NONLINEAR WARPING FUNCTION RECOVERY BY SCAN-LINE SEARCH USING DYNAMIC PROGRAMMING

Fatih Porikli

Mitsubishi Electric Research Laboratories
Cambridge, MA 02139, USA

ABSTRACT

We present a novel solution to the warping recovery problem. Our algorithm has several distinct advantages; it is scalable, it enables effective integration of boundary and continuity constraints, and most importantly it is computationally much less demanding than the previous approaches. In addition, our algorithm accurately detects non-linear warping functions without being restricted to the linearity assumptions and 2-D planar deformations unlike the existing methods. We achieve to formulate the image warping as an optimization process in 1-D scan-line search spaces. We construct the search spaces from block-matching based image distances, and then we traverse minimum cost paths into these search spaces using boundary conditions to determine the horizontal and vertical components of warping for each pixel. Our experiments prove the performance of the proposed algorithm.

1. INTRODUCTION

Discovering the warping function between two images is very important for several applications. For instance, the next generation projector systems require the automatic recovery of the display surface. In such a scenario, a known pattern may be projected onto an unknown surface and the image on the surfaces may be compared to find the warping, which manifest the shape of the surface. Another application is the recovery of optical distortion in outdoors camera setups. The distortion due to the splattered rain drops and disfigured protective glasses in front of the lenses can be solved by finding the warping before and after the distortion.

Although there are previous attempts to address recovery of warping, some methods are limited only for linear distortions [5]. Besides, existing warping recovery algorithms are computationally very demanding [1].

Here, we propose an algorithm to find the nonlinear warping between two images. Unlike the existing approaches, our algorithm is not restricted to the linear warping scenarios. We achieve effective integration of the continuity and boundary conditions. Our algorithm is also scalable without requiring subsampled versions of the input images. Most importantly, our algorithm is computationally much simpler than the two-dimensional dynamic programming approach [4] since we take advantage of one-dimensional scan line structure.

2. SCAN-LINE SEARCH

Scan-line is a 1-D line on the original image. Our objective is to find its correspondence (which may be a curve) in the warped image.

Let an input image and its warping be $I(x,y)$ and $I_w(x,y)$ where $x = 1, ..., M$ and $y = 1, ..., N$. We define a warping function $w(x,y)$ that transfer the intensity values of the original image to the warped image as

$$I_w(x,y) = I(x + w_x(x,y), y + w_y(x,y)) \tag{1}$$

where w_x and w_y represent the horizontal and vertical components of the pixel relocation, respectively. Thus, the warping function is a two-dimensional, real valued mapping of the pixel coordinates. The boundary conditions of the warping function are given as

$$\begin{aligned} w_x(x,y) = 0 \quad &: \quad x = 1, x = M \\ w_y(x,y) = 0 \quad &: \quad y = 1, y = N \\ w_x(x,y) \geq \max(0, x_{max}) \quad &: \quad x < M/2 \\ w_y(x,y) \geq \max(0, y_{max}) \quad &: \quad y < N/2 \\ w_x(x,y) \leq \max(M - x, x_{min}) \quad &: \quad x > M/2 \\ w_y(x,y) \leq \max(N - y, y_{min}) \quad &: \quad y > N/2 \end{aligned} \tag{2}$$

where $[x_{min}, x_{max}]$ and $[y_{min}, y_{max}]$ are the range of horizontal and vertical warping, so called as warping window. These conditions ensure that the warped image coordinates are always within the boundary of the original image.

Principally, the warping function should minimize the aggregated error between the original and projected image for all pixels:

$$\arg\min \sum_{x}^{M} \sum_{y}^{N} |I(x,y) - I_w(x - w_x(x,y), y - w_y(x,y))| \\ x_{min} \leq w_x \leq x_{max}, y_{min} \leq w_y \leq y_{max} \tag{3}$$

It is apparent that the above 2-D minimization problem (even with the continuity constraints) is computationally very demanding [4].

Instead of doing the minimization in the 2-D image plane, we break it down into 1-D horizontal and vertical scan-line processes. We define a scan as $s_y(x), x = 1, ..., M$ (or similarly $s_x(y), y = 1, ..., N$) that corresponds to a row (or column) within the original image.

For a pixel on a scan-line $s_y(x)$, we compute the distances of all possible matches within the warped image within the warping window. One major difference of our algorithm from the previous approaches is that instead of depending only the point comparisons we extend the distance measure to the absolute block differences, i.e. to block-matching. The distance between the original image and warped image blocks is defined as

$$d(I(x,y), I_w(x,y)) = \sum_{m=-\delta}^{\delta} \sum_{n=-\delta}^{-\delta} |I(x + m, y + n) \\ - I_w(x + m, y + n)| \tag{4}$$

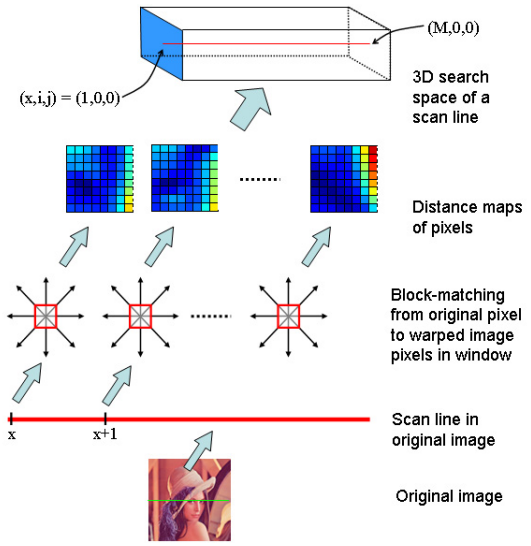

Fig. 1. Scan-line search space is constructed from the combined intensity distances of pixels within a search window centered on them. For each scan-line, a search space is generated.

where δ is the block radius. By computing all the distances within the warping window, we obtain a $[x_{max} - x_{min}, y_{max} - y_{min}]$ non-negative matrix for each pixel on the scan-line as illustrated in Fig. 1. Then, a scan-line search space $S_y(x, i, j)$ is constructed by reindexing the distance matrices

$$S_y(x, i, j) = d(I(x, y), I_w(x + i, y + j)) \tag{5}$$

where $(x_{min} \leq i \leq x_{max}, y_{min} \leq j \leq y_{max}]$. Note that, the $i = 0, j = 0$ axis in this space corresponds to zero-warping, i.e. no change in the pixel locations. The above boundary conditions impose that $S_y(1, 0, 0) = S_y(M, 0, 0) = 0$. One distinct advantage of using a search space is that the integration of continuity and other constraints becomes significantly uncomplicated. We also show that we can easily change the resolution of the warping function.

Now the question becomes how to find the amount of relocation between the original and warped images. It is obvious that the search space captures all the possible warpings of pixels along the scan-line. It is apparent that if there is no continuity requirement, the problem converts itself to an ordinary block-matching, which finds the minimum at each distance matrix for every pixel.

The continuity constraint asserts that the ordering of the pixels before and after the warping should be same. In other words, if a pixel at x is mapped to x_* then the consecutive pixel along the scan-line at $x + 1$ can be mapped to pixels at the same (x) or following pixels $(> x_*)$ but not the $x_* - 1$ or any pixel before it. Otherwise, the ordering of pixels will reverse, and a crossover will happen.

This implies that the warping at $x + 1$ can only be greater than -1 (on either vertical or horizontal directions), i.e. $w_x(x+1, y) \geq w_x(x, y) - 1$ and $w_y(x + 1, y) \geq w_y(x, y) - 1$ on the scan-line direction within the search space.

We will use this property to find a nonlinear warping function in the next section.

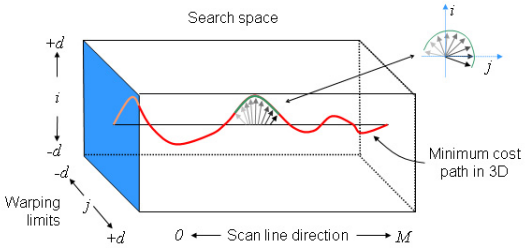

Fig. 2. A minimum cost path within the search space is obtained by dynamic programming. Vertical and horizontal components of the path give the corresponding warping amounts.

3. DETERMINATION OF MINIMUM COST PATH

After we construct the scan-line search space, we find a path that connects the first $(x = 0)$ and last $(x = M)$ pixels such that the total distance on the traversed path is the minimum among all possible paths and the transition between the warping from one pixel to next is bounded by the above property. Since our objective is to find a single minimum cost path traversing from $(1, 0, 0)$ to $(M, 0, 0)$ within S_y by making controlled jumps, we adapt a dynamic programming technique [3].

Dynamic programming is an approach developed to solve sequential, or multi-stage, decision problems [2]. Basically, what dynamic programming approach does is that it solves a multivariable problem by solving a series of single variable problems. The essence of dynamic programming is Richard Bellman's Principle of Optimality. This principle is intuitive: from any point on an optimal trajectory, the remaining trajectory is optimal for the corresponding problem initiated at that point.

Let v be a vertex and e be an edge between the vertices of a directed weighted graph. We associate a cost to each edge $c(e)$. We want to find the minimum cost path by moving from an origin vertex v_1 to a destination vertex v_M. The cost of a path $p(v_1, v_M) = \{v_1, .., v_M\}$ is the sum of its constituent edges

$$C(p(v_1, v_M)) = \sum_{x}^{M} c(v_x) \tag{6}$$

Suppose we know the costs $C(v_1, v_*)$ from v_1 to every other vertex. Let's say v_* is the last vertex the path goes through before v_M. Then, the overall path must be formed by concatenating a path from v_1 to v_*, i.e. $p(v_1, v_*)$, with the edge $e(v_*, v_M)$. Further, the path $p(v_1, v_*)$ must itself be a minimum cost path since otherwise concatenating the minimum cost path with edge $e(v_*, v_M)$ would decrease the cost of the overall path. Notice that $C(v_1, v_*)$ must be equal or less than $C(v_1, v_S)$, since $C(v_1, v_M) = C(v_1, v_*) + c(v_*, v_M)$ and we are assuming all edges have non-negative costs, i.e. $c(v_*, v_M) \geq 0$. Therefore if we only know the correct value of $C(v_1, v_*)$ we can find a minimum cost path.

We modified Dijkstra's algorithm for this purpose. Let Q be the set of active vertices whose minimum cost paths from v_1 have already been determined, and $\vec{p}(v)$ is a back pointer vector that shows the neighboring minimum cost vertex of v. Then the iterative procedure is given as

1. Set $u_1 = v_1$ $Q = \{u_1\}$, $C(u_1) = 0$, $\vec{p}(v_1) = v_1$, and $c(v) = \infty$ for $v \neq u_1$.

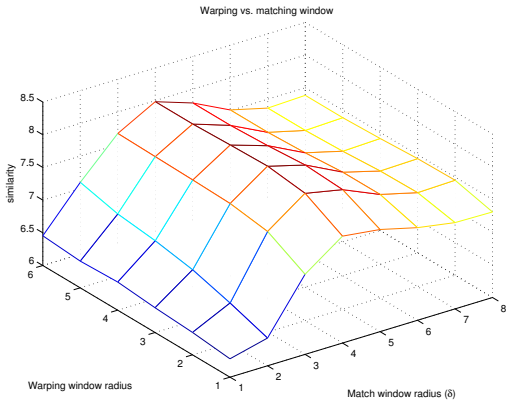

Fig. 3. Size of the matching block is optimum for $\delta = 4$ for most cases. Note that, smaller matching blocks and warping windows increase error.

2. Find u_i that has the minimum cost $c(u_i)$.
3. For each $u_i \in Q$: if v is a connected to u_i, assign $c(v) \leftarrow \min\{c(u_i), C(u_i)+c(v)\}$. If $c(v)$ is changed, assign $\vec{p}(v) = u_i$ and update $Q \leftarrow Q \cup v$.
4. Remove u_i from Q. If $Q \neq \emptyset$ go to step 2.

Then the minimum cost path $p(v_1, v_M) = \{v_1, ..., v_M\}$ is obtained by tracing back pointers by starting from the destination vertex v_M as $v_{M-1} = \vec{p}(v_M)$. The algorithm takes time $O(M^2)$.

We converted the search space to directed weighted graph as follows. Each point (x, i, j) in S_y corresponds to a vertex, $v_{x,i,j} = (x, i, j)$. The edges, connects a point (x, i, j) to other points in the next distance matrix. Due to the continuity constraint, the directional edges are only limited to the vertices in 8^{th}-neighborhood. Thus, 9 possible edges, including the zero-warp case, connects $v_{x,i,j}$ to $v_{x+1,i,j}$, $v_{x+1,i+1,j}$, $v_{x+1,i-1,j}$, $v_{x+1,i,j+1}$, $v_{x+1,i,j-1}$, $v_{x+1,i-1,j+1}$, $v_{x+1,i+1,j-1}$, $v_{x+1,i-1,j-1}$, $v_{x+1,i+1,j+1}$.

The minimum cost path gives the 2-D warping of a 1-D line in the original image. To find the correspondences of the other lines in the original image, we process the consecutive scan-lines accordingly. Fig. 2 depicts the relation between the space and path. Thus, after we scan all the horizontal lines, we obtain a warping function. We repeat this previous process for the vertical scan-lines and assign the pixel-wise mean of both functions as the final result. In summary, the recovery of warping function consists of these stages:

1. Construct scan-line search space for y, start from $S(1, 0, 0)$
2. For (x, i, j), compute costs in from the for 8^{th} neighborhood of $(x + 1, i, j)$
3. When accumulation reaches $(M, 0, 0)$, start back-tracing to determine minimum cost path (warping function is equal to path coordinates)
4. Increase y, repeat from 1
5. Switch the scanning direction from horizontal to vertical (S_y to S_x), repeat above.

Another advantage of the proposed method is that it is possible to scale the resolution of the warping. Unlike the existing approaches, our method does not require to compute the warping at the finest image resolution without subsampling the data, which deteriorates block distances and causes aliasing.

At a lower $(K = M/k, L = N/l)$ resolution, the edges connect (x, i, j) to $(x + k, i, j)$'s, the warp window stays same, and the consecutive scan-line becomes $y = y + l$.

4. EXPERIMENTS AND CONCLUSIONS

We tested the proposed algorithm using synthetic and real images. Fig. 4 - Fig. 7 show various image pairs, extracted warping functions, and recovered images. We manually deformed the original images to generate the warped images. The effect of the block size δ is given in Fig. 3. We compared the similarity of the original image and the reconstructed image from warped version using the extracted warping function for different matching block and warping windows sizes. As visible, the accuracy increases if the warping window size becomes as large as the maximum warping amount, and matching block size around $\delta = 4$.

We compared the computational complexity of our method with the state-of-art methods. In [4], it is shown that their 2-D dynamic programming implementation has $\mathcal{O}(M^3 9^M)$ time complexity. Levin's hidden Markov model based method [1] requires even more operations by expanding to an exponential complexity $\mathcal{O}(M^{4M})$.

On the other hand, our algorithm is remarkably lower time complexity $\mathcal{O}(9(x_{max} - x_{min})(y_{max} - y_{min})(2\delta + 1))^2 M^2 \sim \mathcal{O}(\kappa M^2)$. This is due to using M scan-lines, $9M(x_{max} - x_{min})(y_{max} - y_{min})$ transitions in dynamic programming phase, and $(2\delta + 1))^2$ operations for block matching to construct the distance matrices. For instance, in case of a 5×5 matching window, 11×11 warping window, and $M = N = 256$ image, our algorithm needs only 1.78×10^9 operations, whereas the prementioned 2-D approach demands an exceedingly large number of operations; 3×10^{251} (Using semi-optimal beam-search technique, the number of operations for the 2-D approach decreases to 1.82×10^{10}, however it has still $\mathcal{O}(M^3)$ time complexity).

In summary, we propose a novel, accurate, warping function recovery method that is scalable and proven to be computationally much simpler than the existing methods.

5. ACKNOWLEDGEMENTS

We sincerely thank Dr. Kadir Peker for his valuable comments.

6. REFERENCES

[1] E. Levin and R. Pieraccini, "Dynamic planar warping for optical character recognition", *Proceedings of ICASSP*, 149-152, 1992.

[2] R. Keeney and H. Raiffa, "Decisions with multiple objectives", *Wiley Press*, 1976.

[3] F. Porikli. "Sensitivity characteristics of cross-correlation distance metric and model function", *Proceedings of 37th CISS*, 2003.

[4] S. Uchida and H. Sakoe. "A monotonic and continuous two-dimensional warping based on dynamic programming", *Proceedings of 14th ICPR*, 1, 521-524, 1998.

[5] S. Uchida and H. Sakoe. "Piecewise Linear Two-Dimensional Warping", Proceedings of the 15th ICPR, 3, 538-541, 2000.

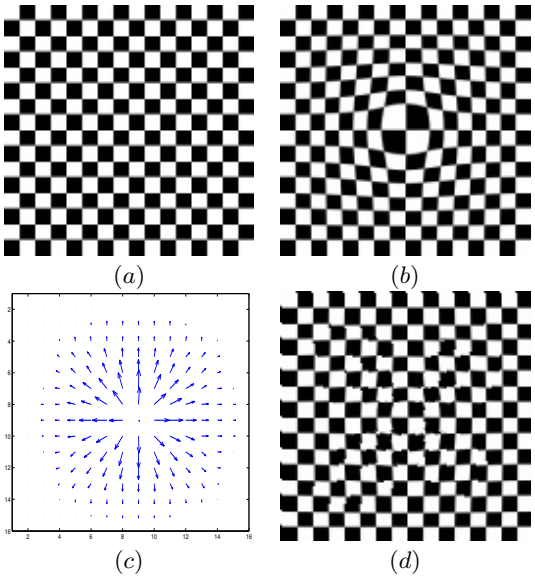

Fig. 4. Checker pattern: (a) original, (b) warped image, (c) warp function, (d) reconstructed image.

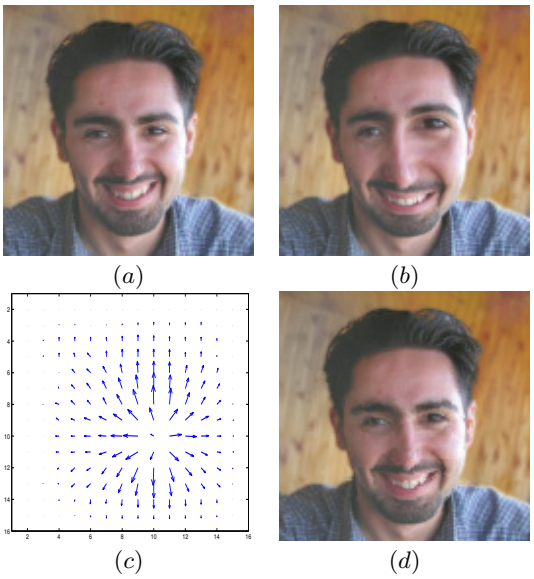

Fig. 6. *Ugur* (a) Original, (b) warped image, (c) warp function, (d) reconstructed image.

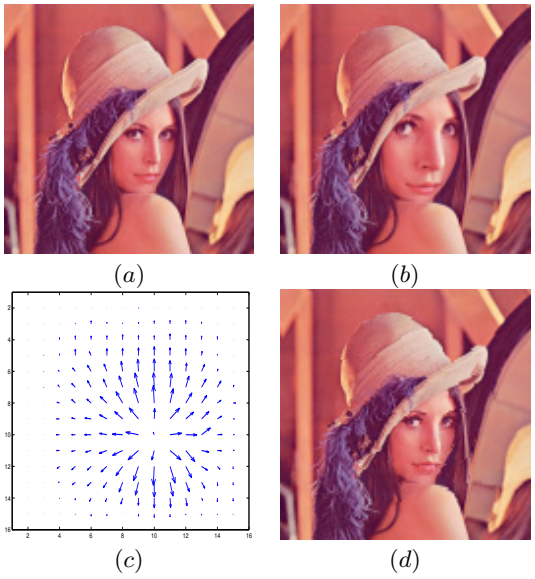

Fig. 5. *Lena*: (a) original, (b) warped image, (c) warp function, (d) reconstructed image.

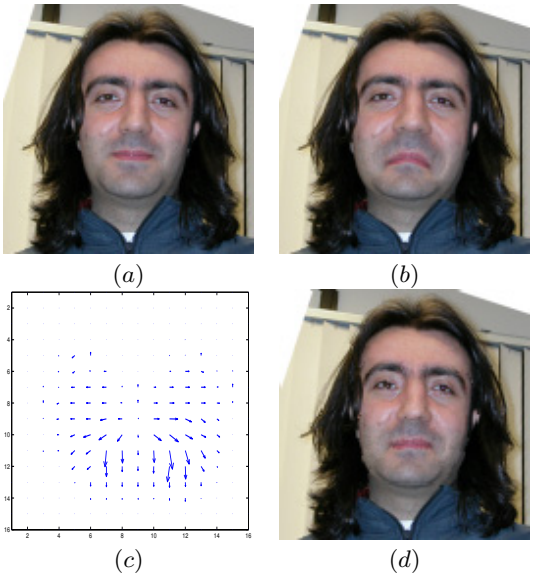

Fig. 7. (a) Original, (b) warped image, (c) warp function, (d) reconstructed image.