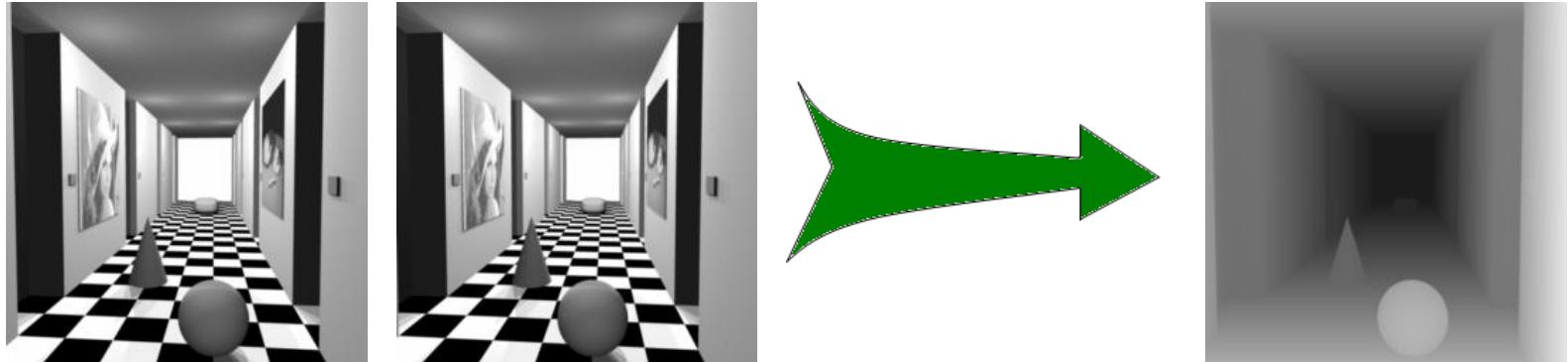
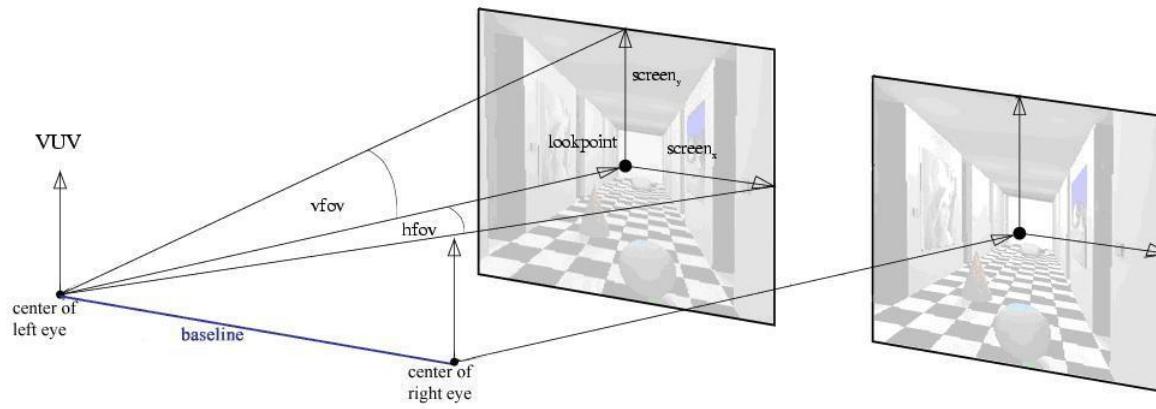


吉祥

# Reconstruction



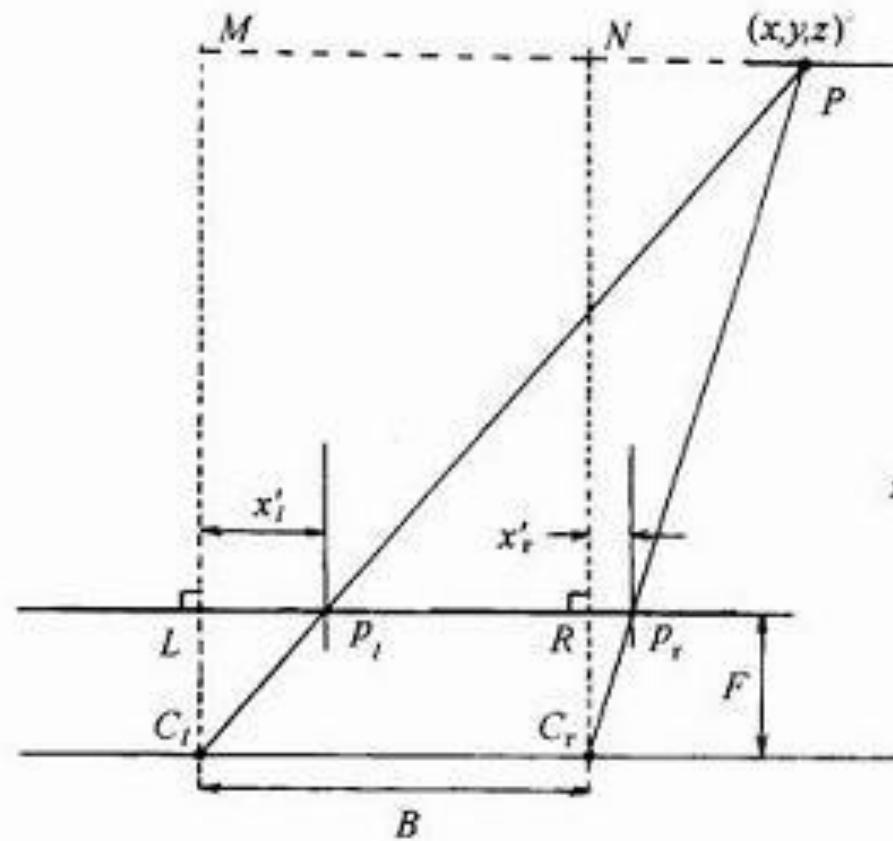
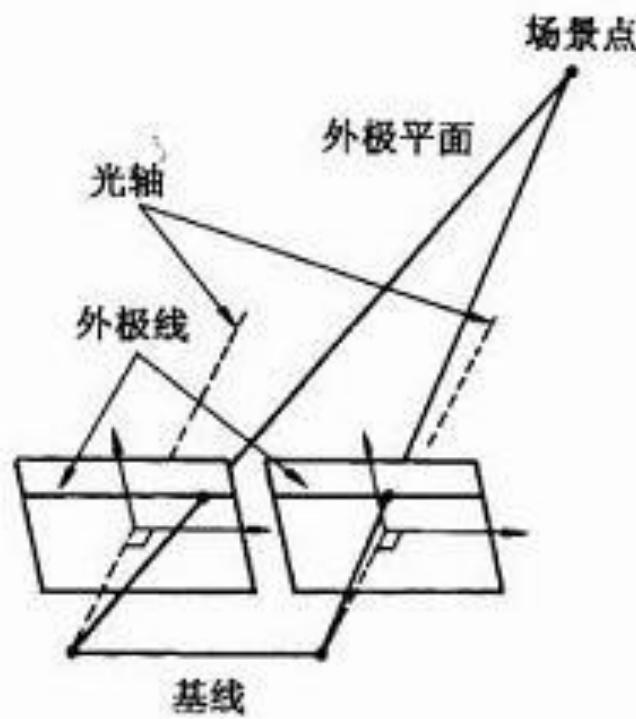
# 立体视觉



# 立体视觉

- 我们将场景中同一点在两个不同图像中的投影点称为**共轭对**
- 其中的一个投影点是另一个投影点的**对应**(correspondence)
- 两幅图像重叠时的共轭对点的位置之差(共轭对点之间的距离)称为**视差**(disparity)
- 通过两个摄像机中心并且通过场景特征点的平面称为**外极**(epipolar)平面
- 外极平面与图像平面的交线称为**外极线**.

# 立体成象



## 立体成象

由相似三角形可得

$$\frac{x}{z} = \frac{x'_l}{F}$$

$$\frac{x - B}{z} = \frac{x'_r}{F}$$

合并两项，可得

$$z = \frac{BF}{x'_l - x'_r}$$

F是焦距，B是基线距离

# 立体成象

- 因此，各种场景点的深度恢复可以通过计算视差来实现。
- 大角度立体方法 —— 提高场景点深度计算精度的有效途径

主要的问题有：

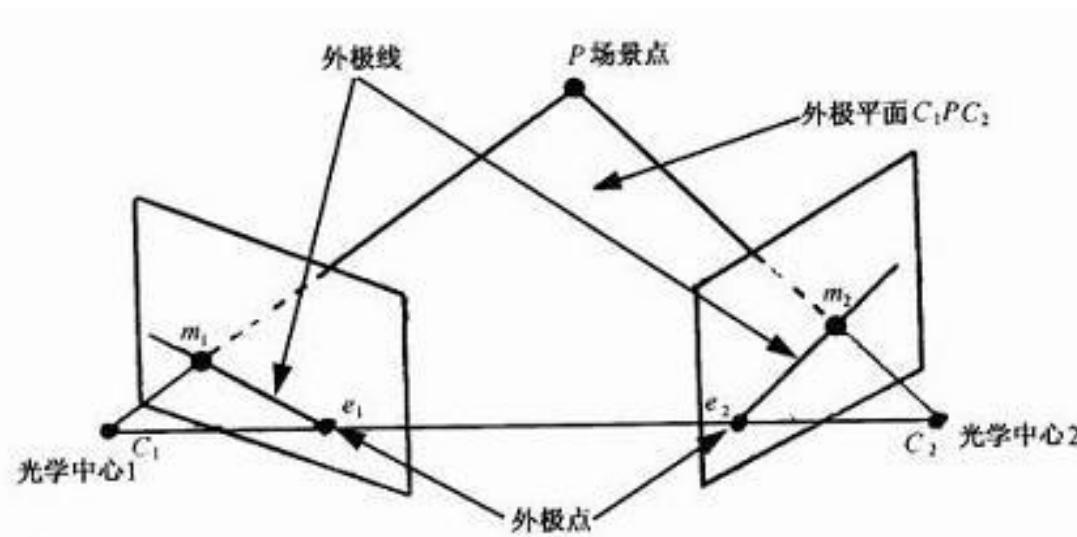
随着基线距离的增加，两个摄像机的共同的可视范围减小

场景点对应的视差值增大，则搜索对应点的范围增大，出现多义性的机会就增大。

由透视投影引起的变形导致两个摄像机获取的两幅图像中不完全相同，这就给确定共轭对带来困难。

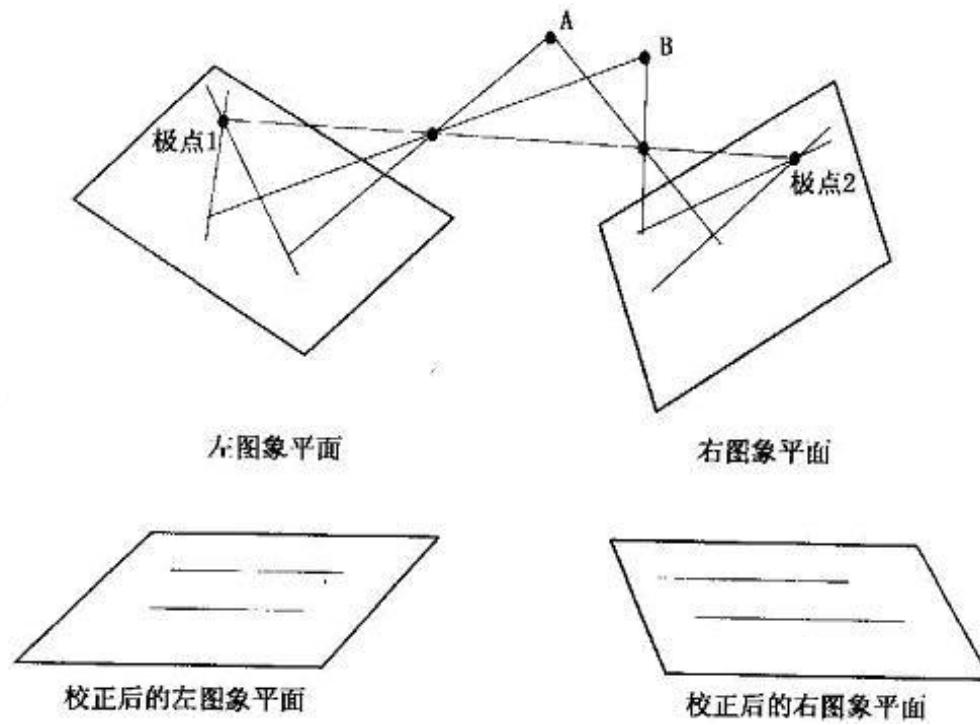
## 立体成像的一般情况

- 在实际中，两条外极线不一定完全在一条直线上，即垂直视差不为零
- 两个摄像机的光轴不平行



# 立体校正

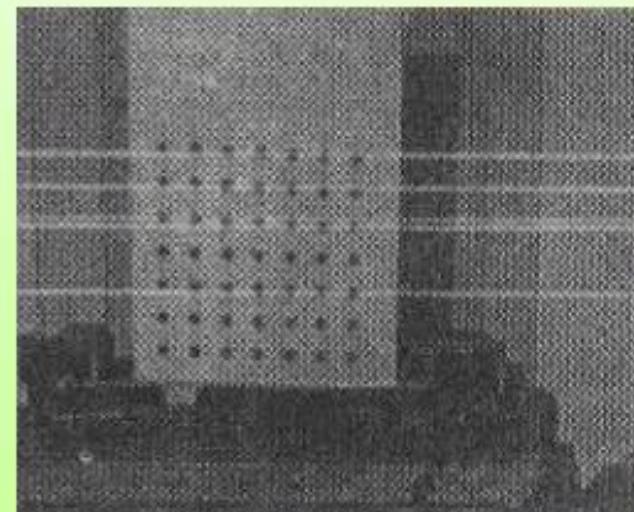
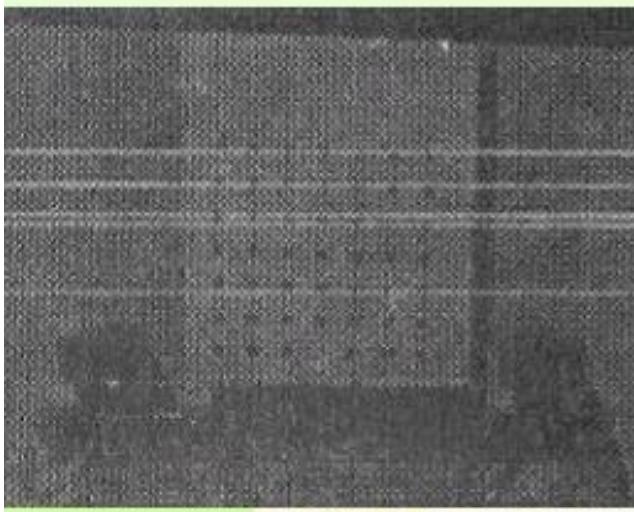
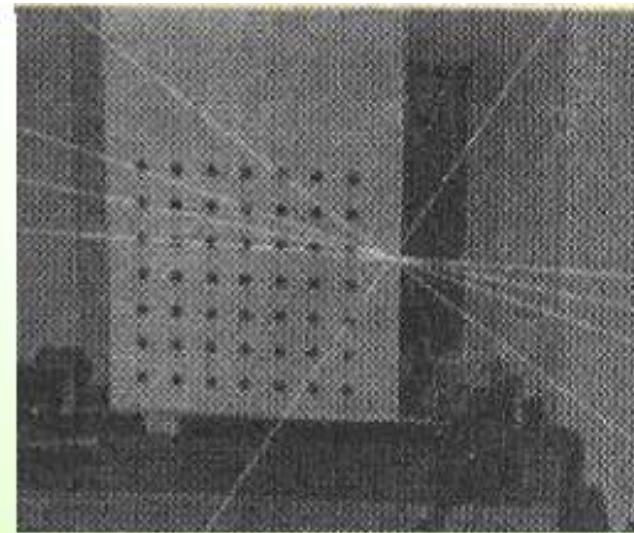
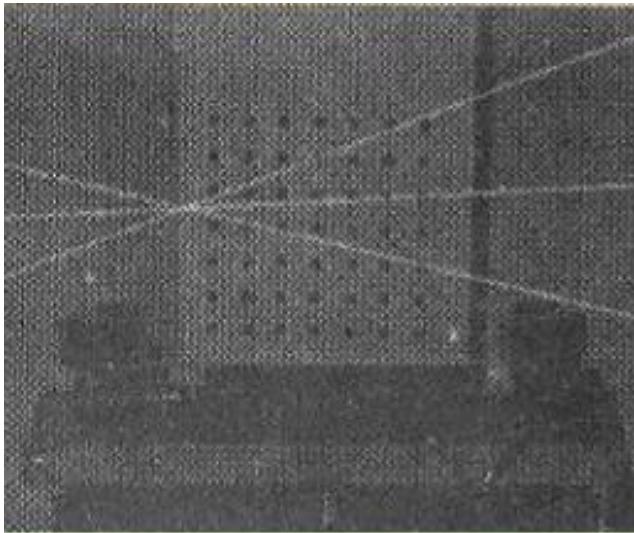
- 立体图像对重新取样，使外极线对应于图像阵列的行



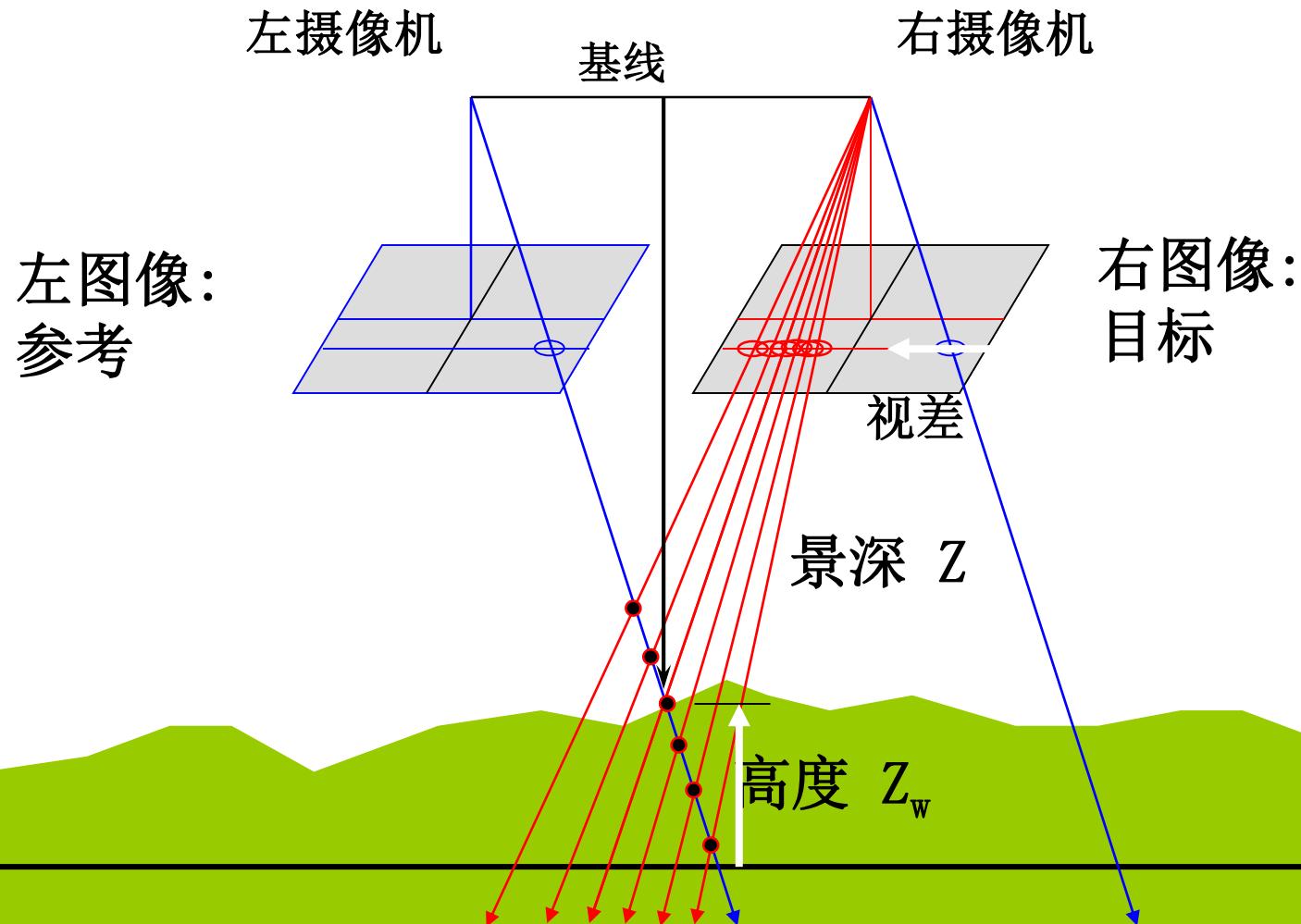
## 立体图像校正

将两图像投影到一个平面上就能得到理想的极线几何。左(右)摄像机中的每一个像素点分别对应于左(右)摄像机坐标系统中的一条射线。

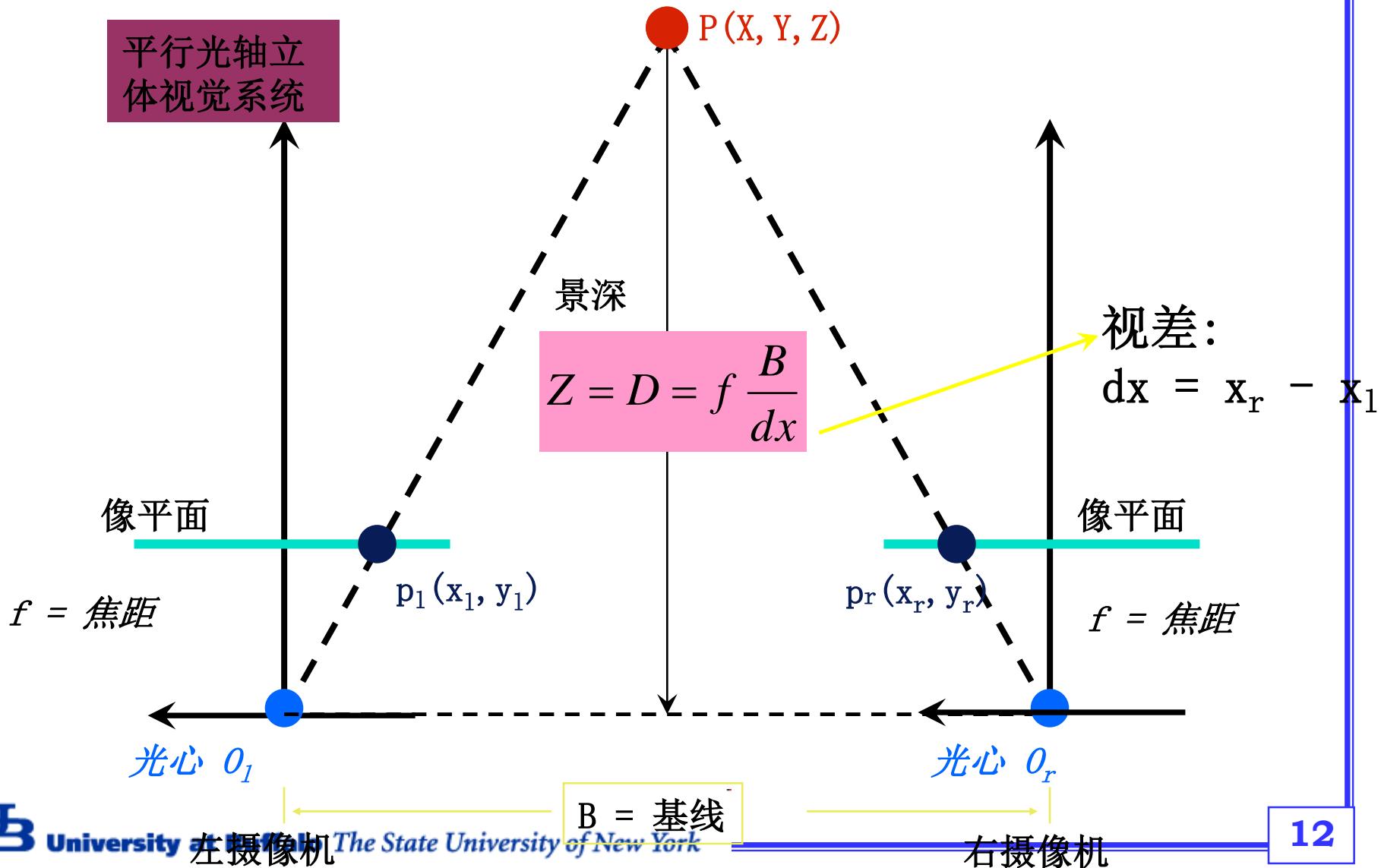
设 $T_1$ 和 $T_r$ 分别表示将左、右摄像机的射线变换到公共平面坐标系的刚体变换，确定每个图像的顶点在公共平面上的位置，创建新的左、右图像网格，将每一个网格点变换回原来的图像上。使用双变量线性内插方法内插像素值可确定公共平面上新的左、右图像中的像素点。



# 平行光轴立体视觉系统



# 视差公式

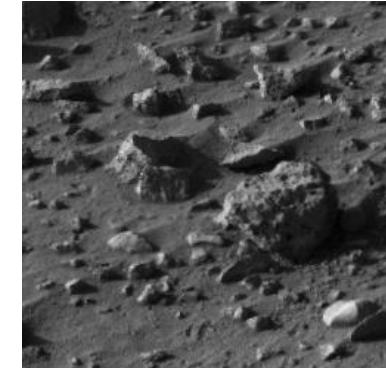
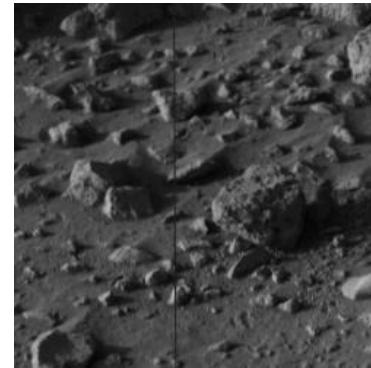


# 立体匹配

- 求解对应问题是立体成象系统的核心内容
- 求解对应问题极富有挑战性，可以说是立体视觉最困难的一步。
- 方法
  - 基于特征（点、线）的匹配(稀疏匹配)
  - 基于区域的匹配(稠密匹配)

## 立体匹配的困难

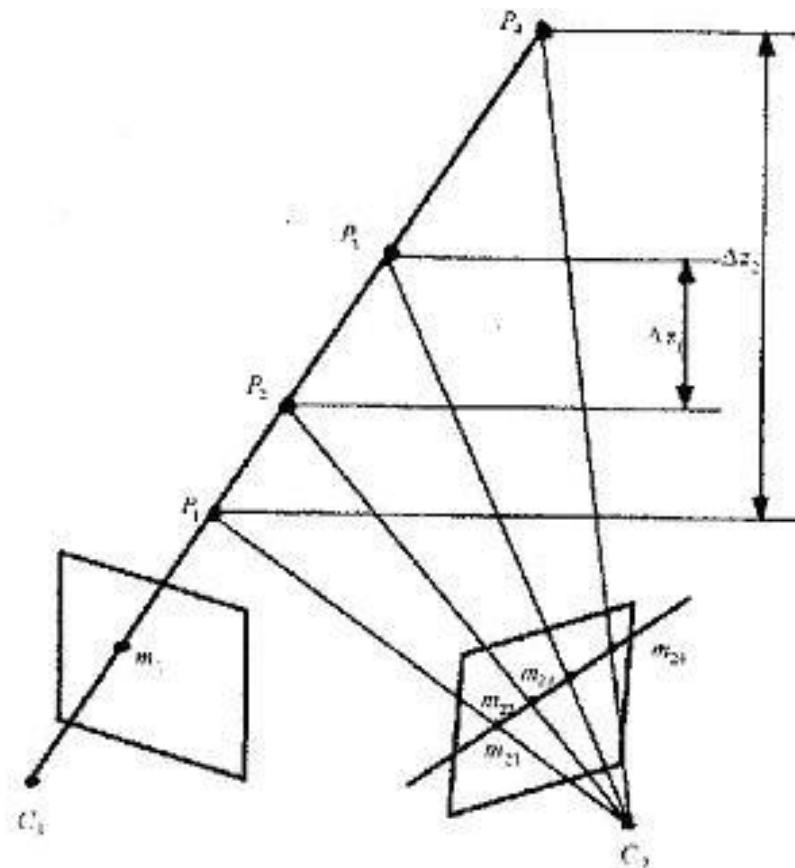
- 场景投影到两幅图像中并不总是一致的
  - 摄像机相关
    - » 图像噪声、不同增益、不同对比度等等...
  - 视点相关
    - » 透视畸变
    - » 遮挡
    - » 镜面反射
- 复杂场景因素
  - » 重复场景
  - » 无纹理区域



引入约束，减少搜索范围

# 外极线约束

- 一幅图像上的特征点一定位于另一幅图像上对应的外极线上
- 将二维搜索转会为一维搜索问题
- 在外极线的一个小邻域内进行搜索



## 一致性约束

对图像进行规范化处理(Normalization)

设参考摄像机和其它摄像机的图像函数分别为

$f_0(i, j)$  和  $f_k(i, j)$ , 则图像窗内规范化图像函数为:

$$\bar{f}_0(i, j) = (f_0(i, j) - \mu_0) / \sigma_0$$

$$\bar{f}_k(i, j) = (f_k(i, j) - \mu_k) / \sigma_k$$

$\mu$  是图像窗内光强的平均值,  $\sigma$  是光强分布参数:

$$\sigma^2 = \frac{1}{mn} \sum_{j=1}^n \sum_{i=1}^m (f(i, j) - \mu)^2$$

相似估价函数为差值绝对值之和  $\varepsilon_k = \sum_{i=1}^n \sum_{j=1}^m |\bar{f}_0(i, j) - \bar{f}_k(i, j)|$



## 顺序约束

- 如果在参考图像中点A在点B的左边 $\Rightarrow$  在目标图像中点A的匹配点也在点B的匹配点的左边
- 对细小物体不成立

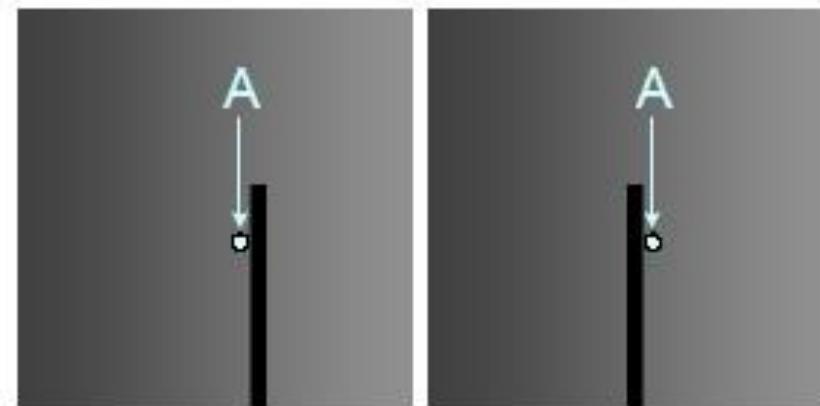


Image from Sun et al. CVPR05

## 其它约束

- 唯一性约束

一幅图像(左或右)上的每一个特征点只能与另一幅图像上的唯一一个特征对应.

- 连续性约束

物体表面一般都是光滑的，因此物体表面上各点在图像上的投影也是连续的，它们的视差也是连续的.

在物体边界处，连续性约束不能成立.



## 特征匹配——稀疏

- 在立体图像对中识别兴趣点 (interesting point), 而后在两幅图像中匹配相对应的点.
- 识别兴趣点 (interesting point)

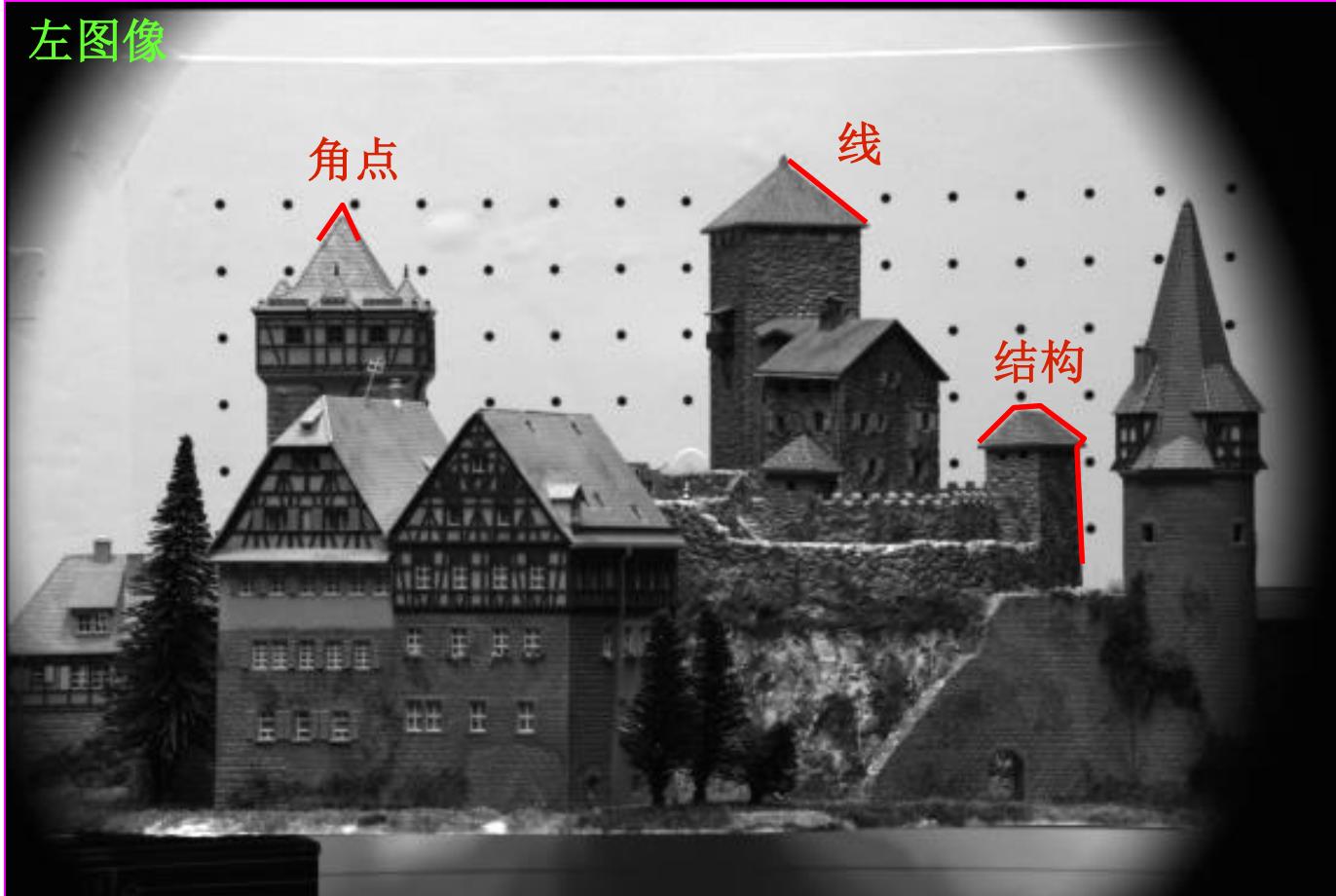
在图像中具有很大变化的区域内寻找兴趣点

在以某一点为中心的窗函数中，计算其在不同方向上的变化量

为避免将多个相邻点选为同一个特征对应的兴趣点，将特征点选在兴趣测度函数具有局部最大值的地方

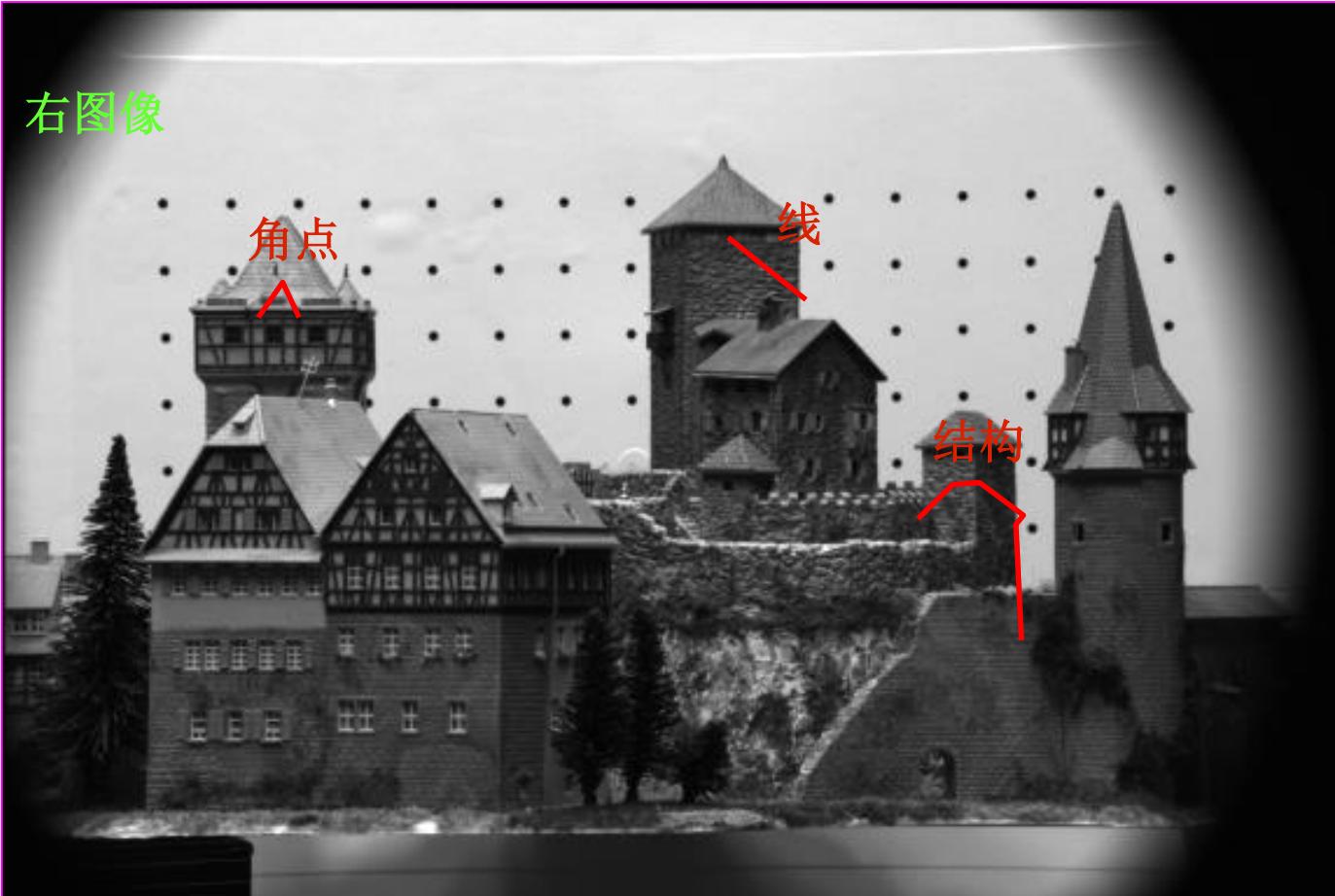
## 特征匹配

左图像



- 对于左图像中的每一个特征...

# 特征匹配



- 在右图像中寻找... 当相似度达到最大时的偏移量就是视差

## 特征匹配——稀疏

- 基于特征的立体匹配算法产生对应于图像特征点的场景稀疏深度图.
- 仅仅能恢复用于求解共轭对的像素子集对应的特征点深度.
- 要想得到其它点的深度值，必须通过使用有关计算方法来估算，如内插值技术.

## ■ 基于边缘特征的立体匹配算法

- (1) 采用标准的摄像机设置，并利用边缘点作为图像特征。
- (2) 采用外极约束使匹配过程简化：为求左图像某扫描行上一个边缘点 $P_{i1}$ 在右图像上的对应点，沿右图像的同一扫描行扫描寻找匹配点即可。根据相容性约束，所求对应点 $P_{ir}$ 也应为一边缘点，且两个边缘点的幅值和方向应保持一致。

## ■ 基于边缘特征的立体匹配算法描述

输入图像： 左边缘图像  $e_l(i, j)$ ,  $1 \leq i \leq I, 1 \leq j \leq J$

右边缘图像  $e_r(i, j)$ ,  $1 \leq i \leq I, 1 \leq j \leq J$

输出图像： 视差图像  $d(i, j)$ ,  $1 \leq i \leq I, 1 \leq j \leq J$

若干标记：

$i_l$  左图像的现行行指标；

$i_r$  右图像的现行行指标；

$j_l$  左图像的现行列指标；

$j_r$  右图像的现行列指标；



## ■ 基于边缘特征的立体匹配算法描述

算法步骤：

(1) 初始化操作：将 $d(i, j)$ 清零，并置：

$$i_1=1, i_r=1, j_1=1 \text{ 和 } j_r=1。$$

(2) 匹配运算：

(2-1) 从左边缘图像的第 $i_1$ 行当前列开始，寻找下一个待匹配的边缘点 $P_{i1}$ 。这里， $P_{i1}$ 的列指标由 $j_1$ 指示；

(2-2) 在右边缘图像的第 $i_r=i_1$ 行上，寻找 $P_{i1}$ 的对应点 $P_{ir}$ 。方法如下：比较候补边缘点和 $P_{i1}$ 的幅值和方向，看是否一致。如果需要的话，可引入顺序约束以进一步减少匹配运算。

上述过程不断进行直到在右图像上找到具有最大一致性的边缘点为止，并将其定为 $P_{i1}$ 的对应点 $P_{ir}$ 。这里， $P_{ir}$ 的列指标由 $j_r$ 指示。然后，根据 $j_1$ 和 $j_r$ 的值计算待匹配的边缘点 $P_{i1}$ 处对应的视差。即置 $d(i_1, j_1)=j_1-j_r$ 。

## ■ 基于边缘特征的立体匹配算法描述

### 算法步骤(续)：

一旦右图像上的一个边缘点被定为左图像上的一个边缘点的对应点，则根据唯一性约束，该点以后将不能和左图像上任何别的边缘点相匹配。

- (2-3) 进行行终止检查。若左图像的现行行上已无待匹配的边缘点，去 (3)；否则，回到 (2-1)。
- (3) 若  $i_1=I$ ，去 (4)；  
否则，置  $i_1=i_1+1$ 、 $j_1=1$  和  $j_r=1$ ，回到 (2-1)；
- (4) 根据视差图像  $d(i, j)$  和摄像机的系统参数，计算各边缘点处的三维坐标。

算法特点：鲁棒，但仅在图像的边缘点处可以获得相应景物的三维位置信息。

## 基于区域相关性的立体匹配(稠密)

- 计算一幅图像的一个小窗函数内的像素与另一幅图像中具有同样的潜在对应特征的小窗函数的像素之间的相关值. 具有最大相关值的小窗区域就是对应区域.
- 只有满足外极线约束的区域才能是匹配区域. 考虑到垂直视差的存在, 应将外极线邻近的像素点也包括在潜在的匹配特征集中.

## 立体匹配评价函数

### SAD—Sum of Absolute Differences

$$\sum_{(i,j) \in W} |I_1(x+i, y+j) - I_2(x+d+i, y+j)|$$

### SSD—Sum of Squared Differences

$$\sum_{(i,j) \in W} (I_1(x+i, y+j) - I_2(x+d+i, y+j))^2$$

### NCC—normalized cross correlation (NCC)

$$\frac{\sum_{(i,j) \in W} I_1(x+i, y+j) \cdot I_2(x+d+i, y+j)}{\sum_{(i,j) \in W} I_1(x+i, y+j)^2 \cdot \sum_{(i,j) \in W} I_2(x+d+i, y+j)^2}$$

# 立体匹配评价函数

## ZSAD—Zero mean SAD

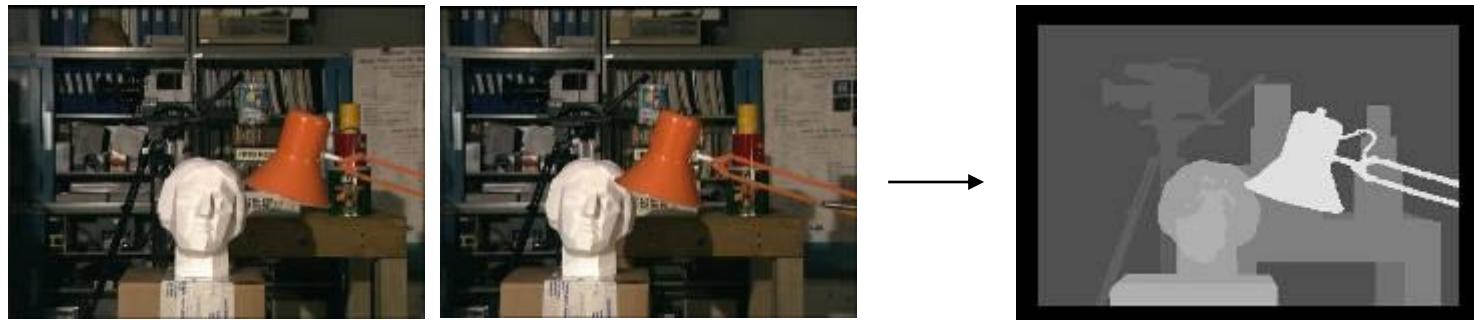
$$\sum_{(i,j) \in W} \left| \left( I_1(x+i, y+j) - \overline{I_1(x, y)} \right) - \left( I_2(x+d+i, y+j) - \overline{I_2(x, y)} \right) \right|$$

## NSAD—Normalized SAD

$$\frac{\sum_{(i,j) \in W} |I_1(x+i, y+j) - I_2(x+d+i, y+j)|}{\sqrt{\sum_{(i,j) \in W} I_1(x+i, y+j)^2} \sqrt{\sum_{(i,j) \in W} I_2(x+d+i, y+j)^2}}$$

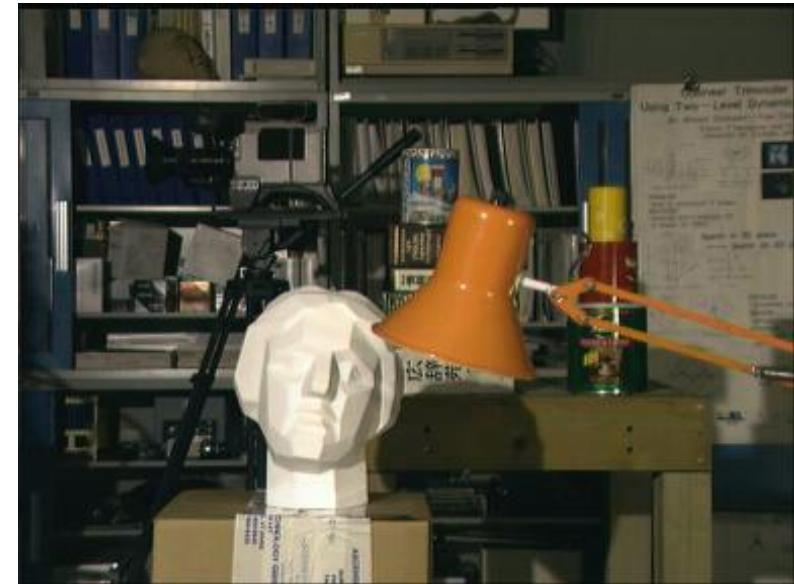
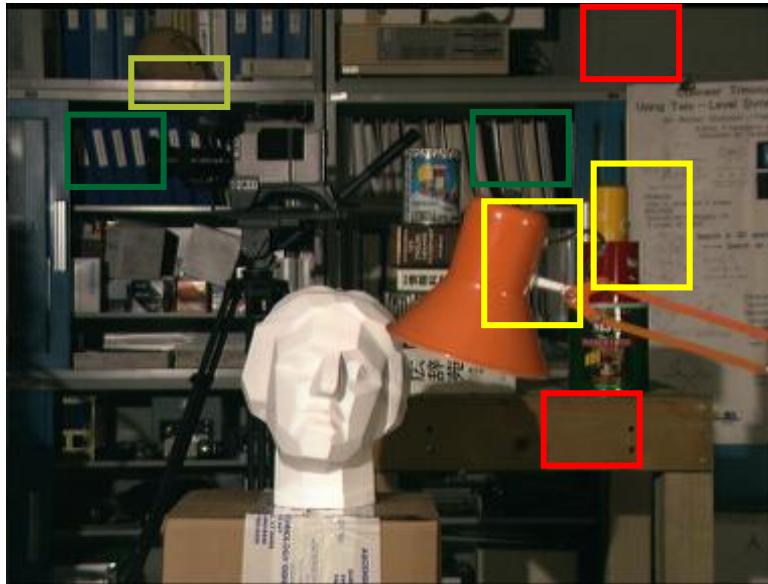
## NZSAD—Normalized ZSAD

# 稠密匹配



- 对参考图像中每一点找到对应于场景中同一点的匹配像素，得到稠密的视差图

## ■ 导致稠密立体匹配困难的原因



- 左、右图像的彩色不一致
- 图像中存在遮挡区域
- 图像中存在镜面反射现象
- 图像中存在重复模式
- 图像中存在无纹理区域

## 算法评估

- 以真实视差场为参照，对计算得到的视差场进行评估，统计视差场的准确度，以此反映匹配方法的性能
- <http://vision.middlebury.edu/stereo>



Algorithm	Avg.	Tsukuba			Venus			Teddy			Cones		
		ground truth			ground truth			ground truth			ground truth		
Rank		nonocc	all	disc									
AdaptingBP [17]	2.8	1.11 <sub>6</sub>	1.37 <sub>3</sub>	5.79 <sub>7</sub>	0.10 <sub>1</sub>	0.21 <sub>2</sub>	1.44 <sub>1</sub>	4.22 <sub>4</sub>	7.06 <sub>2</sub>	11.8 <sub>4</sub>	2.48 <sub>1</sub>	7.92 <sub>2</sub>	7.32 <sub>1</sub>
DoubleBP2 [35]	2.9	0.88 <sub>1</sub>	1.29 <sub>1</sub>	4.76 <sub>1</sub>	0.13 <sub>3</sub>	0.45 <sub>5</sub>	1.87 <sub>5</sub>	3.53 <sub>2</sub>	8.30 <sub>3</sub>	9.63 <sub>1</sub>	2.90 <sub>3</sub>	8.78 <sub>8</sub>	7.79 <sub>2</sub>
DoubleBP [15]	4.9	0.88 <sub>2</sub>	1.29 <sub>2</sub>	4.76 <sub>2</sub>	0.14 <sub>5</sub>	0.60 <sub>13</sub>	2.00 <sub>7</sub>	3.55 <sub>3</sub>	8.71 <sub>5</sub>	9.70 <sub>2</sub>	2.90 <sub>4</sub>	9.24 <sub>11</sub>	7.80 <sub>3</sub>
SubPixDoubleBP [30]	5.6	1.24 <sub>10</sub>	1.76 <sub>13</sub>	5.98 <sub>8</sub>	0.12 <sub>2</sub>	0.46 <sub>6</sub>	1.74 <sub>4</sub>	3.45 <sub>1</sub>	8.38 <sub>4</sub>	10.0 <sub>3</sub>	2.93 <sub>5</sub>	8.73 <sub>7</sub>	7.91 <sub>4</sub>
AdaptOvrSeqBP [33]	9.9	1.69 <sub>22</sub>	2.04 <sub>21</sub>	5.64 <sub>6</sub>	0.14 <sub>4</sub>	0.20 <sub>1</sub>	1.47 <sub>2</sub>	7.04 <sub>14</sub>	11.1 <sub>7</sub>	16.4 <sub>11</sub>	3.60 <sub>11</sub>	8.96 <sub>10</sub>	8.84 <sub>10</sub>
SymBP+occ [7]	10.8	0.97 <sub>4</sub>	1.75 <sub>12</sub>	5.09 <sub>4</sub>	0.16 <sub>6</sub>	0.33 <sub>3</sub>	2.19 <sub>8</sub>	6.47 <sub>8</sub>	10.7 <sub>6</sub>	17.0 <sub>14</sub>	4.79 <sub>24</sub>	10.7 <sub>21</sub>	10.9 <sub>20</sub>
PlaneFitBP [32]	10.8	0.97 <sub>5</sub>	1.83 <sub>14</sub>	5.26 <sub>5</sub>	0.17 <sub>7</sub>	0.51 <sub>8</sub>	1.71 <sub>3</sub>	6.65 <sub>9</sub>	12.1 <sub>13</sub>	14.7 <sub>7</sub>	4.17 <sub>20</sub>	10.7 <sub>20</sub>	10.6 <sub>19</sub>
AdaptDispCalib [36]	11.8	1.19 <sub>8</sub>	1.42 <sub>4</sub>	6.15 <sub>9</sub>	0.23 <sub>9</sub>	0.34 <sub>4</sub>	2.50 <sub>11</sub>	7.80 <sub>19</sub>	13.6 <sub>21</sub>	17.3 <sub>17</sub>	3.62 <sub>12</sub>	9.33 <sub>12</sub>	9.72 <sub>15</sub>
Seqm+visib [4]	12.2	1.30 <sub>15</sub>	1.57 <sub>5</sub>	6.92 <sub>18</sub>	0.79 <sub>21</sub>	1.06 <sub>18</sub>	6.76 <sub>22</sub>	5.00 <sub>5</sub>	6.54 <sub>1</sub>	12.3 <sub>5</sub>	3.72 <sub>13</sub>	8.62 <sub>6</sub>	10.2 <sub>17</sub>
C-SemiGlob [19]	12.3	2.61 <sub>29</sub>	3.29 <sub>24</sub>	9.89 <sub>27</sub>	0.25 <sub>12</sub>	0.57 <sub>10</sub>	3.24 <sub>15</sub>	5.14 <sub>6</sub>	11.8 <sub>8</sub>	13.0 <sub>6</sub>	2.77 <sub>2</sub>	8.35 <sub>4</sub>	8.20 <sub>5</sub>
SO+borders [29]	12.8	1.29 <sub>14</sub>	1.71 <sub>9</sub>	6.83 <sub>15</sub>	0.25 <sub>13</sub>	0.53 <sub>9</sub>	2.26 <sub>9</sub>	7.02 <sub>13</sub>	12.2 <sub>14</sub>	16.3 <sub>9</sub>	3.90 <sub>15</sub>	9.85 <sub>16</sub>	10.2 <sub>18</sub>
DistinctSM [27]	14.1	1.21 <sub>9</sub>	1.75 <sub>11</sub>	6.39 <sub>11</sub>	0.35 <sub>14</sub>	0.69 <sub>16</sub>	2.63 <sub>13</sub>	7.45 <sub>18</sub>	13.0 <sub>17</sub>	18.1 <sub>19</sub>	3.91 <sub>16</sub>	9.91 <sub>18</sub>	8.32 <sub>7</sub>
CostAggr+occ [39]	14.3	1.38 <sub>17</sub>	1.96 <sub>17</sub>	7.14 <sub>19</sub>	0.44 <sub>16</sub>	1.13 <sub>19</sub>	4.87 <sub>19</sub>	6.80 <sub>11</sub>	11.9 <sub>10</sub>	17.3 <sub>16</sub>	3.60 <sub>10</sub>	8.57 <sub>5</sub>	9.36 <sub>13</sub>
OverSeqmBP [26]	14.5	1.69 <sub>23</sub>	1.97 <sub>18</sub>	8.47 <sub>24</sub>	0.51 <sub>18</sub>	0.68 <sub>15</sub>	4.69 <sub>18</sub>	6.74 <sub>10</sub>	11.9 <sub>12</sub>	15.8 <sub>8</sub>	3.19 <sub>8</sub>	8.81 <sub>9</sub>	8.89 <sub>11</sub>
SegmentSupport [28]	15.1	1.25 <sub>11</sub>	1.62 <sub>7</sub>	6.68 <sub>13</sub>	0.25 <sub>11</sub>	0.64 <sub>14</sub>	2.59 <sub>12</sub>	8.43 <sub>24</sub>	14.2 <sub>22</sub>	18.2 <sub>20</sub>	3.77 <sub>14</sub>	9.87 <sub>17</sub>	9.77 <sub>16</sub>
RegionTreeDP [18]	15.7	1.39 <sub>19</sub>	1.64 <sub>8</sub>	6.85 <sub>16</sub>	0.22 <sub>8</sub>	0.57 <sub>10</sub>	1.93 <sub>6</sub>	7.42 <sub>17</sub>	11.9 <sub>11</sub>	16.8 <sub>13</sub>	6.31 <sub>30</sub>	11.9 <sub>27</sub>	11.8 <sub>23</sub>
EnhancedBP [24]	16.6	0.94 <sub>3</sub>	1.74 <sub>10</sub>	5.05 <sub>3</sub>	0.35 <sub>15</sub>	0.86 <sub>17</sub>	4.34 <sub>17</sub>	8.11 <sub>22</sub>	13.3 <sub>19</sub>	18.5 <sub>22</sub>	5.09 <sub>27</sub>	11.1 <sub>23</sub>	11.0 <sub>21</sub>

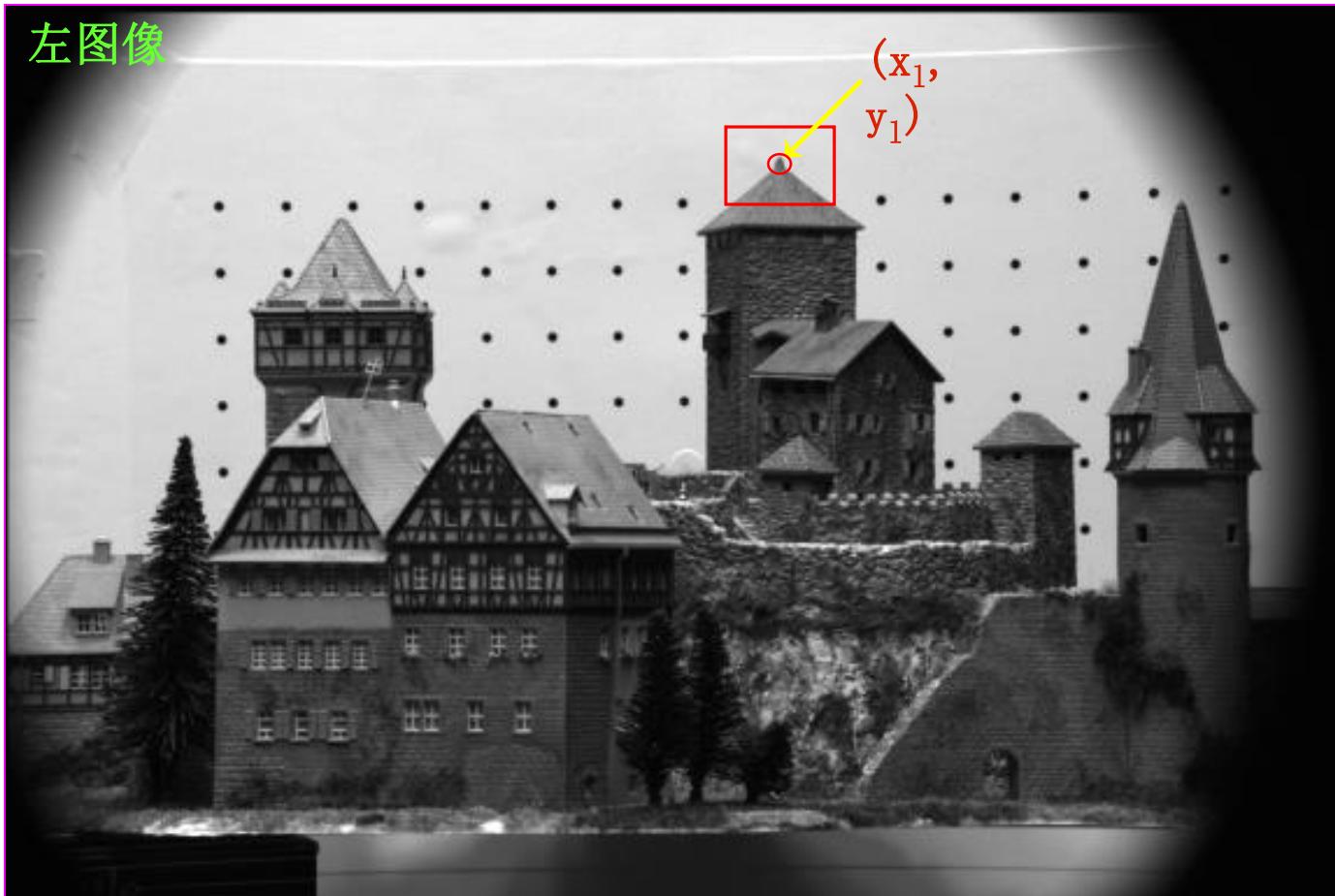
## ■ 立体匹配算法的几个要素



- 基于滑动窗相关和连续性约束的立体匹配算法
- 基于滑动窗相关的动态规划立体匹配算法
- 基于区域特征的图割立体匹配算法
- 基于置信度传播的区域匹配算法

# 经典相关窗法

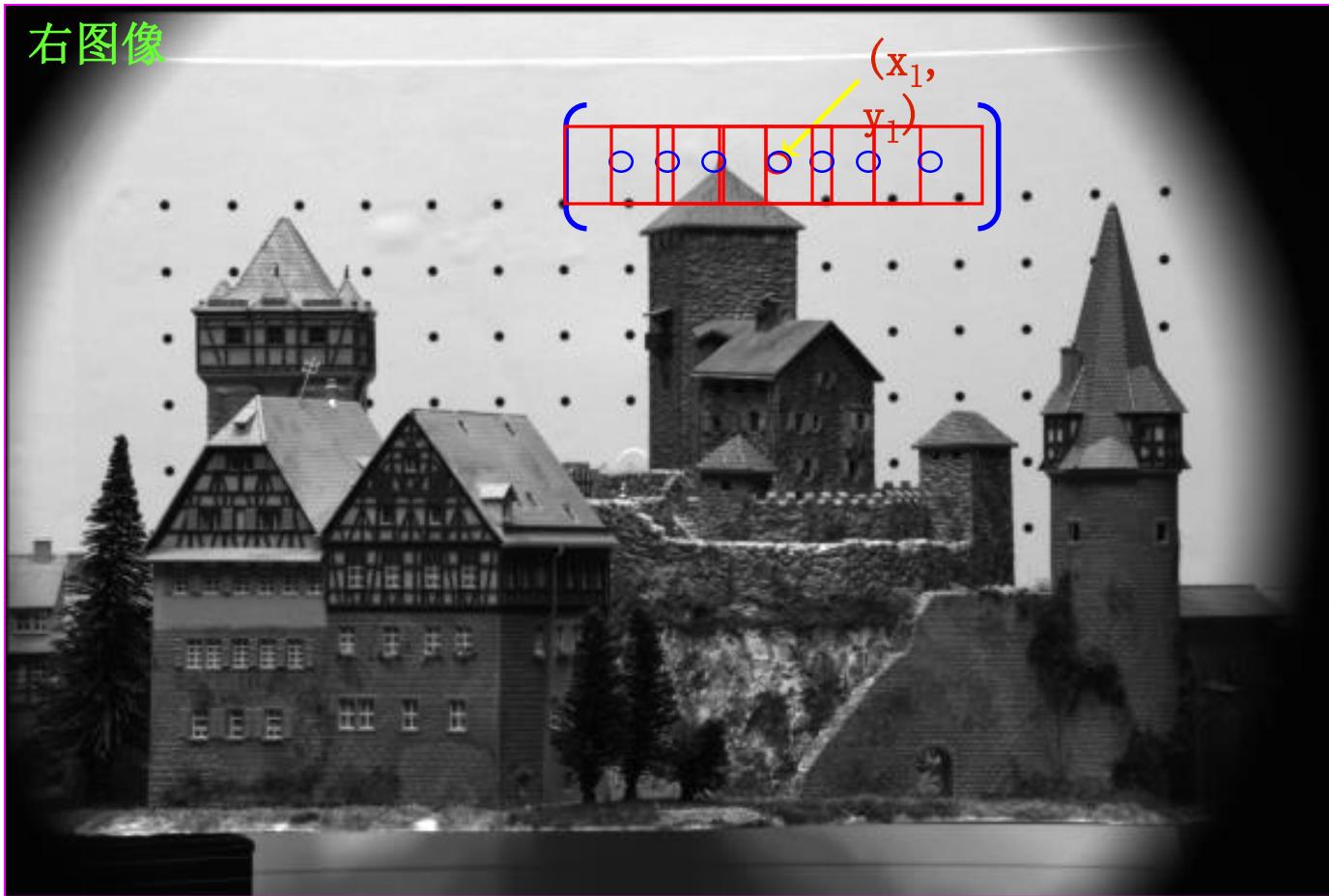
左图像



- 对参考图像中每个点  $(x_1, y_1)$  定义以它为中心的窗口

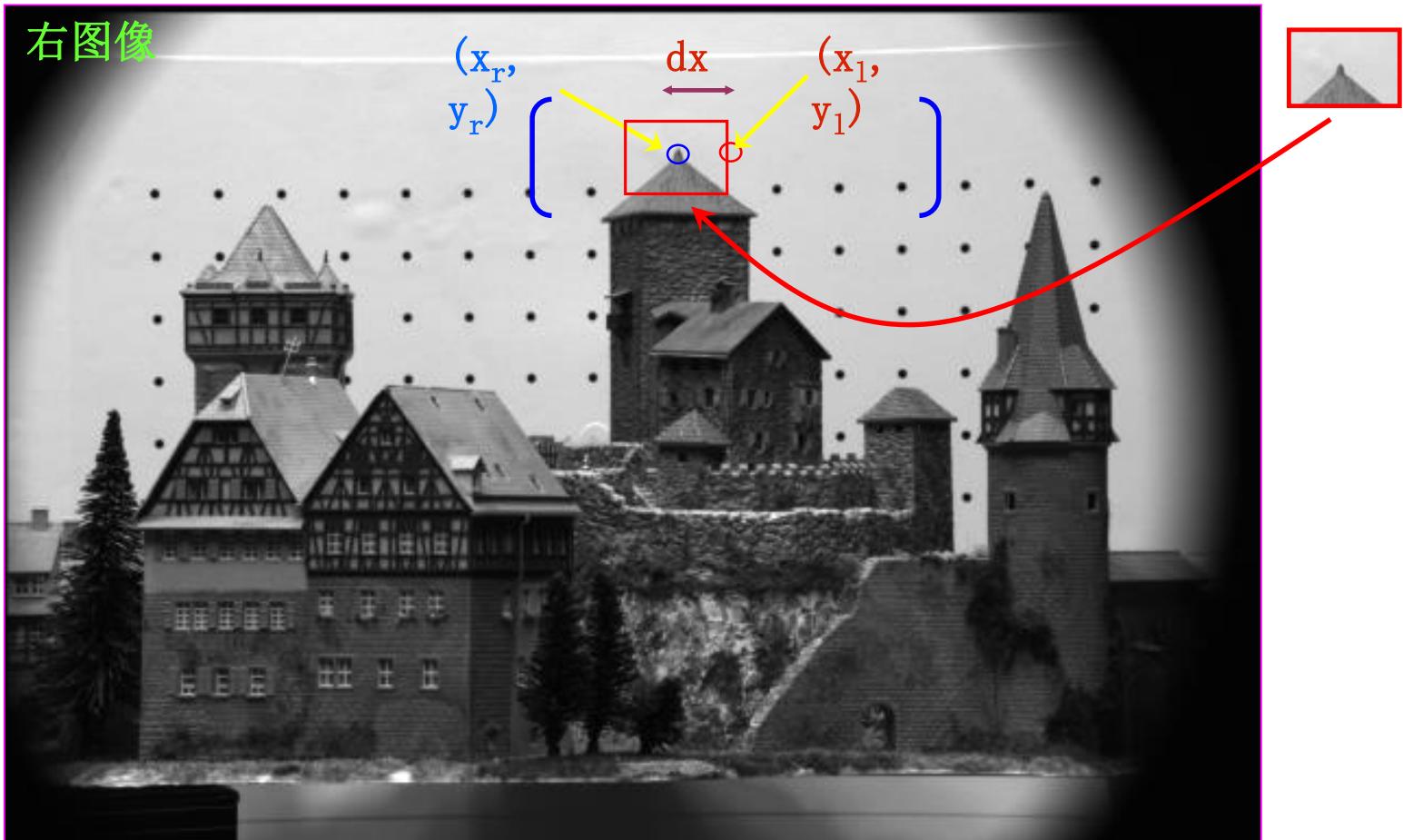
# 经典相关窗法

右图像



- 在目标图像中一定范围搜索匹配点...

# 经典相关窗法

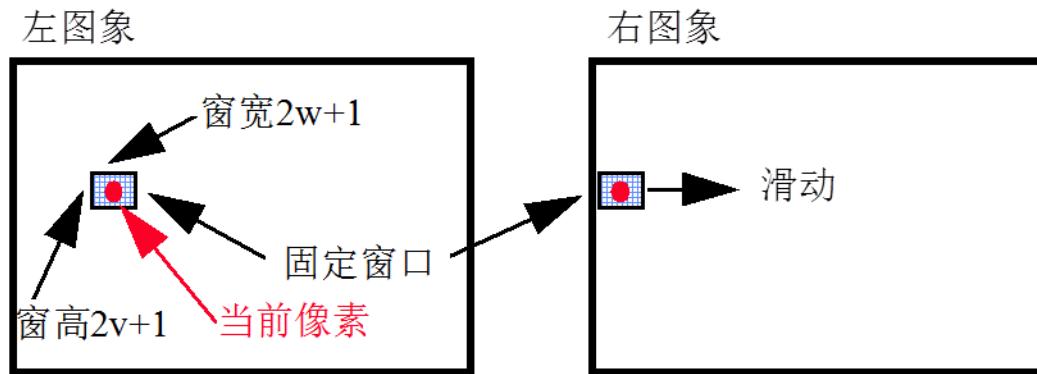


- ...视差就是当相关值达到最大时的偏移量

## ■ 基于相关窗的立体匹配算法

使用特征：

右图所示组合特征



匹配度计算：

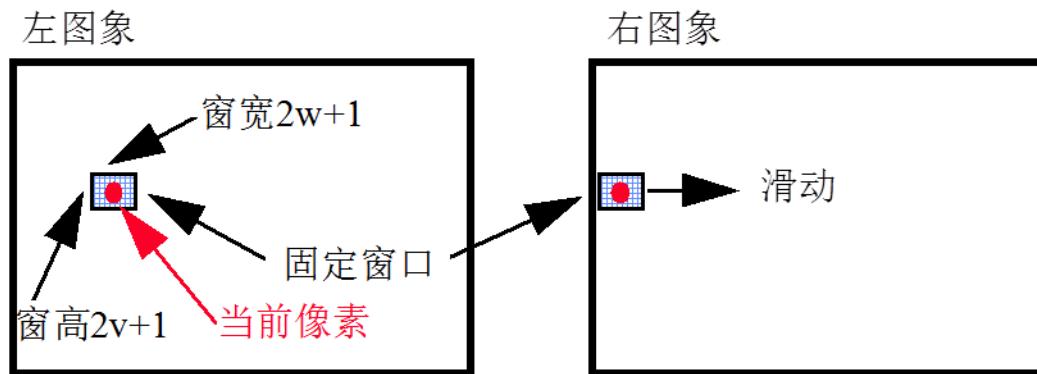
误差函数

$$E(i, j, d) = \frac{\frac{1}{(2v+1)(2w+1)} \sum_{x=i-v}^{i+v} \sum_{y=j-w}^{j+w} (f_l(x, y) - f_r(x + d, y))^2}{\sqrt{\sum_{x=i-v}^{i+v} \sum_{y=j-w}^{j+w} f_l^2(x, y)}}$$

## ■ 基于相关窗的立体匹配算法

使用特征：

右图所示组合特征



匹配度计算：

相关函数

$$R(i, j, d) = \frac{\frac{1}{(2v+1)(2w+1)} \sum_{x=i-v}^{i+v} \sum_{y=j-w}^{j+w} f_l(x, y) f_r(x+d, y)}{\sqrt{\sum_{x=i-v}^{i+v} \sum_{y=j-w}^{j+w} f_l^2(x, y)} \sqrt{\sum_{x=i-v}^{i+v} \sum_{y=j-w}^{j+w} f_r^2(x, y)}}$$

## ■ 基于相关窗的立体匹配算法

输入图像:

左灰度图像  $f_l(i, j)$ ,  $1 \leq i \leq I, 1 \leq j \leq J$   
右灰度图像  $f_r(i, j)$ ,  $1 \leq i \leq I, 1 \leq j \leq J$

输出图像:

视差图像  $d(i, j)$ ,  $1 \leq i \leq I, 1 \leq j \leq J$

若干标记:

$i_l$  左图像的现行行指标;  
 $i_r$  右图像的现行行指标;  
 $j_l$  左图像的现行列指标;  
 $j_r$  右图像的现行列指标;  
 $w$  固定窗口的宽度指标;  
 $v$  固定窗口的高度指标;

## ■ 基于相关窗的立体匹配算法

算法步骤：

- (1) 初始化操作：将 $d(i, j)$ 清零，并置 $i_l=v+1$ ,  $i_r=v+1$ 和 $j_l=w+1$ 。
- (2) 匹配运算：
  - (2-1) 在右图像的第 $i_r=i_l$ 行上寻找与左图像上的当前像素 $P_{i_l} = f_l(i_l, j_l)$ 最相似的对应点 $P_{i_r}$ 。方法如下：以固定尺寸的窗口套住当前像素 $P_{i_l}$ ，并让同一尺寸的窗口顺序滑过右图像的第 $i_r$ 行，在每一个像素位置，计算所定义的误差函数 $E(i_l, j_l, d)$ （也可计算所定义的相关函数 $R(i_l, j_l, d)$ ），直到在右图像上找到最相似的像素点 $P_{i_r}$ 为止。并将 $P_{i_r}$ 定为 $P_{i_l}$ 的对应点。若 $P_{i_r}$ 的列指标由 $j_r$ 指示，那么，置 $d(i_l, j_l) = j_l - j_r$ 。一旦右图像上的一个像素点被定为左图像上的一个像素点的对应点，则根据唯一性约束，该点以后将不能和左图像上任何别的像素点相匹配。同样，也可引入顺序约束以进一步减少匹配运算。
  - (2-2) 做行终止检查：若 $j_l=J-w$ ，去(3)；否则，置 $j_l=j_l+1$ ，回到(2-1)。
  - (3) 若 $i_l=I-v$ ，去(4)；否则，置 $i_l=i_l+1$ 、 $i_r=i_r+1$ 和 $j_l=w+1$ ，回到(2-1)。
  - (4) 根据视差图像 $d(i, j)$ 和摄像机的系统参数，计算各边缘点处的三维坐标。

## ■ 基于相关窗的立体匹配算法

算法特点：

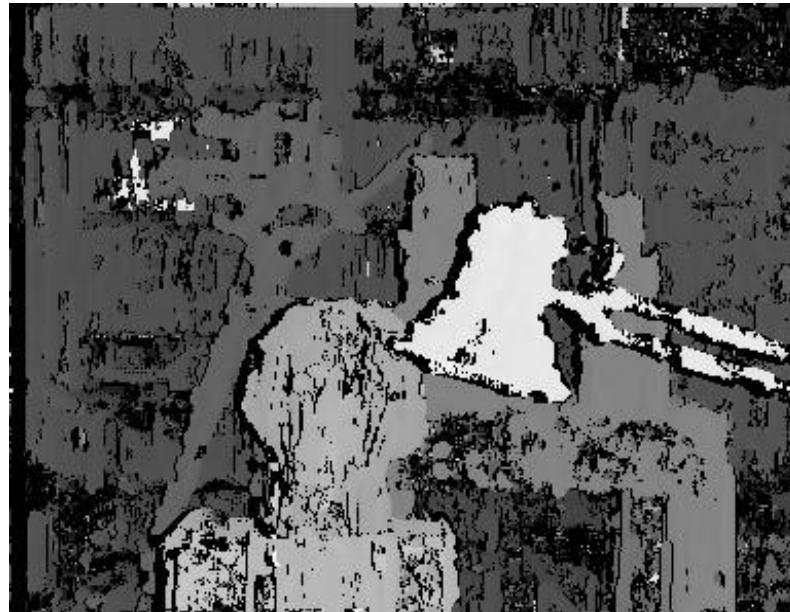
- 可获得稠密视差
- 误匹配可能较多。

误匹配的原因：

- 重复场景
  - 遮挡
  - 局部优化
- (各类匹配算法的共同问题)



## 经典相关窗法结果



# 自适应支持权值法

- 格式塔心理学
  - 相似性准则
  - 接近性准则
- 根据相似性和接近性设计邻近像素对匹配窗中心像素的权值 [Yoon CVPR05]



(a) left support window  
(b) right support window  
(c) color difference  
between (a) and (b)

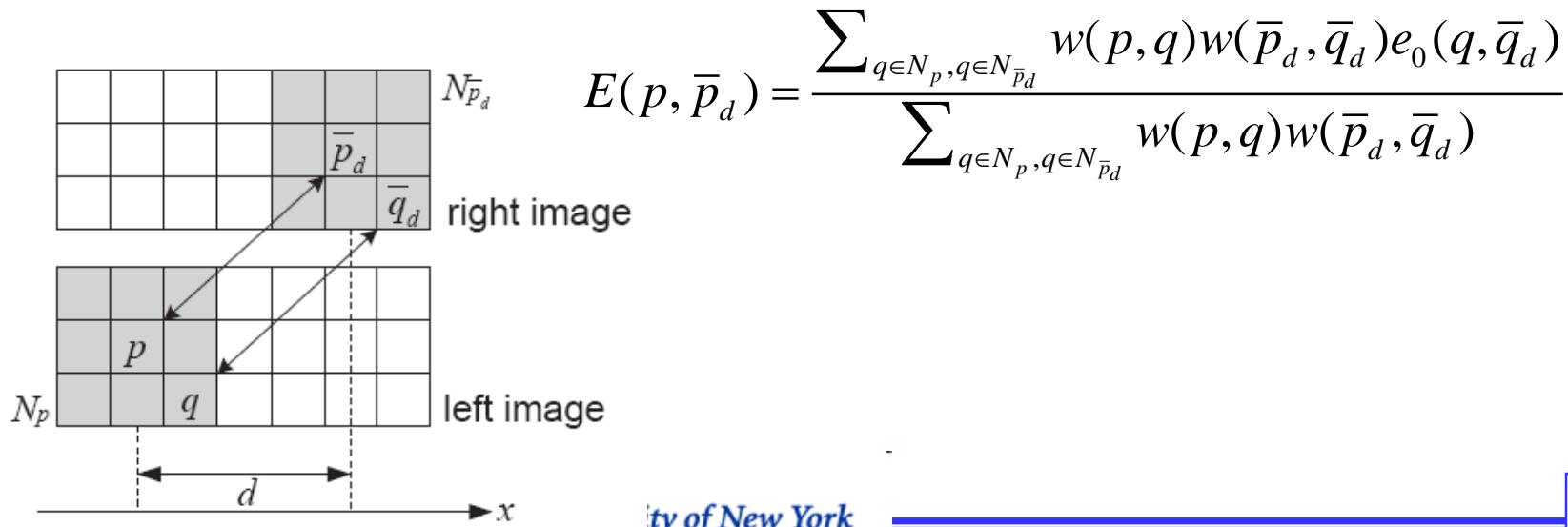
# 自适应支持权值法

- 在CIE Lab颜色空间的相似性:

$$\Delta c_{pq} = \sqrt{(L_p - L_q)^2 + (a_p - a_q)^2 + (b_p - b_q)^2}$$

- 接近性: 欧氏空间距离

- Weights:  $w(p, q) = k \cdot \exp\left(-\left(\frac{\Delta c_{pq}}{\gamma_c} + \frac{\Delta g_{pq}}{\gamma_p}\right)\right)$
- 匹配代价累积公式



# 自适应支持权值法结果



(a) left image



(b) ground truth



(c) shiftable win. [7]



(d) compact win. [3]



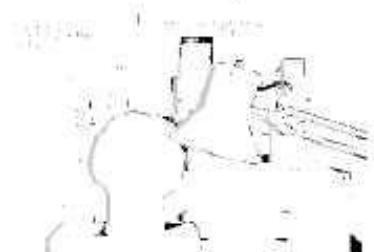
(e) variable win. [4]



(f) Bay. diff. [19]



(g) our result



(h) bad pixels (error > 1)

# 自适应支持权值法结果



(a) left image



(b) ground truth



(c) our result



(d) bad pixels ( $\text{error} > 1$ )

## 自适应窗口加权法结果



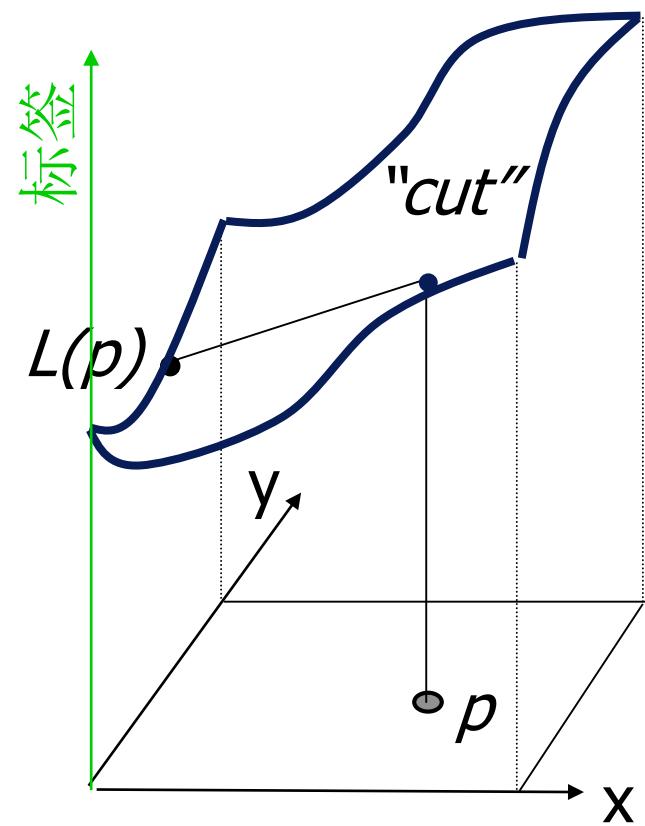
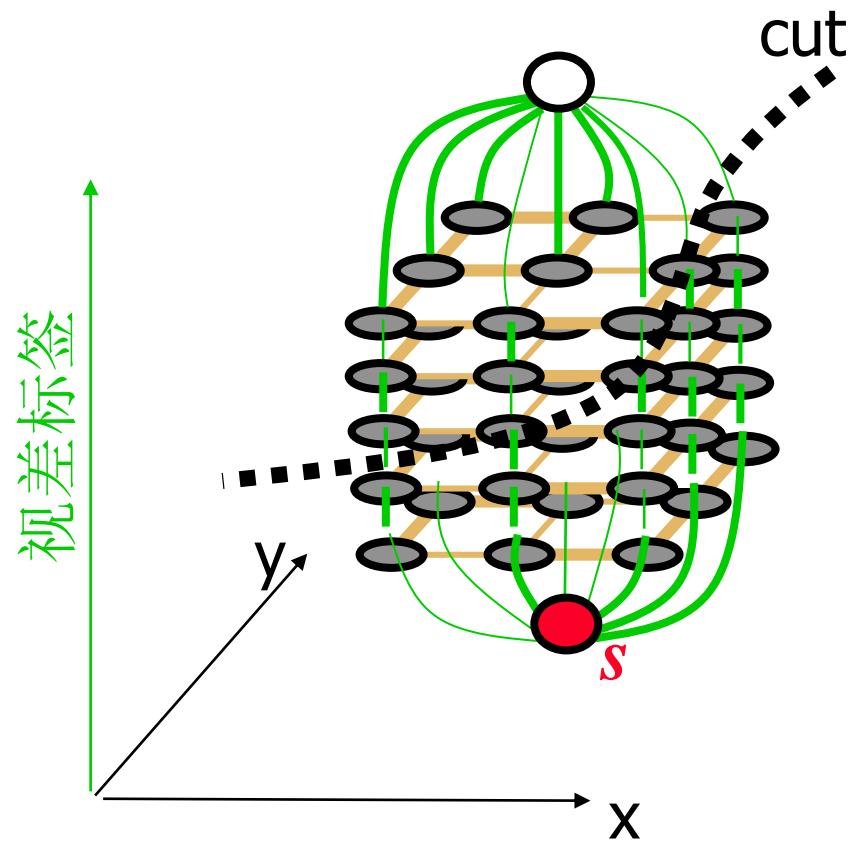
## 马尔科夫随机场

- “贴标签”，标签就是视差
- 给特定像素分配一个标签有分配代价
- 给临近像素分配一对标签有分离代价
- 找到总的分配代价和分离代价之和最小
  - 图割算法
  - 置信传播算法

## 图割算法

- 通过计算赋权图的最小割集求能量最小化
  - 以割代价为能量，求得的割集就是标签，
  - 通过最大流算法使能量迅速减少
- 运行时间与像素数和标签数成线性关系
  - 近似地说，低维多项式

# 图割算法



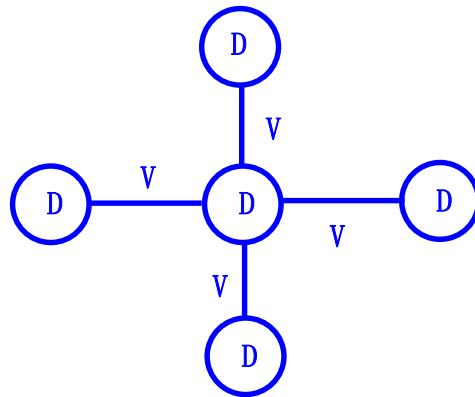
## 图割算法结果

- 在能量中包括遮挡项 [Kolmogorov ICCV01]



# 置信传播算法

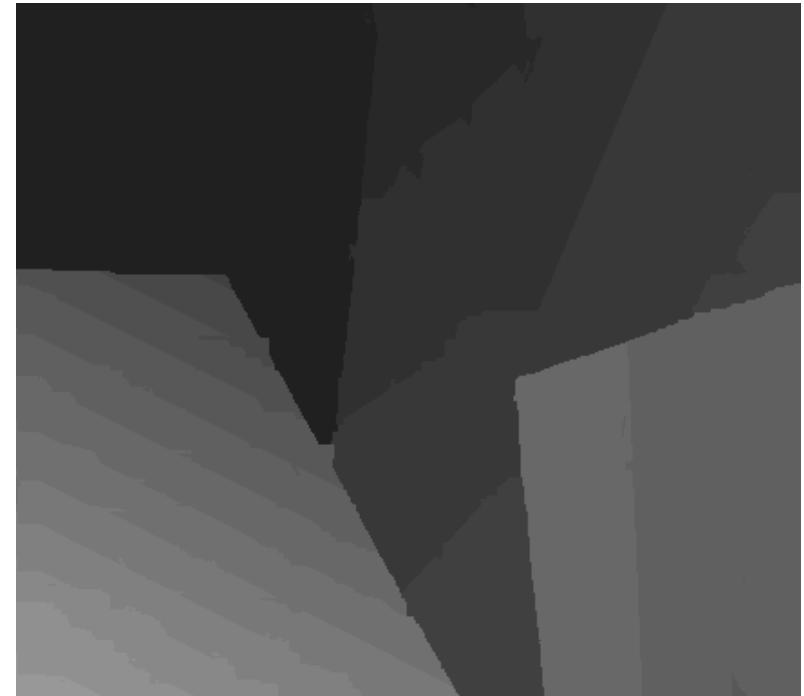
- 在马尔可夫网络上传播消息的迭代推导算法
  - 消息:反映邻近站点变量取值对该站点变量取值的影响
- 对树结构有精确解, 对有环图有较好的近似解



## 对称置信传播算法结果

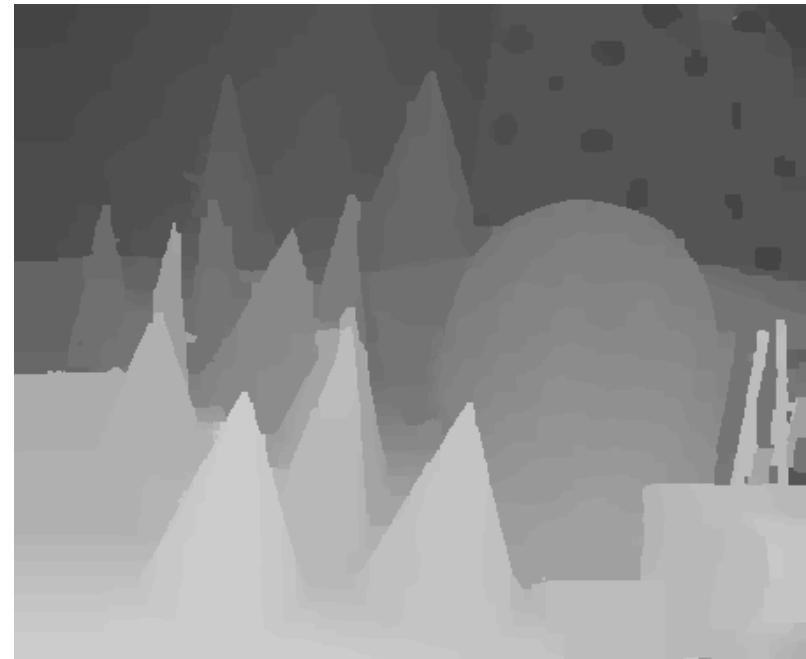
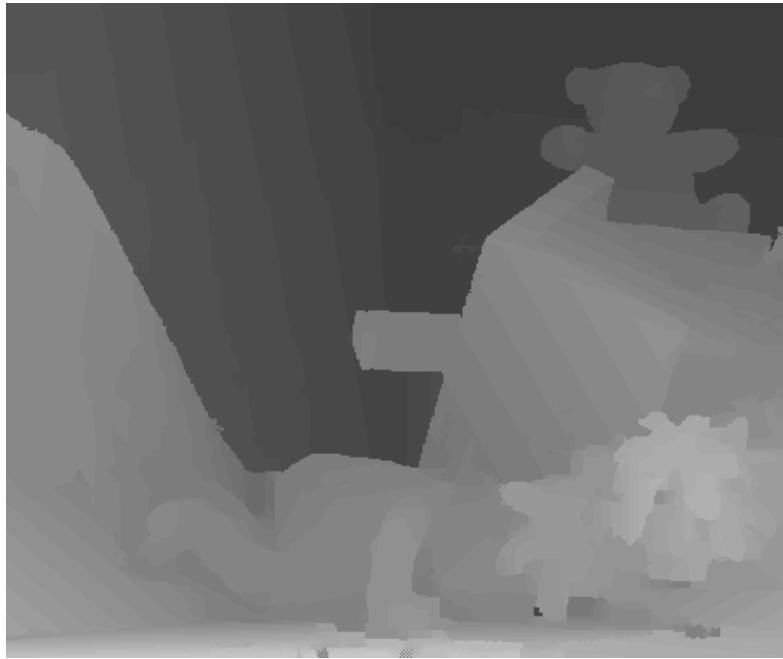


Middlebury评估网站排名第一  
(June 2005)

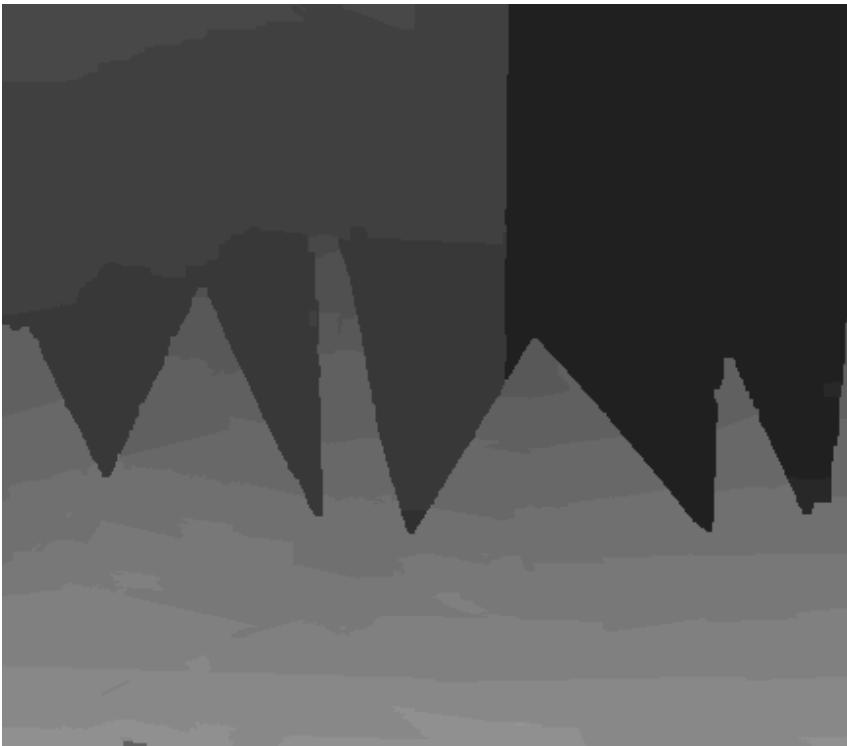


Middlebury评估网站排名第三  
(June 2005)

## 对称置信传播算法结果



## 对称置信传播算法结果



Middlebury评估网站排名第一  
(June 2005)



Middlebury评估网站排名第一  
(June 2005)

# 基于分割区域的算法



- 隐含假设
  - 颜色平滑的区域内部视差能够用平滑的视差模型（常数、平面等等）代替
  - 视差不连续处与分割区域边缘相一致
- 通常步骤
  - 图像分割
  - 初始视差计算
  - 根据初始视差估计每个视差平面参数
  - 根据定义能量以分割区域为整体采用合适算法优化



# 基于分割区域的算法

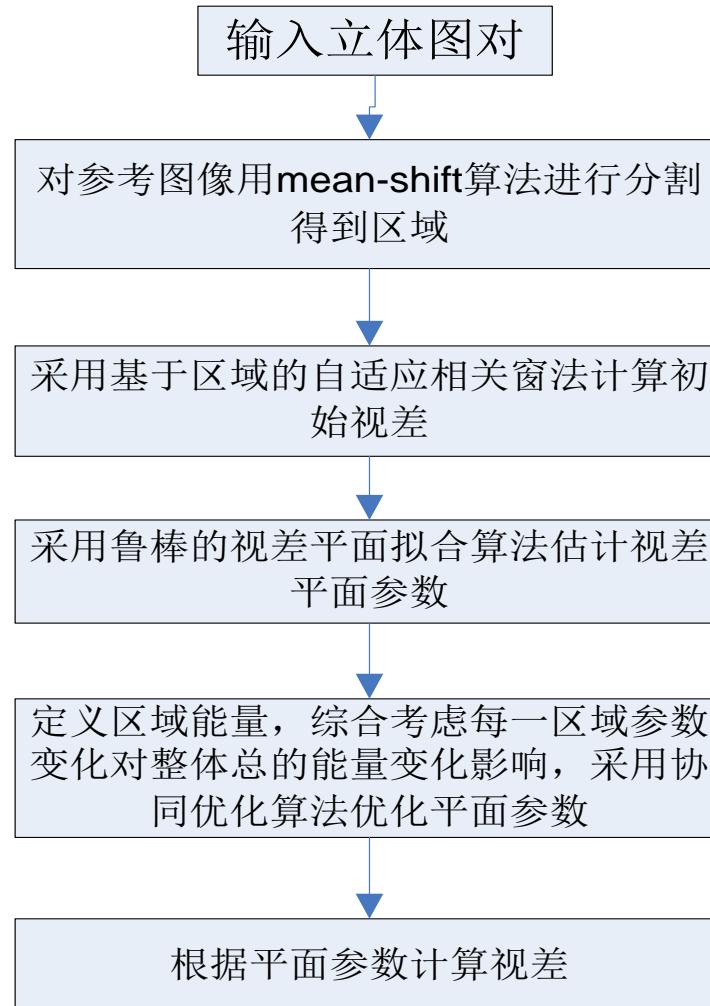
- 优点

- 区域内的平滑是被强制执行的
- 单眼线索所获得的视差边界在很多时候比单纯由视差估计的边界更为准确
- 对遮挡区域匹配的鲁棒性也得到改善
- 效率更高

- 缺点

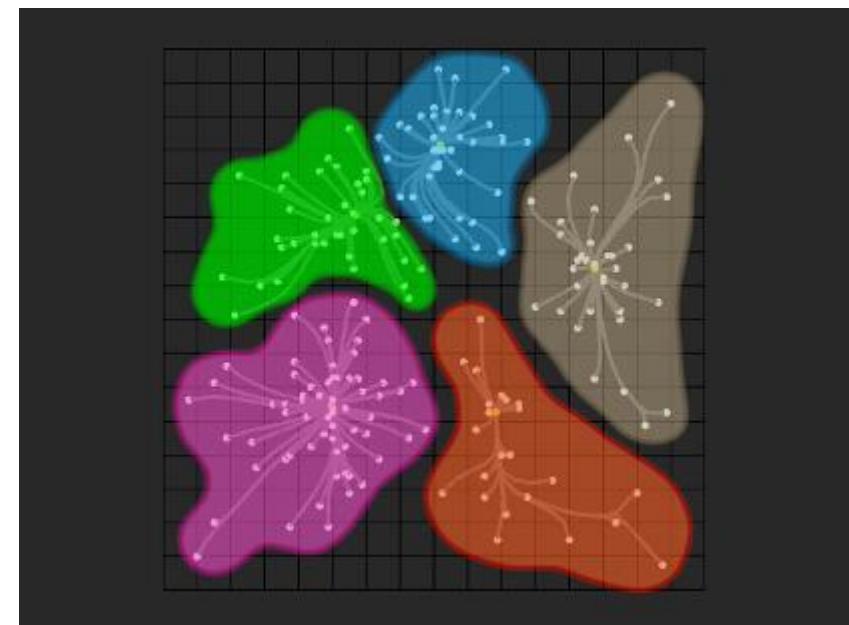
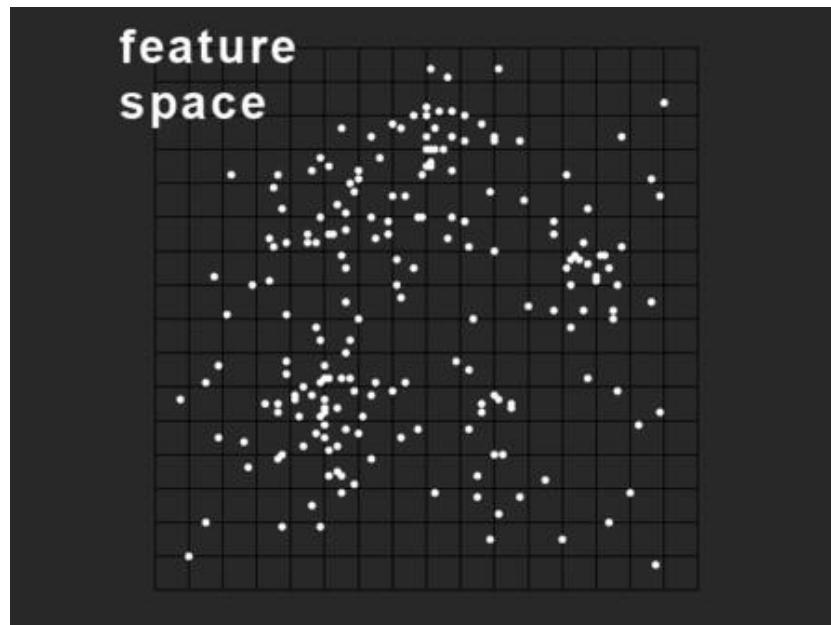
- 基于分割区域的方法的分割假设并不一定总是正确的
- 视差模型可能并不能表示区域真正的视差

# 基于区域间协同优化的立体匹配算法



## 均值偏移 (Mean Shift) 图像分割算法

通过求Mean Shift矢量的方向来得到梯度的方向, 进而通过对其跟踪, 得到密度最大的点, 即聚类算法中的所谓模式点



## 不同带宽对立体匹配的影响



(a)  $h_s = 5$ 、 $h_r = 3$ 、匹配错误率 1.88%



(b)  $h_s = 6$ 、 $h_r = 4$ 、匹配错误率 1.26%

## 不同带宽对立体匹配的影响



(c)  $h_s = 7$ 、 $h_r = 5$ 、匹配错误率 1.13%

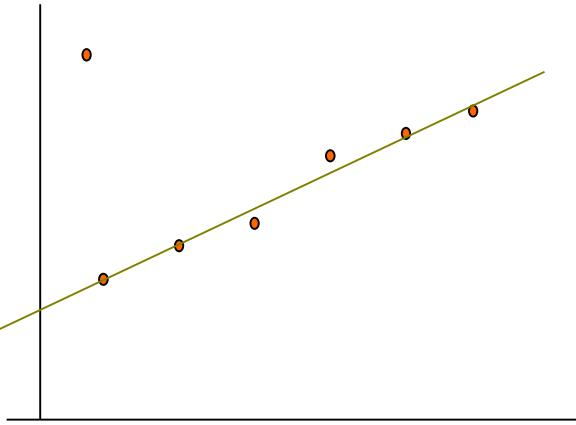


(d)  $h_s = 8$ 、 $h_r = 6$ 、匹配错误率 1.47%

## 鲁棒的视差平面拟合算法

- RANSAC

- 随机地从数据集S中选择s个数据点组成一个样本作为模型的一个例示
- 确定在模型距离阈值 $t$ 内的数据点集为一致集
- 经过N次试验，选择最大一致集，并用的所有点重估模型



## 经过RANSAC平面拟合后的视差图

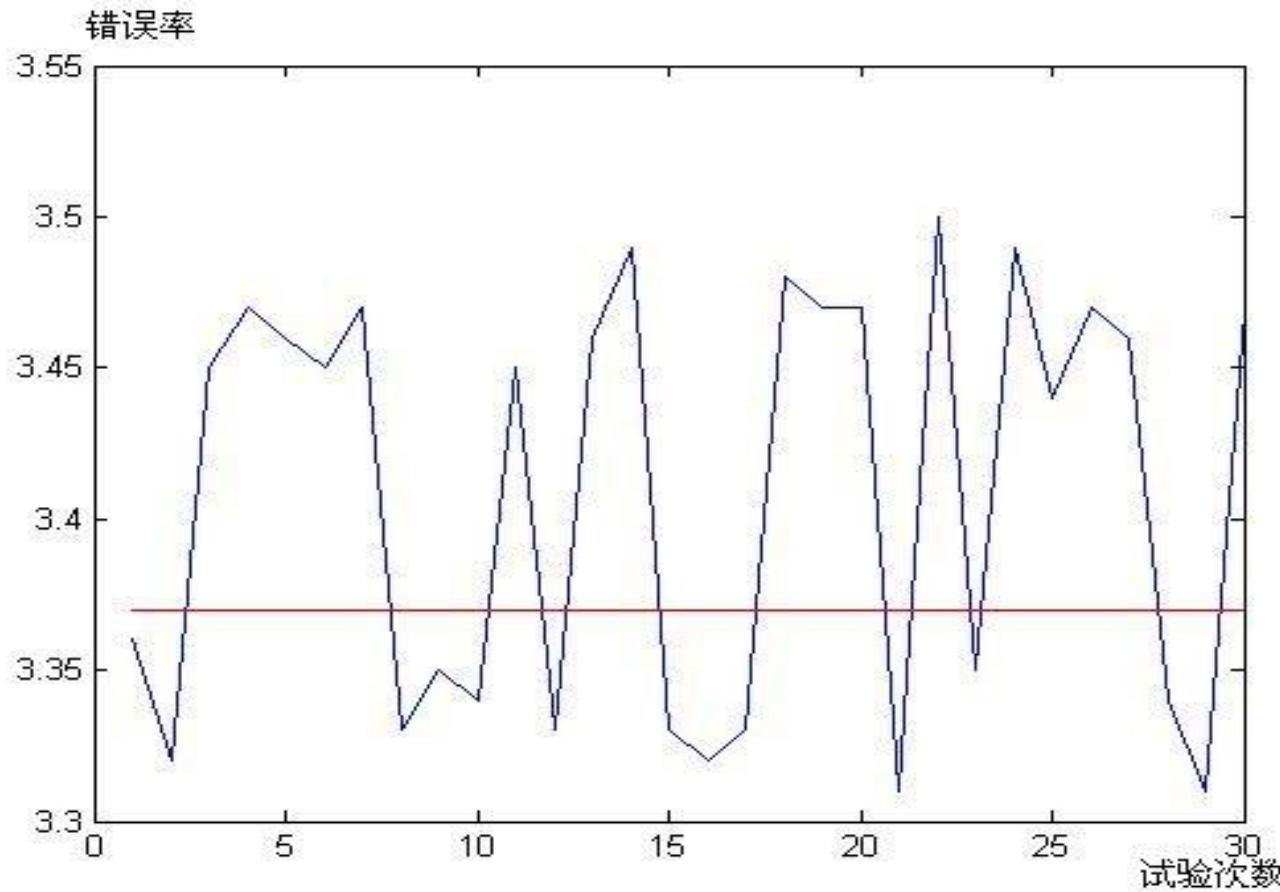


图 5- 3 经过 RANSAC 平面拟合后的视差图

## 基于投票的鲁棒视差平面拟合算法

$$d(x, y) = a * x + b * y + c$$

- 通过对同一行的一对点计算  $\delta d / \delta x$ ，可以得到平面参数a的一个估计
- 将所有同一行的估计值在一维的a参数空间进行投票，并对投票结果进行高斯平滑后从中选出得票最多的值作为参数a的最终估计
- 对同一列的点通过计算  $\delta d / \delta y$ ，可以得到参数b的估计
- 由上式可算出c，采用类似的方法可确定参数c



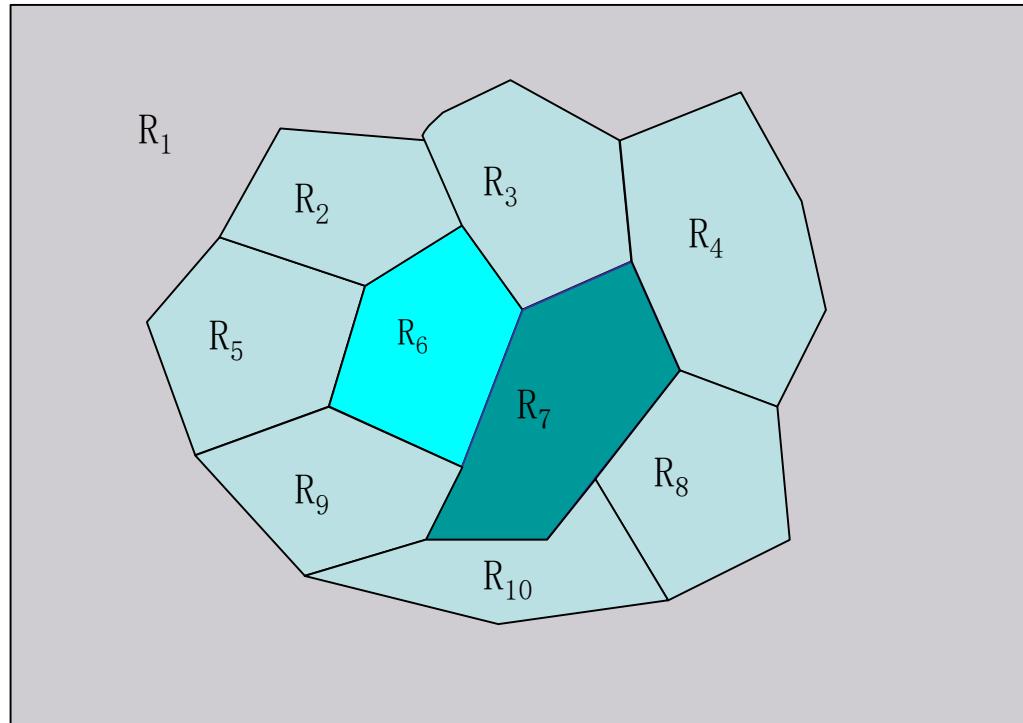
RANSAC方法和投票方法平面拟合结果的比较。其中，  
红色横线表示投票方法的拟合错误率，蓝色折线表示  
RANSAC方法的拟合错误率

## 经过投票平面拟合后的视差图



采用基于投票的平面拟合法得到的视差图

## 协同优化算法的基本原理



$$E(x) = E_1(x^1) + E_2(x^2) + \cdots + E_m(x^m)$$

## 协同优化算法的基本原理

为了使各子目标函数之间的优化结果能够保持一致，协同优化算法在优化每一子目标函数的时候，考虑与之相关联的其它子目标函数的优化结果对其的影响。

$$(1 - \lambda_i)E_i(x) + \lambda_i \sum_{j \neq i} w_{ij} \min_{x_i} E_j(x)$$

其中， $\lambda_i$  表示合作强度，而  $w_{ij}$  则用于刻画传播的力度。

## 协同优化算法的基本原理

### 协同优化算法的迭代方程

$$E_i^{(k)}(x) = \min \left( (1 - \lambda_i^{(k)}) E_i^{(k-1)}(x) + \lambda_i^{(k)} \sum_{j \neq i} w_{ij} E_j^{(k-1)}(x) \right),$$

for  $i = 1, 2, \dots, m$

迭代过程不断进行直至算法收敛或执行完规定的迭代次数。由于对每一个区域的优化结果会在下一次迭代中向周边传播，经过若干次迭代后每一个优化变量在相关子目标函数中的最终取值将会取得一致。

## 基于区域间协同优化的立体匹配

- 每个区域能量项定义

$$E_i = E_{data} + E_{occlude} + E_{smooth}$$

- 其中，第1项是数据能量，第2项是遮挡能量，而第3项是平滑能量。

## 基于区域间协同优化的立体匹配

$$E_{data} = \max_{p \in V_l \text{ and } q \in V_r} (|r(p) - r(q)|, |g(p) - g(q)|, |b(p) - b(q)|)$$

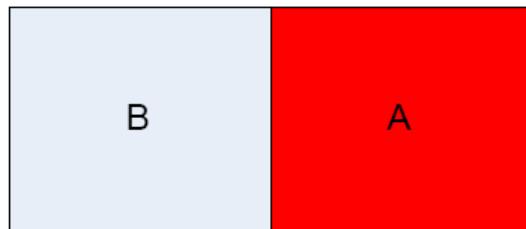
其中， $V_l$ 和 $V_r$ 分别表示当前区域在左右图像上的可见像素集， $p$ 、 $q$ 为左右图像上相匹配的两个对应像素， $r$ 、 $g$ 、 $b$ 表示相应像素的RGB值。

## 基于区域间协同优化的立体匹配

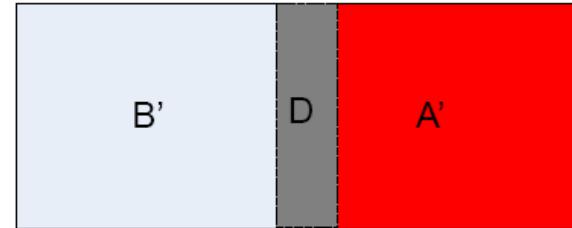
$$E_{smooth} = \sum_{p \in B_c} \begin{cases} \lambda_{disp} & \text{如果 } |d(p) - d(q)| \geq 1 \\ 0 & \text{其它} \end{cases}$$

这里， $B_c$ 表示参考图像上当前区域的边界点集， $N$ 表示和 $B_c$ 近邻的其它区域上的边界点集， $B_c$ 中的 $p$ 、 $N$ 中的 $q$ 为四连通意义上的两个近邻像素， $d(p)$ 、 $d(q)$ 为像素 $p$ 、 $q$ 的视差，而 $\lambda_{disp}$ 为所设置的平滑惩罚常量。

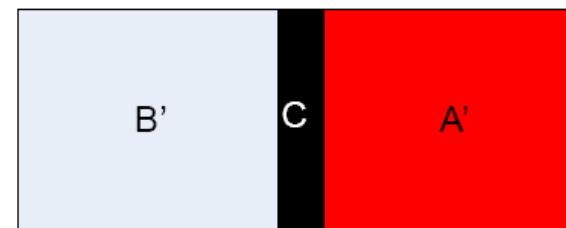
## 基于区域间协同优化的立体匹配



左图像



右图像： A的视差大于B的视差



右图像： A的视差小于B的视差

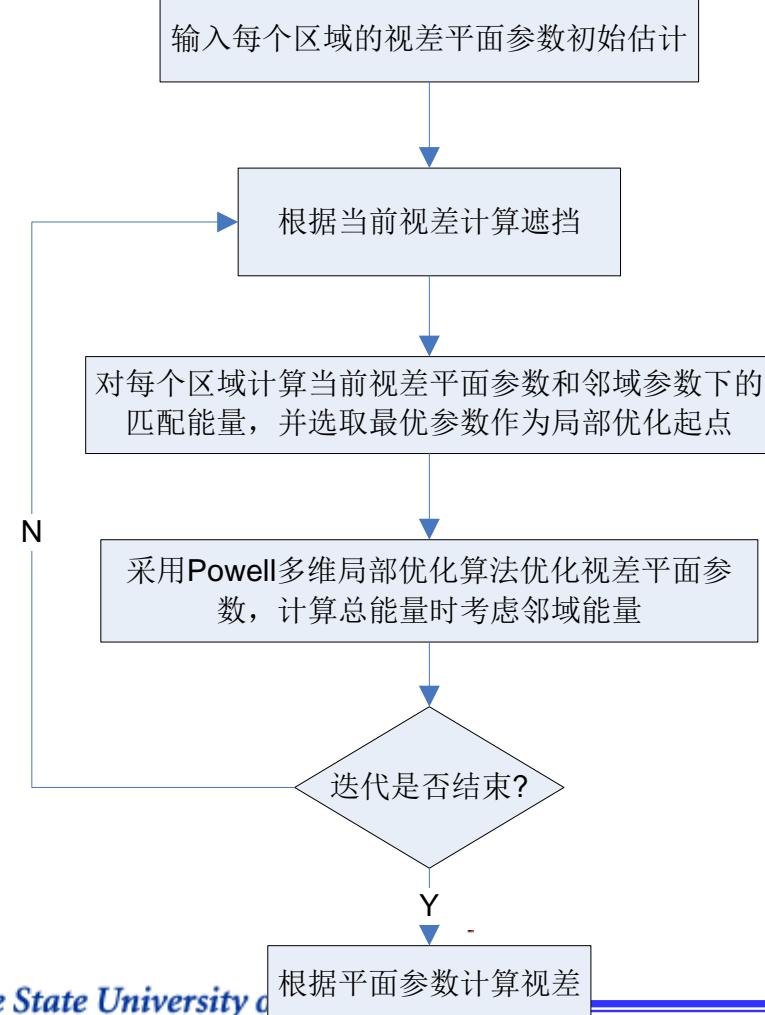
考虑参考图像上的两个相邻区域A和B，A在B的右边。当A区域在相邻边界处的视差大于其左邻域B的视差时，那么根据当前的视差计算结果将A和B映射到右图像时，B'的一部分（图中的D区域）将被A'所遮挡（左遮挡），而当A区域在相邻边界处的视差小于其左邻域B的视差时，映射到右图像后的两个区域A'和B'之间将会出现空隙（图中的C区域），相当于右图像上A'区域有一部分在左图像上被遮挡了（右遮挡）。

## 基于区域间协同优化的立体匹配

$$E_{occlude} = (|Occ_L| + |Occ_R|) \lambda_{occ}$$

其中， $|Occ_L|$  和  $|Occ_R|$  分别表示左遮挡和右遮挡像素的个数，而  $\lambda_{occ}$  表示所设置的遮挡惩罚常量。

# 基于区域间协同优化的立体匹配



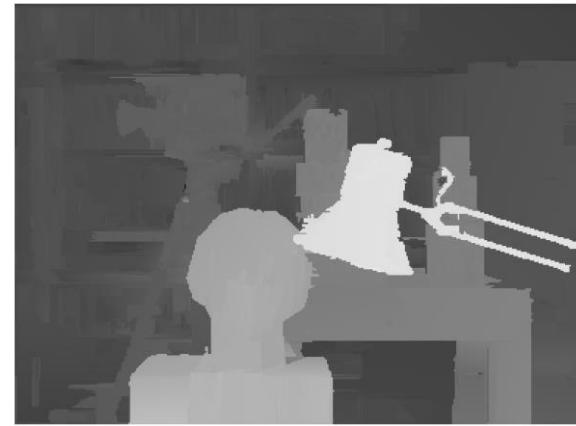
## 基于区域间协同优化的立体匹配

- 计算完每个区域的能量后，选用局部优化方法对所有区域的视差平面参数进行迭代优化
- 当一个区域的平面参数变化时，它不仅会影响本身区域的能量，同时也会对邻域的能量产生影响。
- 这里的邻域包括所有可能带来能量变化的区域，即在视差搜索范围内的所有邻域

## 协同优化算法应用于标准图像对Tsukuba



(a) 第一次  $e=516622.0$



(b) 第二次  $e=487985.0$



(c) 第三次  $e=473195.0$



(d) 第四次  $e=467576.0$

## 基于区域间协同优化的立体匹配

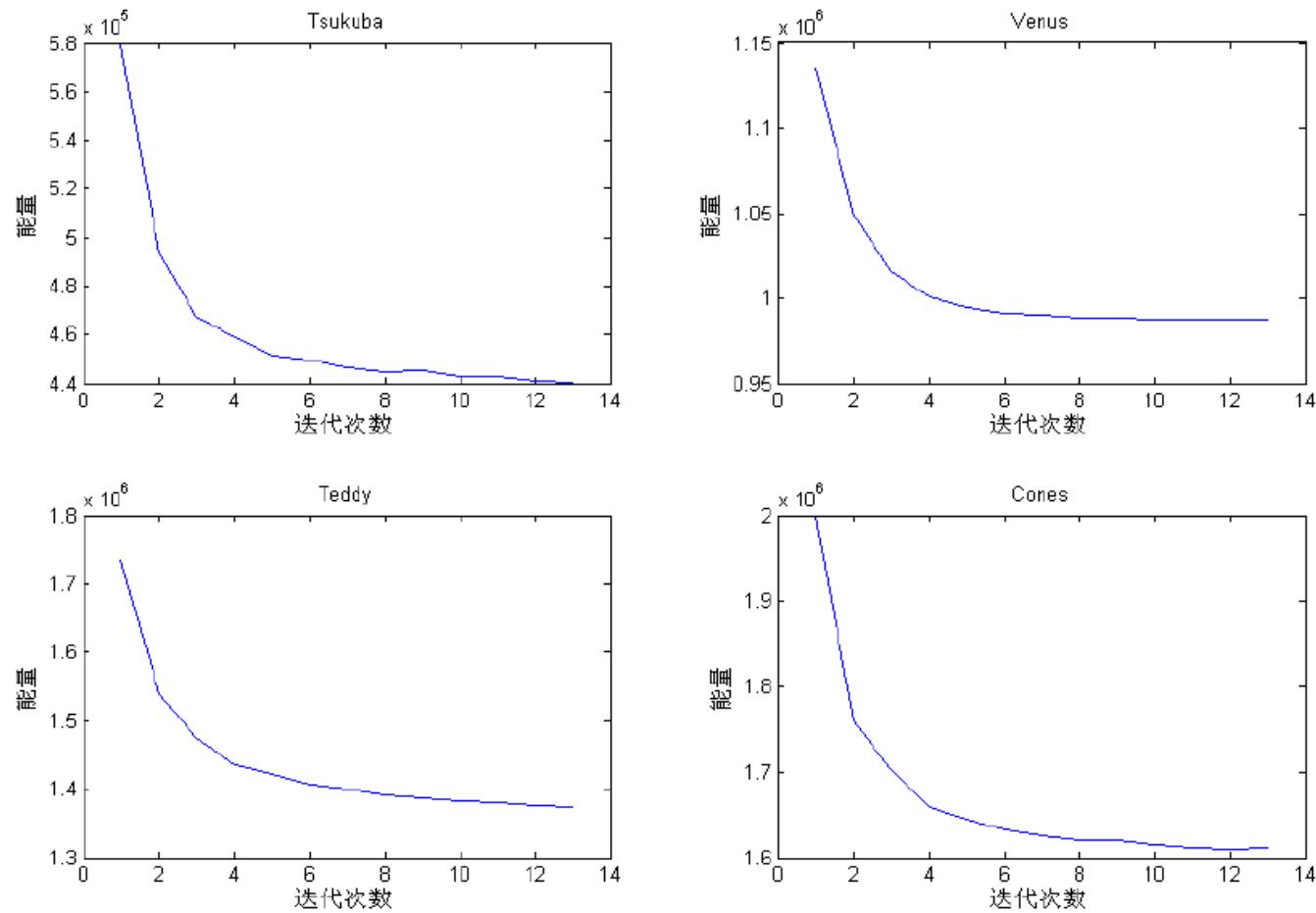
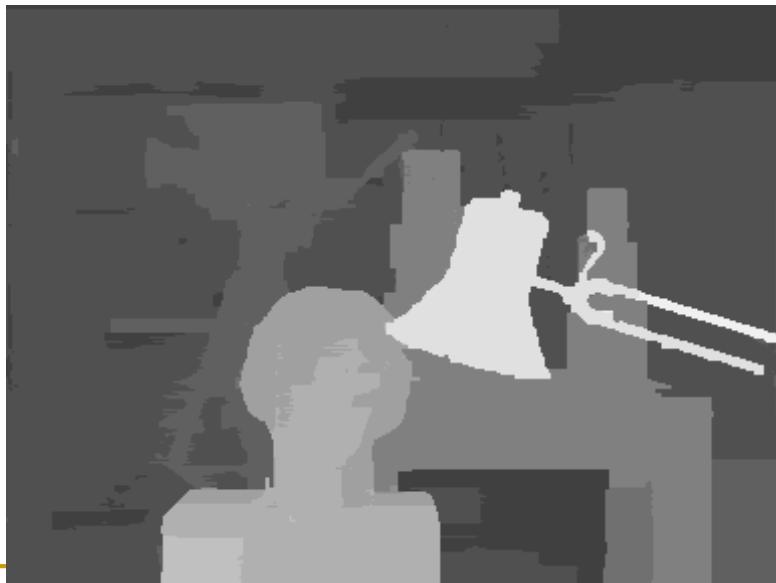
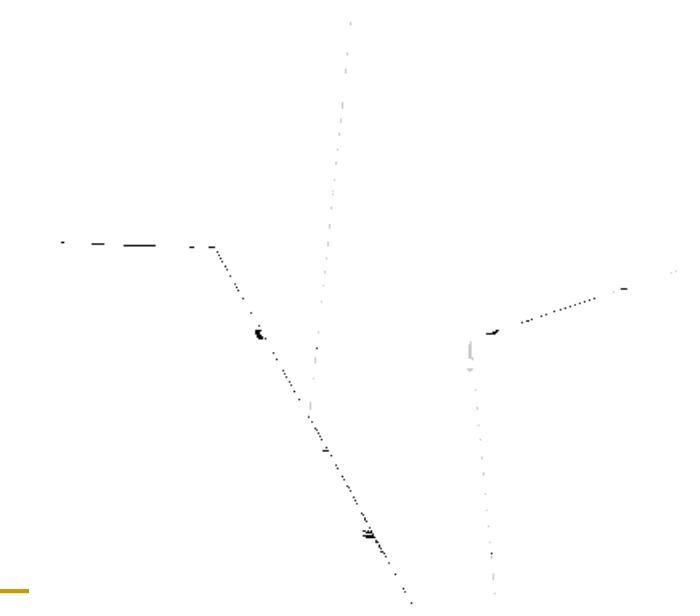
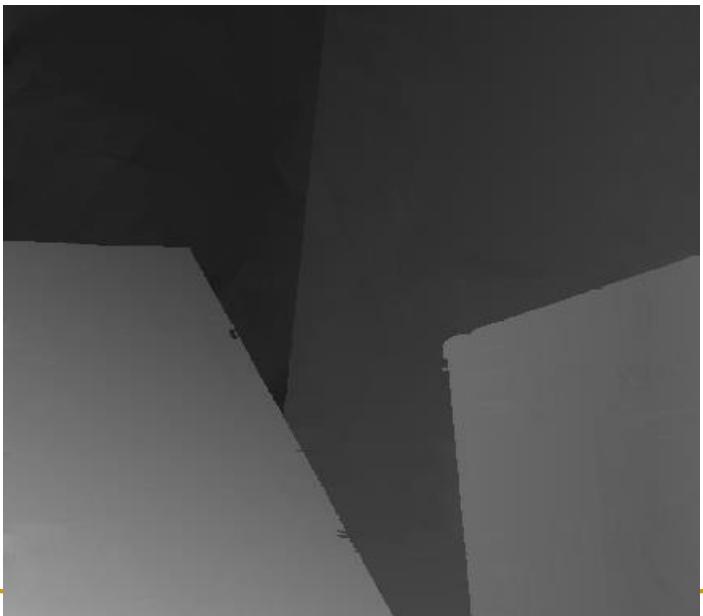
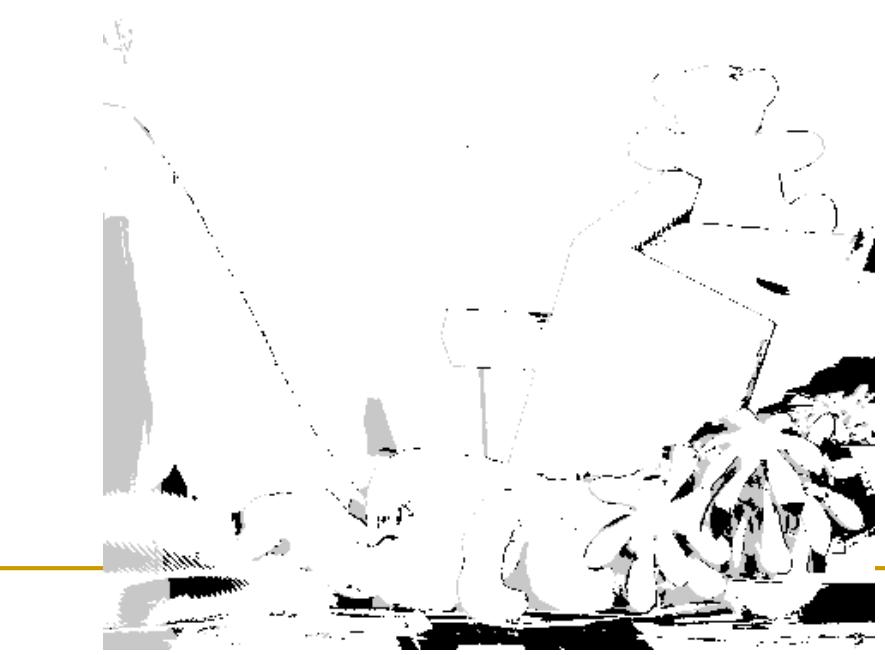
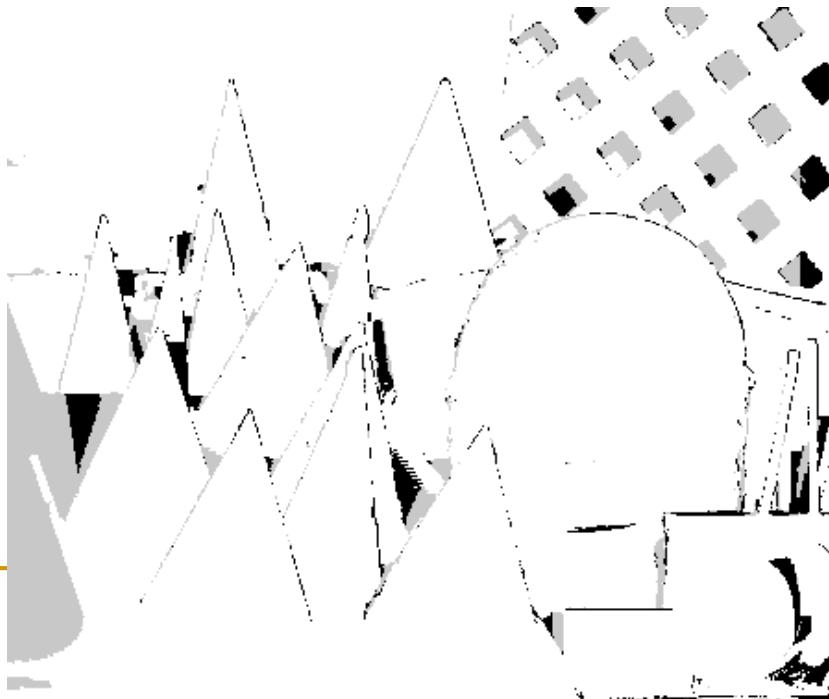


图 5- 9 将协同优化算法应用于 Middlebury 标准图像对能量下降的情况

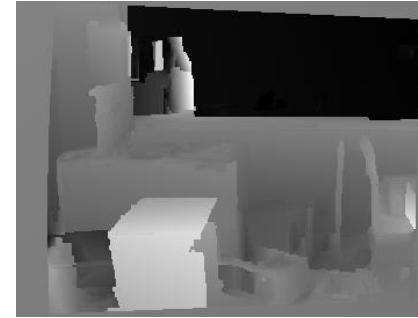








# 实拍图像



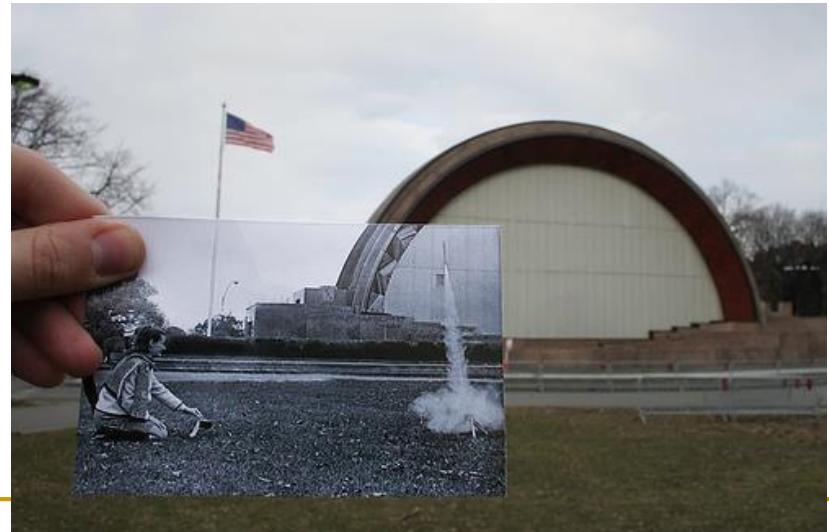
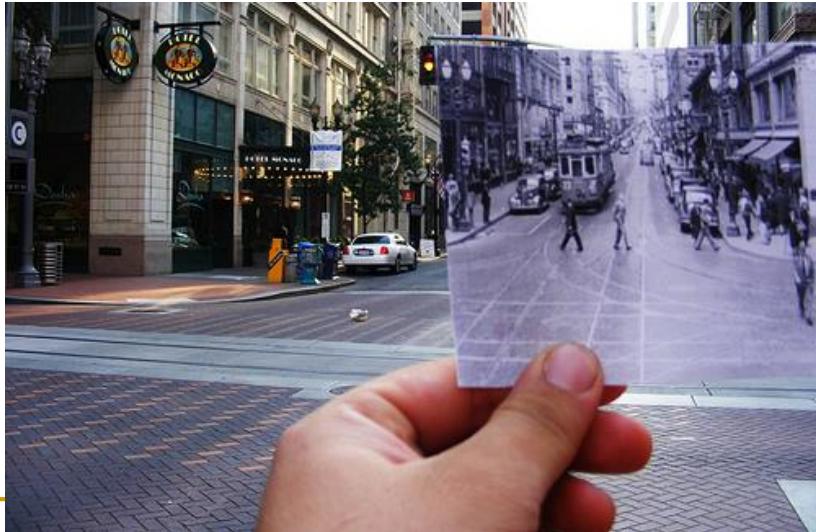
# Schedule

<b>09.02 (3 lectures)</b>	<b>Introduction</b>	<b>10.21 (3 lectures)</b>	<b>Reorganization II</b>
<b>09.09 (3 lectures)</b>	<b>Recall: Image Processing I</b>	<b>10.28 (3 lectures)</b>	<b>Reconstruction I</b>
<b>09.16 (3 lectures)</b>	<b>Recall: Image Processing II</b>	<b>11.04 (3 lectures)</b>	<b>Reconstruction II</b>
<b>09.23 (3 lectures)</b>	<b>Representation I</b>	<b>11.11 (3 lectures)</b>	<b>Recognition I</b>
<b>09.30 (3 lectures)</b>	<b>Representation II</b>	<b>11.18 (3 lectures)</b>	<b>Recognition II</b>
<b>10.07 (3 lectures)</b>	<b>Representation III</b>	<b>11.25 (3 lectures)</b>	<b>Recognition III</b>
<b>10.14 (3 lectures)</b>	<b>Reorganization I</b>	<b>12.02 (1 lecture)</b>	<b>Final Report</b>

# 大作业：（任选）

- 基于图像拼接技术实现**Look into past**
- 基于增强现实技术实现自己的某个想法

# A look into the past



<http://blog.flickr.net/en/2010/01/27/a-look-into-the-past/>

# A look into the past

## ■ Leningrad during the blockade



<http://komen-dant.livejournal.com/345684.html>

# Bing streetside images



<http://www.bing.com/community/blogs/maps/archive/2010/01/12/new-bing-maps-application-streetside-photos.aspx>

# 单应性拟合

- 齐次坐标

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Converting *to* homogenous  
image coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Converting *from* homogenous  
image coordinates

# 单应性拟合

- 齐次坐标

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Converting *to* homogenous  
image coordinates

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

Converting *from* homogenous  
image coordinates

- 单应性方程

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

# 单应性拟合

- 单应性方程

$$\lambda \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$
$$\lambda \mathbf{x}'_i = \mathbf{H} \mathbf{x}_i$$
$$\mathbf{x}'_i \times \mathbf{H} \mathbf{x}_i = 0$$

$$\begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \times \begin{bmatrix} \mathbf{h}_1^T \mathbf{x}_i \\ \mathbf{h}_2^T \mathbf{x}_i \\ \mathbf{h}_3^T \mathbf{x}_i \end{bmatrix} = \begin{bmatrix} y'_i \mathbf{h}_3^T \mathbf{x}_i - \mathbf{h}_2^T \mathbf{x}_i \\ \mathbf{h}_1^T \mathbf{x}_i - x'_i \mathbf{h}_3^T \mathbf{x}_i \\ x'_i \mathbf{h}_2^T \mathbf{x}_i - y'_i \mathbf{h}_1^T \mathbf{x}_i \end{bmatrix}$$

$$\begin{bmatrix} 0^T & -\mathbf{x}_i^T & y'_i \mathbf{x}_i^T \\ \mathbf{x}_i^T & 0^T & -x'_i \mathbf{x}_i^T \\ -y'_i \mathbf{x}_i^T & x'_i \mathbf{x}_i^T & 0^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0$$

3 equations,  
only 2 linearly  
independent

# 直接线性变换

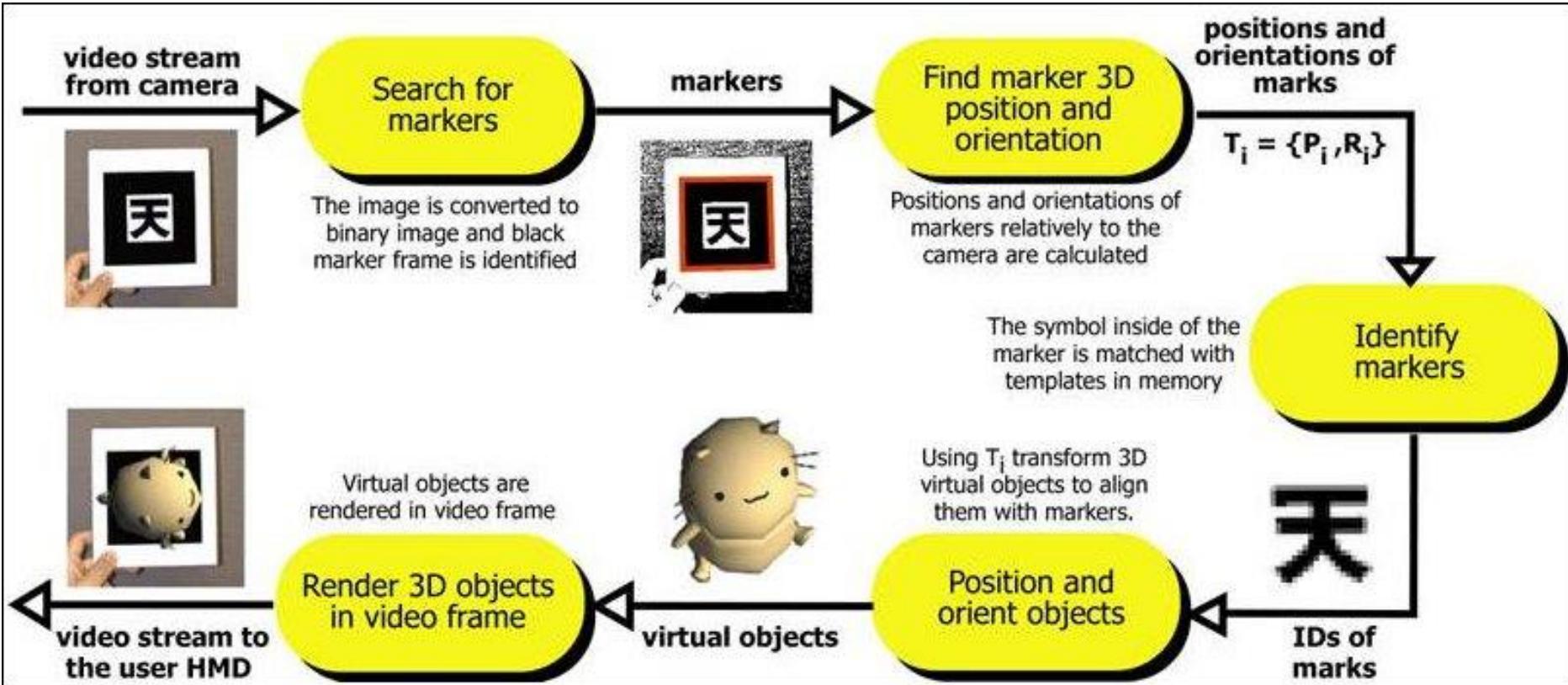
$$\begin{bmatrix} 0^T & \mathbf{x}_1^T & -y'_1 \mathbf{x}_1^T \\ \mathbf{x}_1^T & 0^T & -x'_1 \mathbf{x}_1^T \\ \dots & \dots & \dots \\ 0^T & \mathbf{x}_n^T & -y'_n \mathbf{x}_n^T \\ \mathbf{x}_n^T & 0^T & -x'_n \mathbf{x}_n^T \end{bmatrix} \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{pmatrix} = 0 \quad \mathbf{A}\mathbf{h} = 0$$

- $\mathbf{H}$  有8个自由度 (9个参数, 但尺度是任意的)
- 一个匹配特征对能够建立两个独立方程
- 最少要4个 匹配特征对
- 一般需要多于4个点对: 通过最小二乘得到最优解

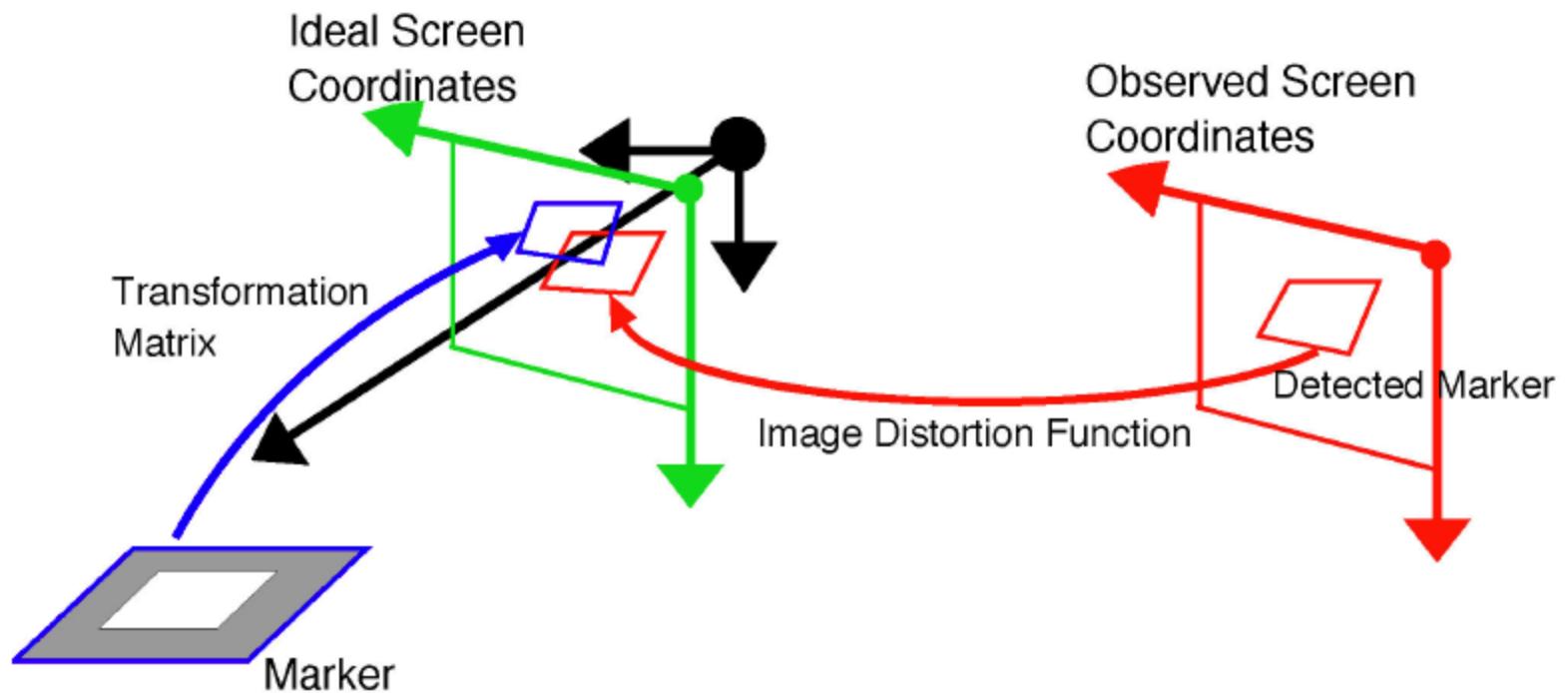
# 增强现实



# 增强现实



# 增强现实



## 考核方式

项目报告：2人一组，从四个候选项目中任选一个，实现并提交项目报告以及源代码。