最终视差图。实验中,对视差稠密度和准确度进行了定量分析,左右一致性检验平均达标率超过了 88%,结果表明,本算法使用边缘提取取代图像分割作为代价聚合向导,很好地解决了低纹理区域立体匹配问题,同时大大提高了匹配效率,获得了准确可靠的视差图,达到了自主导航系统的要求。

关 键 词:三维重建;立体匹配;代价聚合;左右一致性检验;Winner-Take-All

中图分类号:TN911.73 **文献标识码**:A **DOI**:10.3788/YJYXS20132803.0450

New Stereo Matching Method Based Edge Extraction

ZHANG Lai-gang[1,3], WEI Zhong-hui[2], HE Xin[2], SUN Qun[1]

(1. *School of Automobile and Transportation Engineering, Liaochen University, Liaocheng* 252059, *China*, *E-mail*: *zhlgang@163.com*;
2. *Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun* 130033, *China*;
3. *University of Chinese Academy of Sciences, Beijing* 100049, *China*)

Abstract: Computing disparity images for stereo pairs of low texture images is a challenging task because matching costs inside low texture areas of the stereo pairs are almost similar. This problem can not be solved straightforwardly by increasing the size of aggregation windows or by using global optimization methods, e. g. dynamic programming, because those approaches will smooth depth discontinued boundaries as well. This paper proposes a new method that is able to robustly perform stereo matching for low texture stereo images. First, edge detection and Sobel filtering are performed for the images; second, edge maps is used to guide the aggregation of pixel matching costs; Finally, disparity computation and left-right validation(≥88%) are used to get the disparity maps. Experimental results demonstrate that the proposed method utilizes the edge maps computed from the stereo pairs to guide the cost aggregation process in stereo matching, and can produce a larger number of and a better accuracy of reliable disparities for low texture stereo images than the moving average method.

Key words: 3D reconstruction; stereo matching; cost aggregation; left-right validation; winner-take-all

收稿日期:2012-03-01; **修订日期**:2012-03-25

基金项目:山东省自然科学基金(No. ZR2012CO026, No. ZR2011EL038)

作者简介:张来刚(1983—),男,山东聊城人,博士,主要从事视觉测量和立体成像方面的研究。

1 引　　言

计算机视觉是从采集的图像或图像序列中获取对三维世界的真实描述[1],目前已被广泛应用于自主导航领域,例如智能运输系统(ITS)和机器人视觉。立体匹配是计算机视觉研究领域的关键技术。立体匹配主要分为以下几个步骤[2]:(1)像素匹配代价计算;(2)匹配代价聚合;(3)视差计算。基于上述应用,立体匹配速度和视差图质量应该被综合考虑。目前,提高立体匹配速度的方法有:(1)降低计算复杂度[3];(2)利用专用处理器优化匹配算法[4]。

自主导航系统采集的图像往往缺乏纹理,为立体匹配带来了很大的不便,针对此类问题,文献[5-6]利用稀疏视差图进行路面估计,然而稀疏视差图的可靠视差点数目不能满足实际需求。为了获得足够多的视差信息,文献[7]提出了一种"准稠密"(Quasi-Dense)立体匹配算法,但是这两种方法在目标没有清晰边缘的情况下所得结果并不理想。

文献[8,9]提出了利用稠密视差图进行道路估计和目标检测,在匹配过程中采用了启发式搜索选择合适的视差,然而这种方法的有效性很大程度上受限于路面上的预定点("Stabilization Ponits"[8]和"Anchor Points"[9])。文献[10]中提出了利用卡尔曼滤波解决低纹理立体匹配问题的可行性,但是还没有实现。

文献[11]为了解决了低纹理立体匹配问题,采用了增大聚合窗的方法,但是景深不连续的边界处变得模糊不清,此方法可用于路面估计,并不适用于目标检测。在基于窗口的立体匹配算法中,窗口大小的选择应该遵循两个原则:(a)选取的窗口要足够大以至于窗口内的像素灰度值具有一定的规律变化或者说纹理性较好;(b)窗口要足够小以至于窗口内的视差没有变化。针对窗口尺寸选择困难这一问题,文献[12,13]提出了一种自适应窗口立体匹配算法,但是该算法的计算速度不能满足自主导航系统的需求。

近几年,基于图像分割的立体匹配算法被提出[14],获得了能与自适应窗口算法相媲美的匹配结果。以分割后的图像区域作为代价聚合时的导向(约束),位于聚合窗口中心点所属区域以外的像素通过预定的权值降低了对聚合代价的影响。然而,该算法在图像分割时耗费了大量时间。

本文所提出的匹配算法以边缘图(Edge Map)作为代价聚合的导向,其优点是不仅对低纹理图像立体匹配具有鲁棒性,同时节省了图像分割的耗时,其视差图的质量能与文献[14]相媲美。此外,本方法的计算复杂度与聚合窗口的大小无关。本文利用来自于自主导航系统的公开数据库图片对本算法进行了验证,结果表明,与文献[2,15]相比,本算法可以获得更稠密、更准确可靠的视差信息。

2 基于边缘图的立体匹配系统

本算法首先对图像预处理和边缘检测,并计算像素匹配代价;然后在水平和垂直两个方向进行代价聚合;最后利用 WTA 优化算法和左右一致性检验得到最终的视差图,系统框架如图 1 所示。

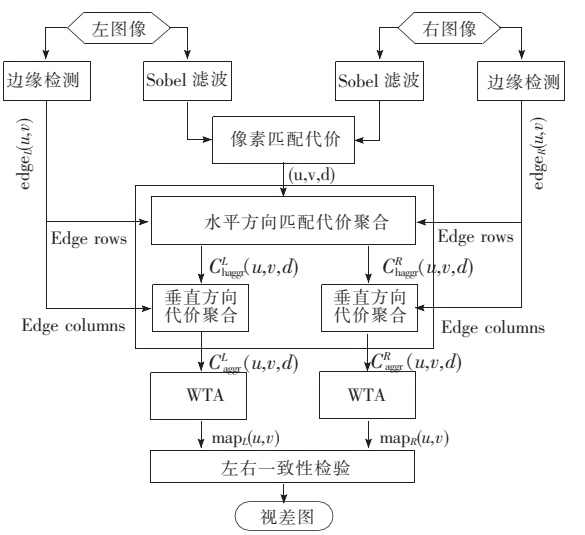

图 1　基于边缘图的立体匹配系统框图

Fig. 1　Proposed stereo matching system

由于自主导航系统采集的图像序列含有大量的噪声、对比度低并且缺乏纹理,因此,在立体匹配之前要对图像进行预处理。由文献[16]可知,Sobel 滤波是一种简单而有效地图像纹理增强方法,本算法采用 Sobel 算子在水平方向上对图像预处理[17]。

接下来,利用像素代价函数估计两个像素之间的相似程度,常用的几种代价函数有[18-20]:像素绝对差异(SAD)、截断像素绝对差异(STAD)和归一化互相关(NCC)。一个好的代价函数应

该具备两个条件:(a)能够有效度量像素间的相似度;(b)计算简单。SAD 算法简单,但是对噪声敏感,噪声像素点的高匹配代价破坏了聚合窗口内的匹配代价均值;STAD 可以减少噪声的影响,但是截断阈值的选择非常困难;NCC 克服了噪声影响,但是由于其采用了开平方运算,所以耗时巨大。

为了克服上述代价函数的缺陷,本文定义了一种新的像素代价函数如式 1 所示,代价函数与 NCC 相似,不同的是没有开平方运算。其中 $\boldsymbol{g}_p=[g_p^r,g_p^g,g_p^b]^T$ 和 $\boldsymbol{g}_q=[g_q^r,g_q^g,g_q^b]^T$ 分别为左右图像像素点 p,q 经 Sobel 滤波后的结果,$u\in[0,U-1]$,$v\in[0,V-1]$,$d\in[0,D-1]$(图像的大小为 $U\times V$,最大搜索视差为 D),输出定义为 $cost_{pix}^L(u,v,d)$ 和 $cost_{pix}^R(u,v,d)$,分别表示从左向右像素匹配代价(left-to-right pixel matching cost)和从右向左像素匹配代价(right-to-left pixel matching cost),两者的关系为:$cost_{pix}^R(u,v,d)=cost_{pix}^L(u+d,v,d)$。

$$f(p,q)=1-\frac{g_p^r g_q^r+g_p^g g_q^g+g_p^b g_q^b}{(|g_p^r|+|g_p^g|+|g_p^b|)+(|g_q^r|+|g_q^g|+|g_q^b|)}\simeq 1-\cos(\boldsymbol{g}_p,\boldsymbol{g}_q) \tag{1}$$

本算法与其他立体匹配算法的主要不同之处在于,利用了边缘图(Edge Map)作为匹配代价聚合的导向,以聚合代价 $cost_{aggr}^L(u,v,d)$ 和 $cost_{aggr}^R(u,v,d)$ 分别计算左右视差图。在进行视差选择时,采用了 WTA(Winters-Take-All)算法。下面分别对边缘检测、代价聚合和 WTA 算法作详细的介绍。

2.1 边缘检测

基于图像分割的自适应窗口立体匹配算法已经获得了比较理想的匹配效果,但是图像分割需要一个良好的分割方法,而且在图像分割的过程中耗费了很多时间[14]。该基于边缘图的立体匹配算法用边缘提取取代了图像分割作为代价聚合的导向,而且经典的边缘检测算法(如:LOG,Sobel 和 Canny 等)使边缘提取比图像分割更加高效。

文献[2,15]的立体匹配算法只利用从左向右聚合代价就可以获得左视差图和右视差图,然而

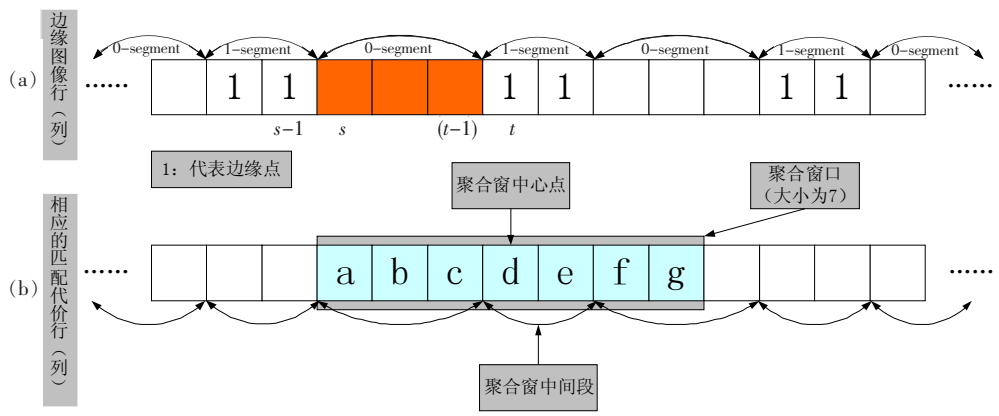

图 2 (a)边缘图像行(列)的区域划分;(b)相应的匹配代价行、聚合窗中心点和中间段;(c)聚合算法实例.

Fig. 2 (a) Segments defined by an edge row/column. (b) Cost row/column corresponding to the edge row/column given in (a), window's mid-point and the window's mid-segment for a window. (c) An example of the proposed aggregation method.

本算法的从左向右聚合代价是在左边缘图的指导下获得的,只能得到左视差图,所以左右原始图像都要进行边缘检测。文献[14]定义位于联合域中同一模式点相关联的所有像素为一个区域,本文定义边缘图行或列上的一组连续像素作为一个区域,如图 2(a)所示,边缘图像行(列)的第 $s-1$ 到第 s 像素值有跳变(1→0),第 $t-1$ 到第 t 像素值有跳变(0→1),而第 s 到第 $t-1$ 像素值没有变化(恒为 0),则 $s\rightarrow t-1$ 被定义为一个区域(0-segment)。

2.2　图像匹配

2.2.1　水平聚合

生成水平方向聚合匹配代价需要两个输入量:像素匹配代价行和相应的左右边缘图像行,两者的对应关系如图 3(a)所示。本算法只使用 $cost^{L}_{pix}(u,v,d)$ 计算聚合代价,以下统称为 $cost(u,v,d)$。水平聚合从像素代价行的左侧至右侧依次完成聚合,假设 $cost(u=r,v=v_i,d=d_j)$ 为将要被聚合窗覆盖的下一个像素匹配代价,那么下一个聚合窗在代价行 $cost(u,v,d)$ 所覆盖的范围是从 $cost(u=r-w+1,v=v_i,d=d_j)$ 到 $cost(u=r,v=v_i,d=d_j)$,其相应的在左右边缘图的聚合窗位置分别为 $edge_L(u=r-W+1,v=v_i)\rightarrow edge_L(u=r,v=v_i)$ 和 $edge_R(u=r-W+1-d_j,v=v_i)\rightarrow edge_R(u=r-d_j,v=v_i)$。

图 3(b)出示了本聚合算法的数据结构,像素代价行、左右边缘图行和左右聚合代价行分别存储于数组 cost_in,l_edge,r_edge,l_cost_out 和 r_cost_out。buffer[l]→buffer[r−1]存储当前窗口下的像素匹配代价,聚合代价存储于 buffer[a]中,当窗口向右移动时,buffer[r]存储将要进入聚合窗的新像素匹配代价。在左右边缘图中,聚合窗可能会覆盖几个不同的区域,如图 2(b)所示(覆盖了 3 个区域),为了下一步的聚合运算,需要对不同的区域进行标记(索引),与之相对应地,图 4 中的 l_sl→l_sr 和 r_sl→r_sr 分别对聚合窗覆盖区域进了标记。左(右)边缘图每个区域的像素数目和相应区域的像素匹配代价和分别存储于 l_scount(r_scount)和 l_scost(r_scost),数组 l_label(r_label)表示左(右)边缘图像素与其所属区域的映射关系,并定义聚合窗口中心点像素所属区域为中间区域(mid−segment)。

水平聚合匹配代价由两部分组成:(a)窗口中间区域(mid−segment)的匹配代价平均值;(b)中间区域以外的匹配代价加权平均值。图 2(c)是本聚合方法的一个实例,聚合窗口大小为 7,中间段像素数目为 2,中间段以外的像素匹配代价与加权系数 alpha(0≤alpha≤1)相乘,降低了其对最终匹配代价的贡献。水平聚合之前需要进行以下初始化工作:

(1)l_sl=l_sr=0;

(2)l_scost[0]等于窗口最左端区域像素匹配代价和;

(3)l_scount[0]=W,W 为聚合窗口大小;

(4)设聚合窗口最左端区域像素的索引值为 0;

同理,r_sl,r_sr,r_scost,r_scount 和 r_label 以同样的方式被初始化。基于以上数据结构,本聚合算法的核心程序如表 1(a)所示,表 1(b)是相应注释。

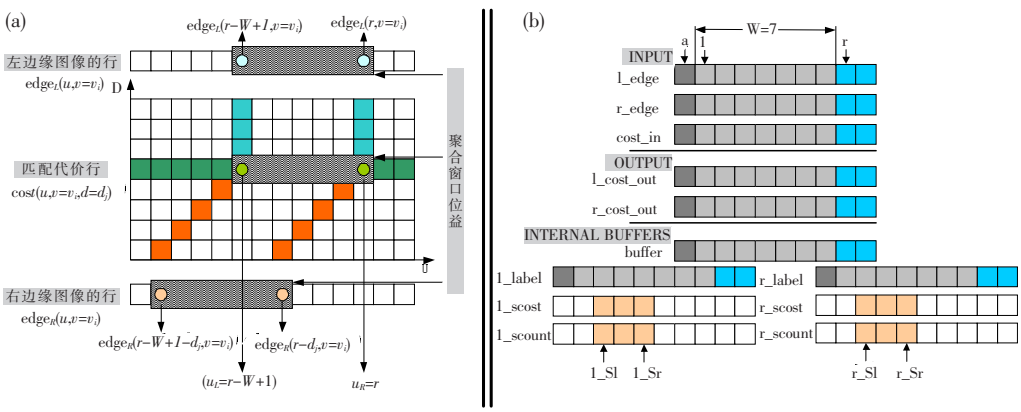

图 3　(a)像素匹配代价行与左右边缘图像行的对应关系;(b)聚合代价算法的数据结构.

Fig. 3　(a) Correspondence between cost rows and edge rows of the left and the right edge map. (b) Data structures used for the proposed aggregation method.

表 1 (a)基于边缘图的代价聚合程序;(b)代价聚合程序的相应注释

Table 1 (a) Cost aggregation guided by edge maps; (b) Explanation of the proposed aggregation method

(a)

```
(a) Add new cost on the rightmost side of the window:
 1.  buffer[r] = cost_in[r];
 2.  aggr_cost = buffer[a] + cost_in[r];

 3.  enter_segment_on_left = l_edge[r-1] ^ l_edge[r];
 4.  if(enter segment on left){
 5.    l_sr= l_sr + 1;
 6.    l_scost[l_sr] =cost_in[r];
 7.    l_scount[l_sr]=1;
 8.   }
 9.  else {
10.    l_scost[l_sr]+= cost_in[r];
11.    l_scount[l_sr] +=1;
12.   }
13.  l_label[r] = l_sr;

14.  enter_segment_on_right = r_edge[r-1-d] ^ r_edge[r-d];
15.  if(enter_segment_on_right){
16.    r_sr = r_sr + 1;
17.    r_scost[r_sr]=cost_in[r];
18.    r_scount[r_sr]=1;
19.   }
20.  else {
21.    r_scost[r_sr]+= cost_in[r];
22.    r_scount[r_sr] +=1;
23.   }
24.    r_label[r] = r_sr;
(b) Remove the element that the window has left:
25.  leave_segment_on_left = l_edge[r-W] ^ l_edge[r-W+1];
26.  if(leave_segment_on_left) l_sl = l_sl +1;
27.  else{ l_scost[l_sl]-= buffer[l]; l_scount -=1; }

28.  leave_segment_on_right = r_edge[r-W-d] ^ r_edge[r-W+1-d];
29.  if(leave_segment_on_right) r_sl = r_sl +1;
30.  else{ r_scost[r_sl]-= buffer[l]; r_scount -=1;}

31.  buffer[l] = aggr_cost - buffer[l];
(c) Compute the average cost for the window:
32.  mid = r - (W DIV 2); seg = l_label[mid];
33.  if((W-l_scount[seg]) != 0){
34.    l_cost_out[mid] = l_scost[seg]/l_scount[seg] +
       alpha*(buffer[l] - l_scost[seg])/(W-l_scount[seg]);
35.   }else l_cost_out[mid] = l_scost[seg]/l_scount[seg];

36.  mid = r - (W DIV 2) - d; seg = r_label[mid];
37.  if((W- r_scount[seg]) != 0){
38.    r_cost_out[mid] = r_scost[seg]/r_scount[seg] +
       alpha*(buffer[l] - r_scost[seg])/(W-r_scount[seg]);
39.   }else r_cost_out[mid] = r_scost[seg]/r_scount[seg];
(d.) Move the window to right:
40.  a++; l++; r++
```

(b)

```
(a.) Add new cost on the rightmost side of the window:
1-2.   Add new cost to the previous accumulated value

3-13.  if new cost lies at the beginning pixel of a segment on left edge map
         Make one new segment at (l_sr + 1)
         Set new segment's cost = new cost
         Set new segment's number of pixels = 1
       else
         Accumulate new cost to the rightmost segment indexed by l_sr
         Increase the total number of pixels of the rightmost segment by 1
       end if
       Fill the mapping from pixel l to segment for the left image

14-24. if new cost lies at the beginning pixel of a segment on right edge map
         Make one new segment at (r_sr + 1)
         Set new segment's cost = new cost
         Set new segment's number of pixels = 1
       else
         Accumulate new cost to the rightmost segment indexed by r_sr
         Increase the total number of pixels of the rightmost segment
       end if
         Fill the mapping from pixel l to segment for the right image

(b.) Remove the element (called "-element" that the window has left:
25-27. if the window has left the leftmost segment identified by l_sl
         Remove the segment by increasing l_sl by 1
       else
         Remove L-element from the segment identified by l_sl
         Decrease the total number of pixels of the leftmost segment by 1
       end if

28-30. if the window has left the leftmost segment identified by r_sl
         Remove the segment by increasing r_sl by 1
       else
         Remove L-element from the segment identified by r_sl
         Decrease the total number of pixels of the leftmost segment by 1
       end if

31.    Remove L-element from the current window
       Save the accumulated value for the current window at buffr[l]

(c.) Compute the average cost for the window:
32-35. The aggregated cost of the current window for left-to-right volume =
         Average cost of the mid-segment +
         alpha*(Average cost of pixels outside of the mid-segment)

36-39. The aggregated cost of the current window for right-to-left volume =
         Average cost of the mid-segment +
         alpha*(Average cost of pixels outside of the mid-segment)
(d.) Move the window to right:
40.    Move the window to right by increasing a, l, and r by 1
```

2.2.2 垂直聚合

垂直聚合是对水平聚合代价在垂直方向上再次聚合,如图 1 所示,垂直聚合分别对 $cost^{L}_{haggr}(u,v,d)$ 和 $cost^{R}_{haggr}(u,v,d)$ 聚合,聚合的过程中分别以左边缘图列 $edge_L(u,v)$ 和右边缘图列 $edge_R(u,v)$ 作为聚合的导向。换言之,与聚合匹配代价列 $cost^{L}_{haggr}(u=u_i,v,d=d_j)$ 相对应的向导边缘列为 $edge_L(u=u_i,v)$,而与聚合匹配代价列 $cost^{R}_{haggr}(u=u_i,v,d=d_j)$ 相对应的向导边缘列为 $edge_R(u=u_i-d_j,v)$。

垂直聚合与水平聚合的计算过程相似,以对 $cost^{L}_{haggr}(u,v,d)$ 的垂直聚合为例,$cost^{L}_{haggr}(u=u_i,v,d=d_j)$ 和 $edge_L(u=u_i,v)$ 分别被存储于数组 cost_in,l_edge,执行过程对应于表 1(a)的 3~13 行,25~27 行和 32~35 行,但是,由于匹配代价聚合列与相应的边缘图向导列位于同一条垂直线上,不存在视差偏移(左右图像标定完毕,不存在垂直视差),所以,程序 14 行聚合窗进入新区域条件和 28 行聚合窗离开一个区域的条件应分别更改为:

enter_segment_on_right=r_edge[$r-1$]^r_edge[r]和 leave_segment_on_right=r_edge[$r-w$]^r_edge[$r-w+1$]。

2.2.3 视差图计算和左右一致性检验

本算法的视差优化策略采用了 WTA 算法,与全局优化算法相比,本算法计算简单,执行效率高。WTA 算法为每一个像素在视差搜索范围内搜索最小聚合匹配代价,与其相对应的视差值即为该像素的最优视差,如式(2)、式(3)。

$$map_L(u,v) = \arg\min cost^{L}_{aggr}(u,v,d) \quad (2)$$

$$map_R(u,v) = \arg\min cost^{R}_{aggr}(u+d,v,d) \quad (3)$$

左右一致性检验用来检测左右图像的公共目标点,剔除错误匹配,从而确定具有高置信度的匹配点,本算法中,左右一致性检验所遵循的原则如表达式(4)。

$$\mathrm{map}_{\mathrm{L}}(u,v)=\begin{cases}\mathrm{map}_{\mathrm{L}}(u,v) & \text{if } \mathrm{map}_{\mathrm{L}}(u,v)=\mathrm{map}_{\mathrm{R}}(u-\mathrm{map}_{\mathrm{L}}(u,v),v)\\ 0 & \text{else}\end{cases} \tag{4}$$

3　实验结果及分析

试验中采用了两种不同图像序列：真实场景图像序列和虚拟场景图像序列。真实场景图像序列共 864 对彩色立体图像对，图像大小为 320×240，大部分图像中的道路区域都缺乏纹理。虚拟场景图像序列有 200 对灰度立体图像对，图像大小为 512×512，其纹理与真实场景图像相比更丰富一些。

我们将本立体匹配算法与文献[2,15]中的算法（参考算法）进行了比较，两算法的参数设置如表 2 所示。图 4(c)、(g)、(k)是由参考算法匹配

表 2　匹配算法参数设置

Table 2　Parameters used for computing disparity images

	本算法	参考算法
真实场景图像	边缘检测：LOG	像素代价函数：SAD
	边缘检测阈值：0.002	代价聚合方法：基于聚合窗代价均值的匹配算法
	聚合窗大小：11×181……	聚合窗大小：11×11
	Alpha：0.2	视差优化算法：WTA
虚拟场景图像	边缘检测：LOG	像素代价函数：SAD
	边缘检测阈值：0.002	代价聚合方法：基于聚合窗代价均值的匹配算法
	聚合窗大小：21×501	聚合窗大小：11×11
	Alpha：0.2	视差优化算法：WTA

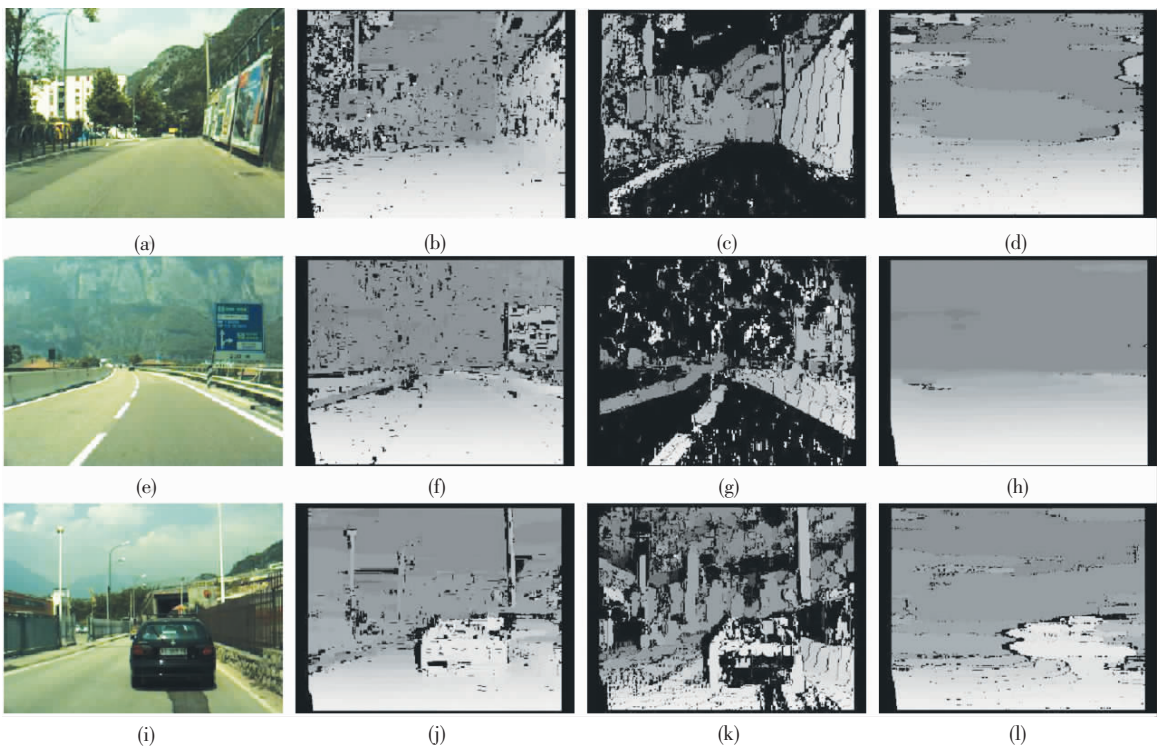

图 4　真实场景图像序列的立体匹配实例.(a,e,i)：左图像；(b,f,j)：本算法视差图；(c,g,k)：参考算法视差图；(d,h,l)：参考算法采用本算法的匹配代价函数和聚合窗匹配视差图.

Fig. 4　Examples of disparity images computed from the real sequence. (a, e, i): Left images. (b, f, j): Disparity images computed by the proposed method. (c, g, k): Disparity images computed by the referred method. (d, h, l): Disparity images computed by the referred method with the cost function and the window size used for the proposed method.

得到的视差图，纹理缺乏的道路区域匹配效果不佳；图 4(d)、(h)、(l)是参考算法采用本算法的匹配代价函数和聚合窗匹配得到的视差图，低纹理区域的立体匹配问题得到了解决，但是深度不连续边界和目标边缘变得模糊了；利用本算法匹配得到了视差图如图 4(b)、(f)、(j)所示，既解决了低纹理匹配问题，又很好地保持了目标边缘的清晰度。由于真实场景图像没有标准视差图，所以本实验使用视差稠密度来衡量算法的有效性，符合左右一致性检验的比率越高，视差越稠密。与参考算法相比，得到的视差图更加稠密，相邻帧之间的视差稠密度变化更小，如图 5 所示。

在对虚拟场景图像的匹配试验中，由于纹理比较丰富，本算法与参考算法都能获得比较稠密的视差图像，但是本算法对低纹理区域的匹配效果更好，如图 6(b)与(c)和(f)与(g)相比较所示。图 6(d)、(h)是参考算法采用本算法的匹配代价函数和聚合窗匹配得到的视差图。在此，除了使

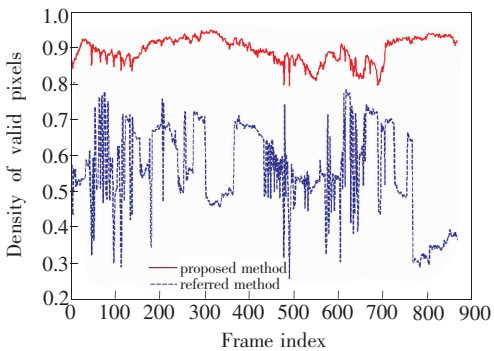

图 5 真实场景视差图稠密度比较

Fig. 5 Density comparison for disparity images computed from the real stereo image sequence

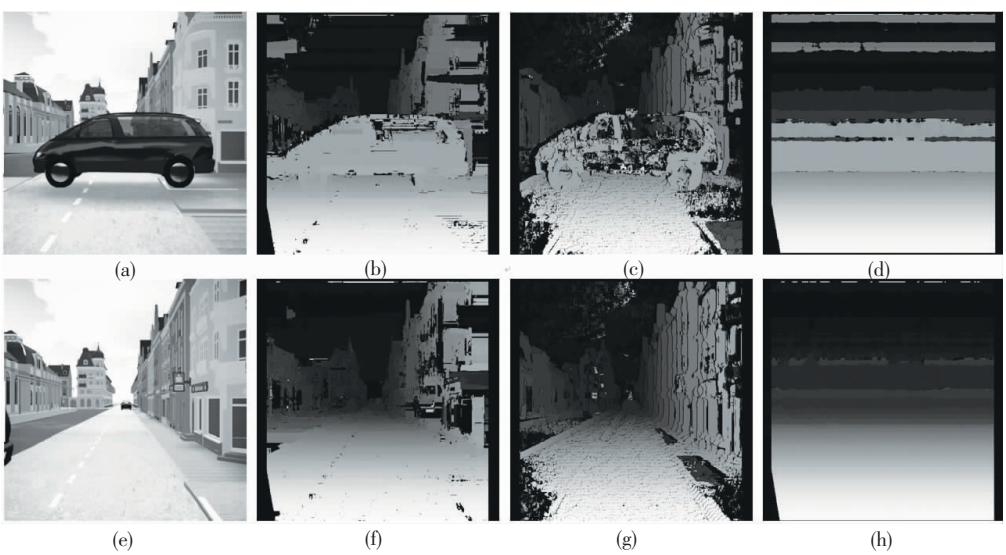

图 6 虚拟场景图像序列的立体匹配实例.(a,e):左图像;(b,f):本算法视差图;(c,g):参考算法视差图;(d,h):参考算法采用本算法的匹配代价函数和聚合窗匹配视差图.

Fig. 6 Examples of disparity images computed from the artificial sequence. (a, e): Left images. (b, f): Disparity images computed by the proposed method. (c, g): Disparity images computed by the referred method. (d, h): Disparity images computed by the referred method with the cost function and the window size used for the proposed method.

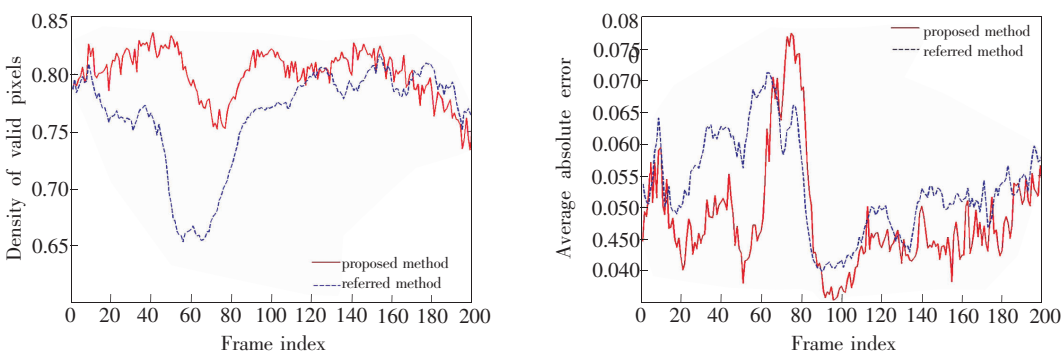

图 7 虚拟场景视差图稠密度和精度比较

Fig. 7 Density and accuracy comparison for disparity images computed from the artificial stereo image sequence

用稠密度衡量视差图质量，我们还利用了 $E_{abs}=\frac{1}{U\times V}\sum_{v=0}^{V-1}\sum_{u=0}^{U-1}|g(u,v)-d(u,v)|$ 来衡量视差图的准确度，式中 $g(u,v)$ 为真实视差，$d(u,v)$ 为本算法（或参考算法）匹配得到的视差，图像大小为 $U\times V$。计算结果如图 7 所示，与参考算法比较，本算法得到了更加稠密、准确的视差图，第 70 帧到第 90 帧出现了比较大的误差，出现这种现象的原因是图像序列中的汽车离开了视场。

4　总结与展望

提出了一种新的基于边缘提取的立体图像匹配算法，对于低纹理图像的匹配有很好的鲁棒性，很好地保留了公路目标，甚至像电线杆、路灯之类小目标的边缘清晰度，此外，在代价聚合过程中，计算复杂度与聚合窗的大小无关，并且以边缘图作为聚合向导，与基于区域分割的匹配算法相比，由于边缘提取比图像分割更加高效，匹配效率大大提高。

本文证明了利用边缘图指导匹配代价聚合的可行性，为基于边缘的立体匹配算法的研究做了一个良好的铺垫。在以后的工作中，准备提出一种新的能够融合图像颜色和梯度信息的匹配代价函数，并进一步利用专用处理器优化聚合算法[4]。

参　考　文　献：

[1] 何友兵，李大海，李良玉，等. 彩色立体图像匹配算法研究 [J]. 液晶与显示，2007，22(4)：418-422.

[2] Scharstein D, Szeliski R. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms [J]. *International J. Computer Vision*, 2002, 47(1/2/3): 7-42.

[3] Sach L T, Hamamoto K, Atsuta K, *et al*. A new coarseto-fine method for computing disparity images by sampling disparityspaces [J]. *Trans. on Electronics, Information and Systems*, 2009, 129(1): 103-111.

[4] W. vander Mark, Gavrila D M. Real-time dense stereo for intelligent vehicles [J]. *Trans. on Intelligent Transport Systems*, 2006, 7(1):38-50.

[5] Se S, Brady M. Ground plane estimation, error analysis and applications [J]. *Trans. on Robotics and Autonomous Systems*, 2002, 39(2):59-71.

[6] Labayrade R, Aubert D, Tarel J P. Real time obstacle detection in stereo vision on non-flat road geometry through v-disparity representation [J]. *Proc. of Intelligent Vehicle Symposium*, 2002, (2):646-651.

[7] Hautiere N, Labayrade R, Perrollaz M, *et al*. Road scene analysis by stereovision: a robust and quasi-dense approach [C]// *Proc of Control, Automation, Robotics and Vision*, Singapore:IEEE,2006:1-6.

[8] Chumerin Nikolay, Van Hulle M M. Ground plane estimation based on dense stereo disparity [C]// *The Fifth International Conference on Neural Networks and Artificial Intelligence*, Minsk, Belarus,2006:1-6.

[9] Zanin M. Localization of ahead vehicle with on-board stereo cameras [C]//*Proc. of Image Analysis and Processing*, Modena:IEEE,2007:111-116.

[10] Suganuma N, Fujiwara N. An obstacle extraction method using virtual disparity image [C]// *Proc. of IEEE Intelligent Vehicles Symposium*, Istanbl, Turkey:IEEE,2007:456-461.

[11] Zhao Z, Katupitiya J, Ward J. Global correlation based ground plane estimation using v-disparity image [C]// *Proc. of Robotics and Automation*, Roma, Italy:IEEE, 2007: 529-534.

[12] *Kanade T, Okutomi M. Stereo matching algorithm with an adaptive window: theory and experiment* [J]. Trans. on Pattern Analysis and Machine Intelligence, 1994, 16(9):920-932.

[13] 尹传历，向长波，宋建中，等. 一种基于自适应窗口和图切割的快速立体匹配算法 [J]. 光学 精密工程，2008，16(6)：1117-1121.

[14] *Gerrits M, Bekaert P. Local stereo matching with segmentation based outlier rejection* [C]// Proc. of Computer and Robot Vision, IEEE,*Canadian:IEEE*, 2006:66-72.

[15] *Muhlmann K, Maier D, Hesser R*, et al. *Calculating dense disparity maps from color stereo images, an efficient implementation* [C]// Proc. of Stereo and Multi-Baseline Vision, IEEE,*Kauai:IEEE*,2001:30-36.

[16] *Broggi A, Caraffi C, Fedriga R I*, et al. *Obstacle detection with stereo vision for off-road vehicle navigation* [C]//Proc. of Computer Vision and Pattern Analysis, IEEE, *San Diego,USA:IEEE*,2005:65-72.

[17] 吴晖辉,张宪民,洪始良. 综合边缘和颜色特征的 IC 类贴装器件的检测 [J]. 光学 精密工程,2009,17(9):2283-2290.

[18] Bank Jasmine, Corke Peter. Quantitative evaluation of matching methods and validity measures for stereo vision [J]. *Trans. on International Journal of Robotics Research*, 2001, 20(7):512-532.

[19] 杨化超,张书毕,张秋昭. 基于 SIFT 的宽基线立体影像最小二乘匹配方法 [J]. 测绘学报,2010,39(2):187-194.

[20] Liang W, Mingwei G, Minglun G, *et al*. How far we can go with local optimization in real-time stereo matching-a performance study on different cost aggregation approaches [C]// *Proc. of 3D Data Processing, Visualization, and Transmission, IEEE*, ChapelHill:IEEE,2006:129-136.

SI 单位的倍数单位

SI 单位的倍数单位是由 SI 词头与 SI 基本单位和 SI 导出单位构成。倍数单位的选取一般应量的数值处于 0.1～1 000 之间。

• SI 词头是用于构成倍数单位的因数符合,不能单独使用。

• 词头符号与所紧接着的单位符号应作为一个整体对待,它们共同组成一个新单位(十进倍数或分数单位),并具有相同的幂次。例如:10 $hm^2=10\times(100\ m)^2=10^5 m^2$;1 $cm^3=(10^{-2}m)^3=10^{-6}m^3$。

• 词头不能重叠使用,即不能使用由几个词头并列组成的组合词头。例如:不能用 mμs(毫微秒),正确的为 ns(纳秒)。

• 摄氏度、平面角和时间单位(s 除外)及 kg 等不得使用词头的倍数单位。例如:“mh”(毫小时),“hkg“(百公斤)是不规范的。