# The Global Patch Collider

Shenlong Wang[1,2] Sean Ryan Fanello[1] Christoph Rhemann[1] Shahram Izadi[1] Pushmeet Kohli[1]

Microsoft Research[1] University of Toronto[2]

## Abstract

*This paper proposes a novel extremely efficient, fully-parallelizable, task-specific algorithm for the computation of global point-wise correspondences in images and videos. Our algorithm, the Global Patch Collider, is based on detecting unique collisions between image points using a collection of learned tree structures that act as conditional hash functions. In contrast to conventional approaches that rely on pairwise distance computation, our algorithm isolates distinctive pixel pairs that hit the same leaf during traversal through multiple learned tree structures. The split functions stored at the intermediate nodes of the trees are trained to ensure that only visually similar patches or their geometric or photometric transformed versions fall into the same leaf node. The matching process involves passing all pixel positions in the images under analysis through the tree structures. We then compute matches by isolating points that uniquely collide with each other* ie. *fell in the same empty leaf in multiple trees. Our algorithm is linear in the number of pixels but can be made constant time on a parallel computation architecture as the tree traversal for individual image points is decoupled. We demonstrate the efficacy of our method by using it to perform optical flow matching and stereo matching on some challenging benchmarks. Experimental results show that not only is our method extremely computationally efficient, but it is also able to match or outperform state of the art methods that are much more complex.*

## 1. Introduction

Correspondence estimation *ie.* the task of estimating how parts of visual signals (images or volumes) correspond to each other, is an important and challenging problem in Computer Vision. Point-wise correspondences between images or 3D volumes can be used for tasks such as camera pose estimation, multi-view stereo, structure-from-motion, co-segmentation, retrieval, and compression *etc*. Due to its wide applicability, many variants of the general correspondence estimation problem like stereo and optical flow have been extensively studied in the literature.

There are two key challenges in matching visual content across images or volumes. First, robust modelling of the photometric and geometric transformations present in real-world data, such as occlusions, large displacements, viewpoints, shading, and illumination change. Secondly, and perhaps more importantly, the hardness of performing inference in the above-mentioned model. The latter stems from the computational complexity of performing search in the large space of potential correspondences and is a major impediment in the development of real time algorithms. A popular approach to handle the problem involves detecting 'interest or salient points' in the image which are then matched based on measuring the euclidean distance between hand specified [24, 34, 20, 7] or learned [38, 35, 23, 32, 30, 6, 39, 29, 31] descriptors that are designed to be invariant to certain classes of transformations and in some cases can also work across different modalties. While these methods generate accurate matches, the computational complexity (quadratic in the number of interest points) of matching potential interest points restricts their applicability to small number of key-points.

An effective strategy to generate dense correspondences is to limit the search space of possible correspondences. For instance, in the case of optical flow by only searching for matches in the immediate vicinity of the pixel location. However, this approach fails to detect large motions/displacements. Methods like [3, 22] overcome this problem by adaptively sampling the search space and have been shown to be very effective for optical flow and disparity estimation [1, 4]. However, they rely on the implicit assumption that the correspondence field between images is smooth and fail when this assumption is violated. Techniques based on algorithms for finding approximate nearest neighbors such as KD-Tree [18, 2] and hashing [22, 9] can be used to search large-displacement correspondences and have been used for initializing optical flow algorithms [37, 1, 36, 25]. However, these approaches search for candidate matches based on the appearance similarity and they are not robust in scenarios when geometric and photometric transformations occurs (see Fig. 2).

In this paper, we address the problem of efficiently generating correspondences that can (1) have arbitrary distribution of magnitudes, (2) and that are between image elements affected by task-dependent geometric and photometric transformations. We propose a novel fully-parallelizable, learned matching algorithm called Global Patch Collider (GPC) to enable extremely efficient computation of global point-wise correspondences. GPC is based on detecting unique collisions between image points using a collection of learned tree structures that act as conditional hash functions. In contrast to conventional approaches that isolate matches by computing distances between pairs of image elements, GPC detects matches by finding which pixel pairs hit the same leaf during traversal through multiple learned tree structures.

The split functions stored at the intermediate nodes of the trees are trained to ensure that visually similar patches fall into the same terminal node. The matching process involves passing all pixel positions in the images under analysis through the tree structures. We then compute matches by isolating points that uniquely collide with each other *ie.* fell in the same empty leaf in multiple trees. We also incorporate a multi-scale top-bottom architecture, which significantly reduces the number of outliers. Content-aware motion patterns are learned for each leaf node, in order to increase the recall of the retrieved matches.

Unlike existing feature matching algorithms, the proposed global patch collider does not require any pairwise comparisons or key-point detection, thus it tackles the matching problem with linear complexity with respect to the number of pixels. Furthermore, its computational complexity can be made independent of the number of pixels by using a parallel computation architecture as the tree traversal for individual image points is decoupled.

We demonstrate the efficacy of our method by applying it on a number of challenging vision tasks, including optical flow and stereo. Not only is GPC extremely computationally efficient, but it is also able to match or outperform more complex state of the art algorithms. To summarize, our contributions are two-fold: firstly, we propose a novel learning based matching algorithm that conducts global correspondence with linear complexity; secondly, we develop a novel hashing scheme by training decision trees designed for seeking collisions.

## 2. Related Work

Our work is similar to correspondence estimation algorithms based on approximate nearest neighbor (ANN) methods, such as KD-Tree [18, 2] or hashing [22, 9]. However, there are two notable differences: (1) GPC is trained to be robust to various geometric and photometrics transformations in the training data, and (2) it isolates potential matches by looking for unique collisions in leaves of decision trees. These leaves act like conditional hash functions.

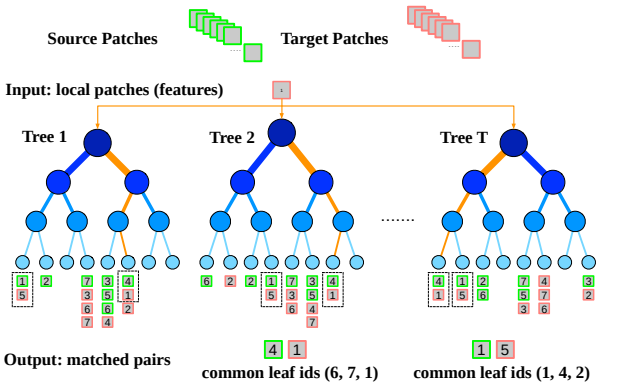


Figure 1. Global Patch Collider (GPC). Local patches traverse each tree in the decision forest, reaching different leaves. If two patches from source and target image hit the same leaf across all trees without collisions with other patches, they are considered as a distinctive correspondence. For instance, source patch 4 and target patch 1 hit the same leaves of all the trees and there is no other patch hit exactly the same leaves across all trees with them, thus it is a distinctive correspondence.

The growing availability of real and synthetic datasets for correspondence problems have led to the proposal of a number of learning based approaches. In one of the earliest works along this direction, Roth and Black [27] showed how optical flow estimates can be improved by incorporating a statistical prior on the distribution of flow in a Field of Experts model. As the size of the available datasets have grown, researchers have started to use high capacity models such as deep convolutional neural network to either learn the pair-wise similarity [29, 38] or learn the end-to-end pipeline directly [16].

The computational architecture of GPC is similar to decision forests [10]. Decision trees have been widely used in various fields of computer vision, such as pose estimation [28], image denoising [14], image classification [5], object detection [21], depth estimation [13, 15], *etc*. However, unlike all these applications, our method does not require classification or regression labels. Our objective function has been especially designed to ensure that visually similar patches (or their perspective transformed versions) will follow the same path in the trees and fall into the same leaf node.

## 3. Global Patch Collider

GPC is a matching algorithm based on finding unique collisions using decision trees as hash function evaluators. Each tree learns to map patches that are in correspondence into the same leaf while separating them from other patches (see Fig. 1). We provide the formal description of the Global Patch Collider (GPC) below.

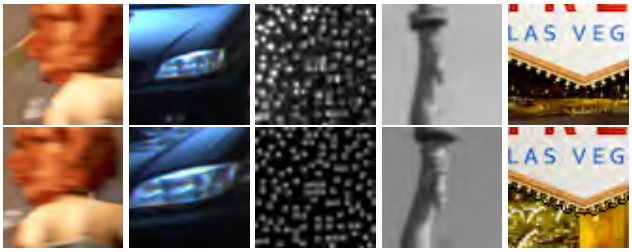


Figure 2. Examples of matched local patches. From left to right: Sintel, Kitti, active stereo, MVS, synthetic. We can see that correspondences are task dependent with different type of variations, e.g. non-rigid transform, scaling, intensity change, rotation, background change, *etc*. It is difficult to propose a generic descriptor that is robust to all kinds of variations, whereas our approach is able to learn those variations directly from the training data.

### 3.1. Formulation

**Single Tree Prediction.** Given two images $I$ and $I'$, our target is to find distinctive local correspondences between pixel positions. Given a local patch $\mathbf{x}$ with center coordinate $\mathbf{p}$ from an image $I$, we pass it into a decision tree $\mathcal{T}$ until it reaches one terminal node (leaf). The id of the leaf is just a hash key for the image patch and is denoted as $\mathcal{T}(\mathbf{x})$.

After processing all the patches, for each leaf $j$, GPC stores a set of patches from source image denoted as $\mathcal{S}_j$ as well as a set of patches from the target image, denoted as $\mathcal{S}'_j$. We will consider two patches to be a correspondence pair if and only if they fall into the same leaf and this leaf contains only one target patch and one source patch. More formally, the set of correspondences could be written as $\mathcal{C}_\mathcal{T}(I, I') = \{(\mathbf{x}, \mathbf{x}')|\mathcal{T}(\mathbf{x}) = \mathcal{T}(\mathbf{x}')$ and $|\mathcal{S}_{\mathcal{T}(\mathbf{x})}| = |\mathcal{S}'_{\mathcal{T}(\mathbf{x})}| = 1\}$.

This decision tree approach can be considered as a hashing function, where correspondent patches are picked by finding distinctive collisions between source and target image in the hash table. Simple binary hash functions can be used instead of decision trees but they would not have the conditional execution structure that decision trees have as only one split function needs to be evaluated at every layer.

**Forest Prediction.** It is worth noting that a simple tree is not discriminative enough to generate a large amount of distinctive pairs. For example, given a 16-layer binary tree, the maximum number of states is 32768, but, if we consider megapixel images, there are millions of patches from one image. Moreover, due to the content similarity, most patches within one image will fall into a small fraction of the leaves (between 6000 to 10000 on Sintel dataset). If we merely increase the depth of the tree we will bring additional computational and storage burdens for training the decision trees. This motivates us to extend the single-tree approach to a hashing forest scheme.

Specifically, instead of searching distinctive pairs that fall into the same leaf, our method seeks distinctive pairs that fall into the same leaf across all the trees in the forest. In particular, two patches are considered as a distinctive match if they reach the same leaves for all the trees and there is not any other patch from both source and target image reach exactly the same leaves. Given two images $I$ and $I'$ and a random forest $\mathcal{F}$, the set of correspondence is formulated as $\mathcal{C}_\mathcal{F}(I, I') = \{(\mathbf{x}, \mathbf{x}')|\mathcal{F}(\mathbf{x}) = \mathcal{F}(\mathbf{x}')$ and $|\mathcal{S}_{\mathcal{F}(\mathbf{x})}| = |\mathcal{S}'_{\mathcal{F}(\mathbf{x})}| = 1\}$. $\mathcal{F}(\mathbf{x})$ is a sequence of leaf nodes $\{j_1, ..., j_T\}$ where $\mathbf{x}$ falls in this forest, and $\mathcal{S}_\mathcal{L}$ represents a set of patches that fall into the ordered leaf nodes sequence $\mathcal{L}$. For a forest with $T$ trees and $L$ layers for each tree, the number of states in total is $2^{L(T-1)}$. In practice, the number of states is between 50k to 200k for a 16-layer-8-tree forest on 0.4-megapixel image from Sintel dataset [8]. Note that our method only seeks unique matched pairs, thus no re-ranking or pairwise comparison is needed.

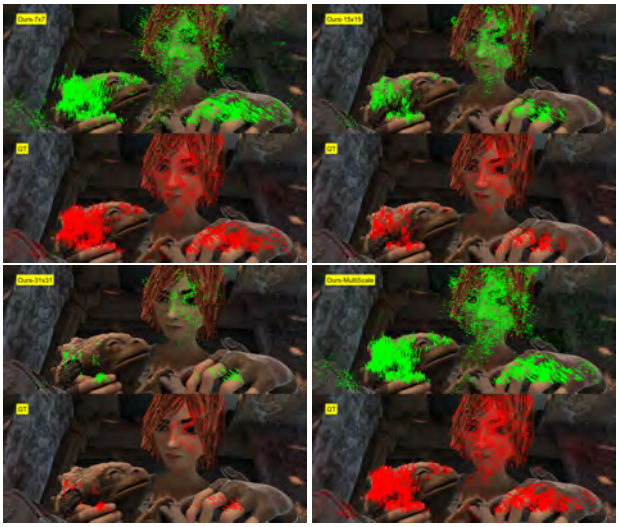

Figure 3. Sparse matching with w/o multi-scale learning. From top-left to bottom right: $7 \times 7$, $15 \times 15$, $31 \times 31$, multi-scale.

**Split Function.** Each split node contains a set of learned parameters $\boldsymbol{\theta} = (\mathbf{w}, \tau)$, where $\mathbf{w}$ is a hyper-plane and $\tau$ represents a threshold value. The split function $f$ is evaluated at a patch $\mathbf{x}$ as:

$$f(\mathbf{x}; \boldsymbol{\theta}) = \text{sign}(\mathbf{w}^T \phi(\mathbf{x}) - \tau) \quad (1)$$

where $\phi(\mathbf{x})$ is the features for $\mathbf{x}$, we will introduce our patch-based features for each task individually. This hyper-plane classifier is commonly adopted in decision forest [10]. Note the sparse hyper-planes can be used to increase efficiency, since only a small fraction of the feature is tested. Furthermore, the nature of random forest allows us to easily process patches and trees in parallel and independently for each pixel.

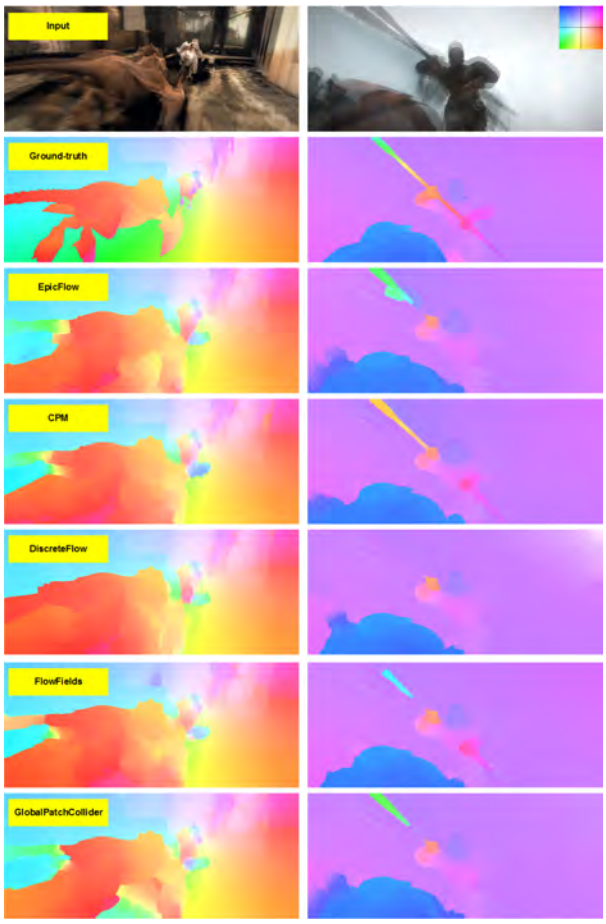


Figure 4. Qualitative comparisons among top-5 algorithms on Sintel optical flow benchmark. Top to bottom: input image (average of two), ground-truth, EpicFlow [26], CPM, FlowFields [1], GlobalPatchCollider (ours).

### 3.2. Training.

**Training Data.** Each tree in the forest is trained independently on a random subset $S$ of the training data. For our correspondence problem, the set $S$ contains triplet samples $(\mathbf{x}, \mathbf{x}_{\text{pos}}, \mathbf{x}_{\text{neg}})$ where $\mathbf{x}$ is a patch in a training source image, $\mathbf{x}_{\text{pos}}$ is the ground-truth correspondent patch in the target image and $\mathbf{x}_{\text{neg}}$ is a negative patch sampled around the ground-truth location in the target image with a random offset.

**Training Objective.** Intuitively, for each node we want to choose the optimal parameters that keep positive and reference patches into the same child node, and that split them from the negative patches. Therefore, for each internal node, we propose to choose the parameters that maximize weighted harmonic mean between precision and recall:

$$\max_{\boldsymbol{\theta}} \frac{\text{precision}(S, \boldsymbol{\theta}) \cdot \text{recall}(S, \boldsymbol{\theta})}{w_1 \text{precision}(S, \boldsymbol{\theta}) + w_2 \text{recall}(S, \boldsymbol{\theta})} \qquad (2)$$

where $w_1 + w_2 = 1$. The optimization task is equivalent to maximize precision if $w_1 = 0, w_2 = 1$ and maximize recall if $w_1 = 1, w_2 = 0$. In practice we choose a small $w_1 \in [0, 0.3]$, since we prefer high-precision matches due to the nature of correspondence problem.

The optimization is conducted in a greedy manner. We randomly sample hyper-planes and for each hyper-plane we choose the best threshold through line-search. Each node selects the hyper-plane and the threshold that maximize our objective function. To further improve the efficiency of the training we share features across nodes within the same layer and updating threshold for each node only. This technique is known in literature as Random Ferns [5].

### 3.3. Extensions.

The described method is very efficient and retrieves very accurate and sparse matches. However, some applications require a denser coverage in order to incorporate smoothness constraints within neighbor pixels. To do so we propose three possible extensions that do not introduce any additional cost in the overall running time. First, we design a multiscale version of the algorithm to increase the coverage across the image and improve the recall of the matched pairs. Secondly, hard pairs are sampled with higher probability during the training stage. Finally we learn motion prior over the patches: this gives a low compute way to disambiguate non-unique matches without performing any expensive re-ranking steps.

**Multi-scale Learning.** Many feature matching methods have difficulties in finding all reliable matches at a fixed scale. For instance, for small local patches, matches are ambiguous due to repetitive patterns or smoothing regions due to the lack of context. This motivates us to utilize information from multiple scales. However, simply stacking multiple features will dramatically increasing the dimension of the hyper-plane which brings difficulty for optimization. Therefore, we proposed a multi-layer learning scheme where the decision trees are organized in a coarse-to-fine manner. The first several layers are required to focus on features at a coarse resolution and they will look into finer resolutions as the tree goes deeper. Tab. 1 shows precision-recall of single-scale approach and multiple scales methods and Fig. 3 depicts the matching results. Compared with a single-scale approach with the same tree architecture, this multi-scale approach achieves better recall at the same level of precision.

**Mining Hard Pairs.** One of the drawback of the greedy training approach is that difficult positive pairs are discarded early once they are split into different internal nodes. In the context of optical flow and stereo, we found these samples are mostly due to large motion. Therefore, when

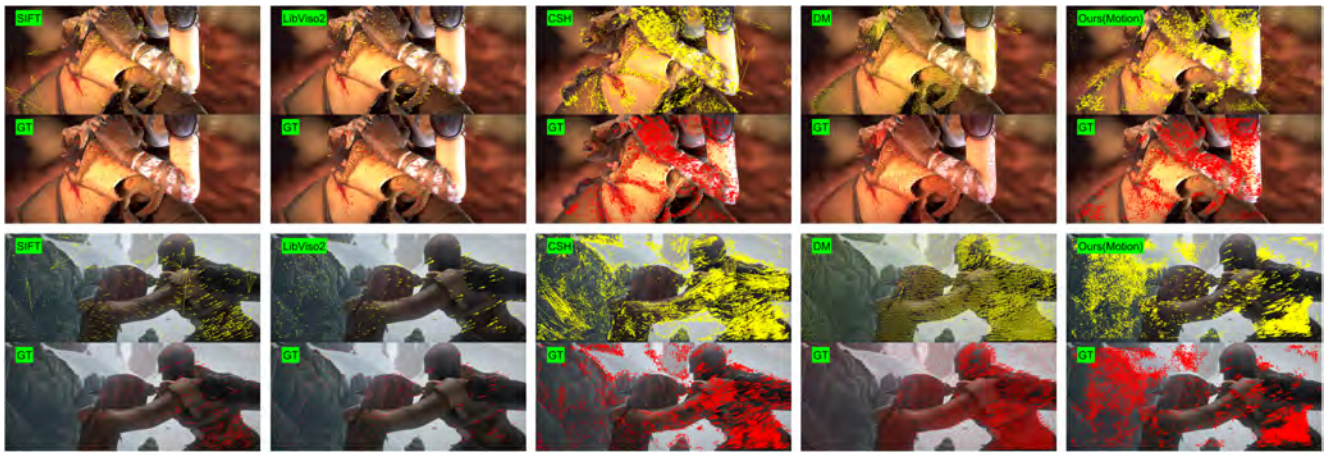


Figure 5. Qualitative comparisons for sparse matching on Sintel flow dataset (zoom-in for better quality). Left to right: Sift, LibViso, CSH, DeepMatching, Ours. Number of matches in Sift and LibViso are not dense enough. CSH and Sift generates too many outliers. DeepMatching has the best coverage but also generates some outliers (background and the arm on top, the man's head on bottom). Our method is almost outlier-free and has most matches.

| Dataset | Optical Flow Sintel (Final, Pr at $k\%$ recall) | | | | |
|---|---|---|---|---|---|
| Method | $1\%$ | $5\%$ | $10\%$ | $25\%$ | $50\%$ |
| Locality Sensitive Hashing | - | - | - | $85.2\%$ | $76.6\%$ |
| Global Patch Collider (single-tree) | - | - | - | $94.5\%$ | $89.5\%$ |
| Global Patch Collider (multi-tree) | $99.8\%$ | $99.3\%$ | $97.5$ | $93.6\%$ | $89.5\%$ |
| Global Patch Collider (+multi-scale) | $99.8\%$ | $99.5\%$ | $99.3$ | $98.1\%$ | $94.7\%$ |

Table 1. Precision at $\%k$ recall under different configurations. Our baseline is a random balanced tree with hyper-plane split function.

sampling training patches we give higher probability to large-displacement patches.

**Motion Pattern Learning.** Our method could be further extended to learn priors of motion patterns. To be specific, at each terminal node, we train a six-layer decision tree to predict whether two patches are true correspondent simply according to the relative motion. This is based on the motivation that motions are highly correlated with local content of images. For instance, boundary patches are more possible to move along the direction perpendicular to the edges than along the edge direction. Fig. 6 depicts the motion priors over different leaves. As we can see the patterns of motion diverse significantly, which justify our approach to using motion features to further boost performance. In the testing stage, we could further utilize non-distinctive patches by predicting whether two patches are likely to be a good match.

### 3.4. Complexity Analysis.

The run-time complexity of the algorithm depends linearly on the size of the image $I$. For instance, in optical flow task, the total complexity of our matching algorithm is

$$O(dTLN) + O(N) \tag{3}$$

where $N$ is the number of patches, $d$ is the number of features examined in each split function, $T$ is the number of trees, and $L$ is the layer of the each tree. To be specific, the forest prediction stage requires $O(dTLN)$ operations and matching stage requires a linear pass over all non-zero states with a maximum number of $N$. Therefore, our method considers all the possible matches globally in linear time and does not require any pairwise comparison. In practice, the parameters for our algorithm are $T = 8, L = 12, d = 27$ for optical flow and $T = 7, L = 12, d = 2$ for stereo. As comparison, KD-tree based matching will takes $O(dN\log N) + O(dN\log N) + O(dmN)$ with an additional tree building step and deep matching approximate takes $O(NN)$ operations.

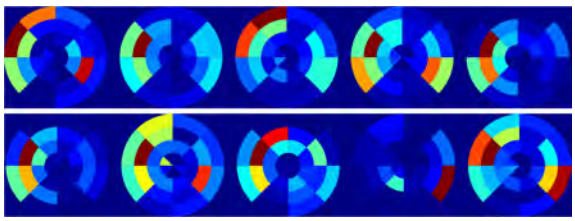

Figure 6. Motion histogram for ten randomly picked leaves. Histogram bins are divided according to motion radius (0, 1, 3, 10) and angle ($-\pi$ to $\pi$).

## 4. Experimental Results

This section presents the results for the proposed Global Patch Collider to the following tasks: (i) optical flow, (ii) structured light stereo matching and (iii) feature matching

**Algorithm 1** Global Patch Collider

**Input:** Image $I$ and $I'$ and the trained decision forest $\mathcal{F}$.
- Get all local patch features $\{\mathbf{x}\}$ and $\{\mathbf{x}'\}$ from source and target images respectively.
- Initialize $\mathcal{C}(I, I')$ with empty set.
- For each patch $\mathbf{x}$ encode and store the forest status $\mathcal{F}(\mathbf{x})$ according to Sec. 3.1.
- Enumerate all the forest status with non-zero number of hits. If there is only one source patch and target patch, add this distinctive pair $(\mathbf{x}_i, \mathbf{x}'_j)$ into the correspondence set $\mathcal{C}(I, I')$.

**Output:** $\mathcal{C}(I, I')$

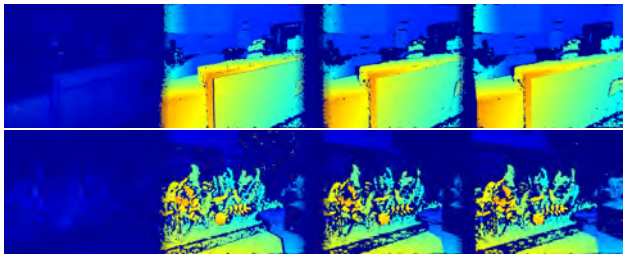

Figure 7. Qualitative comparisons for structured light stereo. Top to bottom: left input, full-iter (random initialization), 1-iter (random initialization), 1-iter (ours). Our initialization can help achieve better results within only one iteration, *e.g.* regions of the computer and the table on the right side of two images.

for widebaseline stereo. For the first problem, we perform evaluations using the popular MPI-Sintel benchmark [8] and the KITTI 2015 Optical flow dataset [17]. We compare our method with current state-of-the-art algorithms. The structured light stereo matching task is conducted over a sequence of infrared stereo images, and compared with the patch-matching stereo algorithm [4]. Finally we adopt the fountain dataset [34] to validate domain transfer ability for the proposed method.

### 4.1. Optical Flow

For optical flow experiments, we evaluate our method on the challenging MPI-Sintel dataset [8]. We first split the training dataset into training (sequence 1-12) and validation (sequence 13-22), where we evaluate the performance of the sparse matching and pick the best hyper-parameters. Reference patches are randomly sampled with higher-probability over large-motion patches (pixel larger than 10). A positive patch is chosen according to the ground-truth flow of center pixel and a negative patch is randomly sampled around ground-truth location with an offset between 3 to 20. This configuration would generate very difficult negative samples due to the local appearance similarity of images. In total we have 4-million patches for training and 1.25 million patches for validation. Three scales are selected for the multi-scale patch collider ($7 \times 7, 15 \times 15, 31 \times 31$). We use Walsh-Hadamard transform (WHT) as feature due to its efficiency and representation power[1]. For each rgb channel we pick the first 9 components, thus our feature dimension in total is 81 for multi-scale collider and 27 for single-scale collider.

| | EPE All | S0-10 | S10-40 | S40+ |
|---|---|---|---|---|
| FlowFields [1] | 5.810 | 1.157 | 3.739 | 33.890 |
| **GlobalPatchCollider** | **6.040** | **1.102** | **3.589** | **36.455** |
| CPM | 6.078 | 1.201 | 3.814 | 35.748 |
| DiscreteFlow [25] | 6.077 | 1.074 | 3.832 | 36.339 |
| EpicFlow [26] | 6.285 | 1.135 | 3.727 | 38.021 |
| TF+OFM [19] | 6.727 | 1.512 | 3.765 | 39.761 |
| Deep+R [12] | 6.769 | 1.157 | 3.837 | 41.687 |
| DeepFlow2 [36] | 6.928 | 1.182 | 3.859 | 42.854 |
| MDP-Flow2 [37] | 8.445 | 1.420 | 5.449 | 50.507 |
| LDOF [7] | 9.116 | 1.485 | 4.839 | 57.296 |
| Classic+NL [33] | 9.153 | 1.113 | 4.496 | 60.291 |

Table 2. Optical Flow Leader-board on Sintel (final) benchmark.

**Precision-Recall.** We first report precision-recall on validation triplets with multiple-configurations in Tab. 1. The balance of precision-recall could be achieved via adjusting the number of layers and the number of intersected trees. As the model becomes complex we could achieve higher precision and lower recall. We compare our method with a random balanced tree baseline with exactly same features and tree architecture but randomly generated hyper-plane. This is essentially equivalent to locality sensitive hashing method [11]. Tab. 1 shows that our learning-based approach clearly out-performs the random baseline in terms both single-tree and forest setting. Furthermore, under the same level of recall, we can see that a multi-scale learning achieves higher precision than the single-scale approach.

**Sparse Matching.** We conduct sparse matching experiments on a subset of our validation data (every 5 frame). Tab. 3 reports the results in terms of endpoint error, inlier percentage as well as number of matches per image. We consider pixel-wise motion estimation with endpoint error larger than 3 pixels as outliers. We also report our algorithm under multiple configurations, namely single-scale, multi-scale and mutli-scale plus motion learning. Several matching methods are picked as competing algorithm. Coherency sensitive hashing [22] is a hashing based PatchMatch algorithm which is designed for dense nearest-neighbor field[2]. SiftMatching [24] is a baseline for sparse matching[3]. LibViso2 [20] is a fast feature matching algorithm which is designed for sparse correspondence with applications in

[1] Since $2^n \times 2^n$ patch size is required for WHT, we extrapolate the additional row and column with padding.

[2] We use the author's implementation. In order to ensure a fair comparison for dense approaches, we only compare pixels available at 'Ours $15 \times 15$' approach when calculating endpoint errors and inlier percentage.

[3] We use the implementation in VLFeat and set PeakThres to be 0 for DoG based keypoint detection.

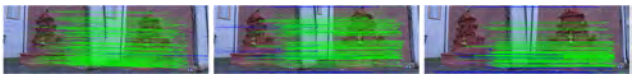

Figure 8. Domain transfer ability for wide-baseline stereo matching. Left to right: matching results across 1, 2, 3 frames respectively. Green lines are inlier and blue lines are outlier.

SLAM, optical flow and stereo[4]. DeepMatching [36] is the state-of-the-art matching algorithm specially designed for optical flow task[5]. From this table we can see that our algorithm achieves the lowest endpoint error and outliers percentage with more number of matched points on average. In terms of coverage in the most difficult case, our method outperforms other feature matching algorithms. Compared with single-scale approach, mutli-scale GPC significantly reduces the endpoint error and outlier percentage, but also decreases the number of matched points in the worst case. If we also consider non-unique hits with motion learning, the proposed method reaches the same accuracy level with multi-scale method while keeps a reasonable number of matches. Qualitative comparisons of sparse matching are shown in Fig. 5. Two failure cases of our global patch collider are shown in Fig. 10. Although motion and multi-scale learning is introduced to increase coverage, in some cases (*e.g.* in presence of motion blur and rotation) our method may fail in capturing some large transformed regions.

**Dense Flow.** Once we compute sparse matches, we use the state-of-the-art interpolation method EpicFlow [26] to generate dense flow results on the Sintel testing benchmark. Tab. 2 shows the qualitative results of top-8 optical flow methods on Sintel testing benchmark (final) as well as other three popular algorithms[6]. Please note all the top-5 methods use EpicFlow as post-processing and the original EpicFlow uses DeepMatching [36] as sparse initialization. The proposed GPC is ranked second among all the optical flow methods. In particular our proposed approach achieves the best results over pixels with motion between 10 pixels and 40 pixels. It is worth noting that our matching algorithm is the only method which does not need pairwise similarity comparison or re-ranking from multiple proposals. Fig. 4 shows qualitative comparison of all the competing algorithms over the testing dataset.

## 4.2. KITTI Optical Flow.

We evaluate our algorithm on the KITTI 2015 Optical flow dataset [17]. To be specific we follow the same configuration used in the Sintel dataset. In the training stage, we trained a GPC with 8 trees, with 12 layers. Each layer learns a specific scale from ($7\times7$, $15\times15$ and $31\times31$) with

[4]We use the author's implementation.

[5]We use the author's implementation.

[6]This is a snapshot of Sintel benchmark on Nov. 10 2015. For latest results, please refer to http://sintel.is.tue.mpg.de/.

| Method | EPE | Inlier % | Mean # | Min # | Max # |
|---|---|---|---|---|---|
| CSH | 5.7427 | 86.39% | dense | dense | dense |
| Sift Matching | 3.3814 | 92.60% | 1120 | 61 | 2393 |
| LibViso | 1.5577 | 92.42% | 848 | 45 | 1805 |
| Deep Matching | 3.0844 | 87.36% | 5945 | **2008** | 6818 |
| Ours $15\times15$ | 1.8796 | 94.69% | 21048 | 973 | 93934 |
| Ours M-Scale | **1.2809** | **97.29%** | 16813 | 27 | 88309 |
| Ours M-Scale+Motion | 1.3626 | 96.17% | **26131** | 890 | **169736** |

Table 3. Sparse matching performance on Sintel Dataset.

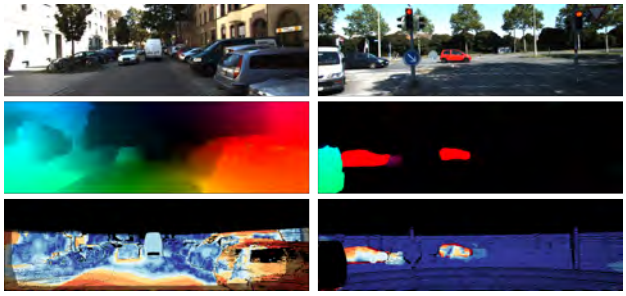

Figure 9. Results on KITTI optical flow. From top to bottom: input image, flow estimation, flow error.

| **Error** | **Fl-bg** | **Fl-fg** | **Fl-all** |
|---|---|---|---|
| All / All | 30.60 % | 33.09 % | 31.01 % |
| Noc / All | 20.09 % | 28.92 % | 21.69 % |

Table 4. Performance on KITTI flow 2015 benchmark

27 dimensional feature for each scale. In the testing stage, our sparse matching is conducted with GPC and we used EpicFlow [26] to obtain the final dense optical flow, with the standard hyper-parameters for the KITTI dataset. The average number of matches per image is 14563, whereas the minimum number of matches is 2542. Tab. 4 shows the results. In general our method is comparable with sparse matching + EpicFlow, but orders of magnitude faster in the matching stage. We also show some qualitative results in Fig. 9.

## 4.3. Structured Light Stereo Matching

For the stereo matching task, we collected 2200 infrared stereo images in indoor scene with a Kinect depth sensor based on structured illumination. The reference pattern is recovered using the calibration procedure in [15]. The pattern and Kinect images are rectified so that disparity is along horizontal line [15]. We used 1000 frames as training set and the rest as test test. The GPC patch size is set to $7\times7$. For this scenario, we use the following pixel-wise difference test as split function: $f(\mathbf{x},\theta)=\text{sign}(\mathbf{x}(i)-\mathbf{x}(j)-\tau)$. Each internal node calculates the pixel-wise intensity different at two pixel offsets $(i,j)$ and the binary decision is made whether the difference is smaller than the threshold $\tau$. This relative feature is illumination invariant and requires little computation. In the training stage, we trained a 10-tree forest with 16 layers for each tree over 1 million triplets ran-

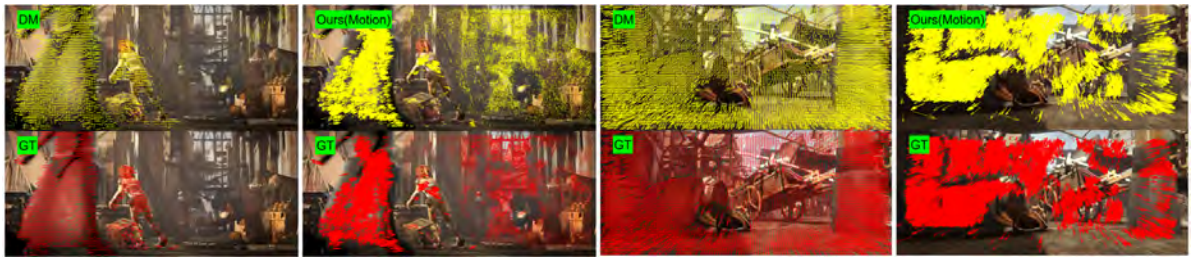


Figure 10. Failure cases of Global Patch Collider. Left: compared with deep matching, which explicitly models rotation, our method failed in capturing rotation of the basket; Right: both deep matching and our patch collider failed in capturing non-rigid deformation of the bat.

| Baseline | Sparse EPE | Outlier % | Mean # |
|---|---|---|---|
| Ours | 1.30 | 1.28% | 3754 |
| Ours (high-recall) | 1.41 | 1.79% | 8369 |
| Ours (motion) | **1.00** | **0.88%** | **9883** |

Table 5. Sparse matching performance on IR Stereo data w/o motion.

| Baseline | 1-iter | | 2-iter | |
|---|---|---|---|---|
| Random | 1.5940 | 36.81% | 1.5698 | 36.59% |
| Ours | **1.5863** | **35.26%** | **1.5642** | **34.90%** |

Table 6. PatchMatch based on dense stereo results w/o initialization with global patch collider.

domly collected from the training set. In practice we found choosing the first 7 layers will already ensure a good balance between precision and recall for this task, since the number of possible matches is greatly reduced by epipolar constraint. For each internal node, we generate 1024 random proposals and pick the best one which maximizes our objective defined in Eq. (2). Tab. 5 shows the average endpoint error and outlier percentage under different configurations. Pixels with disparity error larger than 1 are considered as outliers. In this table 'Ours' represents our standard unique-collisions based matching, 'Ours (high-recall)' represents reducing the complexity of the tree architecture (6-layer, 8-tree) in order to generate comparable number of matches before inducing 'motion' prior. Please note that in this 1-dimensional matching case, 'motion' learning is conducted simply as training a 1D-Gaussian binary classifier with disparity as input for each node. With this prior and non-unique collisions our method further increases both accuracy and recall. We also conducted dense stereo reconstruction experiment by using our sparse matching as initialization for PatchMatch based stereo [4]. In Fig. 7, we show PatchMatch results after 1-iteration with our method initialization and random initialization respectively. As we can see our method could generate more completed results and even comparable with the quality of full-iteration of PatchMatch. Quantitative comparisons are shown in Tab. 6.

### 4.4. Feature Matching for Wide-baseline Stereo

We also conduct an experiment to show the domain transfer ability of GPC. To be more specific, we trained a Global Patch Collider over Sintel dataset for optical flow task and used it for matching correspondence on EPFL wide-baseline stereo data [34]. Given the camera poses, we use our patch collider to find matches across two images then discard those violating the epipolar constraints. Errors are measured in the 3D space by projecting the two matched points back into world coordinate using the GT depth. A match pair is considered as outlier if the $\ell_2$-error is larger than 0.15m in 3D space. Fig. 8 depicts an example of matches across 1 frame, 2 frames and 3 frames respectively. Green lines are inliers and blue lines are outliers. We also consider Sift as a baseline approach and reported the quantitative results on average over all frames on 'Fountain' subset in Tab. 7. From the table and figure we can see that our method achieves better results for small baseline cases, but the performance dropped over wide-baseline case. This is expected since the GPC model is trained on the Sintel dataset, where large viewpoint changes and significant patch deformations barely happen on adjacent frames.

| Method | Sift | | | Ours | | |
|---|---|---|---|---|---|---|
| Frame Diff | 1 | 2 | 3 | 1 | 2 | 3 |
| EPE | 0.12 | 0.21 | 1.07 | 0.04 | 0.17 | 0.94 |
| Inlier% | 96% | 92% | 71% | 98% | 87% | 53% |
| Mean # | 669 | 239 | 71 | 6933 | 1143 | 200 |

Table 7. Quantitative analysis of our method's domain transfer ability on wide-baseline stereo matching (trained on Sintel).

## 5. Conclusion

This paper proposes a novel algorithm, the Global Patch Collider, for the computation of global point-wise correspondences in images. The proposed method is based on detecting unique collisions between image points using a collection of learned tree structures that act as conditional hash functions. Our algorithm is extremely efficient, fully-parallelizable, task-specific and does not require any pairwise comparison. Experiments on optical flow and stereo matching validates the performance of the proposed method. Future work includes high level applications such as hand tracking, and nonrigid reconstruction of deformable objects.

## References


[1] C. Bailer, B. Taetz, and D. Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *ICCV*. 1, 4, 6

[2] L. Bao, Q. Yang, and H. Jin. Fast edge-preserving patchmatch for large displacement optical flow. In *TIP*, 2014. 1, 2

[3] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized patchmatch correspondence algorithm. In *ECCV*. 2010. 1

[4] M. Bleyer, C. Rhemann, and C. Rother. Patchmatch stereo-stereo matching with slanted support windows. In *BMVC*, 2011. 1, 6, 8

[5] A. Bosch, A. Zisserman, and X. Munoz. Image classification using random forests and ferns. In *ICCV*, 2007. 2, 4

[6] H. Bristow, J. Valmadre, and S. Lucey. Dense semantic correspondence where every pixel is a classifier. In *ICCV*, 2015. 1

[7] T. Brox and J. Malik. Large displacement optical flow: descriptor matching in variational motion estimation. In *PAMI*, 2011. 1, 6

[8] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012. 3, 6

[9] Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu. Large displacement optical flow from nearest neighbor fields. In *CVPR*, 2013. 1, 2

[10] A. Criminisi, J. Shotton, and E. Konukoglu. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. In *Now*, 2012. 2, 3

[11] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SOCG*, 2004. 6

[12] B. Drayer and T. Brox. Combinatorial regularization of descriptor matching for optical flow estimation. In *BMVC*, 2015. 6

[13] S. R. Fanello, C. Keskin, S. Izadi, P. Kohli, D. Kim, D. Sweeney, A. Criminisi, J. Shotton, S. Kang, and T. Paek. Learning to be a depth camera for close-range human capture and interaction. In *ACM SIGGRAPH and Transaction On Graphics*, 2014. 2

[14] S. R. Fanello, C. Keskin, P. Kohli, S. Izadi, J. Shotton, A. Criminisi, U. Pattacini, and T. Paek. Filter forests for learning data-dependent convolutional kernels. In *CVPR*, 2014. 2

[15] S. R. Fanello, C. Rhemann, V. Tankovich, A. Kowdle, S. Orts Escolano, D. Kim, and S. Izadi. Hyperdepth: Learning depth from structured light without matching. In *CVPR*, 2016. 2, 7

[16] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *ICCV*, 2015. 2

[17] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 6, 7

[18] K. He and J. Sun. Computing nearest-neighbor fields via propagation-assisted kd-trees. In *CVPR*, 2012. 1, 2

[19] R. Kennedy and C. J. Taylor. Optical flow with geometric occlusion estimation and fusion of multiple frames. In *EMMCVPR*, 2015. 6

[20] B. Kitt, A. Geiger, and H. Lategahn. Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *IV*, 2010. 1, 6

[21] P. Kontschieder, S. R. Bulò, A. Criminisi, P. Kohli, M. Pelillo, and H. Bischof. Context-sensitive decision forests for object detection. In *NIPS*, 2012. 2

[22] S. Korman and S. Avidan. Coherency sensitive hashing. In *ICCV*, 2011. 1, 2, 6

[23] L. Ladickỳ, C. Häne, and M. Pollefeys. Learning the matching function. In *arXiv preprint arXiv:1502.00652*, 2015. 1

[24] D. G. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999. 1, 6

[25] M. Menze, C. Heipke, and A. Geiger. Discrete optimization for optical flow. In *GCPR*, 2015. 1, 6

[26] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, 2015. 4, 6, 7

[27] S. Roth and M. J. Black. On the spatial statistics of optical flow. In *IJCV*, 2007. 2

[28] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, 2011. 2

[29] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. M. Noguer. Discriminative learning of deep convolutional feature point descriptors. In *ICCV*, 2015. 1, 2

[30] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. In *PAMI*, 2014. 1

[31] S. Singh, A. Gupta, and A. Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV*, 2012. 1

[32] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua. Ldahash: Improved matching with smaller descriptors. In *PAMI*, 2012. 1

[33] D. Sun, S. Roth, and M. J. Black. Secrets of optical flow estimation and their principles. In *CVPR*, 2010. 6

[34] E. Tola, V. Lepetit, and P. Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. In *PAMI*, 2010. 1, 6, 8

[35] T. Trzcinski, M. Christoudias, P. Fua, and V. Lepetit. Boosting binary keypoint descriptors. In *CVPR*, 2013. 1

[36] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *ICCV*, 2013. 1, 6, 7

[37] L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. In *PAMI*, 2012. 1, 6

[38] S. Zagoruyko and N. Komodakis. Learning to compare image patches via convolutional neural networks. In *CVPR*, 2015. 1, 2

[39] J. Žbontar and Y. LeCun. Computing the stereo matching cost with a convolutional neural network. In *CVPR*, 2015. 1