浙江工业大学

本科毕业设计论文

外文翻译

题目：双目视觉立体匹配算法设计与实现

作者姓名 王 灏

指导教师 宣琦研究员

专业班级 通信工程 1301

学　　院 信息工程学院

提交日期 2017 年 2 月 28 日

通过训练卷积神经网络比较图像块的立体匹配

Jure Zbontar，卢布尔雅那大学计算机与信息科学学院

Yann LeCun，纽约大学 Courant 数学科学研究所

摘要:我们提出了一种从经过矫正的图像对中提取深度信息的方法。我们的方法集中在许多立体视觉算法的第一阶段：匹配代价计算。我们通过使用卷积神经网络进行小图像块的相似性度量来解决这个问题。我们使用相似和不相似的图像对构建二元分类数据集来进行有监督的训练。针对这个任务，我们检测了两个网络架构：一个是速度导向的， 另一个是准确性导向的。卷积神经网络的输出用于初始化立体匹配代价。一系列后处理步骤如下：基于交叉的代价聚合，半全局匹配，左右一致性检查，亚像素增强，一个中值滤波器和一个双边滤波器。我们用 KITTI 2012，KITTI 2015 和 Middlebury 的立体视觉数据集来评估我们的方法，并发现它在所有的三大数据集中的表现均优于其他方法。

关键词：立体视觉，匹配代价，相似性学习，监督学习，卷积神经网络

1．介绍

考虑以下问题：给定由不同水平位置的相机拍摄的两个图像，我们希望计算左图像中每个像素的视差 d。 视差指的是同一对象在左图像和右图像中的水平位置的差异——同一对象在左图像中的位置为（x，y），在右图像中的位置为（x-d，y）。 如果我们知道一个对象的视差，我们可以使用以下关系计算它的深度 z：

$$z = \frac{fB}{d}$$

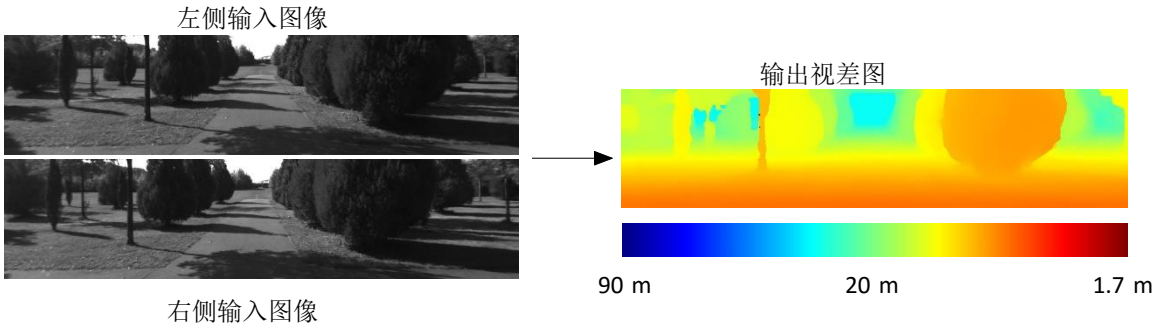

图 1： 输入是来自左侧和右侧相机的一对图像。 两个输入图像的差异主要在对象的水平位置上（其他的差异是由反射，遮挡和透视失真引起的）。注意靠近相机的对象比远离相机的对象有更大的视差。右图的输出是一个密集的视差图，暖色表示更大的视差值（和较小的深度值）

其中 f 是相机的焦距，B 是相机中心之间的距离。图 1 描述了输入和通过我们的方法处理的输出。

上述的立体匹配问题在许多领域中都是非常重要的，例如自动驾驶，机器人技术，中间视图生成和 3D 场景重建。 根据 Scharstein 和 Szeliski （2002）的分类，一个典型的视觉算法包括四个步骤：匹配代价计算，代价聚合，优化，视差精化。 根据 Hirschmüller 和 Scharstein （2009）的分类，我们认为前两个步骤是计算匹配代价和后两个步骤为立体视觉法。我们工作的重点是计算好匹配代价。

我们建议对图像块对训练卷积神经网络（ LeCun 等人，1998 年），其中真实的视差是已知的（例如，由激光雷达或结构光获得）。网络的输出用于初始化匹配代价。我们还需进行许多不是新颖的但是必要的后处理步骤以取得良好效果。匹配代价是由具有相似图像强度的邻近像素通过基于交叉的代价聚合的方式组合而成。平滑约束通过半全局匹配进行，同时左右一致性检查被用来检测和消除遮挡区域中的误差。我们进行亚像素增强并应用中值滤波器和双边滤波器以获得最终视差图。

本文的贡献是：

- 基于卷积神经网络的为计算立体匹配代价的两种架构的描述;
- 一种方法，伴随其源代码，在 KITTI 2012，KITTI 2015 和 Middlebury 立体视觉数据集中具有最低的错误率;
- 实验分析了数据集大小的重要性，与其他方法相比的错误率，以及不同超参数设置下精度和运行时间之间的权衡。

本文延伸了我们以前的工作（Zbontar 和 LeCun，2015 年），包括一个新架构

的描述，两个新数据集的结果，更低的错误率以及更彻底的实验。

2．相关工作

在引入大型立体数据集如 KITTI 和 Middlebury 之前，相对较少立体视觉算法使用标签信息来得到他们模型的参数;在这一节，我们回顾一下一些以前的做法。有关立体视觉算法的一般概述，请参阅 Scharstein 和 Szeliski（2002）。

Kong 和 Tao (2004) 采用平方距离的总和来计算初始匹配代价。然后他们训练了一个模型来预测三个类别的概率分布：初始视差正确的，由于前景目标过大导致初始视差不正确的，并且由于其他原因导致初始视差不正确的。预测概率被用来调整初始匹配代价。Kong 和 Tao(2006)随后延伸了他们的工作，通过组合由计算归一化的不同的窗口大小和中心的互相关获得的预测。Peris 等人（2012）用 AD-Census（Mei 等，2011）进行初始化匹配代价，并使用多类线性判别分析来得知从计算的匹配代价到最终视差的映射。

标签数据还用于得到概率图形模型的参数。Zhang 和 Seitz（2007）使用一种替代优化算法来估算马尔科夫随机场超参数的最优值。Scharstein 和 Pal（2007）构建了一个新的 30 个立体对的数据集，并使用它来得到条件随机场的参数。Li 和 Huttenlocher（2008）提出了一个带非参数代价函数的条件随机场模型，并使用结构化支持向量机来得到模型参数。

最近的工作（Haeusler 等人，2013; Spyropoulos 等人，2014）集中在估计计算的匹配代价的置信度。Haeusler 等人（2013）使用了一种随机森林分类器来组合若干置信度度量方式。同样的，Spyropoulos 等人（2014）训练了一个随机森林分类器来预测匹配代价的置信度，并且使用预测结果作为马尔科夫随机场中的软约束来减少立体视觉法的误差。

一个计算匹配代价的相关问题是得到局部图像描述符（Brown 等人，2011; Trzcinski 等人，2012; Simonyan 等人，2014; Revaud 等人，2015; Paulin 等人，2015；Han 等人，2015; Zagoruyko 和 Komodakis，2015）。这两个问题共享一个公共子任务：测量图像块之间的相似性。Brown 等人（2011）提出了一个总体框架来得到图像描述符并使用鲍威尔的方法来选择良好的超参数。为得到局部图像描述符已经提出了几种解决方法，例如 Boosting 优化（Trzcinski 等人，2012），

凸优化（Simonyan 等人，2014 年），分层移动象限的相似性（Revaud 等人，2015），卷积内核网络（Paulin 等人，2015），和卷积神经网络（Zagoruyko 和 Komodakis，2015；Han 等人，2015）。Zagoruyko 和 Komodakis（ 2015）的工作，Han 等人（2015），特别是，非常类似于我们自己，不同的主要是网络的架构;具体地，包括合并和子采样以考虑更大的块尺寸和更大的视点变化。

3．匹配代价

典型的立体视觉算法首先计算在每个位置 **p** 考虑所有视差 d 的匹配代价。一种计算匹配代价的简单方法是绝对差的总和：

$$C_{SAD}(\mathbf{p}, d) = \sum_{q \in Np} |I^L(\boldsymbol{q}) - I^R(\boldsymbol{q} - d)| \quad (1)$$

其中，$I^L(\boldsymbol{q})$和$I^R(\boldsymbol{p})$是位置 **p** 在左右图像中的图像强度，N_p是以 P 为中心的固定的矩形窗口内的位置集合。

我们使用粗体小写字母 **p** 和 **q** 表示图像位置。粗体小写 **d** 表示向量的视差 d，即 **d**=（**d**，0）。我们为超参数的名称使用 typewriter 字体。例如，我们将使用 patch size 字体来表示附近区N_p的大小

等式（1）可以理解为测量匹配左图以位置 **p** 为中心的图像块与右图以位置 **p–d** 为中心的图像块的代价。

我们希望当两个图像块围绕相同的 3D 点时代价小，反之则代价大。

既然好的和坏的匹配的示例可以从公开可用数据集构建（例如，KITTI 和 Middlebury 立体数据集），我们可以尝试通过监督学习方法来解决匹配问题。受到卷积神经网络在视觉问题中成功应用的启发，我们用它来评估两个小图像块匹配程度。

3.1 构造数据集

我们使用来自 KITTI 或 Middlebury 立体视觉数据集的地面实况视差图来构建二元分类数据集。在每个真实视差已知的图像位置处，我们提取一个消极的和一个积极的训练示例。这确保了数据集包含相等数量的正例和负例。一个积极的

示例是指一对图像块，其中一个来自左边图像，一个来自右边图像，两者的中心像素是同一个 3D 点，而一个消极的示例是指一对不同于前者的图像块。以下部分详细描述数据集构建步骤。

用$< P^L_{n\times n}(\mathbf{p}), P^R_{n\times n}(\mathbf{q}) >$表示一对图像块，其中$P^L_{n\times n}(\mathbf{p})$为左图中以位置 **p**=（x，y）为中心的 n×n 图像块，$P^R_{n\times n}(\mathbf{q})$是右图中以位置 **q** 为中心的 n×n 图像块，d 表示位置 **p** 处的正确视差。一个消极的示例的获取是通过将右图像块的中心设置为

$$\mathbf{q} = (\mathrm{x} - \mathrm{d} + o_{neg}, \mathrm{y})$$

其中o_{neg}是从区间[dataset_neg_low, dataset_neg_high]或它的原点对应区间[−dataset_neg_high, −dataset_neg_low]中选出的。随机偏移o_{neg}确保了生成的图像块以不同与前者的 3D 点为中心。

一个积极示例的获取则是通过设置

$$\mathbf{q} = (\mathrm{x} - \mathrm{d} + o_{pos}, \mathrm{y})$$

其中o_{pos}是从区间[−dataset_pos, dataset_pos]中随机挑选的。上式中包含o_{pos}而不是将其设为 0 的原因与后续使用的立体视觉法有关。特别是，我们发现当网络将低匹配代价分配给良好匹配和相近的匹配时基于交叉的代价聚合表现更好。在我们的实验中，超参数 dataset_pos 永远不会大于一个像素。

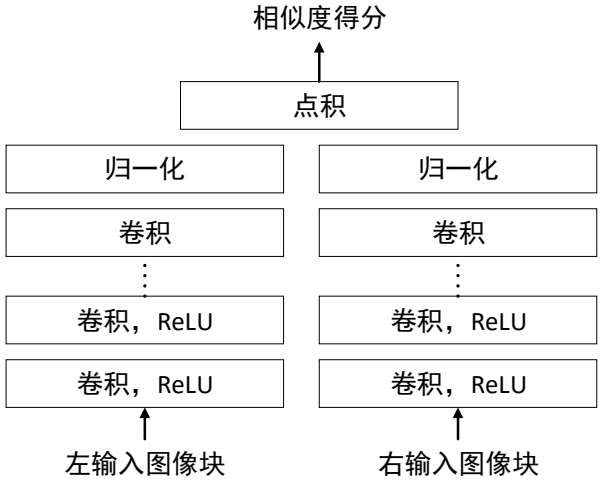

图 2： 这个快速架构是一个暹罗网络。 这两个子网络由多个卷积层和紧随其后的整流线性单元（缩写为 “ReLU”）组成。通过从每次输入的两个图像块中提取向量并计算它们之间的余弦相似性来获得相似度得分。 在上图和我们的具体实现中，余弦相似性计算可分为两个步骤：归一化和点积。

因为每个位置仅需要执行一次归一化操作，所以这种方法减少了运行时间（详见 3.3 节）

3.2 网络架构

为了比较图像块上的相似度，我们提出了两种网络架构。第一种网络架构比第二种运行速度更快，但产生的视差图不太准确。在这两种架构下，网络的输入是一对小图像块，输出是它们之间的相似性的度量。两个架构都包含一个可训练的用特征向量表示每个图像块的特征提取器。图像块之间的相似性通过特征向量上而不是原始图像强度值衡量。快速架构使用固定的相似性度量来比较两个特征向量，而精确架构试图从特征向量中得到一个更好的相似性度量。

3.2.1 快速架构

第一种架构是暹罗网络，即两个共享权重子网加在头部（Bromley 等人，1993）。子网络由多个带整流线性单元（除最后一层外）的卷积层组成。两个子网输出一个带有输入图像块属性的向量。得到的两个向量比较使用余弦相似性度量来产生网络的最终输出的。图 2 提供了该架构的概述。

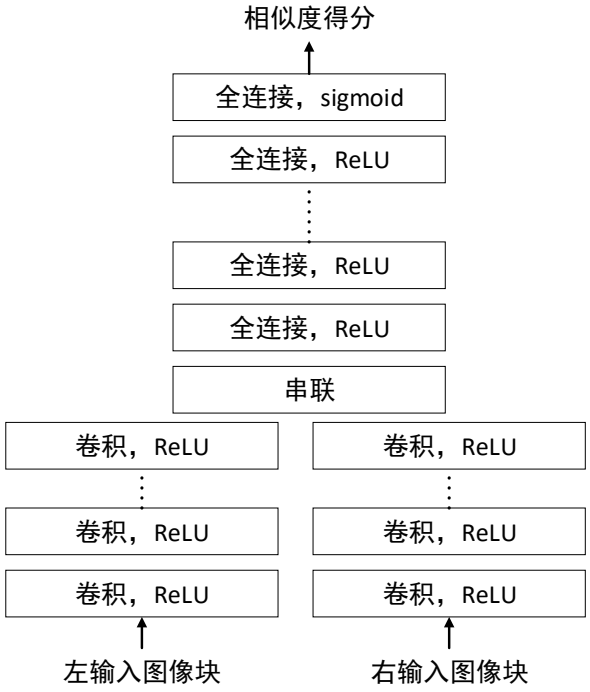

图 3： 精确架构从两个卷积特征提取器开始。提取的特征向量串联并通过多个全连接层比较。输入是两个图像块，输出是介于 0 和 1 之间的单个实数，

我们称这个实数为输入图像相似性的度量。

通过使铰链损耗最小化来训练网络。损失是通过以同一图像位置为中心的含积极和消极两类的示例对来计算的。设s_+是积极实例网络的输出，s_-是消极实例网络的输出，并让余量 m 为正实数。该实例对的铰链损失定义为 max（0，$m+s_- - s_+$）。当积极实例的相似性大于消极实例至少余量 m 时，损失为零。在我们的实验中，我们设置余量为 0.2。

该架构的超参数是每个子网中卷积层的数量（num_conv_layers）,卷积核的大小（conv_kernel_size）,每层中的特征映射的数量（num_conv_feature_maps），以及输入图像块的大小（input_patch_size）。

3.2.2 精确架构

第二种架构是从第一种架构中通过用一些全连接层替换余弦相似性度量得到的（见图 3）。

这种架构的变化增加了运行时间，但降低了错误率。两个子网包括很多卷积层，在每层之后紧跟着整流线性单元。两个结果向量串联并通过多个紧跟整流线性单元全连接层前向传播。最后一个全连接层经过 sigmoid 非线性变换输出一个数字，这个数字就是两图像块的相似度得分。

我们使用二元交叉熵损失函数来进行训练。令 s 表示一个训练示例的网络输出，t 表示该训练示例的类别;如果示例属于积极示例的类别，则 $t=1$，如果示例属于消极示例的类别，则 $t=0$。该示例的二元交叉熵损失被定义为$t\log(s) + (1 - t)\log(1 - s)$。

两个架构使用不同的损失函数是基于实验性的证据。虽然我们宁愿对两个架构使用相同的损失函数，但实验表明在精确架构中，二元交叉熵损失函数表现得比铰链损耗函数更好。另一方面，虽然快速架构的最后一步是余弦相似度计算，但交叉熵损失不直接适用。

精确架构的超参数是在每个子网络中卷积层的数量（num_conv_layers），每个层中的特征映射的数量（num_conv_feature_maps），卷积内核的大小（conv_kernel_size），输入图像块的大小（input_patch_size），每个全连接层中的单元数（num_fc_units），以及全连接层数（num_fc_layers）。

3.3 计算匹配代价

网络的输出用于初始化匹配代价：

$$C_{CNN}(\boldsymbol{p},d) = -s(< P^L(\boldsymbol{p}),\ P^R(\boldsymbol{p}-d) >)$$

其中，$s(< P^L(\boldsymbol{p}),\ P^R(\boldsymbol{p}-d) >)$是当网络输入图像块$P^L(\boldsymbol{p})$和$P^R(\boldsymbol{p}-d)$时的输出。负号将相似度分数转换为匹配代价。要计算整个匹配的成本代价张量$C_{CNN}(\boldsymbol{p},d)$，我们仅需要对每个图像位置和每个需考虑的视差执行前向传递。以下三个实现细节保证了运行时间可管理：

- 两个子网络的输出每个位置只需计算一次，并且不需要针对每个需考虑的差异重新计算。
- 通过传播全分辨率图像，可以为单次前向传递中的所有像素点计算两个子网络的输出，而不仅仅是一个小图像块。在整个 w×h 图像上执行单次前向传递比执行 w*h 次小图像块的前向传递更快，因为许多中间结果可以重复使用。
- 全连接层在精确架构中的输出也可以在单次前向传递中计算。这是通过用一个 1×1 内核的卷积层替换每个全连接层实现的。我们还需要执行每个需考虑的视差的前向传递;对于 KITTI 数据集，最大视差 d 为 228，对于 Middlebury 数据集，最大视差为 400。因此，网络的全连接部分需要运行 d 次，这是精确架构的瓶颈。

为计算一对图像的匹配代价，我们每张图片在子网上运行 1 次，在全连接层运行 d 次，其中 d 是需考虑的最大视差。这个理解在设计网络的架构中是重要的。我们可能会选择两个图像在提交给网络之前级联的架构，但这意味着在运行时的大量代价，因为整个网络将需要运行 d 次。这种理解也导致了快速架构的发展，快速架构中唯一需要运行 d 次的层是特征向量的点积。

4. 立体视觉法

卷积神经网络的原始输出不足以产生准确的视差图，特别是在低纹理区域和遮蔽区域存在明显的误差。通过应用一系列后处理来改善视差图的质量，这就是

我们称之为立体视觉法的步骤。我们所使用的立体视觉方法是受到 Mei 等（2011）的论文的影响，其中包括基于交叉的代价聚合，半全局匹配，左右一致性检查，亚像素增强，中值和双边滤波器。

4.1 基于交叉的代价聚合

来自相邻像素的信息可以通过对一个固定窗口内的匹配代价进行平均来组合。但这种方法在深度不连续点附近失效，在那些点违反了窗口内深度恒定的设定。我们可能更喜欢自适应为每个像素选择邻域的方法，那样信息仅从相同的物理对象的像素中获取。在基于交叉的代价聚合（Zhang 等人，2009）中，我们在每个位置周围建立了一个本地邻域，邻域中包括类似图像强度值的像素，我们希望这些像素属于同一个对象。

该方法开始于在每个位置构建直立十字; 用这个十字来定义本地支持区域。只要以下两个条件成立，左臂P_l就在位置 **p** 向左延伸：

- $|\mathrm{I}(\mathbf{p}) - \mathrm{I}(P_l)|$ < cbca_intensity; 位置 P 和P_l的图像强度应该是类似，它们的差异应小于 cbca_intensity。
- 对$||\mathbf{p} - \mathrm{P_l}||$ < cbca_distance; 位置 P 和P_l的水平距离（在上臂或下臂时为垂直距离）小于 cbca_distance 像素。

右臂，下臂和上臂的构建都是与前者类似的。一旦四个臂确定的，我们就可以计算支持区域 U（**p**）为铺设在位置 **p** 垂直臂上所有的位置 **q** 的水平臂的并集（见图 4） 。

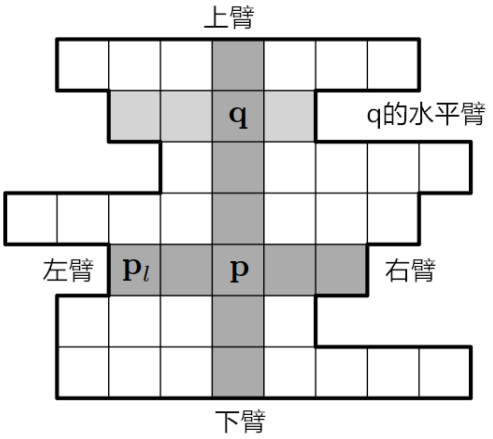

图 4： **p** 的支持区域是铺设在位置 p 垂直臂上所有的位置 q 的水平臂的并集

Zhang 等人（2009）认为，聚合应考虑立体对的两张图片的支持区域。令U^L型和U^R表示在左，右图片的支持区域。我们定义共同支持区域 Uð 为：

$$U_d(\boldsymbol{p}) = \{q | q \in U^L(\boldsymbol{p}), q - d \in U^R(\boldsymbol{p} - \boldsymbol{d})\},$$

在共同支持区域内对匹配成本进行平均：

$$C^0_{CBCA}(\boldsymbol{p}, d) = C_{CNN}(\boldsymbol{p}, d),$$

$$C^i_{CBCA}(\boldsymbol{p}, d) = \frac{1}{|U_d(\boldsymbol{p})|}\sum_{q \in U_d(\boldsymbol{p})} C^{i-1}_{CBCA}(\boldsymbol{q}, d),$$

其中 i 是迭代数。我们重复平均次数。因为支持区域是重叠的，每次迭代计算结果都可能改变。我们在快速架构中跳过基于交叉的代价聚合，因为它对于实现低错误率没有决定性作用而且计算开销相对较大。

4. 2 半全局匹配

我们通过对视差图像实施平滑约束来改善匹配代价。根据 Hirschmüller（2008）所说，我们定义一个依赖于视差图 D 的能量函数 E(D)的：

$$\mathrm{E(D)} = \sum_{\boldsymbol{p}}(C^4_{CBCA}\big(\boldsymbol{p}, D(\boldsymbol{p})\big) + \sum_{q \in N_{\boldsymbol{p}}} P_1 \cdot 1\{|D(\boldsymbol{p}) - D(\boldsymbol{q})| = 1\}$$

$$+ \sum_{q \in N_{\boldsymbol{p}}} P_2 \cdot 1\{|D(\boldsymbol{p}) - D(\boldsymbol{q})| > 1\}),$$

其中 1{·} 表示指标函数。第一项惩罚了高匹配代价的视差。当相邻像素的出现一个视差不同点时，第二项增加一个罚因子 P_1 。当相邻像素的视差不同点超过一个时，第三项增加了一个更大的罚因子 P_2 。

我们可以通过动态规划在单一方向上实现 E（D）最小化，而不是同时在所有方向上最小化。因为在我们没有优化的方向上没有激励来使视差图平滑，所以这个解决方案会引入不必要的拖尾效应。在半全局匹配中，我们最小化单个方向的能量，并重复几个方向，最后取平均来得到最终结果。虽然 Hirschmüller（2008）建议选择 16 个方向，但我们只沿两个水平方向和两个垂直方向优化; 添加对角线 方向并没有提高我们系统的准确性。为了最小化方向 $\mathbf{r}$ 上的 E（D），我们定义了一个具有以下递推关系的匹配代价$C_r(\boldsymbol{p}, d)$：

$$C_r(\boldsymbol{p}, d) = C^4_{CBCA}(\boldsymbol{p}, d) - \min_k C_r(\boldsymbol{p} - \boldsymbol{r}, k) + \min\{C_r(\boldsymbol{p} - \boldsymbol{r}, d), C_r(\boldsymbol{p} - \boldsymbol{r}, d - 1) +$$

$$P_1, C_r(\boldsymbol{p} - \boldsymbol{r}, d + 1) + P_1, \min_k C_r(\boldsymbol{p} - \boldsymbol{r}, k) + P_2\},$$

第二项是减法是为了防止$C_r(\boldsymbol{p},d)$的值过大以致于影响优化视差图。

惩罚因子P_1和P_2是根据图像梯度为使视差中的跳跃与图像的边缘一致而设置的。令D_1=$|I^L(\boldsymbol{p})-I^L(\boldsymbol{p}-\boldsymbol{r})|$和$D_2$=$|I^R(\boldsymbol{p}-d)-I^R(\boldsymbol{p}-d-\boldsymbol{r})|$是我们优化方向上两个相邻位置之间的图像强度的差异。我们按照以下规则设定P_1和P_2：

$$\begin{array}{lll} P_1=\text{sgm\_P1}, & P_2=\text{sgm\_P2} & \text{if } D_1<\text{sgm\_D}, D_2<\text{sgm\_D};\\ P_1=\text{sgm\_P1}/\text{sgm\_Q2}, & P_2=\text{sgm\_P2}/\text{sgm\_Q2} & \text{if } D_1\geq\text{sgm\_D}, D_2\geq\text{sgm\_D};\\ P_1=\text{sgm\_P1}/\text{sgm\_Q1}, & P_2=\text{sgm\_P2}/\text{sgm\_Q1} & \text{otherwise.}\end{array}$$

超参数 sgm_P1 和 sgm_P2 为视差图中的不连续点设置了基础惩罚因子。如果D_1或D_2中的一个表示高图像梯度，基础惩罚因子由 sgm_Q1 因子减少，如果D_1和D_2同时表示高图像梯度，则通过较大的 sgm_Q2 因子来减少。当进一步考虑两个垂直方向时，P_1的值将通过 sgm_V 因子减小;在地面的真实数据中，视差的微小变化在垂直方向上比在水平方向上更频繁，而且惩罚因子应该更小。

最终代价$C_{SGM}(\boldsymbol{p},d)$是在所有四个方向取平均值计算得到的：

$$C_{SGM}(\boldsymbol{p},d)=\frac{1}{4}\sum_{\boldsymbol{r}}C_r(\boldsymbol{p},d).$$

正如在上一节所说，在半全局匹配之后，我们重复进行基于交叉的代价聚合。 超参数 cbca_num_iterations_1 和 cbca_num_iterations_2 确定半全局前后基于交叉的代价聚合的迭代次数。

4.3 计算视差图

视差图 D(p)是通过胜者全拿的策略计算得到的，即寻找使 C(**p**，d)最小的视差 d，

$$\mathrm{D}(\mathbf{p})=\arg\min_{d}C(\boldsymbol{p},d).$$

4.3.1 插值

插值步骤试图解决从左右两张图片中预测得到的视差图之间的冲突。令D^L表示通过以左图为参考得到的视差图——这是到目前为止情况，即，D^L(P)=D(P) ,令D^R表示以右图为参考得到的视差图。D^L和D^R有时在一些特殊点的视差是不同的。我们通过左右一致性检查来检测这些冲突。我们通过应用以下规则来按顺序标记每个位置 **p**：

$$\begin{array}{ll} \text{正确} & \text{if } |d - D^R(\mathbf{p} - \mathbf{d})| \leq 1 \text{ for } d = D^L(\mathbf{p}), \\ \text{不匹配} & \text{if } |d - D^R(\mathbf{p} - \mathbf{d})| \leq 1 \text{ for any other } d, \\ \text{遮挡} & \text{其他情况} \end{array}$$

对于标记为遮挡的位置，我们希望从背景获取新的视差值。我们向左移直到我们找到一个标记为正确的位置并用它的值来内插。对于标记为不匹配的位置，我们在 16 个不同方向中寻找最近的正确像素，并取这些视差的中值进行插值。 我们称插值的视差图为D_{INT}。

4.3.2 亚像素增强

亚像素增强提供了增加立体视觉算法的分辨率的简单方法。我们通过相邻代价拟合二次曲线来获得新的视差图：

$$D_{SE}(\boldsymbol{p}) = d - \frac{C_+ - C_-}{2(C_+ - 2C + C_-)},$$

其中 d=$D_{INT}(\boldsymbol{p})$，$C_- = C_{SGM}(\boldsymbol{p}, d-1)$，C=$C_{SGM}(\boldsymbol{p}, d)$，$C_+ = C_{SGM}(\boldsymbol{p}, d+1)$.

4.3.3 细化

立体视觉方法的最后步骤包括一个 5×5 中值滤波器和下列的双边滤波器：

$$D_{BF}(\boldsymbol{p}) = \frac{1}{W(\boldsymbol{p})} \sum_{\boldsymbol{q} \in \boldsymbol{N_p}} D_{SE}(\boldsymbol{q}) \cdot g(||\boldsymbol{p} - \boldsymbol{q}||) \cdot 1\{|I^L(\boldsymbol{p}) - I^L(\boldsymbol{q})| < blur\_threshold\}$$

其中 g（x）是标准差为 blur_threshold 的零均值正态分布概率密度函数，W（p）是归一化常数，

$$\mathrm{W}(\boldsymbol{p}) = \sum_{\boldsymbol{q} \in \boldsymbol{N_p}} g(||\boldsymbol{p} - \boldsymbol{q}||) \cdot 1\{|I^L(\boldsymbol{p}) - I^L(\boldsymbol{q})| < blur\_threshold\}$$

双边滤波器的作用是平滑视差图而不模糊边缘。D_{BF}为我们的立体视觉方法的最终输出。

5．实验

我们在实验中使用了三个立体视觉数据集：KITTI 2012，KITTI 2015 和 Middlebury。表 1，2 和 4 中的测试集误差率是通过提交所生成的视差图到在线

评估服务器后获得。所有的其他错误率是通过将数据集分成两部分计算得到，我们使用一部分用于训练，另一部分用于验证。

表 1: 截至 2015 年 10 月 KITTI 2012 数据集的最高排名方法。“设置”列提供对视差图如何计算的解释：“F”表示光流的使用，“MV”表示两个以上的时间相邻图像，“MS”表示使用对极几何计算光流。“错误率”列表示错误分类像素的百分比，“运行时间”列表示处理一对图像所需的时间（以秒为单位）。

排名	方法		设置	错误率	运行时间
1	**MC-CNN-acrt**	**Accurate architecture**		2.43	67
2	Displets	Güney and Geiger (2015)		2.47	265
3	MC-CNN	Žbontar and LeCun (2015)		2.61	100
4	PRSM	Vogel et al. (2015)	F, MV	2.78	300
	MC-CNN-fst	**Fast architecture**		2.82	0.8
5	SPS-StFl	Yamaguchi et al. (2014)	F, MS	2.83	35
6	VC-SF	Vogel et al. (2014)	F, MV	3.05	300
7	Deep Embed	Chen et al. (2015)		3.10	3
8	JSOSM	Unpublished work		3.15	105
9	OSF	Menze and Geiger (2015)	F	3.28	3000
10	CoR	Chakrabarti et al. (2015)		3.30	6

表 2: 截至 2015 年 10 月 KITTI 2015 排行榜上的领先排名。“设置”，“错误率”和“运行”列意义和表 1 相同。

排名	方法		设置	错误率	运行时间
1	**MC-CNN-acrt**	**Accurate architecture**		3.89	67
	MC-CNN-fst	**Fast architecture**		4.62	0.8
2	SPS-St	Yamaguchi et al. (2014)		5.31	2
3	OSF	Menze and Geiger (2015)	F	5.79	3000
4	PR-Sceneflow	Vogel et al. (2013)	F	6.24	150
5	SGM+C+NL	Hirschmüller (2008); Sun et al. (2014)	F	6.84	270
6	SGM+LDOF	Hirschmüller (2008); Brox and Malik (2011)	F	6.84	86
7	SGM+SF	Hirschmüller (2008); Hornacek et al. (2014)	F	6.84	2700
8	ELAS	Geiger et al. (2011)		9.72	0.3
9	OCV-SGBM	Hirschmüller (2008)		10.86	1.1
10	SDM	Kostková and Sára (2003)		11.96	60

5.1 KITTI 立体视觉数据集

KITTI 立体数据集（Geiger 等人，2013; Menze 和 Geiger，2015）是从安装在汽车车顶上两个相隔约 54 厘米的摄像机拍摄的经整流的图像对的集合。图像是在白天晴朗和多云的天气下，在 Karlsruhe 市中心和郊区开车记录的。图像分辨率为 1240×376。旋转激光扫描仪安装在左侧相机后面，记录地面的真实深度，标记了约 30%的图像像素。

测试集的地面真实视差并未公开，而是提供了一个在线排行榜宫研究人员在

测试集上评估他们的算法。每隔三天允许提交一次。错误率是计算那些真实视差和预测视差相差超过三个像素的像素的比例。这就意味着，例如，错误的容许范围为 3 厘米则转换成物理距离为距离相机 2 米和 80 厘米的容许范围，就是距离相机 10 米。

存在两个 KITTI 立体视觉数据集：KITTI 2012 和较新的 KITTI 2015。为了完成立体视觉的计算任务，他们几乎是相同的，较新的数据集的改进了光流任务的一些方面。2012 年的数据集包含 194 个训练图和 195 个测试图像，而 2015 年的数据集包含 200 个训练图和 200 个测试图。较新的数据集介绍了一个微妙但重要的区别：运动车辆是密集标记的，并且汽车玻璃也被包括在评估中。这强调了该方法在反射面上的表现。

在 KITTI 2012 数据集表现最好的方法已经列在表 1 上。我们的精确架构排名第一，仅有 2.43％的误差率。在排行榜第三名是我们之前的工作成果（Zbontar 和 LeCun，2015），误差率为 2.61％。误差从 2.61％减少到 2.43％得益于两个改变——扩大了数据集（见第 5.4 节）和倍增了卷积层的数量，同时减少内核大小从 5×5 到 3×3。排在第二位的方法（Güney 和 Geiger，2015）用的是我们之前的方法计算匹配代价（Zbontar 和 LeCun，2015）。测试快速架构的错误率是 2.82％，这将是足够在排行榜上排到第五位。精确结构处理单一图像对的运行时间是 67 秒，而快速架构为 0.8 秒。图 5 包含了一对来自 KITTI 2012 数据集的例子，以及按我们的方法进行预测的结果。

表 2 列出了 KITTI 2015 数据集的领先者。我们方法中精确架构的错误率为 3.89％，而快速架构为 4.46％，占据了排行榜的第一和第二位。由于每篇论文只被允许提交一次，所以只有精确架构的结果出现在排行榜上。图 6 是 KITTI 2015 数据集按我们的方法产生视差图。

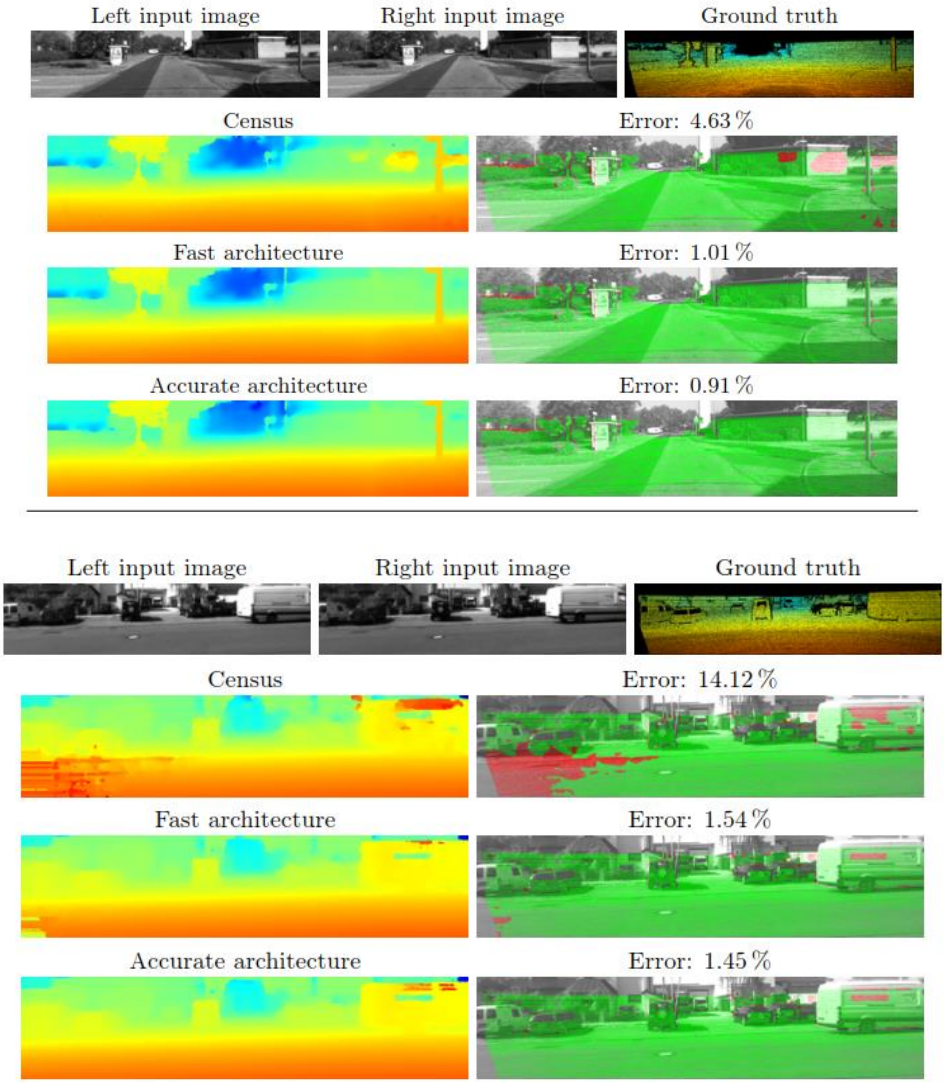

图 5: 对 KITTI 2012 数据集预测视差图的实例。注意图像的某些区域（顶部例子的白墙，底部例子的沥青）如何导致统计变换出现问题。快速和精确架构都有更好的表现，精确架构平均更少出现错误。

Year	Number of Image Pairs	Resolution	Maximum Disparity
2001	8	380×430	30
2003	2	1800×1500	220
2005	6	1400×1100	230
2006	21	1400×1100	230
2014	23	3000×2000	800

表 3: 五个 Middlebury 立体视觉数据集的总结。“Number of Image Pairs”列只计算有地面真实视差的图像对的数量。2005 年和 2014 年的数据集还包含了一些保留地面真实视差的图像对;这些图像对构成了测试数据集。

5.2 Middlebury 立体视觉数据集

Middlebury 立体视觉数据集的图像对来自光照可控的室内场景。数据通过结

构光来测量真实视差，视差的密度和精度都比在 KITTI 数据集好。该数据集共发布了五个独立的数据集，分别是在 2001 年，2003 年，2005 年，2006 年和 2014 年（Scharstein 和 Szeliski，2002，2003; Scharstein 和 Paul，2007; Hirschmüller 和 Scharstein，2007;Scharstein 等人，2014）。在本文中，我们指的 Middlebury 数据集是所有的五个数据集的串联;表 3 是各个数据集的情况总结。

在 2005 年，2006 年，和 2014 年数据集的每一个场景都是根据多种光照条件和快门曝光方式拍摄的，有一组典型的图像对是同一个场景在四种光照条件和七种快门曝光方式下拍摄的共计 28 张图片。

一个类似 KITTI 的在线排行榜，显示了所有提交算法的排名表。参与者只有一次提交结果的机会。这条规则比允许每三天提交一次的 KITTI 数据集更严格。测试数据集包含 15 幅来自 2005 年和 2014 年数据集的图像。

数据集提供全分辨率，半分辨率和四分之一分辨率的图片。误差率的计算是根据全分辨率的;如果算法输出的是半分辨率或四分之一分辨率的视差图，它们在计算错误率之前上采样。由于显卡内存有限，我们选择半分辨率的图片来运行我们的算法。

通过标准的校准程序来矫正图像，像 OpenCV 库中的那样，会在 Middlebury 数据集中导致多达九个像素的垂直视差错误（Scharstein 等人，2014）。2014 年数据集中的每个立体图像对都整流了两次：一次是使用标准的不完美的方法，另一次是采用严格的 2D 对应来达到完美的矫正（Scharstein 等，2014）。我们使用不完美的矫正图像对来训练深度学习网络，因为十五对测试图像中仅有 2 对是完美矫正的。

误差率是计算真实视差与预测视差相差超过两个像素的像素点的百分比;这相当于半分辨率中的错误容限是一个像素。在默认情况下，评估服务器上的错误率仅仅计算非遮挡的像素点。最终在线报告上的错误率是十五对测试图像的加权平均，权重由数据集的作者设置。

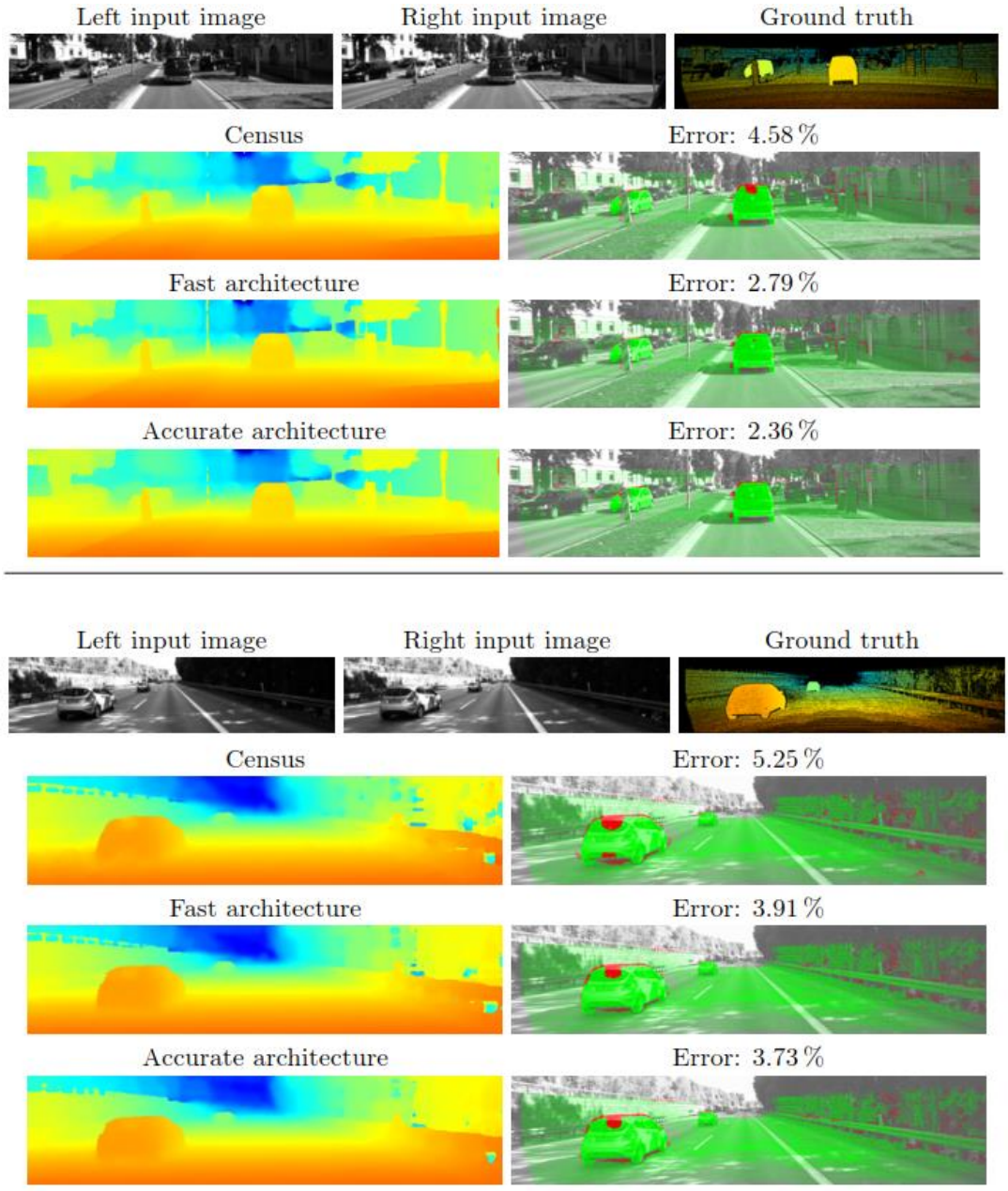

图 6: 在 KITTI 2015 数据集的预测实例。可以看出运动车辆在 KITTI 2015 数据集中是密集标记的。

表 4: 截至 2015 年 10 月，在 Middlebury 立体视觉数据集中的排名前十的算法，“Error”列是上采样成全分辨率图片后的加权平均误差。“Runtime”是处理一对图像的运行时间，以秒为单位。

Rank	Method		Resolution	Error	Runtime
1	**MC-CNN-acrt**	**Accurate architecture**	Half	8.29	150
2	MeshStereo	Zhang et al. (2015)	Half	13.4	65.3
3	LCU	Unpublished work	Quarter	17.0	6567
4	TMAP	Psota et al. (2015)	Half	17.1	2435
5	IDR	Kowalczuk et al. (2013)	Half	18.4	0.49
6	SGM	Hirschmüller (2008)	Half	18.7	9.90
7	LPS	Sinha et al. (2014)	Half	19.4	9.52
8	LPS	Sinha et al. (2014)	Full	20.3	25.8
9	SGM	Hirschmüller (2008)	Quarter	21.2	1.48
10	SNCC	Einecke and Eggert (2010)	Half	22.2	1.38

表 4 包含了一张第三名的快照，以及最新版本的 Middlebury 排行榜。我们的算法排在第一，是 8.29％的误差率，并在大幅领先第二名 MeshStereo 的算法，它的误差率是 13.4％。图 7 是我们根据 Middlebury 数据集中的一个图像对计算的视差图。

5. 3 深度学习细节

我们用训练数据集中所有可用的图像对构建了一个二元分类数据集。该数据集包含了 KITTI 2012 的 2500 万实例和 KITTI 2015 的 1700 万实例，以及 Middlebury 数据集的 3800 万实例。

在训练时，网络的输入是一批 128 对的图像块。在测试期间，输入是整个的左和右图像。在训练期间，我们也可以使用整个的图像，这将使我们能够实现 3.3 节中所说的速度优化。我们更愿意使用图像块训练的原因是：这是更容易控制批次的大小，这些例子可以被混合在一起，使得同一个批次的图像块可以分别来自几个不同图像，同时这也比较容易维持同一批内积极实例和消极实例的数量相同。

我们使用动量项为 0.9 的小批梯度下降来减少损失。

我们训练迭代 14 次，对于精确架构学习率的初始设置为 0.003，快速架构的初始学习率为 0.002。学习率在 11 次迭代时减少 10 倍。迭代次数，初始学习率以及学习率衰减计划，都作为超参数并用交叉验证进行优化。每个图像都通过减去平均值并除以像素强度值的标准差来预处理。立体图像对的左和右图分别预处理。我们的初步实验表明，使用彩色信息不会改善视差图的质量;因此，我们把所有的彩色图片都转换成灰度图。

立体视觉方法的后处理步骤在 CUDA（Nickolls 等人,2008）中得到实施，该

网络的训练是使用从 cuDNN 库（Chetlur 等人，2014）的卷积例程在 Torch 环境中完成的（Collobert 等人，2011）。OpenCV 库（ Bradski， 2000）被用于在数据集扩增步骤的仿射变换。

超参数通过手动搜索和简单的自动化脚本优化。表 5 是我们选择的超参数。

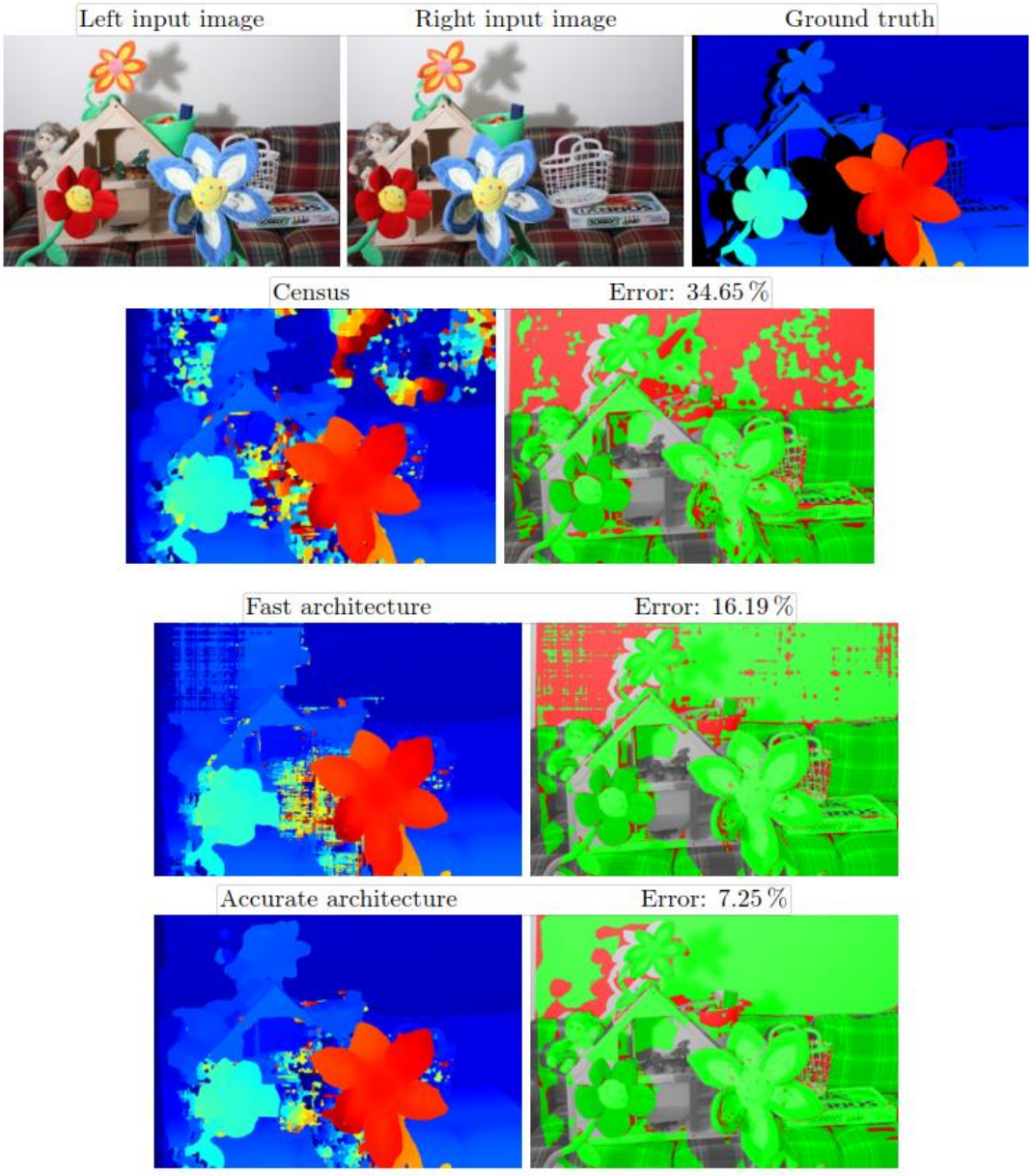

图 7: Middlebury 数据集中一个特别难的图像对示例；白色的背景墙几乎是无纹理。精确架构能够区分大多数的内容。快速架构做的并没有那么好但仍比统计方法更好。

表 5: 我们用于快速和精确架构的超参数值（缩写“FST”和“ACRT”）。需要注意的是超参数有关图像强度的值（cbca_intensity 和 sgm_D）仅适用于经过预处理的图像，而不是强度值范围为 0 到 255 的原始图像。

	KITTI 2012		KITTI 2015		Middlebury	
Hyperparameter	fst	acrt	fst	acrt	fst	acrt
input_patch_size	9×9	9×9	9×9	9×9	11×11	11×11
num_conv_layers	4	4	4	4	5	5
num_conv_feature_maps	64	112	64	112	64	112
conv_kernel_size	3	3	3	3	3	3
num_fc_layers		4		4		3
num_fc_units		384		384		384
dataset_neg_low	4	4	4	4	1.5	1.5
dataset_neg_high	10	10	10	10	6	18
dataset_pos	1	1	1	1	0.5	0.5
cbca_intensity		0.13		0.03		0.02
cbca_distance		5		5		14
cbca_num_iterations_1		2		2		2
cbca_num_iterations_2		0		4		16
sgm_P1	4	1.32	2.3	2.3	2.3	1.3
sgm_P2	223	32	42.3	55.8	55.9	18.1
sgm_Q1	3	3	3	3	4	4.5
sgm_Q2	7.5	6	6	6	8	9
sgm_V	1.5	2	1.25	1.75	1.5	2.75
sgm_D	0.02	0.08	0.08	0.08	0.08	0.13
blur_sigma	7.74	6	4.64	6	6	1.7
blur_threshold	5	6	5	5	2	2

5. 4 数据集扩增

通过反复训练实例的变换来扩增数据集是一种常用的技术，用来减少网络的泛化误差。这种变换是在训练时进行，不会影响运行时性能。我们随机旋转，缩放和剪切训练的图像块;我们还改变其亮度和对比度。因为是在提取图像后才对图像块变换，所以扩增数据集的步骤不改变地面真实视差图或破坏矫正。

变换的参数是每对图像对随机选取，每经过一次迭代后，相同的实例被第二次提供给网络，新的随机参数也被选择。我们对左右图像选择略有不同的变换参数;例如，我们会旋转左图 10 度而右图是 14 度。不同的数据集从不同类型的变换中受益，在某些情况下，使用错误的转换会增加错误率。在 Middlebury 数据集中，我们充分利用的既有事实，即所有的可用图像是在不同的光照条件和不同的快门曝光下拍摄的。这些数据集的扩增参数同样被用于 KITTI 2012 和 KITTI 2015 数据集中。

Middlebury 测试数据集中有两张值得一提图片：图片“教室”，它的右图曝光不足，因此，比左侧更暗；图片“非洲鼓”，它的左和右图像分别在不同光照条

件下拍摄的。为了处理这两种情况，我们花了 20％的时间训练快门曝光或灯光安排不同的情况下的左右图像。

我们通过在左右图像块之间加一个小的垂直视差来改善 Middlebury 数据集中的不完美矫正。

在描述数据扩增的步骤之前，让我们介绍一些符号：在下面，typewriter 字体的词被用来表示定义一个集合的超参数的名称，而 italic（斜体）字体的同一个词被用来表示从集合中随机抽取的数字。例如，rotate 是定义一组可能的旋转角度的集合，rotate 是从集合中随机抽取的数字。下表是数据扩增步骤的介绍：

- 左图像块旋转 rotate 度，右图像块旋转 rotate+rotate_diff 度。
- 左图像块缩放 scale 倍，右图像块缩放 scale*scale_diff 倍。
- 左图像块在水平方向上拉伸 horizontal_scalescale 倍，右图像块拉伸 horizontal_scalescale*horizontal_scalescale_diff 倍。
- 左图像块在水平方向上剪切 horizontal_shear，右图像块在水平方向上剪切 horizontal_shear+horizontal_shear_diff。
- 在垂直方向上调整右图像块 vertical_disparity.的视差
- 调整由左和右图像的补丁的亮度和对比度设定为：

$$\mathcal{P}^L \leftarrow \mathcal{P}^L \cdot contrast + brightness \text{ and}$$
$$\mathcal{P}^R \leftarrow \mathcal{P}^R \cdot (contrast \cdot contrast\_diff) + (brightness + brightness\_diff),$$

  逐项适当的加法和乘法。

表 6 包含使用的超参数和数据扩增步骤是如何影响验证误差的。

表 6: 管理数据扩增的超参数以及它们是如何影响验证误差的。“Error”列统计未使用某个特殊数据扩增步骤时的验证误差。最后两行统计使用或不使用数据扩增的验证错误。例如，如果没有数据扩增，KITTI 2012 的验证误差是 2.73％;如果除了旋转以外的所有步骤都被使用，验证误差为 2.65％;如果使用全部数据扩增步骤则验证误差为 2.61％。

	KITTI 2012		Middlebury	
Hyperparameter	Range	Error	Range	Error
rotate	$[-7, 7]$	2.65	$[-28, 28]$	7.99
scale			$[0.8, 1]$	8.17
horizontal_scale	$[0.9, 1]$	2.62	$[0.8, 1]$	8.08
horizontal_shear	$[0, 0.1]$	2.61	$[0, 0.1]$	7.91
brightness	$[0, 0.7]$	2.61	$[0, 1.3]$	8.16
contrast	$[1, 1.3]$	2.63	$[1, 1.1]$	7.95
vertical_disparity			$[0, 1]$	8.05
rotate_diff			$[-3, 3]$	8.00
horizontal_scale_diff			$[0.9, 1]$	7.97
horizontal_shear_diff			$[0, 0.3]$	8.05
brightness_diff	$[0, 0.3]$	2.63	$[0, 0.7]$	7.92
contrast_diff			$[1, 1.1]$	8.01
No data set augmentation		2.73		8.75
Full data set augmentation		2.61		7.91

在 KITTI 2012 数据集中，数据扩增把验证误差从 2.73%降低到了 2.61%，而在 Middlebury 数据集中是从 8.75%降至 7.91%。

5. 5 运行时间

我们在一台带有图形处理器单元 NVIDIA Titan X 的电脑上运行我们的应用并测量运行时间。表 7 包含三个数据集在不同的超参数设置下的运行时间:KITTI，Middlebury 半分辨率，和一个新的，称为 Tiny 的虚拟数据集，这是我们用来验证我们算法用于自主驾驶或机器人时的性能。我们测量运行时间的图片大小是：针对 KITTI 数据集的是 1242×350 并含有 228 个视差水平，针对 Middlebury 数据集的是 1500×1000 并含有 200 个视差水平，以及针对 Tiny 数据集的是 320×240 并含有 32 个视差水平。

表 7: 用来计算匹配代价的时间（秒），即花的没有任何后处理步骤卷积神经网络中的时间。 这个时间不包括计算匹配代价两次：一次是当左图像被取为参考图像，另一次是右图像被取为参考图像。我们测量作为四个控制网络架构的超参数的函数的运行时间;例如，前六个行包含网络中卷积层的数量从一个增加到六个的运行时间。表中的最后一行包含整个算法的运行时间，包括后处理步骤。和前面一样，快速和精确架构的缩写为“fst”和“acrt”。

Hyperparameter		KITTI		Middlebury		Tiny	
		fst	acrt	fst	acrt	fst	acrt
num_conv_layers	1	0.23	66.1	0.70	74.9	0.01	1.7
	2	0.26	66.2	0.82	75.2	0.01	1.8
	3	0.30	66.3	0.97	75.4	0.02	1.8
	4	0.34	66.4	1.11	75.6	0.03	1.8
	5	0.38	66.5	1.24	75.7	0.03	1.9
	6	0.42	66.7	1.37	76.0	0.04	1.9
num_conv_feature_maps	16	0.09	59.4	0.27	64.8	0.01	1.6
	32	0.15	60.4	0.51	66.2	0.01	1.6
	48	0.25	61.5	0.94	68.2	0.02	1.7
	64	0.34	62.7	1.24	70.0	0.03	1.7
	80	0.44	64.0	1.63	72.0	0.04	1.8
	96	0.53	65.3	1.93	73.9	0.04	1.8
	112	0.61	66.4	2.28	75.7	0.05	1.8
	128	0.71	67.7	2.61	77.8	0.06	1.9
num_fc_layers	1		16.3		25.3		0.5
	2		32.9		50.7		0.9
	3		49.6		75.7		1.4
	4		66.4		101.2		1.8
	5		82.9		126.4		2.3
num_fc_units	128		17.4		21.4		0.6
	256		38.5		44.9		1.1
	384		66.4		75.7		1.8
	512		101.0		113.3		2.7
No stereo method		0.34	66.4	1.24	75.7	0.03	1.8
Full stereo method		0.78	67.1	2.03	84.8	0.06	1.9

表 7 表明，快速架构比精确架构速度快高达 90 倍。此外，快速架构的运行时间在 KITTI 中为 0.78 秒，在 Middlebury 中为 2.03 秒，在 Tiny 数据集中为 0.06 秒。我们还可以看到在精确架构中全连接层占据了大多数的运行时间，控制卷积层的数量和特征映射的数目的超参数对运行时间仅有很小的影响。

训练次数取决于数据集的大小和结构，但从来没有超过两天。

5.6 匹配代价

我们认为，我们方法的低错误率是由于卷积神经网络而不是更优秀的立体视觉方法。我们通过用三个标准的计算匹配代价的方法更换卷积神经网络验证这种说法：

- 绝对差之和方法根据等式（1）计算匹配代价，也就是说，两个图像块之间的匹配代价是计算在相应的位置之间的图像强度的绝对差值之和。我

们使用 9×9 的图像块。

- 统计变换方法（Zabih 和 Woodfill，1994）认为每个图像的位置是比特向量。此向量的大小是一个超参数，它的值，在测试几次后，我们设置为 81。这个向量通过计算被裁剪成围绕感兴趣位置的 9×9 的图像块，并比较图像块中每个像素的强度值和中心像素的强度值。如果中心像素更亮，相应位就被置位。匹配代价被计算为两次统计变换所得向量之间的汉明距离。
- 归一化的互相关方法是用下面等式定义的基于窗口的方法：

$$C_{NCC}(\boldsymbol{p}, d) = \frac{\sum_{\boldsymbol{q}\in \boldsymbol{N_p}} I^L(\boldsymbol{q}) I^R(\boldsymbol{q}-\boldsymbol{d})}{\sqrt{\sum_{\boldsymbol{q}\in \boldsymbol{N_p}} I^L(\boldsymbol{q})^2 \sum_{\boldsymbol{q}\in \boldsymbol{N_p}} I^R(\boldsymbol{q}-\boldsymbol{d})^2}}$$

归一化互相关匹配代价是计算左右图像块的余弦相似度，左和右图像块被看作是向量，而不是矩阵。这是在快速架构的最后两层（归一化和点积）计算的同一个函数。邻域$\boldsymbol{N_p}$被设定为围绕 **p** 的 11×11 的方形窗口。

表 8 的“sad”，“cens”和“ncc”列是上述三个方法在 KITTI 2012,KITTI 2015 和 Middlebury 数据集的结果。表 8 中最后一排的验证误差被用来比较五种方法。对所有这三个数据集，准确架构的效果最好，其次是快速架构，其后是统计变换。这是对所有三个数据集表现最好的三个方法。他们的错误率分别在 KITTI 2012 中为 2.61％，3.02％和 4.90％; 在 KITTI 2015 中为 3.25％，3.99％和 5.03％ 2015 年; 在 Middlebury 中是 7.91％，9.87％和 16.72％。绝对差之和的方法和归一化互相关方法的匹配代价产生较大错误率的视差图。 我们的方法和统计变换的视觉对比见图 5,6,7。

表 8: 当从立体视觉方法中去除某个特定的后处理步骤时，验证误差的数值。表的最后两行应有不同于之前的解释：它们是原始卷积神经网络的验证误差和完整的立体视觉方法后的验证误差。例如，如果我们排除半全局匹配，在 KITTI 2012 数据集中快速架构的错误率为 8.78％率和在应用整个立体视觉算法后为 3.02％。我们的简写“fst”为快速架构“acrt”的精确建筑，“sad”为绝对值之和方法 “cens”为统计变换方法，而“ncc”为归一化互相关匹配代价。

	KITTI 2012				
	fst	acrt	sad	cens	ncc
Cross-based cost aggregation	3.02	2.73	8.22	5.21	8.93
Semiglobal matching	8.78	4.26	19.58	8.84	10.72
Interpolation	3.48	2.96	9.21	5.96	11.16
Subpixel Enhancement	3.03	2.65	8.16	4.95	8.93
Median filter	3.03	2.63	8.16	4.92	9.00
Bilateral filter	3.26	2.79	8.75	5.70	9.76
No stereo method	15.70	13.49	32.30	53.55	22.21
Full stereo method	3.02	2.61	8.16	4.90	8.93
	KITTI 2015				
	fst	acrt	sad	cens	ncc
Cross-based cost aggregation	3.99	3.39	9.94	5.20	8.89
Semiglobal matching	8.40	4.51	19.80	7.25	9.36
Interpolation	4.47	3.33	10.39	5.83	10.98
Subpixel Enhancement	4.02	3.28	9.44	5.03	8.91
Median filter	4.05	3.25	9.44	5.05	8.96
Bilateral filter	4.20	3.43	9.95	5.84	9.77
No stereo method	15.66	13.38	30.67	50.35	18.95
Full stereo method	3.99	3.25	9.44	5.03	8.89
	Middlebury				
	fst	acrt	sad	cens	ncc
Cross-based cost aggregation	9.87	10.63	43.09	29.28	33.89
Semiglobal matching	25.50	11.99	51.25	19.51	35.36
Interpolation	9.87	7.91	41.86	16.72	33.89
Subpixel Enhancement	10.29	8.44	42.71	17.18	34.12
Median filter	10.16	7.91	41.90	16.73	34.17
Bilateral filter	10.39	7.96	41.97	16.96	34.43
No stereo method	30.84	28.33	59.57	64.53	39.23
Full stereo method	9.87	7.91	41.86	16.72	33.89

5. 7 立体视觉方法

立体视觉方法包括多个后处理步骤：基于交叉的代价聚合，半全局匹配，插值，亚像素增强，中值和双边滤波器。我们进行了一系列实验，在实验中我们依次排除上述的步骤并记录验证误差（见表 8）。

表 8 的最后两行暗示了立体视觉方法中后处理步骤的重要性。我们看到，如果所有的后处理步骤都被去除，精确架构的验证误差在 KITTI 2012 中从 2.61%

提高到 13.49%，在 KITTI 2015 中从 3.25%升到 13.38%，在 Middlebury 中从 7.91%升至 28.33%。

在立体视觉方法的所有后处理步骤中，半全局匹配对验证误差的影响最强。如果我们将其去除，在 KITTI 2012 中验证误差从 2.61%上升到 4.26%年，在 KITTI 2015 中从 3.25%上升到 4.51%，在 Middlebury 中从 7.91%上升到 11.99%。

我们没有使用左右一致性检查来消除 Middlebury 中遮蔽区域的误差。在精确架构中使用左右一致性检查，错误率从 7.91%提高到了 8.22%，这就是我们决定将其删除的原因。

5.8 数据集大小

我们使用了一个监督学习的方法来衡量图像块之间的相似性。因此，我们自然要问数据集的大小是如何影响视差图质量的。要回答这个问题，我们用较小的训练集重新训练我们的网络，这个小训练集是通过选择一组随机的实例获得（见表 9）。

我们观察到，随着我们增加训练示例的数量，验证误差减少了。这些实验表明了一个改善我们立体视觉方法结果的简单策略：收集更大的数据集。

表 9: 验证误差是与训练集大小相关的函数

	KITTI 2012		KITTI 2015		Middlebury	
Data Set Size (%)	fst	acrt	fst	acrt	fst	acrt
20	3.17	2.84	4.13	3.53	11.14	9.73
40	3.11	2.75	4.10	3.40	10.35	8.71
60	3.09	2.67	4.05	3.34	10.14	8.36
80	3.05	2.65	4.02	3.29	10.09	8.21
100	3.02	2.61	3.99	3.25	9.87	7.91

5.9 迁移学习

到目前为止，训练集和验证集都是从同一个立体视觉数据集中产生的，无论是在 KITTI 2012，还是 KITTI 2015 或 Middlebury 数据集。为了评估我们算法在迁移学习方面的表现，我们再次进行实验，其中验证误差是计算在一个不同于之前训练时的数据集。例如，我们使用 Middlebury 数据集训练匹配代价神经网络，

并在 KITTI 2012 数据集中评估它的的性能。这些实验展示给我们一些算法在现实世界中的应用时预期的表现，因为没有可用的标签数据，所以在现实世界中是不可能训练出专门的网络的。这些实验的结果展示于表 10 中。

表 10 中的一些结果出人意料。例如，当使用 Middlebury 训练数据并在 KITTI 2012 中验证的误差比用 KITTI 2015 训练时更低，即使 KITTI 2012 数据集相比于 Middlebury 明显更类似 KITTI 2015。此外，当使用 KITTI 2015 训练网络时，使用快速架构的 KITTI 2012 的验证误差比使用精确架构时更低。

由 Middlebury 数据集训练的匹配代价神经网络，能更好的迁移到 KITTI 数据集中。其验证误差类似于 KITTI 数据集训练的网络的验证误差。

表 10: 当训练集和测试集不同时的验证误差。例如，当 Middlebury 数据集用于训练快速架构并使用 KITTI 2012 数据集测试时，验证误差为 3.16%。

		Test Set					
		KITTI 2012		KITTI 2015		Middlebury	
		fst	acrt	fst	acrt	fst	acrt
Training Set	KITTI 2012	3.02	2.61	4.12	3.99	12.78	11.09
	KITTI 2015	3.60	4.28	3.99	3.25	13.70	14.19
	Middlebury	3.16	3.07	4.48	4.49	9.87	7.91

5. 10 超参数

寻找一组好的超参数是一个艰巨的任务——搜索空间随着超参数的数量成指数增长并且没有梯度可循。为了更好地理解每个超参数对验证误差的影响，我们进行了一系列的实验，在实验中我们改变一个超参数的值，同时保持其他超参数固定为默认值。结果在表 11 中，我们可以观察到增加网络的大小提高了泛化性能，但只在一种特殊情况下，因为数据集的大小，泛化性能开始真的下降。

注意，num_conv_layers 超参数隐性控制着图像块的大小。例如，一个 3×3 内核的卷积层的网络意味着尺寸 3×3 的图像块，而有五个卷积层的网络意味着尺寸 11×11 的图像块。

表 11: 一系列超参数设置下的验证误差。

Hyperparameter		KITTI 2012		KITTI 2015		Middlebury	
		fst	acrt	fst	acrt	fst	acrt
num_conv_layers	1	5.96	3.97	5.61	4.06	20.74	12.37
	2	3.52	2.98	4.19	3.45	12.11	9.20
	3	3.10	2.72	4.04	3.27	10.81	8.56
	4	3.02	2.61	3.99	3.25	10.26	8.21
	5	3.03	2.64	3.99	3.30	9.87	7.91
	6	3.05	2.70	4.01	3.38	9.71	8.11
num_conv_feature_maps	16	3.33	2.84	4.32	3.51	11.79	10.06
	32	3.15	2.68	4.12	3.35	10.48	8.67
	48	3.07	2.66	4.06	3.32	10.20	8.47
	64	3.02	2.64	3.99	3.30	9.87	8.12
	80	3.02	2.64	3.99	3.29	9.81	7.95
	96	2.99	2.68	3.97	3.27	9.62	8.03
	112	2.98	2.61	3.96	3.25	9.59	7.91
	128	2.97	2.63	3.95	3.23	9.45	7.92
num_fc_layers	1		2.83		3.50		8.52
	2		2.70		3.31		8.33
	3		2.62		3.30		8.06
	4		2.61		3.25		8.00
	5		2.62		3.29		7.91
num_fc_units	128		2.72		3.36		8.44
	256		2.65		3.28		8.03
	384		2.61		3.25		7.91
	512		2.60		3.23		7.90
dataset_neg_low	1.0	3.00	2.76	3.97	3.35	9.84	8.00
	1.5	3.00	2.71	3.97	3.33	9.87	7.91
	2.0	2.99	2.63	3.98	3.31	9.98	8.08
	4.0	3.02	2.61	3.99	3.25	10.20	8.66
	6.0	3.06	2.63	4.05	3.28	10.13	8.86
dataset_neg_high	6	3.00	2.72	3.98	3.30	9.87	8.59
	10	3.02	2.61	3.99	3.25	9.97	8.23
	14	3.04	2.61	4.02	3.25	10.00	8.05
	18	3.07	2.60	4.06	3.23	9.98	8.11
	22	3.07	2.61	4.05	3.24	10.16	7.91
dataset_pos	0.0	3.04	2.67	4.00	3.26	9.92	7.97
	0.5	3.02	2.65	3.99	3.28	9.87	7.91
	1.0	3.02	2.61	3.99	3.25	9.86	8.04
	1.5	3.04	2.62	4.04	3.27	10.00	8.34
	2.0	3.04	2.66	4.04	3.29	10.16	8.51

6. 总结

我们提出了两个卷积神经网络架构来学习图像块相似性的测量并应用到立

体匹配的问题中。

我们实现方法的源代码在 https://github.com/jzbontar/mc-cnn。在线仓库中包含计算视差图，训练网络以及立体视觉方法的后处理步骤的过程。

精确架构产生更低错误率的视差图，比任何先前公布在 KITTI 2012，KITTI 2015 和 Middlebury 数据集的方法更低。快速架构计算视差图比精确架构快将近 90 倍，并且只有小幅的错误率增长。这些结果表明，卷积神经网络非常适合计算立体视觉匹配成本，甚至可以适应那些需要实时性能的应用中。

在一个已经被充分研究的问题中，一个相对简单的卷积神经网络胜过先前所有的立体视觉算法的事实是对现代机器学习方法力量的一个相对重要的示范。

参考文献

[1] G. Bradski. The OpenCV library. Dr. Dobb's Journal of Software Tools, 2000.

[2] Jane Bromley, James W Bentz, L´eon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore,Eduard S¨ackinger, and Roopak Shah. Signature verification using a siamese time delay neural network. International Journal of Pattern Recognition and Artificial Intelligence, 7(04):669–688, 1993.

[3] Matthew Brown, Gang Hua, and Simon Winder. Discriminative learning of local image descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(1): 43–57, 2011.

[4] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(3):500–513, 2011.

[5] Ayan Chakrabarti, Ying Xiong, Steven J. Gortler, and Todd Zickler. Low-level vision by consensus in a spatial hierarchy of regions. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[6] Zhuoyuan Chen, Xun Sun, Yinan Yu, Liang Wang, and Chang Huang. A deep visual correspondence embedding model for stereo matching costs. IEEE International Conference on Computer Vision (ICCV), 2015.

[7] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran,

Bryan Catanzaro, and Evan Shelhamer. cuDNN: Efficient primitives for deep learning. CoRR, abs/1410.0759, 2014.

[8] Ronan Collobert, Koray Kavukcuoglu, and Cl´ement Farabet. Torch7: A matlab-like environment for machine learning. In BigLearn, NIPS Workshop, 2011.

[9] Nils Einecke and Julian Eggert. A two-stage correlation method for stereoscopic depth estimation. In Digital Image Computing: International Conference on Techniques and Applications (DICTA), pages 227–234, 2010.

[10] Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient large-scale stereo matching. In Proceedings of the 10th Asian Conference on Computer Vision - Volume Part I, ACCV'10, pages 25–38. Springer-Verlag, Berlin, Heidelberg, 2011.

[11] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: the KITTI dataset. International Journal of Robotics Research (IJRR), 2013.

[12] Fatma Guney and Andreas Geiger. Displets: Resolving stereo ambiguities using object knowledge. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[13] Ralf Haeusler, Rahul Nair, and Daniel Kondermann. Ensemble learning for confidence measures in stereo vision. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2013.

[14] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. MatchNet: Unifying feature and metric learning for patch-based matching. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[15] Heiko Hirschmu¨ller. Stereo processing by semiglobal matching and mutual information. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(2):328–341, 2008.

[16] Heiko Hirschmu¨ller and Daniel Scharstein. Evaluation of cost functions for stereo matching. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007.

[17] Heiko Hirschmu¨ller and Daniel Scharstein. Evaluation of stereo matching costs on images with radiometric differences. IEEE Transactions on Pattern Analysis and

Machine Intelligence, 31(9):1582–1599, 2009.

[18] Michael Hornacek, Andrew Fitzgibbon, and Carsten Rother. SphereFlow: 6 DoF scene flow from RGB-D pairs. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2014.

[19] Dan Kong and Hai Tao. A method for learning matching errors for stereo computation. British Machine Vision Conference (BMVC), 2004.

[20] Dan Kong and Hai Tao. Stereo matching via learning multiple experts behaviors. British Machine Vision Conference (BMVC), 2006.

[21] Jana Kostkov´a and Radim S´ara. Stratified dense matching for stereopsis in complex scenes. British Machine Vision Conference (BMVC), 2003.

[22] Jedrzej Kowalczuk, Eric T Psota, and Lance C Perez. Real-time stereo matching on CUDA using an iterative refinement method for adaptive support-weight correspondences. IEEE Transactions on Circuits and Systems for Video Technology, 23(1):94–104, 2013.

[23] Yann LeCun, L´eon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, 1998.

[24] Yunpeng Li and Daniel P Huttenlocher. Learning for stereo vision using the structured support vector machine. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2008.

[25] Xing Mei, Xun Sun, Mingcai Zhou, Haitao Wang, Xiaopeng Zhang, et al. On building an accurate stereo matching system on graphics hardware. IEEE International Conference on Computer Vision Workshops (ICCV Workshops), pages 467–474, 2011.

[26] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[27] John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with CUDA. Queue, 6(2):40–53, 2008.

[28] Mattis Paulin, Matthijs Douze, Zaid Harchaoui, Julien Mairal, Florent Perronin, and Cordelia Schmid. Local convolutional features with unsupervised training for

image retrieval. In IEEE International Conference on Computer Vision (ICCV), pages 91–99, 2015.

[29] Martin Peris, Atsuto Maki, Sara Martull, Yasuhiro Ohkawa, and Kazuhiro Fukui. Towards a simulation driven stereo vision system. In 21st International Conference on Pattern Recognition (ICPR), pages 1038–1042, 2012.

[30] Eric T Psota, Jedrzej Kowalczuk, Mateusz Mittek, and Lance C Perez. Map disparity estimation using hidden markov trees. IEEE International Conference on Computer Vision (ICCV), 2015.

[31] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Deepmatching: Hierarchical deformable dense matching. ArXiv e-prints, 1(7):8, 2015.

[32] Daniel Scharstein and Chris Pal. Learning conditional random fields for stereo. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2007.

[33] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. International Journal of Computer Vision, 47(1-3): 7–42, 2002.

[34] Daniel Scharstein and Richard Szeliski. High-accuracy stereo depth maps using structured light. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June.2003.

[35] Daniel Scharstein, Heiko Hirschmu¨ller, York Kitajima, Greg Krathwohl, Nera Ne˘si´c, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. German Conference on Pattern Recognition (GCPR), September 2014.

[36] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Learning local feature descriptors using convex optimisation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(8):1573–1585, 2014.

[37] Sudipta N Sinha, Daniel Scharstein, and Richard Szeliski. Efficient high-resolution stereo matching using local plane sweeps. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2014.

[38] Aristotle Spyropoulos, Nikos Komodakis, and Philippos Mordohai. Learning to

detect ground control points for improving the accuracy of stereo matching. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2014.

[39] Deqing Sun, Stefan Roth, and Michael J Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. International Journal of Computer Vision, 106(2):115–137, 2014.

[40] Tomasz Trzcinski, Mario Christoudias, Vincent Lepetit, and Pascal Fua. Learning image descriptors with the boosting-trick. In Advances in neural information processing systems, pages 269–277, 2012.

[41] Christoph Vogel, Konrad Schindler, and Stefan Roth. Piecewise rigid scene flow. IEEE International Conference on Computer Vision (ICCV), 2013.

[42] Christoph Vogel, Stefan Roth, and Konrad Schindler. View-consistent 3D scene flow estimation over multiple frames. European Conference on Computer Vision (ECCV), September 2014.

[43] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3D scene flow estimation with a piecewise rigid scene model. International Journal of Computer Vision, pages 1–28, 2015.

[44] Koichiro Yamaguchi, David McAllester, and Raquel Urtasun. Efficient joint segmentation, occlusion labeling, stereo and flow estimation. European Conference on Computer Vision (ECCV), September 2014.

[45] Ramin Zabih and John Woodfill. Non-parametric local transforms for computing visual correspondence. European Conference on Computer Vision (ECCV), 1994.

[46] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[47] Jure Zbontar and Yann LeCun. Computing the stereo matching cost with a convolutionalˇ neural network. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[48] Chi Zhang, Zhiwei Li, Yanhua Cheng, Rui Cai, Hongyang Chao, and Yong Rui. Meshstereo: A global stereo model with mesh alignment regularization for view interpolation. IEEE International Conference on Computer Vision (ICCV), 2015.

[49] Ke Zhang, Jiangbo Lu, and Gauthier Lafruit. Cross-based local stereo matching using orthogonal integral images. IEEE Transactions on Circuits and Systems for Video Technology, 19(7):1073–1079, 2009.

[50] Li Zhang and Steven M Seitz. Estimating optimal parameters for MRF stereo from a single image pair. IEEE Transactions on Pattern Analysis and Machine Intelligence, 29(2): 331–342, 2007.