SubPatch: Random kd-tree on a Sub-sampled Patch Set for Nearest Neighbor Field Estimation

Fabrizio Pedersoli [a], Sergio Benini [a], Nicola Adami [a], Masahiro Okuda [b], Riccardo Leonardi [a]
[a] Dept. of Information Engineering, University of Brescia, Italy
[b] Faculty of Environmental Engineering, University of Kitakyushu, Japan

ABSTRACT

We propose a new method to compute the approximate nearest-neighbors field (ANNF) between image pairs using random kd-tree and patch set sub-sampling. By exploiting image coherence we demonstrate that it is possible to reduce the number of patches on which we compute the ANNF, while maintaining high overall accuracy on the final result. Information on missing patches is then recovered by interpolation and propagation of good matches. The introduction of the sub-sampling factor on patch sets also allows for setting the desired trade off between accuracy and speed, providing a flexibility that lacks in state-of-the-art methods. Tests conducted on a public database prove that our algorithm achieves superior performance with respect to PatchMatch (PM) and Coherence Sensitivity Hashing (CSH) algorithms in a comparable computational time.

Keywords: Dense correspondence, kd-tree, image reconstruction.

1. INTRODUCTION

Computing the *nearest-neighbor field* (NNF) to find dense correspondences between image patches (i.e., for every patch in a target image, find the closest patch in the source image by minimizing a distance *d*) is becoming a key aspect in many computer vision and graphic applications, such as image retargeting, completion, and reshuffling. However, the computation of an optimal NNF using brute force approach becomes unfeasible as the image size increases, since for an image with $M \times N$ pixels and patch size w, the complexity becomes $O(w^2(MN)^2)$. For this reason research efforts have been recently directed towards the development of effective techniques to solve the *approximate nearest-neighbor field* (ANNF). In this case, the match found for each query patch is not guaranteed to be the optimal one, but it is found much faster than with exact NNF methods.

1.1 Previous work

One of the most notable works concerning ANNF on image regions is the PM algorithm [1], together with its generalized version [2]. The principal idea is that good matches for patches can be found by random sampling at a first stage, and then good matches can be iteratively propagated to neighbors considering the natural coherence of the image. The iterative process, which is repeated until convergence, consists of two phases. The first is *propagation*, which consider the above and left neighbor mappings to check if they improve the current one. The second is *random search*, in which the current match is perturbed in concentric neighborhoods by multiple scales of random offsets, in order to avoid to end up in local minima.

Other techniques to compute ANNF rely on locality sensitive hashing (LSH) [3]. This method limits the number of candidates for each query by hashing objects to ensure that the probability of collision is higher for those that are close to each other. Coherency sensitive hashing (CSH) in [4] relies on hashing to seed the initial matching of query patches, and on image coherence to propagate good matches. Specifically, information is propagated to nearby patches in the image as in [1], and also to similar patches that were hashed to the same bin by Walsh-Hadamard (WH) hashing functions.

In the past kd-trees have been employed to solve the problem of exact NNF [5, 6], since when dealing with low dimensional data, feasible solutions could be found. In case of increased data dimensionality, variants of kd-trees have been proposed to solve the associated ANNF problems. For example, the method in [7] creates multiple kd-trees whose splitting directions are randomly chosen and simultaneously searched. The work in [8] presents the priority search k-means tree that clusters the data points using the full distance across all dimensions, in contrast to the randomized kd-tree algorithm which only partitions the data based on one dimension at a time. Recently the algorithm in [9] which proposes

Seventh International Conference on Machine Vision (ICMV 2014), edited by Antanas Verikas, Branislav Vuksanovic, Petia Radeva, Jianhong Zhou, Proc. of SPIE Vol. 9445, 944514
 · doi: 10.1117/12.2180577

a kd-tree variant to find *(1+ε)* approximate nearest neighbors, has been employed in [10] in order to estimate an ANNF based on nine features from each image patch. After a colour adaptation between image pairs, the mean and the max values of R, G and B channels, the mean value of the *x* and *y* gradients, and the projection on two WH bases are chosen as the dimensions exploited by a kd-tree. The obtained results are further improved by a propagation step similar to PM, and a random affine transform applied to the source image; the whole procedure is then iterated until convergence.

1.2 Paper aims

All above mentioned solutions to the ANNF problem present pros and cons. PM suffers from possibly ending up in bad correspondences, since it initially considers only some random portions of the image. Conversely kd-trees are potentially able to find matches in any image location. This means that the PM algorithm tends to be usually faster, while tree-based methods are generally slower and more accurate. Hashing methods are accurate too, but in order to be fast they are bound to rely on features that can be computed in an efficient way, using for example the gray-code kernel recursive technique [11].

In this paper we combine the advantages of both PM and kd-tree approaches, with the goal of achieving superior ANNF accuracy in a comparable computational time. In order to accomplish this goal, we introduce a fast estimation by randomized kd-trees on a sub-sampled version of the complete patch set to estimate the initial ANNF. Then missing patches are reconstructed by interpolation followed by a propagation phase, by using locality considerations similar to the PM. Performed tests demonstrate the superiority of the approach, where the sub-sampling factor acts as a control parameter in the tradeoff between accuracy and speed. The reminder of this paper is organized as follows: Section 2 provides the overall methodology, while Section 3 describes the tests which have been carried out to assess performance and potentialities of the proposed approach. Concluding remarks are gathered in Section 4.

2. SUBPATCH ALGORITHM

Natural images present spatial correlation that is particularly high within the range of patch width. For this reason we can compute an ANNF on regularly spaced patches instead of on the whole dense set. The main idea of the SubPatch algorithm is therefore to find accurate matches on a subset of patches by using random kd-tree, compute the remaining ones by interpolation, and improve matches by propagation of the good ones as in PM algorithm. The proposed solution allows to obtain superior performance with respect to state-of-the-art solutions, increasing flexibility allowing to select the desired tradeoff between accuracy and speed.

2.1 Workflow

Source and target images I_S and I_T act as inputs of the SubPatch algorithm. The whole process consists of four steps, as illustrated in Figure 1: *patchization*, *nearest-neighbour initialization*, *interpolation*, and *propagation*. Other important parameters are the down-sampling factor *s*, the patch width *w*, and the number of iteration N_{iter}. The output is the approximated nearest-neighbor field (ANNF), which technically consists of a 3D matrix that in dimension 1 and 2 stores the *i* and *j* positions of the nearest neighbor patch, respectively, while dimension 3 contains the distance *d* between corresponding patches.

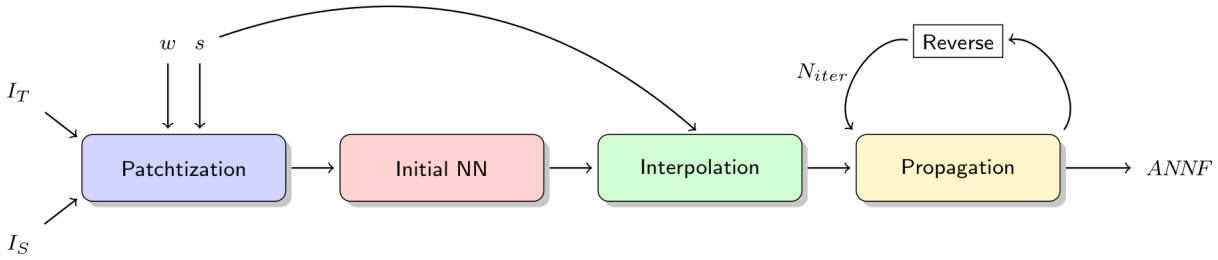


Figure 1: The general workflow of the SubPatch algorithm.

2.2 Patch down-sampling

Let us define a squared patch of width w from an image $I \in \mathbf{N}^{M \times N}$ having the top-left corner in position (i,j) as:

$$p_{i,j} = \left[I_{i,j}, I_{i,j+1}, , I_{i,j+w-1}, I_{i+1,j+1}, , I_{i+1,j+w}, , I_{i+w-1,j}, I_{i+w-1,j+1}, , I_{i+w-1,j+w-1} \right]^T \quad (1)$$

where $i \in [0, 1, \ldots, M-w-1]$ and $j \in [0, 1, \ldots, N-w-1]$. Previous approaches always considered a dense set of patches, i.e. patches that are spatially separated by one pixel only. We then reduce the cardinality of patches in both images, by applying a down-sampling factor s on both rows and columns. Therefore the down-sampled set of patches $P_{h,k}$ is a matrix where each element corresponds to a patch, such as:

$$P_{h,k} = \begin{pmatrix} p_{0,0} & p_{0,s} & \cdots & p_{0,N^{(s)}} \\ p_{s,0} & p_{s,s} & \cdots & p_{s,N^{(s)}} \\ \vdots & & \ddots & \\ P_{M^{(s)},0} & P_{M^{(s)},s} & \cdots & P_{M^{(s)},N^{(s)}} \end{pmatrix} \quad (2)$$

where $N^{(s)} = \lfloor (N-w-1)/s \rfloor$ and $M^{(s)} = \lfloor (M-w-1)/s \rfloor$. As a consequence, the initial ANNF will be s^2 smaller than a dense one.

2.3 Initial ANNF estimation

The initial ANN is computed by searching for each element of the patch subset of the target image I_T the approximated nearest neighbor among the patch subset of I_S by using random kd-tree [7]. In order to build the forest, each patch of width w is represented as a $3w^2$ dimensional feature vector which accounts for all RGB pixel values.

Random kd-tree solution has been chosen since it is very efficient in finding the nearest-neighbors of high dimensional features vectors. This algorithm builds multiple kd-trees whose splitting directions are chosen randomly within the top five dimensions with highest variance. The nearest neighbor of a query is found simultaneously searching in all the trees according to a priority queue. This queue orders the leaves to be visited starting from the branch that has less distance from the decision boundary from all trees. The key aspect of random kd-tree is that having multiple kd-trees, the probability that in one of them the query patch and the nearest neighbor one are in the same cell increases. For this reason we limit the number of leaves to be visited, thus obtaining a consistent speed improvement.

The estimated initial ANNF is accurate enough to ensure the possibility of further propagation of good matches between neighbors. Since each patch is potentially matched with all others, the kd-tree method does not suffer of the problem of ending up with bad matches due to limited image "visibility". In addition it avoids the inconvenient of working with low dimensional feature vectors that reduces patch information and often leads to inaccurate nearest neighbors. Here we compute the initial ANNF using the random kd-tree implementation provided by FLANN[1] library.

2.4 Interpolation and good matches propagation

The initial estimation of ANNF is up-sampled by a factor s on both dimensions. To retrieve the original resolution ANNF we apply two subsequent steps: an initial interpolation followed by a propagation of good matches. The missing values due to patch sub-sampling are recovered by means of zero-order hold interpolation (ZOH filter).

Thanks to the coherence of natural images, the interpolated ANNF is further improved by propagating the good matches through an iterative procedure that scans the image in a row-major order. The scan order is toggled at each iteration from top-left to bottom-right. To determine the nearest-neighbor of a given query patch, two additional

[1] http://www.cs.ubc.ca/research/flann/

candidate patches are considered. These are located in correspondence of the two best matches in I_S of the spatial neighbors of the query patch in I_T, as shown in Figure 2. The best match is found among the candidates, minimizing the following distance:

$$argmin_{h,k} \left\| p_{i,j}(I_S) - p_{h,k}(I_T) \right\|_2^2 \quad (h,k) = \left\{ ANNF(i-1,j) + (1,0), ANNF(i,j-1) + (0,1) \right\} \tag{3}$$

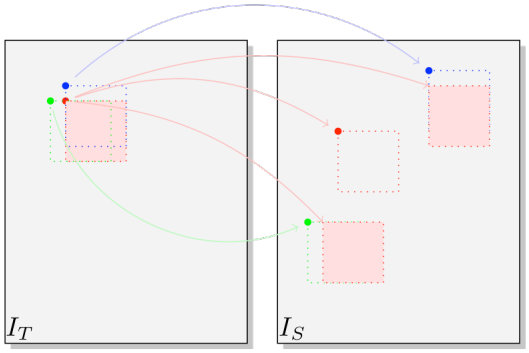


Figure 2: Propagation of good matches. The best match for a given query patch (in red) in I_T is computed evaluating the distances of two additional candidate patches in I_S and choosing the closest one. Additional candidates are found in correspondence of the matching positions of the top and left neighbors of the given query patch.

3. EXPERIMENTS

We measure the performance of our method by evaluating the RMSE and SSIM of image reconstruction using a public dataset[2] of 133 full HD image pairs, which are taken 30 frames apart from famous movies, and that has been used in [1] and [10]. The RMSE is computed as the sum of the root mean square error of reconstruction of the three image channels. SSIM [12] instead is the structural similarity measure and assess the perceptual quality of reconstruction.

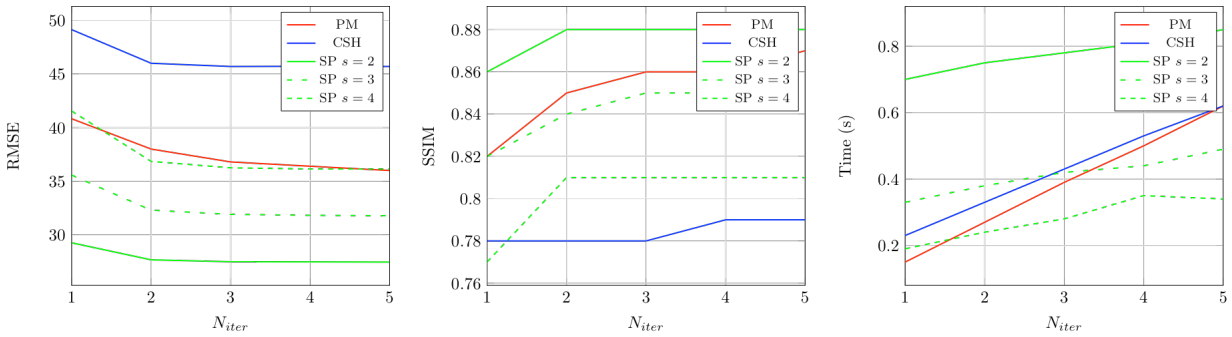
We have chosen to compare our results only with algorithms whose original authors implementations are freely available, PM [1] and CSH [4]. In order to stimulate research and ensure reproducibility of results, SubPatch source code has been made available too on our public github repository[3].

Evaluation and comparison consider the following three aspects: mean RMSE, SSIM, and computational time. Using patch of width $w=8$, we create 4 kd-trees using a maximum number of checks equal to 8. Results are reported in Figure 3, where we assess performance by varying the number of iterations N_{iter} and sub-sampling factor s.

As shown in Figure 3, SubPatch in general outperforms both algorithms in term of RMSE and SSIM. As an additional advantage, varying the down-sampling factor, we can exchange accuracy with speed. For instance with a number of iteration greater than 2 using $s=3$ we can be slightly better than PM in term of performance and be slight faster, up to half time faster with $N_{iter}=5$. It can also be noticed that in general we converge faster than PM. With $N_{iter}=2,3,4,5$ the values of RMSE change only about 1% while PM does 5%. That is because our initial estimation of the ANNF is already pretty robust after the first iteration. Conversely PM, performing an initial random guess, needs more iterations to improve the ANNF. CSH instead is in average 17% worse than PM in term of accuracy with a comparable execution time.

2 http://www.eng.tau.ac.il/~simonk/CSH/VidPairs_DataSet/VidPairs_DataSet.zip

3 https://github.com/fpeder/SubPatch



(a) RMSE (b) SSIM (c) time

Figure 3. Comparison in terms of average RMSE, SSIM, and processing time per image.

4. CONCLUSION

We developed an algorithm for estimating the ANNF between two images that outperforms PM and CSH. The proposed solution computes the ANNF at different scales combining random kd-tree, interpolation and propagation. The improvements over PM are mainly provided by the use of a more reliable initial estimation, while the use of a multi-resolution architecture allows for quick computations of ANNF with respect to both tested competitors. Moreover, the down-sampling parameter is used to trade off accuracy and speed. This leads to interesting scenarios and gives the flexibility that lacks in other methods. In future work we will investigate how our method ca be used to effectively encode images in a scalable fashion and privacy preserving modality.

5. REFERENCES

[1] Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D., "Patchmatch: A randomized correspondence algorithm for structural image editing," ACM Transactions on Graphics, 24 (2009).

[2] Barnes, C., Shechtman, E., Goldman, D. and Finkelstein, A., "The generalized patchmatch correspondence algorithm," in proceedings of ECCV 2010, 2943 (2010).

[3] Gionis, A., Indyk, P., and Motwani, R., "Similarity search in high dimensions via hashing," in proceedings of VLDB 1999, 518529 (1999).

[4] Korman, S. and Avidan, S., "Coherency sensitive hashing." in proceedings of ICCV 2011, 16071614 (2011).

[5] Bentley, J., "Multidimensional binary search trees used for associative searching," Communications of the ACM, 509517 (1975).

[6] Friedman, J., Bentley, J. and Finkel, R. "An algorithm for finding best matches in logarithmic expected time," ACM Transactions on Mathematical Software, 209226 (1977).

[7] Silpa-Anan, C. and Hartley, R., "Optimised KD-trees for fast image descriptor matching," in proceedings of CVPR 2008, 18 (2008).

[8] Muja, M. and Lowe, D., "Scalable nearest neighbor algorithms for high dimensional data," IEEE Transactions on Pattern Analysis and Machine Intelligence, (2014).

[9] Arya, S. Mount, D., Netanyahu, N., Silverman, R., and Wu, A., "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," journal of the ACM, 891923 (1998).

[10] Ramakanth, S. and Babu, R., "FeatureMatch: A general ANNF estimation technique and its applications," IEEE Transactions on Image Processing, 2193 (2014).

[11] Ben-Artzi, G., Hel-Or, H. and Hel-Or, Y., "The gray-code filter kernels," IEEE Transactions on Pattern Analysis and Machine Intelligence, 382 (2007).

[12] Wang, Z., Bovik, A., Sheik, H., and Simoncelli, E., "Image quality assessment: from error visibility to structural similarity," IEEE Transactions on Image Processing, 600612 (2004).