Fundamental Principles on Learning New Features for Effective Dense Matching

Feihu Zhang, *Student Member, IEEE*, Benjamin W. Wah, *Fellow, IEEE*

***Abstract*—In dense matching (including stereo matching and optical flow), nearly all existing approaches are based on simple features, such as gray or RGB color, gradient or simple transformations like census, to calculate matching costs. These features do not perform well in complex scenes that may involve radiometric changes, noises, overexposure and/or textureless regions. Various problems may appear, such as wrong matching at the pixel or region level, flattening/breaking of edges and/or even entire structural collapse. In this paper, we propose two fundamental principles based on the consistency and the distinctiveness of features. We show that almost all existing problems in dense matching are caused by features that violate one or both of these principles. To systematically learn good features for dense matching, we develop a general multi-objective optimization based on these two principles and apply convolutional neural networks (CNNs) to find new features that lie on the Pareto frontier. By using two-frame optical flow and stereo matching as applications, our experimental results show that the features learned can significantly improve the performance of state-of-the-art approaches. Based on the KITTI benchmarks, our method ranks first on the two stereo benchmarks and is the best among existing two-frame optical-flow algorithms on flow benchmarks.**

***Index Terms*—Image Feature, CNN, Dense Matching, Optical Flow, Stereo Matching, Matching Cost.**

I. Introduction

STEREO matching, optical flow and other dense-matching applications have always been hot issues in computer vision. In the past, a number of methods have been developed to solve these problems. These methods consist of three steps: extracting features and their descriptors, computing the matching cost and/or aggregation [1]–[4], and applying matching algorithms [5]–[7] to minimize some energy functions.

In recent years, there is a lot of attention on the last two steps. However, little has been done on feature extraction that is critical in dense-matching. The most popular features for stereo matching and optical flow are still limited to some kind of color space or gradient values [8]. Although these simple features are fast to compute and flexible (like in scaling and subpixel interpolation), they are easily influenced by radiometric changes, noise, overexposure and the scene environment (as shown in Fig. 1). This is also the major reason why some of the best methods that work well on benchmarks of simple indoor scenes [9] report limited success on benchmarks of complicated outdoor scenes [10]. On the other hand, the popular sparse features used for shape matching and object detection (including SIFT [11] and SURF [12]) which are scale and radiometric invariant met their limitations on performance improvement and flexibility when directly applied to dense matching. These methods were not designed for dense matching from the beginning, and they usually involve a complex step on label densifying [13].

Instead of developing new features for dense matching, some recent methods [14], [15] introduce convolutional neural networks (CNNs) to compare the similarity of a pair of patches and use the similarity score as the matching cost. These help achieve high accuracy when used in some stereo matching methods [14]. To address their high computational cost, Zbontar *et al*. proposed a faster framework with some sacrifice in accuracy [16]. However, as its time complexity depends on the displacement space, it still cannot be used for optical flow and other complex algorithms, such as continuous matching with slanted surface or subpixel accuracy. (The time complexity is $O(KNM)$, with size K of displacement space, N pixels, and computation complexity M of CNN.)

The primary problem in developing better dense matching algorithms is to identify good features. There has been little work in this area. A direct approach [8], [17] collects the error rates when employing one type of features in a specific algorithm. The rates, however, are not useful for designing feature extractors because they cannot provide quantitative information on the features of each pixel and/or region. Also, it is impractical to use them as targets because not only is it time consuming to run matching algorithms during feature extraction, the error rate of one matching algorithm cannot represent the feature's performance in other algorithms.

In this paper, we identify two fundamental principles on good features that each pixel should possess in order to be effective for dense matching. These principles help understand the requirements on good features for dense matching, as well as identifying the weaknesses of existing algorithms (as they violate one or both of these principles).

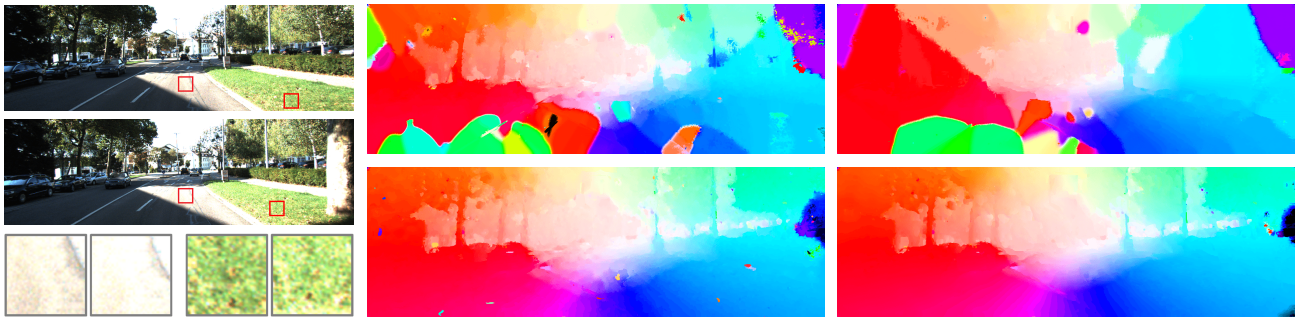
The first principle, the *Consistency Principle*, states that a feature point (a pixel/location where the feature performs well) should own the same or similar feature descriptors (such as RGB values when color is used as the feature) when it appears in different image views (such as the left and right views of a stereo pair). For example, many existing features like color or gradient are highly influenced by noise, radiometric variance, scaling change, translation and/or rotation. As illustrated in Fig. 1, such external disturbances can easily break the consistency of features between different views.

The second principle, the *Distinctiveness Principle*, states that a feature point should be different enough with respect

The authors are with the Chinese University of Hong Kong (e-mail: hi.yexu@gmail.com, bwah@cuhk.edu.hk).

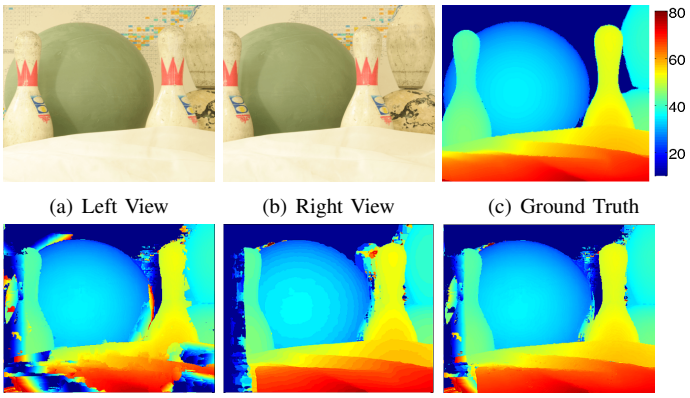
Research was supported in part by the National Grand Fundamental Research 973 Program of China No. 2014CB340401.

Manuscript received XX 2016; revised XX 2017; accepted XX, 2017.



(a) Inputs & inconsistency examples (b) Original & improved CostFilter [1] (c) Original & improved PMBP [5]

Fig. 1: An illustration of the Consistency Principle using optical flow as an example. Results on flow field are visualized by the color coding technique used in [10]. (a) Input views (from KITTI dataset [10]). Enlarged red windows, illustrating color inconsistencies, are usually caused by radiometric variations, view-angle changes and over exposure. (b) Original CostFilter [1] with simple color+gradient features and the improved results using our proposed feature. (c) Original and improved PMBP [5] by using 32-channel fast-flow features to improve these matching methods. (See Section V for a description of the features.)



(a) Left View (b) Right View (c) Ground Truth

(d) PatchMatch [18] (e) MC-CNN [14] (f) [18] with Our Feature

Fig. 2: An illustration of the Distinctiveness Principle using stereo matching as an example. (a) Left and (b) right views from Middlebury dataset [9]. (c) Ground truth (with disparity values visualized). (d) Original patchmatch stereo [18] with color+gradient features. (e) Poor subpixel accuracy produced by MC-CNN [14] that cannot be used for continuous stereo matching. (f) Significant improvements when our 32-channel fast stereo features are embedded in the original patchmatch algorithm [18].

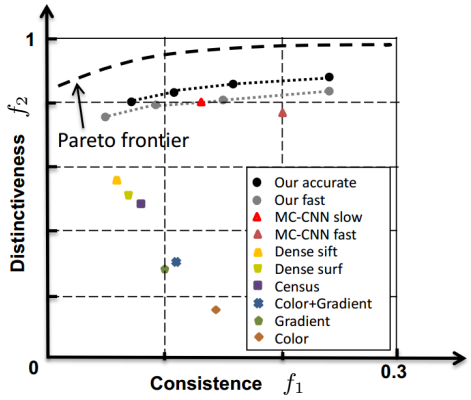


Fig. 3: Referring to details in Section VI-D, an illustration of Pareto optimality and a comparison of the solutions found in stereo matching. The x axis shows the consistence f_1 (Eq. (6)), the smaller the better. The y axis shows the distinctiveness f_2 (Eq. (9)), the larger the better. The target of the search is to find the Pareto optimal frontier with trade-offs between f_1 and f_2. The results of ten feature methods are plotted, and four non-dominated solutions with each of our CNN solvers are shown. Data is collected using the KITTI datasets [21] and shown in Table VII. The dash curve illustrates a hypothetical Pareto frontier for some CNN architecture.

to other points/pixels in its surrounding regions. For instance, when using color as the feature, pixels at a corner are unique, whereas those in smooth regions are not distinct. In large smooth regions, the principle is violated in many state-of-the-art features [5], [18], [19] because the color of all the pixels in these regions are very similar. As illustrated in Fig. 2, the lack of distinctiveness lead to wrong matches in these regions.

The above two principles can guide us in finding good features. To facilitate the search of such features, we formulate in Sec. III a multi-objective optimization that incorporates both principles as objectives under the search space imposed by the CNN architecture. Our formulation aims to study the tradeoffs between these principles and to find Pareto-optimal solutions [20] in the search space that are optimal in the sense that no solution on the Pareto frontier is better than another when all the objectives are considered together (illustrated in Fig. 3).

We implement the search using a loss function that incorporates the objectives on the two principles and train a CNN (that acts as the feature extractor) to look for new features. This implementation is found to be effective for identifying a set of dominating solutions closest to the Pareto frontier.

The features learned possess the following good properties. a) *High accuracy with robustness to noise and radiometric changes*. Our evaluations on benchmarks show that our feature-based method outperforms the current best. b) *Flexible for use in almost all existing dense-matching algorithms*. These are similar to RGB values (*e.g.* scaling and subpixel interpolation are possible for continuous and pyramid-matching algorithms). c) *Low memory requirement and fast speed*. The feature extractor consists of five convolutional layers and can be implemented efficiently (*e.g.*, a 16-channel fast feature can be extracted in 20 fps). Also, the new features do not increase the time complexity of the original matching algorithms.

In short, the major contribution of this paper lies in the development of two fundamental principles for characterizing good features in dense matching. The tradeoffs between the two principles can be implemented as a multi-objective optimization problem. We propose to use a CNN solver as a

feature extractor to find good features on the Pareto frontier.

The rest of paper is organized as follows. After reviewing previous work and their challenges in Section II, we formulate and analyze the two fundamental principles in Section III and propose a general model for designing good features in Section IV. Section V shows the details of our CNN solver, which includes a fast version and an accurate version learned by different architectures. The experimental results in Section VI demonstrate that our approach is effective and efficient. We further analyze our model and solutions in Section VI-D and discuss an extension of our CNN architectures in Section VII. Finally, Section VIII concludes the paper.

II. Related Work and Challenges

A. Feature Extraction in Dense Matching

A dense matching framework usually consists of three steps: extracting local feature descriptors, calculating and aggregating matching costs, and matching the descriptors by minimizing some energy functions. Based on the available features, matching cost can be defined as the distance or difference between two feature descriptors. Common functions used include the sum of absolute or squared distances (SAD/SSD), and normalized cross-correlation (NCC) as well as their mixed or truncated versions [17].

Cost aggregation and matching algorithms have been well studied in the past. Many powerful approaches have been proposed, including fast cost-aggregation methods [1]–[4], [22] and matching algorithms [5]–[7], [23]. They are effective and perform well under some simple synthetic or indoor scenes. However, they may perform poorly when applied to complicated outdoor scenes (for instance, the KITTI datasets [21]) because the simple color or gradient-based features they employ for calculating matching costs cannot address the nuances in complex scenes. Various problems, such as wrong matches in pixels or regions, flattening/breaking in edges, and/or collapse of the whole structure, may arise when involving radiometric changes, noises, overexposure and/or textureless regions.

In general, features form the foundation of dense matching approaches. In state-of-the-art dense matching schemes, features are more important than the matching-cost function used. As shown in Fig's 1 and 2, good features can lead to impressive results, whereas improper ones may lead to structural collapses. To this end, we focus in this paper on the development of methods for finding good features.

As discussed above, one simple and widely used feature is the color-space (*e.g.*, RGB) and/or gradient values. Its biggest advantage is that it can be flexibly and efficiently used in every dense matching scheme. For instance, it can be used for scaling and subpixel interpolations to produce sub-pixel accuracy in some continuous and pyramid dense matching approaches [5], [19], [24]. However, it is easily influenced by radiometric changes, noise, overexposure and textureless regions (as shown in Fig. 1).

As a compensation, some illuminance invariant features have been proposed, including Laplacian of Gaussian, photometric correlation by bilateral filter [25], and Adaptive Normalized Cross Correlation (ANCC) [26]. However, they are still not robust enough for challenging outdoor scenes. Moreover, they may bring outliers and smoothness effects to some object boundaries.

Some approaches have introduced radiometric and scale invariant SIFT [11] or SURF [12] to their dense matching algorithms [13]. These features have been successfully used in object detection or scene recognition. However, for dense matching, they are not reliable in every region because it is necessary to employ a complex densifying procedure to generate dense matching maps. This requirement limits their effectiveness and accuracy improvements.

Zbontar, *et al*. are the first to introduce CNNs to compute the matching-cost matrix [14]. Their method uses a trained siamese CNN architecture to learn the patch similarity score. It skips the feature-extraction step and directly uses the score as the matching cost. Such a method has largely improved the accuracy of some algorithms (like SGM [6]). However, it sacrifices many good properties of the traditional feature-based framework; for instance, it cannot be used in continuous or pyramid matching algorithms [5], [18], [27]. Also, its time complexity and memory requirement rely on the displacement space and are issues in many large displacement dense matching applications.

In short, existing dense-matching applications generally find good features by trying them one after another and by adjusting their parameters in order to achieve good performance on some benchmarks. This is the reason why one effective matching scheme may degenerate greatly when applied to a new untested environment/dataset. This has happened to many schemes (*e.g.* [5], [18], [19], [27]) that cannot produce good results on the challenging KITTI dataset, although they have done well on others. In this paper, we develop a new approach that considers the properties of good features before choosing a feature for dense matching. Without relying on trial-and-error, our method identifies good features through optimization.

B. Multi-Objective Optimization and Pareto Optimality

A general multi-objective optimization is defined as follows:

$$\min_{\phi \in C} F = \begin{bmatrix} f_1(\phi) \\ ... \\ f_k(\phi) \end{bmatrix}. \quad (1)$$

Solving Eq. (1) amounts to finding a representative set of Pareto optimal solutions. *Pareto optimality* or *Pareto efficiency* is a state of allocation of resources from which it is impossible to reallocate in order to make any one objective better off without making at least another objective worse off. The *Pareto frontier* is the set of all Pareto efficient solutions that do not dominate each other. The following four classes of methods are widely recognized approaches.

A) *Classical methods based on scalarizing.* Scalarizing Eq. (1) entails reformulating it into a single-objective optimization problem in such a way that optimal solutions to the single-objective problem are Pareto optimal solutions to the original multi-objective problem [28]. Using different scalarization parameters, different Pareto optimal solutions can be produced. There are two popular scalarization methods.

a) Linear scalarization (weighted-sum) methods [29]:

$$\min_{\phi \in C} F = \sum_{i=0}^{k} \omega_i f_i(\phi), \tag{2}$$

which can be extended to more general non-linear forms.

b) ϵ-constraint methods [29] optimize one of the objectives and reformulates the remaining as constraints:

$$\begin{aligned} &\min_{\phi \in C} \ f_j(\phi) \\ &\text{s.t.} \quad f_i(\phi) \leq \epsilon_i, i = 1...k, i \neq j, \ \phi \in C. \end{aligned} \tag{3}$$

B) *Methods based on lexicographic ordering* assume that the multiple objectives in Eq. (1) can be ranked in order of importance, with f_1 being the most important to the decision maker and f_j, the least important. The following sequence of optimization problems are then solved one at a time:

$$\begin{aligned} &\min_{\phi \in C} \ f_l(\phi) \\ &\text{s.t.} \quad f_j \leq \mathbf{y}_j^*, \ j = 1...k, \ j \neq l, \ \phi \in C, \end{aligned} \tag{4}$$

where $\mathbf{y}_j^*$ is the optimal value of the above problem with $l = j$.

C) *Methods based on evolutionary multi-objective optimization (EMO)* simulate the natural evolution by an iterative computation process. An initial population is first created according to a predefined scheme. Then a loop (generation) consisting of evaluation, selection, recombination, and/or mutation is executed a number of times until some termination condition is met. The best individuals left in the population are output as Pareto optimal solutions. Examples of popular EMO algorithms include NSGA-II [30] and SPEA-2 [31].

D) *Other methods.* In some special cases, no-preference methods [32] and interactive methods [33] can be used.

In this paper, we solve Eq. (1) using CNNs trained by a back-propagation solver with mini-batch gradient descent. As gradient values must be propagated backward in every iteration, only the formulations based on weighted sum in Eq. (2) and ϵ-constraint in Eq. (3) are applicable. Using a weighted sum is easy because its gradient can be directly calculated. However, for the ϵ-constraint method, it is impossible to discard solutions out of the ϵ-constraint in the mini-batch-based training scheme. The only possibility is to give a heavy penalty when $f_i(\phi) > \epsilon_i$ and zero penalty otherwise in order to make most of the training samples within the ϵ-constraint. This is similar to the widely used hinge-loss method.

III. Metrics of Features Satisfying Principles

To develop superior features for dense matching, we first present the metrics on features that satisfy the consistency and distinctiveness principles discussed in Section I.

ϕ	– feature extractor;
$\phi(p)$	– feature descriptor at pixel p;
l_0	– ground truth displacement;
l_i	– candidate displacement in label space S ($l_i \in S$); (*e.g.* for stereo, l_i is the candidate disparity value).

For stereo matching, stereo images are well rectified, with displacements in the x direction. In contrast, displacements for optical flow occur in both the x and y directions. In evaluating features for dense matching, there must be a related image pair (such as a stereo pair) and a correspondence map that describes the displacement between the corresponding pixels in the two images. Such a correspondence map provides the ground-truth displacement l_0 at each pixel.

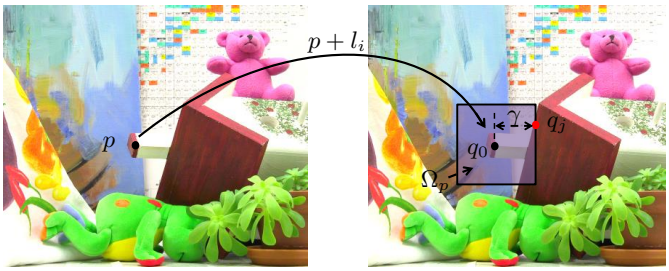


Fig. 4: An illustration of the definition in Eq. (9). Here, $q = p + l_0$ is the best matched location for p. The red pixel $q_j = p + l_j$ is the closest outlier to q. Hence, the width and height of Ω_p is $\gamma = \|q - q_j\|_\infty = \|l_0 - l_j\|_\infty$.

In all stereo-matching and optical-flow datasets, ground truths are usually available and obtained by radar (for natural images). New features can be learned and validated on the training data based on the ground truths and then generalized to test data with the ground truths hidden.

For any pixel p in one view and q_i in another ($q_i = p + l_i$ as shown in Fig. 4), let $d(p, q_i)$ be the distance between the feature descriptors of p and q_i. For convenience, we set:

$$d(p, q_i) = d(\phi(p), \phi(q_i)) = d(p, p + l_i) = d(p, l_i). \tag{5}$$

Although there are many popular functions for $d(p, q_i)$, such as $L-1$ distance $\|\phi(p) - \phi(q_i)\|_1$ and $L-2$ distance $\|\phi(p) - \phi(q_i)\|_2$, we introduce in Eq. (15) of Section V our own distance functions that can be specialized for dense matching applications and tailored to different feature extractors.

We then define the metrics for the two principles below.

1) *Consistency Principle.* The principle requires a feature point to own the same or similar feature descriptors when it appears in different views. For $d(p, q)$ defined in Eq. (5), the principle can be stated in terms of the consistence measure.

$$f_1(\phi) = d(p, q) = d(p, l_0) = 0 \ \text{ or } \ \approx 0. \tag{6}$$

To increase the consistency of features between different views, f_1 must be as small as possible. In practice, large $d(p, q)$ in some regions will lead to poor accuracy in matching algorithms. For example, regions with noises and illuminance changes always have larger average $d(p, q)$.

2) *Distinctiveness Principle.* For pixel p in an image, let $p_i \neq p \in \Omega_p$, where Ω_p is the surrounding region (called *distinctive region*) centered at p in the same image (to be defined more formally later). The principle requires a feature to be distinct enough with respect to other pixels in its surrounding region. It can be stated formally as follows:

$$\forall \ p_i \in \Omega_p, \ p_i \neq p, \ d(p, p_i) > m, \tag{7}$$

where m is the threshold related to the vision system. For instance, the threshold is known as the just-noticeable difference (JND) in human vision systems [34], [35]. There are plenty of studies that focus on JND thresholds, which decide whether $d(p, p_i)$ is "good." Note that the performance of a feature is

TABLE I: Error rates under different precisions based on the results of MC-CNN [14] from the KITTI 2012 stereo benchmarks. The results illustrate that more errors appear around ground truths.

Error Threshold	Error of All Regions	Increased Error Rate
2 pixels	5.45 %	1.82%
3 pixels	3.63 %	0.78%
4 pixels	2.85 %	0.46%
5 pixels	2.39 %	–

highly influenced by the size of its distinctive region $|\Omega_p|$ (the larger the better). For example, Eq. (7) will not be satisfied for pixels in a smooth region and $|\Omega_p| \approx 0$.

Since the consistency principle defines a relation between different views, whereas the distinctiveness principle states a condition between a pixel and its neighborhood in the same view, we can deduce from the consistency principle $d(p_i, q_i) \approx 0 \Rightarrow d(p, q_i) \approx d(p, p_i)$ and $d(p, q) \approx 0$, where $q = p + l_0$ is the best matched location for p. Based on Eq. (6), we have $d(p, q_i) = d(p, l_i)$ and $d(p, q) = d(p, l_0)$. The distinctiveness principle between different views can be formulated as follows.

$$d(p, l_i) - d(p, l_0) > m' \quad \forall q_i = p + l_i \in \Omega_q, \; l_i \neq l_0, \tag{8}$$

where m' is the new JND threshold that varies in different matching algorithms.

As shown in Fig. 4, we define the *distinctive region* $\Omega_p = \Omega_q$ with width/height of $2\gamma^*$ and ground truth displacement l_0 as:

$$\Omega_p = \{p + l_i \mid l_i \in S, \; \|l_i - l_0\|_\infty < \gamma^*\} \text{ with } \gamma^* = \max \gamma$$
$$\text{s.t. } d(p, l_i) - d(p, l_0) > m', \forall \, \|l_i - l_0\|_\infty < \gamma, l_i \neq l_0 \in S.$$

We now define f_2 as a metric (normalized to the [0,1] range) for the distinctiveness principle.

$$f_2(\phi) = |\Omega_p|/|S|, \tag{9}$$

where $|S|$ is the number of candidates in the search space S. The value is better if it is larger. For example, when it is 1, the feature at p is distinctive in the entire displacement space.

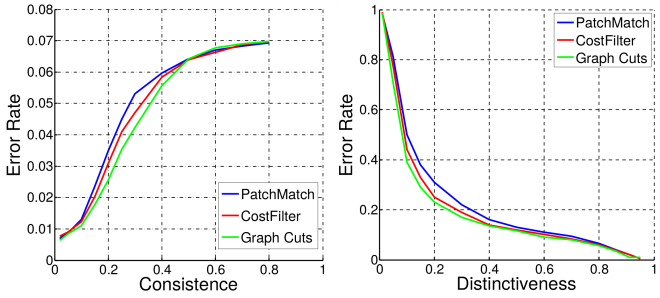
Before understanding the definition of f_2, two important properties of any dense matching algorithms must be emphasized. Firstly, outliers (for pixel p, l_j is an outlier displacement for p if $d(p, l_j) - d(p, l_0) \leq m'$) are more likely to appear around a ground truth because pixels close to each other in an image usually share similar texture, illuminance and color conditions that make them hard to differentiate. Secondly, an outlier close to a ground truth has significant influence on the accuracy of matching, not because of its high similarity to the ground truth but also their spatial proximity. These facts make it hard to find correct matches among outliers close to a ground truth.

Table I illustrates the above observations on some dense matching results. It shows that 1.82% pixels are matched at the wrong locations 2-3 pixels away from the ground truths; 0.78% are wrongly matched 3-4 pixels away; but only 0.46% are wrongly matched 4-5 pixels away.

The definition of f_2 is designed based on the properties above. The value of f_2 is decided by the outlier closest to the ground truth and defines the size of the distinctive region in which there is no outlier.

(a) (b)

Fig. 5: An illustration of the distinctiveness principle with two matching-cost maps. Each grid corresponds to one candidate that matches q_i for p, with its color representing the value of $d(p, q_i)$ (the brighter the larger). Our target is to find the center grid corresponding to ground truth q_0, where any grid darker than the center is an outlier. (a) $f_2 = 0$, with 24 outliers close to the ground truth. (b) $f_2 \approx 0.5$, with more than 100 outliers that are all far from the ground truth.



(a) Effect of consistency (b) Effect of distinctiveness

Fig. 6: Effects of each principle on accuracy of the matching results. (a) The effect of the consistency principle (f_1) on accuracy of the stereo matching results. Only regions where the distinctiveness principle is well satisfied is counted ($f_2 \geq 0.95$). (b) The influence of the distinctiveness principle (f_2) on result accuracy. Only regions where the consistency principle is well satisfied is counted ($f_1 \leq 0.02$). We use the evaluation method suggested in [8], [17] to collect the accuracy data. Different lines represent different matching algorithms, including a local method Costfilter [1], a global method graph cuts and a continuous matching algorithm Patchmatch [18].

As illustrated in Fig. 5(a) with $f_2 = 0$, although there are only 24 outliers, it is hard for any matching algorithm to find the center best matches. However, in Fig. 5(b) with $f_2 \approx 0.5$, there are more than 100 outliers, but it is much easier to find the center best matches. When the candidate displacement drops into the distinctive region, the matching algorithm will be guided to converge to the center pixel during cost aggregation.

Both principles are indispensable for characterizing good features. As an illustration, we collect statistics on the effect of one when that of the other is controlled. For example, we measure the effect on $d(p, q)$ by keeping the left view fixed and by adding noise and radiometric changes to the other. We then collect the data from those regions where distinctiveness is well satisfied. Fig. 6(a) shows how changing $f_1(\phi)$ can influence the performance of a feature (in terms of result accuracy) when $f_2(\phi)$ is fixed. The accuracy statistics is collected using the evaluation method suggested in [8], [17]. Likewise, we choose those regions where the consistency principle is satisfied and then collect the accuracy of these regions to see its relationship to f_2. Fig. 6(b) shows the trend on result accuracy due to changing $f_2(\phi)$ when $f_1(\phi)$ is limited to a small value.

IV. Optimization Model for Finding Good Features

Based on f_1 in Eq. (6) and f_2 in Eq. (9), we can formulate the feature extraction problem as a multi-objective optimization over C, a set of possible calculations/transformations (such as Laplacian of Gaussian (LoG) and Sobel operator) over pixels:

$$\min_{\phi \in C} F = \begin{bmatrix} f_1(\phi) \\ -f_2(\phi) \end{bmatrix}. \tag{10}$$

Eq. (10) defines the objective on good features for dense matching. Its solution $\{\phi_1, ...\phi_n\}$ (feature extractors) is a set of Pareto optima that are decision vectors whose objective cannot be improved in any dimension without degrading the other. Specifically, solution ϕ_1 dominates ϕ_2 ($\phi_1 \succ \phi_2$) when

$$\begin{aligned} & f_1(\phi_1) < f_1(\phi_2) \ \& \ f_2(\phi_1) \geq f_2(\phi_2) \\ \text{or} \quad & f_1(\phi_1) \leq f_1(\phi_2) \ \& \ f_2(\phi_1) > f_2(\phi_2). \end{aligned} \tag{11}$$

Solutions that are not dominated by others are called non-dominated, and the set of all such solutions form a Pareto-optimal set or frontier. Our target is to systematically find one or more Pareto-optimal solutions for Eq. (10).

In Eq. (10), f_1 can be easily computed when given the feature descriptors ϕ. However, f_2 is expensive to evaluate because, as defined in Eq. (9), we need to traverse all candidate displacement values in the whole displacement space S in order to calculate f_2 for each pixel. Moreover, f_2 is non-differentiable with respect to ϕ.

To reduce the complexity of calculating f_2, we develop f_2' to replace f_2 by using the following trend information on f_2. According to Eq. (9), we have the following observations.

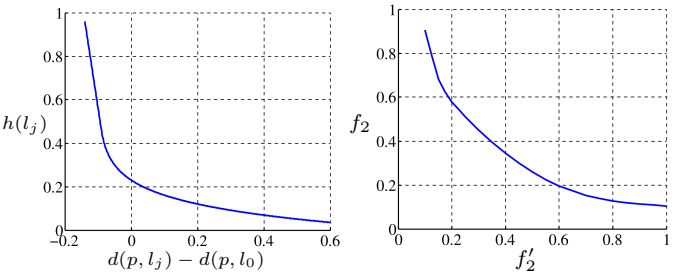
1) A higher $f_2(\phi)$ comes with a lower outlier rate. As shown in Fig. 4, Ω_p and f_2 for p are decided by the closest outlier l_j to ground truth displacement l_0. Given any outlier distribution, f_2 is, therefore, directly proportional to the outlier rate.

2) A high $f_2(\phi)$ also means that all the outliers are far from ground truth l_0; that is, the value of γ^* in Eq. (9) should be as large as possible.

Hence, to increase f_2, we can decrease the outlier rate as well as control the distribution of outliers in order to make them all far from the ground truth. Fortunately, these targets can be easily achieved using $f_2'(\phi)$ defined for each pixel p:

$$\begin{aligned} f_2'(\phi) \quad &= \quad \frac{1}{|U|} \frac{\sum_{l_j \in U} w_j h(l_j)}{\sum_{l_j \in U} w_j} \\ \text{where} \quad & h(l_j) = \delta \cdot h_1(l_j) + (1-\delta) \cdot h_2(l_j) \\ & h_1(l_j) = -\tau \cdot \log(d(p,l_j) - d(p,l_0) + \tau) \\ & h_2(l_j) = -\frac{\tau \left(d(p,l_j) - d(p,l_0) + \tau\right)}{\epsilon} \\ & \qquad -\tau \cdot \log(\epsilon) + \tau \\ & \delta = \begin{cases} 1 & \text{if } d(p,l_j) - d(p,l_0) + \tau > \epsilon \\ 0 & \text{otherwise} \end{cases} \\ & w_j = \exp\left(-\frac{\|l_j - l_0\|_\infty}{r}\right). \end{aligned} \tag{12}$$

Here, $U \subseteq S$ is a small subset from the whole displacement space S chosen to reduce the high computational cost of f_2. $|U|$ represents the number of samples in U; w_j is a popular spatial Gaussian weight for controlling the distribution of outliers (to give different penalties to make them far from l_0); τ is used to adjust the penalty to sample l_j; and h is a piecewise function of h_1 and h_2 chosen as follows. h_1 is a slightly altered log function of the hinge/margin $\Delta d = d(p,l_j) - d(p,l_0)$: it gives large penalties if Δd is small or negative, and small penalties otherwise. h_2 is a supplementary function for h_1: it has the same value and derivative/gradient at the intersection point with h_1, and helps set an upper bound to the derivative/gradient of h_1 to avoid it becoming infinity. ϵ is a very small value chosen to avoid h_2 being infinity or an imaginary number. As $d(p,l) = d(\phi(p), \phi(p+l))$ is differentiable with respect to ϕ, h and f_2' are continuous and differentiable, although they are both piecewise functions. In short, the components of f_2' are chosen to give large penalties to closer outliers, while having small penalties to non-outliers.



(a) Trend of $h(l_j)$ (b) Trend of f_2' with respect to f_2

Fig. 7: An illustration of the monotonic behavior of f_2' with respect to f_2. (a) Monotonic trend of $h(l_j)$ with respect to $d(p,l_j) - d(p,l_0)$. (b) Corresponding monotonic relation between f_2 and f_2'. Data for f_2 and f_2' is collected by sampling from the KITTI datasets. To calculate f_2, we use $m' = 0$. Note that different m' does not change the monotonic behavior.

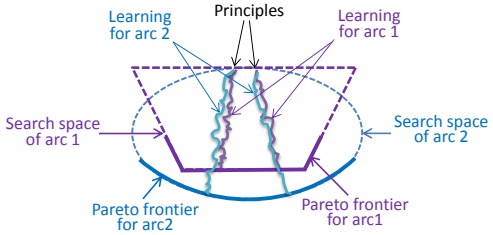


Fig. 8: An illustration of the orthogonal relationship among the principles on consistency and distinctiveness, network architectures, as well as training/optimization procedures in our learning system.

Fig. 7(a) shows that $h(l_j)$ is effective for reducing the outlier rate. When l_j is an outlier, it means that $d(p,l_j) - d(p,l_0)$ is small or even negative, and $h(l_j)$ gives it a higher penalty. On the other hand, as different matching algorithms have different JND (m'), it is hard to set a fixed value for m'. Instead of setting a fixed value like hinge loss, the definition of $h(l_j)$ can increase generalizability of the features learned and remove the influence of m' in our method.

Fig. 7(b) shows that f_2 is monotonically decreasing with increasing f_2'. We calculate f_2' by sampling p and l_j from the KITTI datasets with the condition of f_1<0.02. f_2 is then calculated by brute-force sampling. Both the sampled values of f_2 and f_2' are then averaged over a set of pixels. In this and the following experiments, we used r=10, τ=0.1, ϵ=0.1τ. These parameters are used to adjust the penalty and gradient of the function, as they help limit the maximum of

the derivative/gradient's absolute value $\|\frac{\partial(f_2')}{\partial(\Delta d)}\|=\frac{\tau}{\epsilon}=10$ and set the baseline $\|\frac{\partial(f_2')}{\partial(\Delta d)}\|=1$ if $\Delta d=0$. These choices are not unique, and other reasonable values can also be used.

Finally the original objective in Eq. (10) is transformed to include f_1 and f_2' that are differentiable with respect to ϕ:

$$\min_{\phi\in C} F' = \begin{bmatrix} f_1(\phi) \\ f_2'(\phi) \end{bmatrix}. \tag{13}$$

V. Implementing Feature Extractors Using CNNs

Recently, CNNs trained by back-propagation [36] have been found to perform well on large-scale computer-vision tasks, with superior performance on feature extraction for classification and recognition [37], [38]. In this section, we develop a method based on CNNs to solve Eq. (13).

In the search space defined by a given CNN architecture, the multi-objective formulation in Eq. (13) identifies the possible tradeoffs between the two principles in this search space. The network optimization/training procedure is used to find a path in this space towards those solutions that are on or close to the Pareto frontier with the best tradeoffs. That is, different CNN architectures will lead to different Pareto frontiers as well as tradeoffs. Fig. 8 illustrates the orthogonal relationship among the principles, the CNN architecture, and the learning algorithm. The architectural setting defines the corresponding search space and limits where solutions can be found.

Among the principles, the architectures and the learning algorithms, we find the architectures to be the most flexible because they are always designed empirically. More advanced architectures usually lead to an expanded solution space and better solutions. However, when limited by time and computational resources, it is impossible to try all possible architectures one by one in order to find the best solution.

It is important to point out that our focus in this paper is on the tradeoffs between the proposed principles. With this in mind, we employ simple 5-layer network architectures as base settings in order to verify the effectiveness of the principles, and further evaluate in Sec. VII two promising architectures besides the original settings. Since the network architectures are orthogonal to the tradeoffs studied, the two principles proposed in this paper are always applicable in any architecture, as they define good features for dense matching and provide guidance for any learning system.

A. CNN Architectures Studied

Fig. 9 shows our proposed CNN architectures for stereo and optical flows, and Table II lists the configurations of two versions, one designed for speed and the other for accuracy.

For the version designed for speed, we employ a one-branch fully-connected CNN structure in which both views share the same model for feature extraction. The input data can be any $k \times k$ $(k > 10)$ sub-images. We find that this approach increases the consistency of the features obtained and helps training converge faster. With only one branch, we do not need to exchange the left and right views as well as to compute the feature maps twice to obtain the matching results for both views. To accelerate convergence without affecting accuracy,

TABLE II: Configurations of the CNN architectures studied for stereo and optical flows. Two versions, one designed for speed and the other for accuracy, are used. In these architectures, padding is not employed, and batch normalization [39] is embedded before each ReLU and Max-Pooling layer. The output of each layer is set as $n-$channel.

Arc	Stereo		Optical Flow	
	One-branch Fast	Two-branches Accurate	One-branch Fast	Two-branches Accurate
Input	$k \times k \times 1$ ——	$11 \times 11 \times 1$ normalize	$k \times k \times 1$ ——	$31 \times 31 \times 1$ normalize
Layers	$\begin{bmatrix}\text{conv}, 3\times 3\\ \text{ReLU}\end{bmatrix}\times 4$	$\begin{bmatrix}\text{conv}, 3\times 3\\ \text{ReLU}\end{bmatrix}\times 4$	$\begin{bmatrix}\text{conv}, 5\times 5\\ \text{ReLU}\end{bmatrix}\times 4$	$\begin{bmatrix}\text{conv}, 3\times 3\\ \text{pooling}, 2, 2\end{bmatrix}\times 3$
	$[\text{conv}, 3\times 3]\times 1$	$[\text{conv}, 3\times 3]\times 1$	$[\text{conv}, 5\times 5]\times 1$	$[\text{conv}, 3\times 3]\times 1$
Out	Sigmoid	——	Sigmoid	——

we use the sigmoid function after the last layer to restrict the range of values to (0,1).

For the version designed for accuracy, we employ a symmetric two-branch architecture, one for the left view and another for the right. Its inputs are fixed-size patch pairs at each pixel. Before they are input to the network, we reduce the influence of radiometric changes between the views by normalizing both patches (through subtracting the mean and dividing by the standard deviation). To extract features for the whole images after training, we use mirror-padding in each convolutional layer for pixels in image borders.

B. Loss Functions

To solve Eq. (13), we employ the following loss function ℓ to train the feature extractor. For the input left view I and the corresponding right view I',

$$\ell = \frac{1}{|I|}\sum_{p\in I}\left((1-\lambda)f_2' + \lambda f_1^3\right), \tag{14}$$

where $|I|$ is the number of valid training samples, and f_1 and f_2' are defined in Eq's (6) and (12), respectively. We use λf_1^3 to control the consistency principle in order to achieve different non-dominated solutions. Different solutions can be found by adjusting λ and/or by changing the form of f_1^3.

Our loss function follows the weighted-sum method while using a cubic form of f_1. This realizes the target by using a small penalty when $f_1(\phi)$ is relatively small and a heavy penalty when $f_1(\phi)$ is relatively large. By giving different values to λ, we can obtain a set of non-dominated solutions.

Before deciding on the form of the loss function, we tried $d = 1, 2, 3, 4$, respectively, in f_1^d. In each case, we collected several solutions by adjusting λ. Finally, we found that the current cubic form works the best.

In solving Eq. (13), it is impossible to get all the non-dominated solutions on the continuous Pareto frontier, which does not have a closed form. Due to the high cost in training CNNs, we find four non-dominated solutions for each architecture. More solutions can be obtained by changing λ and by training the CNN again. These solutions are non-dominated with respect to each other; that is, no one solution is better than another, and each may perform differently under

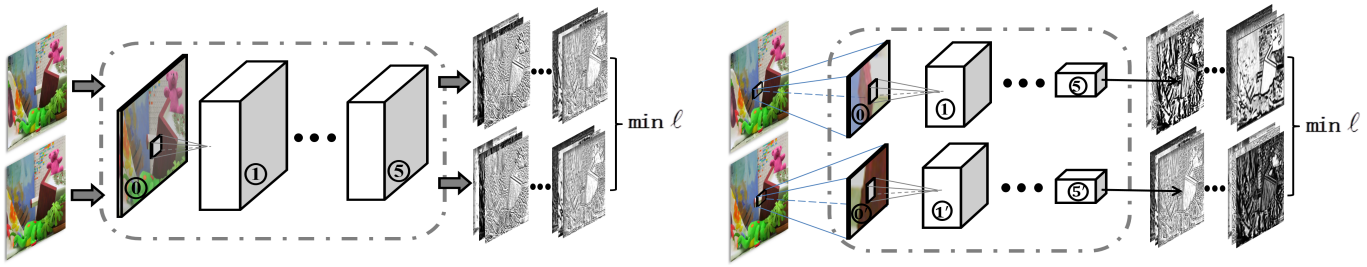


Fig. 9: CNN architectures used as feature extractors. *Left panel*: fast architecture, with left and right views using the same network as feature extractors. Inputs can be images of any size. *Right panel*: two-branch accurate architecture, with both left and right views having its own network branch as feature extractor. Inputs must be 11×11 or 31×31 patches; $M\times N$ images must be sliced to MN patches before input.

different matching algorithms. For example, in some local matching algorithms with the Winner-Take-All strategy (such as CostFilter [1]), the feature when setting $\lambda = 0$ gets the best accuracy. In other global matching algorithms (such as PMBP [5]), spatial propagation (such as PatchMatch [18]) and pyramid matching (pyramid-MDP [24]), features by setting λ in $[0.4, 0.6]$ perform a little better.

Based on the fast and accurate CNN architectures, we can define the distance-measurement function (cost function) of two feature descriptors $d(p, l_i) = d(\phi(p), \phi(p + l_i)) = d(\phi(p), \phi(q_i))$ as follows:

$$d(p,l_i) = \begin{cases} \frac{1}{2}\left\| \frac{\phi(p)}{\|\phi(p)\|_2} - \frac{\phi(p+l_i)}{\|\phi(p+l_i)\|_2} \right\|_2^2 & \text{fast} \\ \text{sigmoid}\left(\alpha \begin{bmatrix} \phi(p) \\ \phi(p+l_i) \end{bmatrix} + \beta\right) & \text{accurate.} \end{cases} \quad (15)$$

For the fast (*resp*., accurate) architecture, we use the first (*resp.*, second) form as the cost function. Here, α and β are both learned parameters that can be implemented using one fully connected layer. The time complexities of both functions in Eq. (15) are $O(n)$, where n is the number of channels in the feature descriptor. More complex forms are not employed, as they highly influence a matching algorithm's complexity.

These two functions both restrict the range of distance values to [0,1]. There are two main reasons for using this range. Firstly, restricting the distribution of distance values will help adapt them easier in different matching algorithms, as it is easy to adjust parameter settings in different matching algorithms (such as setting the balance between matching-cost and smoothness terms). Secondly and more importantly, this helps reduce the influence of some extreme outliers. Without restricting the matching cost, $d(p, l_0)$ of the best matches in some difficult cases can be very large and will significantly influence the neighborhoods in cost aggregation of matching algorithms, leading a whole region to wrong matches.

a) Preprocessing of training data: Training data are constructed from the KITTI stereo and optical-flow datasets [10]. For the fast architecture, each whole image is normalized by subtracting its mean and dividing by its standard deviation. As mentioned before, the fast architecture can use any size of sub-images as inputs. To ensure the same size of training-image pairs, all training images are sliced into overlapping 71 × 71 sub-image pairs. The center of each pair is required to be highly matched (*e.g.* the KITTI 2012 dataset contains 194 image pairs that can be sliced to about 200,000 sub-image pairs). In contract, for the accurate architecture, training data are sliced into 11 × 11 (stereo) or 31 × 31 (flow) patch pairs as in [14]. During the training process, all invalid regions and pixels, such as occlusions and pixels without ground-truth displacements, are neglected.

TABLE III: Approaches and parameter ranges for training-data augmentation on the KITTI datasets

Types	Parameters	Ranges	
		stereo	optical flow
Radiometric	*contrast*	$[0.8, 1.2]$	$[0.8, 1.2]$
	constrast_dif	$[-0.15, 0.15]$	$[-0.25, 0.25]$
	brightness	$[-0.3, 0.3]$	$[-0.3, 0.3]$
	brightness_dif	$[-0.2, 0.2]$	$[-0.4, 0.4]$
Scale&Rotation	*scale*	$[0.9, 1.1]$	$[0.75, 1.25]$
	scale_dif	$[-0.1, 0.1]$	$[-0.25, 0.25]$
	angle	$[-10°, 10°]$	$[-20°, 20°]$
	angle_dif	$[-10°, 10°]$	$[-20°, 20°]$
Noise	*mean*	$[-0.05, 0.05]$	$[-0.05, 0.05]$
	stdev	$[0, 0.2]$	$[0, 0.2]$

As proposed in Eq. (12), we use a small sample displacement set $U \subseteq S$ to improve training efficiency. In this paper, $U = U_1 \cup U_2$ contains two parts: U_1 is a sample set of integer-precision displacements and U_2 a set of floating/sub-pixel displacements. $d(p, l_j)$ and $d(p, l_0)$ can be easily achieved if $l_j \in U_1$. If $l'_j \in U_2$, we use the bilinear interpolation to obtain $d(p, l'_j)$ and $d(p, l'_0)$. (l'_0 is also floating-precision if $l'_j \in U_2$.)

C. Details of Implementing our Learning Algorithms

b) Data-set augmentation: Augmenting the training data set is commonly used to reduce generalization errors. To make the learned feature robust to imperfections such as noise, illuminance changes and view rotations, for sub-images I^l and I^r of the, respectively, left and right views, we control their radiometric variance by $I^l_n = I^l \times \text{constrast} + \text{brightness}$ and $I^r_n = I^r \times (\text{constrast} + \text{constrast\_dif}) + (\text{brightness} + \text{brightness\_dif})$. Further, to make the feature extractor robust to noise, Gaussian noises (controlled by their mean and standard deviation) are randomly added to each sub-image pairs during training. Likewise, each pixel in the training data is scaled and rotated by the scale and rotation parameters. Table III shows the ranges in which the random values are taken for these radiometric, noise, scaling and rotation parameters.

c) Learning platform and parameters: We modified the CNN platform Caffe [41] to implement our own learning procedures. For training-data sampling, we set $U = \{l_i | l_i \in S, |l_i - l_0|_\infty > 3\}$ and the sampling numbers $|U_1| = 2$ and $|U_2| = 1$. Mini-batch gradient descent with the momentum term set to 0.9 is used to minimize the loss. The batch size is set at 256 (*resp.*, 64) for the accurate (*resp.*, fast) architecture. The models are trained for a total of 800K iterations with

TABLE IV: Performance evaluation (in error rate %) and comparisons of different features in various matching algorithms

Methods & Applications		Color	Gradient	Color+Gradient	Census	SIFT*	SURF*	MC-CNN [14]		Our Feature	
								fast	accurate	fast	accurate
CostFilter [1]	stereo	30.19%	23.1%	21.51%	11.54%	9.13%	9.91%	6.49%	6.04%	6.01%	**5.67%**
	flow	39.94%	28.31%	27.5%	19.1%	17.51%	18.19%	×	×	10.47%	**9.34%**
particle BP [7]	stereo	21.9%	14.1%	13.92%	10.21%	8.87%	9.19%	6.17%	5.49%	5.51%	**5.04%**
PatchMatch [18]	stereo	19.92%	14.81%	14.25%	×	×	×	×	×	4.74%	**4.37%**
	flow	23.9%	15.51%	13.35%	×	×	×	×	×	5.74%	**5.17%**
PMBP [5]	stereo	17.99%	14.1%	12.95%	×	×	×	×	×	4.64%	**3.83%**
	flow	21.79%	14.61%	10.5%	×	×	×	×	×	5.46%	**4.81%**
SGM [6]	stereo	11.75%	6.91%	7.01%	5.17%	4.92%	5.01%	3.41%	3.09%	3.24%	**3.01%**
pyramid-MDP [24]	flow	15.9%	13.41%	11.15%	×	×	×	×	×	5.84%	**5.17%**

∗ Keypoint-detection step is discarded for SIFT [11] and SURF [12] features (refer to the dense SIFT implementation in [40]). Descriptors are generated at each pixel in order to run above dense matching algorithms.

the learning rate initially set to 0.01, and then decreased by a factor of 10 at the 300K$^{\text{th}}$ and the 600K$^{\text{th}}$ iterations.

VI. Experimental Results[1]

A. Parameter settings

There are two key parameters in our learning system. The first is n, the number of channels in the CNN architecture. It denotes the length of the feature descriptor that highly influences the accuracy and efficiency of the features found. We set n to different integers (64, 128, 256) and train the CNN to get the feature descriptors. Section VI-E analyzes the effect of n on complexity. The second key parameter is λ in Eq. (14) that balances the weights between consistency and distinctiveness. Different λ will lead to different non-dominated solutions. In our experiments, we set λ is to $\{0.8, 0.6, 0.4, 0\}$ to get four sets of non-dominated features.

All the other parameters, including learning rate, training iterations, batch-size and data augmentation parameters, are set empirically. Reasonable settings are decided using trial-and-error, background experience, and suggestions from related works. Until now, there is no easy way to automatically set the best values for these parameters in learning CNNs.

B. Evaluations of Features Found by Our Approach

The learned features (in 256 channels) are evaluated in various existing dense matching (stereo matching or two-frame optical-flow) algorithms. The features found by our method are flexible and can be used for subpixel-location interpolation and size/shape scaling for each channel, allowing them to be used in almost all existing dense matching algorithms. One or two algorithms are selected for each of the 5 existing classical frameworks and their performance is compared using different features. The matching algorithms studied include cost-filtering methods [1], global-matching algorithms [7], semi-global matching algorithms [6], continuous (subpixel accuracy/slanted surface) matching algorithms [5], [18], and pyramid-matching algorithms [24]. The features (or the matching-cost method) for comparisons include Census (used in [6]), color, gradient, color+gradient [18], Dense SIFT [11]/SURF [12] and MC-CNN matching cost [14].

In our evaluations, test data are chosen from the KITTI 2012 (stereo and flow) dataset: 194 training-image pairs are randomly divided into 4 sets (each with 48 or 49 two-view pairs), with 3 sets chosen for training and the rest for testing.

Table IV presents the performance results, which are evaluated by the 3-pixel-threshold error rate of the non-occluded regions and calculated by:

$$er = \frac{1}{|I|}\sum_{p\in I}\delta_p \tag{16}$$

$$\delta_p = \begin{cases} 1 & \text{if } \|l(p) - l_0(p)\|_2 > 3 \\ 0 & \text{otherwise,} \end{cases}$$

where $|I|$ is the number of valid pixels in an image; $l(p)$ is the matching result at pixel p; and $l_0(p)$ is the ground-truth displacement at p. The results show that our features found can significantly improve the performance of existing dense matching algorithms, especially with respect to popular features like color, gradient or census.

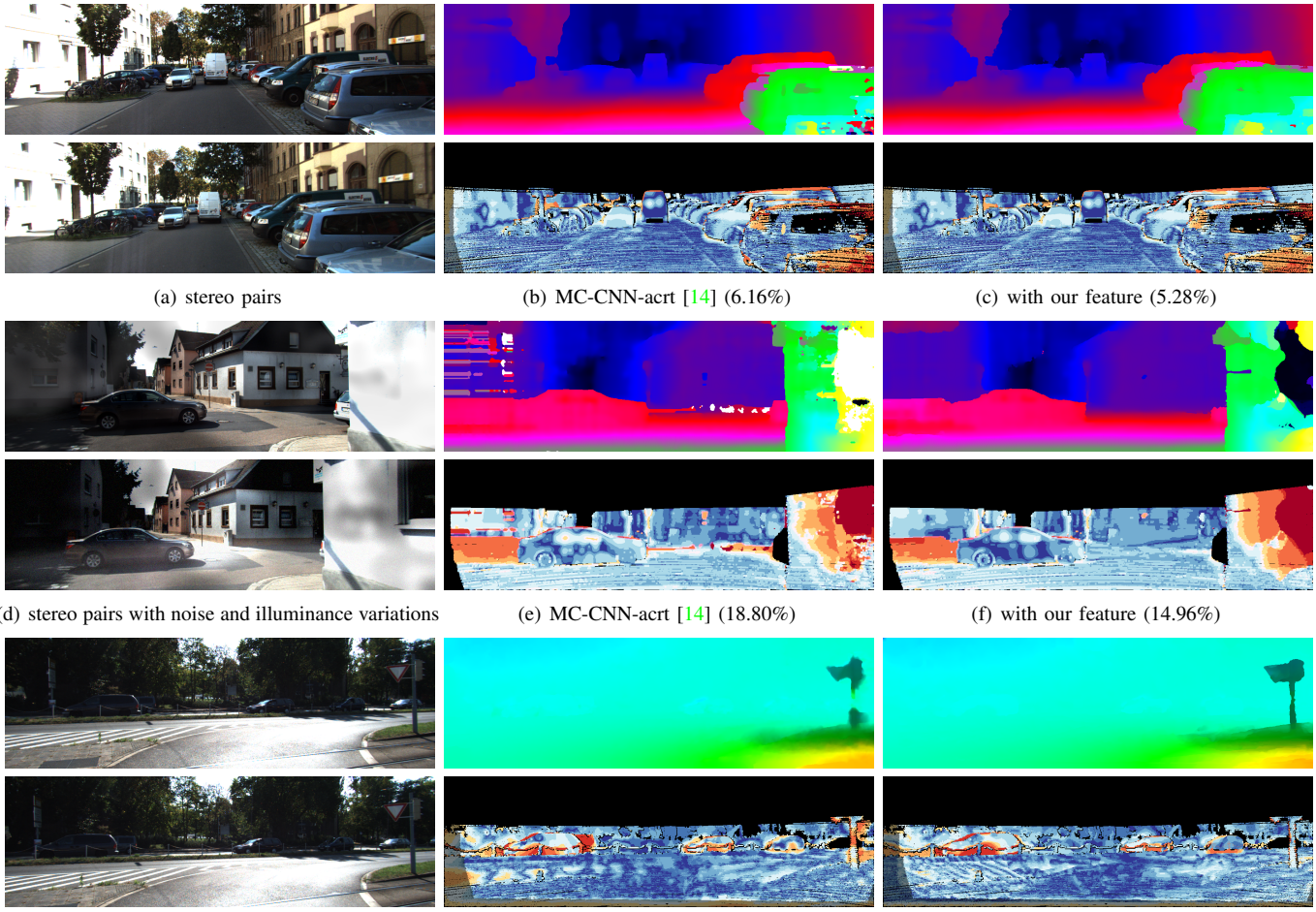
When compared to the current best MC-CNN matching-cost computation [14], our features perform better in all matching algorithms tested. Fig. 10 illustrates the improvements using examples from the KITTI datasets. It shows that matching algorithms using our features perform better in some large smooth regions such as cars. Further, our features are more robust to illuminance variations and noises. More importantly, they are general and can be used in all 9 stereo and optical-flow matching algorithms. In contrast, MC-CNN [14] can only be used in the three integer-precision stereo-matching algorithms due to its time complexity, as it needs to use all possible patch pairs to calculate the matching-cost matrix. This requirement is impractical for optical flow and continuous or pyramid matching algorithms.

C. Evaluations of Our Features Using KITTI Benchmarks

We evaluate our features using the stereo and optical-flow benchmarks by embedding it in one of the existing matching frameworks. For stereo evaluations, the whole stereo framework in [14] with the matching-cost matrix calculated by our accurate 256-channel features are employed. The depth edges are further refined by a weighted median filter (with radius=11, $\epsilon = 0.05$).

Table V shows the evaluation results on the stereo benchmarks. When evaluated on non-occlusion areas, our feature with the SGM stereo matching algorithm (CNNF+SGM) ranks

[1]We provide the demo code at https://github.com/feihuzhang/CNNF.



(a) stereo pairs (b) MC-CNN-acrt [14] (6.16%) (c) with our feature (5.28%)

(d) stereo pairs with noise and illuminance variations (e) MC-CNN-acrt [14] (18.80%) (f) with our feature (14.96%)

(g) optical flow pairs (h) PMBP [5] (7.99%) (i) with our feature (4.88%)

Fig. 10: Improvements using features found by our method. (b)∼(c), (e)∼(f) and (h)∼(i) are visualized disparity/optical-flow results (upper: color coded by KITTI stereo/flow benchmarks) and visualized error map (lower) corresponding to stereo/flow pairs (a) (d) (g), respectively. 3-pixel threshold error rates are reported in the captions. Gaussian noises (only right view) and irregular illuminance variations (both views) are added to the stereo pair of (d) to evaluate the robustness of the features to noise and illuminance variations.

TABLE V: Stereo matching evaluations (non-occlusion areas) on two KITTI stereo benchmarks with ranking data, error rates and run times reported. The stereo matching algorithm [14] with our learned feature achieves the current best accuracy.

KITTI 2012 Benchmark				KITTI 2015 Benchmark			
Method	Rank	Error Rate	Run Time (Device)	Method	Rank	Error Rate	Run Time (Device)
Our CNNF+SGM	**1**	**2.28%**	71s (GPU TESLA K40)	**Our CNNF+SGM**	**1**	**3.04%**	71s (GPU TESLA K40)
PCPB [42]	2	2.36%	68s (GPU Titan X)	Displets v2 [43]	2	3.09%	265s (CPU >8 cores)
Displets v2 [43]	3	2.37%	265s (CPU >8 cores)	PCPB [42]	3	3.17%	68s (GPU Titan X)
MC-CNN-acrt [14]	5	2.43%	67s (GPU Titan X)	MC-CNN-acrt [14]	4	3.33%	67s (GPU Titan X)
cfusion [44]	6	2.46%	70s (GPU Titan X)	Content-CNN [45]	6	4.00%	1s (GPU Titan X)

top in the KITTI 2012 and 2015 stereo benchmarks.[2]

For optical-flow evaluations, we only test our features on classical two-frame optical-flow applications. Since our method does not use extra information like multi-view frames, stereo/depth and/or semantic segmentation results, for fair comparison, methods like motion stereo flows and multi-view flows are not compared. PMBP [5] with slanted surface assumption (just like the PMBP-stereo) are implemented to collect the initial results for both the left and right views. Consistency checking as that in [14] is used to find occlusions and invalid regions which are recovered by an iterative weighted median filter (with radius=20, ϵ=0.075). (We applied the weighted median filter several times until all the invalid regions are filled with displacement values.) Finally, we employ a 5×5 median filter and a bilateral filter (with radius=11, ϵ=0.1). The above three post-refinement steps are repeated twice to collect the final results for benchmark evaluations.

The original PMBP [5] was run on a CPU with one thread. Its computational complexity is KNR^2, where K is a constant, N is the number of pixels in an image, and R is the patch width. It is slow and cannot be parallelized due to the spatial-propagation step. We use PMBP instead of other improved

[2]With respect to the public KITTI benchmarks at http://www.cvlibs.net/datasets/kitti/index.php, our visualized results and ranking data are available at (by clicking to access each) (a) 2012 stereo benchmarks; (b) 2015 stereo benchmarks; (c) 2012 flow bechmark; and (d) 2015 flow benchmarks.

TABLE VI: Optical-flow evaluations on two KITTI flow benchmarks with ranking data, error rates and run times reported. The optical-flow algorithm [5] with our learned feature achieves the current best accuracy among state-of-the-art two-frame optical-flow algorithms.

KITTI 2012 Benchmark				KITTI 2015 Benchmark			
Method	Rank	Error Rate	Running Time (Device)	Method	Rank	Error Rate	Run Time (Device)
Our CNNF+PMBP	**11**	**4.70%**	30min (CPU 1 core)	**Our CNNF+PMBP**	**4**	**12.26%**	45min (CPU 1 core)
PatchBatch-s	12	4.81%	60s (GPU)	SOF [46]	7	16.81%	6min(CPU)
CNN-HPM [47]	13	4.89%	23s (GPU)	JFS [48]	8	17.07%	13min (CPU)
PatchBatch [15]	15	5.29%	50s (GPU)	PatchBatch-s	10	17.69%	60s (GPU)
PH-Flow [49]	21	5.76%	800s (CPU)	PatchBatch [15]	14	21.69%	50s (GPU)

TABLE VII: Comparison of consistency and distinctiveness with respect to different features on the KITTI 2012 dataset

Feature Types	Stereo Matching		Optical Flow	
	f_1	f_2	f_1	f_2
Color	0.14	0.17	0.23	0.09
Gradient	0.10	0.29	0.16	0.14
Grad+Color	0.11	0.31	0.19	0.17
Census	0.08	0.49	0.15	0.28
dense sift	0.06	0.57	0.11	0.39
dense surf	0.07	0.52	0.13	0.32
MC-CNN fast [14]	0.20	0.77	×	×
MC-CNN acc [14]	0.13	0.80	×	×
Our Four Sets of	0.05	0.76	0.06	0.51
Non-dominated	0.09	0.79	0.11	0.55
Fast Features	0.15	0.81	0.17	0.58
Found	0.24	0.83	0.25	0.60
Our Four Sets of	0.07	0.80	0.05	0.57
Non-dominated	0.11	0.83	0.1	0.62
Accurate Features	0.16	0.86	0.16	0.65
Found	0.24	0.88	0.25	0.68

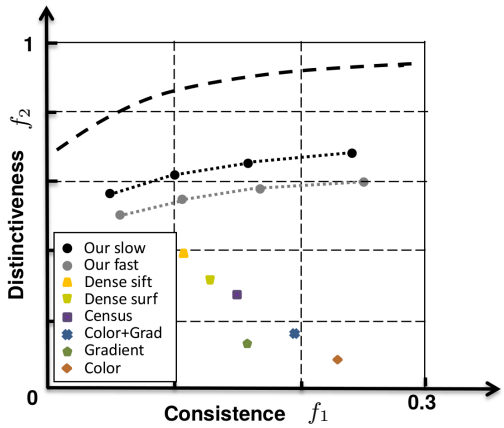


Fig. 11: Comparisons of various approaches based on the consistency and the distinctiveness principles (for optical flow). The x-axis represents f_1, the smaller the better. The y-axis shows f_2, the larger the better. Table.VII further shows the coordinate data. The figure shows two sets of four non-dominated solutions of our method, one set for the fast architecture and the other for the accurate architecture.

algorithms based on PMBP because it is the first and the most classical version among these algorithms. There are some faster versions, such as GC-LSL [50] and SPMBP [19] that achieved speed through limiting the candidate displacement space or discarding the spatial-propagation step. Our features can be easily introduced in all these algorithms to improve their accuracy. When our features are embedded, using half of the patch width slightly affects accuracy, but the whole algorithm can be sped up by $2 \sim 3$ times.

Table VI shows the ranks and the error-rate information. Our method ranks 11$^{\text{th}}$ on the KITTI 2012 benchmarks and 4$^{\text{th}}$ on the 2015 optical-flow benchmarks. More importantly, the results show that our method is the current best two-frame optical-flow method without using extra multi-view, motion stereo or semantic segmentation reinforcement.

D. Analysis and Comparisons of Results

In this subsection, we compare different features and explain why our features learned can outperform other state-of-the-art approaches. Features are compared with respect to how they satisfy the consistency and the distinctiveness principles by calculating f_1 (Eq. (6)) and f_2 (Eq. (9)). For f_1, we first normalize their values by subtracting $\min(f_1)$ and dividing by $\max(f_1) - \min(f_1)$, before comparing the average normalized values. Then, f_2 is calculated by assuming integer displacements. JND m' in f_2 is set to zero, as different values of m' do not change the relative partial order of solutions as well as the monotonic behavior of f_2 (as illustrated in Fig. 7). For example, our feature always dominates the others regardless of whether $m' = 0$ or $m' = 0.1$.

Table VII shows the values of f_1 and f_2 when setting λ to 0.8, 0.6, 0.4 and 0, respectively, to get four non-dominated solutions on the KITTI 2012 dataset. Fig's 3 and 11 further depict all the (f_1, f_2) tuples. The results show that the features found by our method are the closest to the Pareto frontier that achieve the best tradeoffs between consistency and distinctiveness. For stereo matching, we get the following dominance order: our feature (fast or accurate) $\succ$ MC-CNN (fast or accurate) $\succ$ dense sift $\succ$ dense surf $\succ$ Census $\succ$ gradient (or color+gradient) $\succ$ color. Such a dominance order is also visible in Fig. 3. For optical flows, we get similar conclusions as shown in Fig. 11.

E. Analysis of Time Complexities

In this subsection, we analyze the time complexities of our feature-extraction procedure and those of using it in existing matching algorithms. Each convolutional manipulation of one pixel in a channel is set as one basic unit of computation time (while ignoring activations, like ReLU, max-pooling and Sigmoid computations due to their negligible times). For example, for an n-channel fast-feature extractor, we have five convolutional layers for an N-pixel image. According to the CNN architectures in Table II, the time complexity for the fast-feature extractor is $O(n^2N)$, as it needs $(4n^2 + n)N$ units of convolutional computations. Table VIII lists the time complexities and elapsed times for all the feature extractors.

The time complexities of using our features in matching algorithms directly depend on the complexities of the cost functions in Eq. (15). Since both are linear with n channels

TABLE VIII: Efficiency and time complexity of our feature extractors without the matching algorithm. The running times are elapsed times for just one view (resolution: 1242×375). The numbers in brackets are the CPU run times.

Feature Extractor	Time Complexity	Running Time		
		$n = 16$	$n = 64$	$n = 128$
fast-stereo	$O(4n^2N)$	0.05s (4s)	0.15s (10s)	0.3s (19s)
fast-flow	$O(4n^2N)$	0.1s (7s)	0.4s (21s)	0.7s (37s)
acc-stereo	$O(84n^2N)$	3s	12s	21s
acc-flow	$O(195n^2N)$	7s	23s	35s

TABLE IX: Elapsed time (1-core CPU) when embedded in different matching algorithms.

Matching Alogirthm	Original Running Time	Running Time With Our Feature		
		$n = 16$	$n = 64$	$n = 128$
CostFilter [1]	8s	13s	27s	41s
PMBP flow [5]	800s	500s	1300s	2100s
Particle BP [7]	5s	7s	15s	27s
SGM [6]	3s	5s	11s	20s

TABLE X: Marginal effects of training data size on error rate, using Costfilter [1] in stereo matching with a 64-channel fast architecture as the feature extractor.

Number of Images (Samples)	f_1	f_2	Error Rate
100 (∼10M)	0.095	0.76	6.43%
146 (∼15M)	0.094	0.80	6.31%
200 (∼20M)	0.094	0.82	6.25%
300 (∼30M)	0.093	0.83	6.20%
346 (∼35M)	0.093	0.83	6.18%

of the feature descriptor, the time complexity when using our feature is $O(nM)$ if the complexity of the matching algorithm is $O(M)$. Table IX compares the run times for the matching algorithms before and after using our features.

F. Effects Due To Increased Training Samples

Appropriately increasing training samples will improve accuracy. Besides data augmentations, more training data can be introduced from other datasets. For example, we can combine the KITTI 2012 and KITTI 2015 datasets and get 394 image pairs. Table X illustrates that the number of training samples has a marginal reduction on error rates.

VII. Improvements with Advanced Architectures

In general, more complex/advanced CNN architectures will improve the accuracy of results. However, better accuracy always comes at higher computational and memory costs. To use the recently proposed residual and inception strategies, extra layers must be added to the original 5-layer network. For the residual strategy, we employ more advanced 4D transformed setting [51] instead of the original one-branch setting [52]. For the inception strategy, a four-branch inception block with residual connections similar to [53] is tested.

Fig. 12 shows these architectural blocks. Limited by available computational resources, only fast 64-channel architectures are tested to illustrate the performance differences. Table XI and Fig. 13 lead to the following observations.

a) Residual and inception strategies are all effective for getting results with better accuracy. But the memory and time

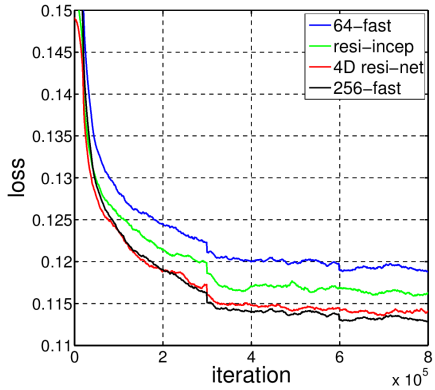


Fig. 13: Convergence of training in different network settings as illustrated by stereo matching. 4D-residual and residual inception blocks are implemented in the original 64-channel fast architectures to replace the original five convolutional layers.

required to train them increase significantly (2∼4 times when compared to the simple 5-layer net).

b) When compared to a 256-channel simple architecture, a 64-channel 4D-residual net consumes more memory and similar training time, although it does not lead to much improved results. Hence, with limited computing resources, it is better to first increase the number of channels, before employing a more complex architecture.

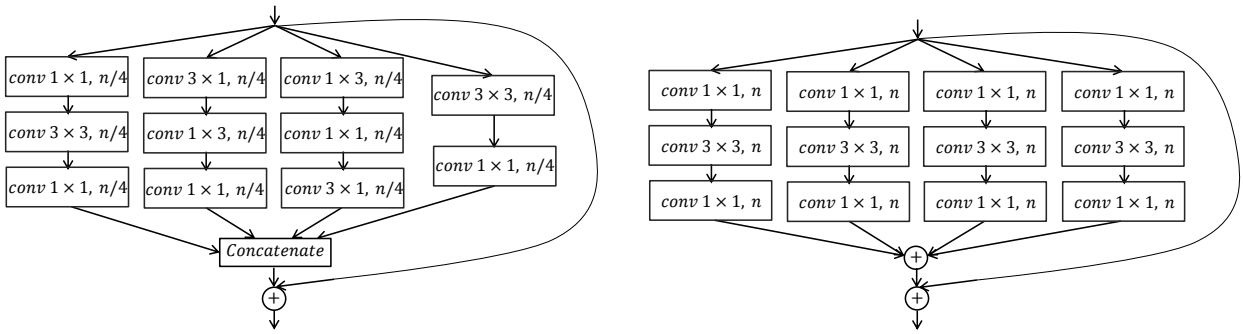
c) Further improvements on accuracy are possible with more complex architectures, such as replacing the original 5×5 convolutional layers in architectures for optical flow by two 3×3 residual inception blocks (Fig. 12(a)) or 4D-residual blocks (Fig. 12(b)). However, these will consume more memory and training time ($6 \sim 10$ times) and may require several powerful GPUs to train the models in a reasonable amount of time.

In short, improvements due to network architectures are always possible when computational resources are plenty. Their study is orthogonal to the tradeoffs between consistency and distinctiveness that define the properties of good features.

VIII. Conclusions and Future Work

This paper solves two important problems in dense matching. a) What are good features that lead to better performance? b) How can these good features be found systematically? We have developed a general model for finding good features in dense matching based on two fundamental principles: consistency and distinctiveness. Their tradeoffs lead to a multi-objective formulation, which is solved by using convolutional neural networks as feature extractors to learn new features. We have demonstrated that our learned features outperform all existing features or matching-cost computation methods using the challenging KITTI datasets.

There are several directions orthogonal to the two principles proposed that can be studied in the future. In this paper, we have focused only on two-frame matching applications (stereo matching and optical flows). Other applications, such as multi-view matching/flow or reconstruction, can lead to further improvements. Secondly, our approach is based on existing matching frameworks and does not develop new approaches specialized for the features learned. The design of such new matching schemes by ameliorating some existing work will be beneficial. Thirdly, the limitations of our method appear in



(a) Residual Inception Block (b) 4D-Residual Block

Fig. 12: Advanced architectural settings studied. Each of the original-layer sets can be replaced by one of the above blocks. Batch normalization (BN) is performed right after each convolution, followed by ReLU (except the last one that is performed after adding the residual connection). Before and after all the five blocks, there is an extra $[1 \times 1, n]$ convolutional layer. (a) Inception-Residual block similar to that in [53]. (b) More advanced 4D-residual block similar to that in [51].

TABLE XI: Performance evaluations and comparisons of different network architectures in the feature learning system

Architectural Setting	Memory (GB)	Stereo Matching				Optical Flow			
		param	speed (fps)	(f_1, f_2)	error rate	param	speed (fps)	(f_1, f_2)	error rate
fast-64	0.7	0.15M	6.5	$(0.094, 0.73)$	6.31%	0.41M	2.7	$(0.12, 0.50)$	10.89%
Res-Incep	3.2	0.14M	4.1	$(0.096, 0.76)$	6.21%	0.24M	2.3	$(0.12, 0.52)$	10.74%
4D Res-Net	9.2	0.91M	1.8	$(0.093, 0.78)$	6.09%	1.7M	0.7	$(0.12, 0.55)$	10.55%
fast-256	2.7	2.4M	0.9	**(0.091,0.79)**	**6.01%**	8.2M	0.4	**(0.11,0.55)**	**10.47%**

those reflection regions (such as windows of a car) that are prevalent in most existing work. In these regions, no useful matching information can be found. To further improve the performance in these regions, semantic segmentation information would be helpful (as done in [43], [54]). Finally, designing new CNN architectures and learning algorithms will improve the accuracy of results, leading to better features found. Using parallel computers with GPUs will further enable effective explorations of large search spaces in manageable time.

REFERENCES

[1] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2011, pp. 3017–3024.

[2] Q. Yang, "A non-local cost aggregation method for stereo matching," in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 1402–1409.

[3] M. Gong, R. Yang, L. Wang, and M. Gong, "A performance study on different cost aggregation approaches used in real-time stereo matching," *Int'l J. of Computer Vision*, vol. 75, no. 2, pp. 283–296, 2007.

[4] X. Mei, X. Sun, W. Dong, H. Wang, and X. Zhang, "Segment-tree based cost aggregation for stereo matching," in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 313–320.

[5] F. Besse, C. Rother, A. Fitzgibbon, and J. Kautz, "PMBP: Patchmatch belief propagation for correspondence field estimation," *Int'l J. of Computer Vision*, vol. 110, no. 1, pp. 2–13, 2014.

[6] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, 2008.

[7] J. Sun, N.-N. Zheng, and H.-Y. Shum, "Stereo matching using belief propagation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 7, pp. 787–800, 2003.

[8] H. Hirschmüller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2007, pp. 1–8.

[9] D. Scharstein and R. Szeliski, "Middlebury stereo vision page," *Online at http://www.middlebury.edu/stereo*, vol. 6, 2002.

[10] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int'l J. of Robotics Research*, p. 0278364913491297, 2013.

[11] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int'l J. of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[12] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Computer vision and image understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[13] G. Zhang, J. Jia, T.-T. Wong, and H. Bao, "Consistent depth maps recovery from a video sequence," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, no. 6, pp. 974–988, 2009.

[14] J. Zbontar and Y. LeCun, "Computing the stereo matching cost with a convolutional neural network," in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1592–1599.

[15] D. Gadot and L. Wolf, "Patchbatch: a batch augmented loss for optical flow," in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 4236–4245.

[16] J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *J. of Machine Learning Research*, vol. 17, pp. 1–32, 2016.

[17] H. Hirschmüller and D. Scharstein, "Evaluation of stereo matching costs on images with radiometric differences," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 31, no. 9, pp. 1582–1599, 2009.

[18] M. Bleyer, C. Rhemann, and C. Rother, "Patchmatch stereo-stereo matching with slanted support windows." in *British Machine Vision Conf. (BMVC)*, vol. 11, 2011, pp. 1–11.

[19] Y. Li, D. Min, M. S. Brown, M. N. Do, and J. Lu, "SPM-BP: Sped-up patchmatch belief propagation for continuous MRFs," in *Proc. of IEEE Int'l Conf. on Computer Vision*, 2015, pp. 4006–4014.

[20] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Trans. on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[21] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *Int'l J. of Computer Vision*, vol. 92, no. 1, pp. 1–31, 2011.

[22] D. Min, J. Lu, and M. N. Do, "Joint histogram-based cost aggregation for stereo matching," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 35, no. 10, pp. 2539–2545, 2013.

[23] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," *ACM Trans. on Graphics*, vol. 28, no. 3, p. 24, 2009.

[24] L. Xu, J. Jia, and Y. Matsushita, "Motion detail preserving optical flow

estimation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 9, pp. 1744–1757, 2012.

[25] A. Ansar, A. Castano, and L. Matthies, "Enhanced real-time stereo using bilateral filtering," in *Proc. 2nd Int'l Symp. on 3D Data Processing, Visualization and Transmission (3DPVT)*. IEEE, 2004, pp. 455–462.

[26] Y. S. Heo, K. M. Lee, and S. U. Lee, "Illumination and camera invariant stereo matching," in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2008, pp. 1–8.

[27] S. Xu, F. Zhang, X. He, X. Shen, and X. Zhang, "PM-PM: patchmatch with Potts model for object segmentation and stereo matching," *IEEE Trans. on Image Processing*, vol. 24, no. 7, pp. 2182–2196, 2015.

[28] C.-L. Hwang and A. S. M. Masud, *Multiple objective decision making-methods and applications: a state-of-the-art survey*. Springer Science & Business Media, 2012, vol. 164.

[29] K. Deb, "Multi-objective optimization," in *Search Methodologies*. Springer, 2014, pp. 403–449.

[30] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[31] M. Kim, T. Hiroyasu, M. Miki, and S. Watanabe, "SPEA2+: Improving the performance of the strength Pareto evolutionary algorithm 2," in *Parallel problem solving from nature-PPSN VIII*. Springer, 2004, pp. 742–751.

[32] M. Zeleny and J. L. Cochrane, *Multiple Criteria Decision Making*. Univ. of South Carolina Press, 1973.

[33] K. Miettinen, F. Ruiz, and A. Wierzbicki, "Introduction to multiobjective optimization: Interactive approaches," *Multiobjective Optimization*, pp. 27–57, 2008.

[34] Y. Zhao, Z. Chen, C. Zhu, Y.-P. Tan, and L. Yu, "Binocular just-noticeable-difference model for stereoscopic images," *IEEE Signal Processing Letters*, vol. 18, no. 1, pp. 19–22, 2011.

[35] A. Liu, W. Lin, M. Paul, C. Deng, and F. Zhang, "Just noticeable difference for images with decomposition model for separating edge and textured regions," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 20, no. 11, pp. 1648–1652, 2010.

[36] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[38] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[39] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[40] A. Vedaldi and B. Fulkerson, "VLFeat: An open and portable library of computer vision algorithms," in *Proc. 18th Int'l Conf. on Multimedia*. ACM, 2010, pp. 1469–1472.

[41] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd Int'l Conf. on Multimedia*. ACM, 2014, pp. 675–678.

[42] A. Seki and M. Pollefeys, "Patch based confidence prediction for dense disparity map," in *British Machine Vision Conf. (BMVC)*, 2016.

[43] F. Guney and A. Geiger, "Displets: Resolving stereo ambiguities using object knowledge," in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 4165–4175.

[44] V. Ntouskos and F. Pirri, "Confidence driven TGV fusion," *arXiv preprint arXiv:1603.09302*, 2016.

[45] W. Luo, A. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2016, pp. 5695–5703.

[46] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3061–3070.

[47] C. Bailer, K. Varanasi, and D. Stricker, "CNN based patch matching for optical flow with thresholded hinge loss," *arXiv preprint arXiv:1607.08064*, vol. abs/1607.08064, 2016.

[48] J. Hur and S. Roth, "Joint optical flow and temporally consistent semantic segmentation," in *ECCV Workshops*, 2016, pp. 163–177.

[49] J. Yang and H. Li, "Dense, accurate optical flow estimation with piecewise parametric model," in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1019–1027.

[50] T. Taniai, Y. Matsushita, and T. Naemura, "Graph cut based continuous stereo matching using locally shared labels," in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1613–1620.

[51] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," *arXiv preprint arXiv:1611.05431*, 2016.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[53] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-V4, inception-resnet and the impact of residual connections on learning," *arXiv preprint arXiv:1602.07261*, 2016.

[54] L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black, "Optical flow with semantic segmentation and localized layers," *arXiv preprint arXiv:1603.03911*, 2016.