BoostMotion: Boosting a discriminative similarity function for motion estimation

Shaohua Kevin Zhou[†], Jie Shao[‡], Bogdan Georgescu[†], and Dorin Comaniciu[†]
[†]Integrated Data Systems Department, Siemens Corporate Research, Princeton NJ 08540
[‡]Center for Automation Research, University of Maryland, College Park, MD 20742

Abstract

Motion estimation for applications where appearance undergoes complex changes is challenging due to lack of an appropriate similarity function. In this paper, we propose to learn a discriminative similarity function based on an annotated database that exemplifies the appearance variations. We invoke the LogitBoost algorithm to selectively combine weak learners into one strong similarity function. The weak learners based on local rectangle features are constructed as nonparametric 2D piecewise constant functions, using the feature responses from both images, to strengthen the modeling power and accommodate fast evaluation. Because the negatives possess a location parameter measuring their closeness to the positives, we present a location-sensitive cascade training procedure, which bootstraps negatives for later stages of the cascade from the regions closer to the positives. This allows viewing a large number of negatives and steering the training process to yield lower training and test errors. In experiments of estimating the motion for the endocardial wall of the left ventricle in echocardiography, we compare the learned similarity function with conventional ones and obtain improved performances. We also contrast the proposed method with a learning-based detection algorithm to demonstrate the importance of temporal information in motion estimation. Finally, we insert the learned similarity function into a simple contour tracking algorithm and find that it reduces drifting.

1. Introduction

Motion estimation is fundamental to computer vision. Underlying any motion estimation algorithm lie two ingredients: similarity function and spatiotemporal smoothing. While many existing approaches [7, 11, 20] investigated the latter, we concentrate on the former. A similarity function is a two-input function $s(\mathtt{I}, \mathtt{I}')$ that measures how closely the test patch $\mathtt{I}'$ is visually similar to the template patch $\mathtt{I}$. A typical motion estimation algorithm performs the following: given two consecutive frames and a target point (u, v) whose motion vector to be measured from $t-1$ to t, one finds the shift that has the (local) maximum similarity.

$$(\delta\hat{u}, \delta\hat{v}) = \arg \max_{(\delta u, \delta v) \in \mathcal{N}} s(\mathtt{I}_{t-1}(u, v), \mathtt{I}_t(u+\delta u, v+\delta v)), \tag{1}$$

where $\mathtt{I}_t(u, v)$ is a local patch extracted from the frame t, centered at (u, v), and $\mathcal{N}$ is the searching window.

For applications where the observed appearance undergoes complex changes in an application-dependent fashion, motion estimation is challenging due to lacking an appropriate similarity function. Similarity functions proposed in the literature are mostly generic and inadequate for handling complex appearance variations. As a motivating example, consider a stress echocardiographic video (stress echo), a series of 2D ultrasound images of the human heart captured after the patient undergoes exercise or takes special medicine. We focus on wall motion analysis to characterize the functionality of the heart. To be specific, we measure the motion of the endocardium of the left ventricle (LV). As shown in Figure 1, the LV endocardium presents severe appearance changes over a cardiac cycle due to nonrigid deformation, imaging artifacts like speckle noise and signal dropout, movement of papillary muscle (which is attached to the LV endocardium but not a part of the wall), respiratory interferences, unnecessary probe movement, etc. In the experiments, we illustrate the ineffectiveness of generic similarity functions when applied to estimating the motion in the stress echo sequences.

Section 2 gives a brief review of similarity functions as well as learning-based tracking algorithms. Generic similarity functions work for certain imaging scenarios but break down for others, depending on the way in which the appearance changes. For example, the sum of squared distance (SSD) works best for isotropic Gaussian noise, the sum of absolute distance (SAD) for the Laplacian noise, and the CD_2 [5] for fully developed speckle noise. On one hand, there does not exist a *global* and generic similarity function universally good for all scenarios; on the other hand, it is very likely that different *local* regions of the image are best suited for different similarity functions. This strongly motivates us to take a boosting approach [9, 10] that combines locally good similarity function (still weak though) into a

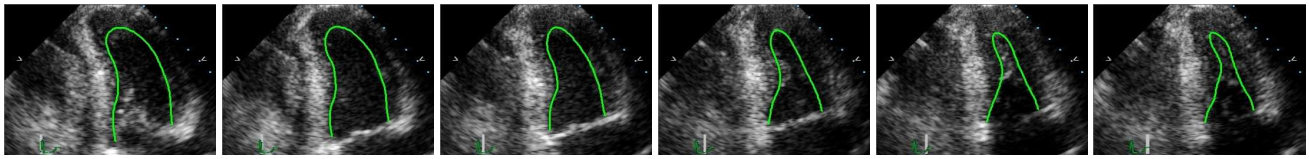

Figure 1. An example of stress echo sequence with the annotation of the LV endocardium. Due to the rapid heart rate (177 beats per second), the LV appearance varies significantly within a short period of six consecutive frames.

globally strong similarity function that works best for the scenario exemplified by an annotated database.

From another perspective, one wish to have a similarity function whose response map is highly peaked at the correct location. In the extreme, the similarity function operates as a discriminative function: positive if the centers of two paired image patches are in correspondence and negative if out of correspondence.

$$s(\mathtt{I}, \mathtt{I}') = \begin{cases} 1 & \text{if } (\mathtt{I}, \mathtt{I}') \text{ is a corresponding pair;} \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

This fits in the concept of discriminative learning. By treating the image pairs in correspondence as positives and the rest as negatives, the similarity function becomes a discriminative function that separates two classes. Given an annotated video database, we can learn such a discriminative function using examples extracted from the database.

Inspired by the above arguments, we propose to learn a discriminative similarity function using the boosting framework. We invoke the LogitBoost algorithm [10] to selectively combine weak learners into one strong similarity function. We associate a weak learner with an Haar-like local rectangle feature [17, 22] to accommodate fast computation. The weak learner takes an image pair as input and uses the two feature responses collected from both images. We construct the weak learner as a nonparametric 2D piecewise constant function of the two feature responses in order to strengthen its modeling power, thereby bringing savings in both training speed and storage requirement. Section 3 addresses the boosting procedure and weak learners. Because we boost a similarity function for motion estimation, we call the proposed approach *BoostMotion*.

Selecting negatives is crucial to the training accuracy and consequently influences the motion estimation accuracy. The negatives implicitly possess a location parameter measuring their closeness to the positives. To leverage the additional distance structure of the negatives, we present a location-sensitive cascade training procedure that bootstraps negatives for later stage of the classifier cascade from the regions closer to the positives. This allows not only viewing a large number of negatives as in the regular cascade training [22] but also steering the training process with respect to the motion estimation accuracy. In section 4, we empirically show that the location-sensitive cascade yields lower training and test errors than the regular one.

In section 5, we compare the proposed similarity function with conventional similarity functions using the stress echo sequences and obtain improved performance when estimating the motion of the LV endocardium. We also contrast the BoostMotion approach, which takes a pair of images as input, with a learning-based detection algorithm, which takes a single image patch as input, and demonstrate the importance of temporal information in motion estimation. Then we insert the BoostMotion module into a naive tracker that estimates the motion vector frame by frame and hence is prone to drift. In the experiment of tracking regular echo sequences, we show that using the discriminant similarity function reduces drifting. Section 6 summarizes the paper.

2. Related work

In this section, we briefly review related work on (i) similarity functions commonly used in the motion estimation algorithms and (ii) learning-based visual tracking.

2.1. Similarity function

The similarity functions for motion estimation proposed in the literature can be roughly categorized as (i) intensity-based, (ii) histogram-based, and (iii) application-specific.

Intensity-based similarity function. It includes sum of square distance (SSD) [4, 14, 19], sum of absolute distance (SAD), and normalized cross correlation (NCC).

The SSD similarity function corresponds to the 'brightness constancy' assumption.

$$SSD: \quad s(\mathtt{I}, \mathtt{I}') = ||\mathtt{I} - \mathtt{I}'||_2 \quad (3)$$

where $||.||_2$ takes the L_2 norm. The SSD is also equivalent to assuming an isotropic Gaussian noise model.

The SSD can be generalized in two ways. First, it can be based on images derived from the original ones. For example, if the gradient image is used, this corresponds to the 'gradient constancy' assumption [4]. Second, the L_2 norm can be replaced by an L_p norm. If $p = 1$, the similarity function becomes the SAD function which is equivalent to assuming a Laplacian noise model.

The NCC function is defined as

$$NCC: \quad s(\mathtt{I}, \mathtt{I}') = \frac{(\mathtt{I} - \mu(\mathtt{I})) \bullet (\mathtt{I}' - \mu(\mathtt{I}'))/N}{\sigma(\mathtt{I})\sigma(\mathtt{I}')}, \quad (4)$$

where $\bullet$ denotes the dot product, N is the number of pixel in the image $\mathtt{I}$, and $\mu(.)$ and $\sigma(.)$ take the sample mean and standard deviation, respectively.

Histogram-based similarity function. Assuming that the histogram of $\mathtt{I}$ is given by $h(\mathtt{I})$, the Bhattacharyya distance [8] is defined as

$$BHA: \;\; s(\mathtt{I}, \mathtt{I}') = \int \sqrt{h(\mathtt{I})h(\mathtt{I}')}d\lambda. \qquad (5)$$

Other histogram-based similarity functions include the χ^2 distance, earth moving distance, KL divergence, etc.

Application-specific similarity function. In the paper, we focus on ultrasound images. The ultrasound-specific CD_2 similarity function proposed by Cohen and Dinstein [5] is specially designed for handling a fully-developed speckle noise in an ultrasound image and shown to be effective by Boukerroui *et al*. [3]. Denoting $\tilde{\mathtt{I}} = \log(\mathtt{I})$, the CD_2 function is defined as

$$CD_2: \;\; s(\mathtt{I}, \mathtt{I}') = \sum\{\tilde{i} - \tilde{i}' - \log[\exp(2\tilde{i} - 2\tilde{i}') + 1]\}, \quad (6)$$

where $\tilde{i}$'s are the pixels belonging to the image $\tilde{\mathtt{I}}$.

2.2. Learning-based visual tracking

It should be emphasized that visual tracking is different from motion estimation. The latter concerns only two successive frames; while the former concerns the motion estimation of a whole video sequence. While a naive tracker estimates motion recursively frame by frame, tracking is more than motion estimation as the above naive tracker is prone to drift away. To overcome drifting, the tracking algorithm does the following: (i) updates the appearance model strategically and/or (ii) performs temporal smoothing/fusion.

Underlying learning-based tracking algorithms lies a data-driven procedure. In most cases, a binary classifier (except [13]) that discriminates the object of interest and the background is learned, whether offline or online. If the classifier is learned offline, this solves tracking as a detection problem [17, 22]. In [16], temporal smoothing is enforced by casting the detector output as the observation likelihood in a particle filter setting. Avidan [1] proposes a support vector tracking algorithm that learns a support vector machine (SVM) from the training data using the polynomial kernel. The SVM score, after the Taylor expansion, is analytically maximized for every frame. In [24], Williams *et al*. build on the relevance vector machine to perform tracking, where temporal fusion is applied. In [13], Lepetit *et al.* artificially generates exemplars (using affine transform or 3D model) for each feature point that is treated as a class and use 1-NN neighbor searching to determine the class label.

If the classifier is learned online, the appearance model updating is embedded into the classifier. In the work of Collins and Liu [6], the appearance model is represented by a set of features that are selected online based on the variance ratio of the log likelihood function, which is empirically estimated. Ensemble tracking [2] developed by Avidan invokes the AdaBoost to learn the classifier. After tracking, the classifier is updated by adding the recently tracked results. Since the AdaBoost is a feature selection process, the ensemble tracker also represents the appearance model by features that are updated over time.

The approaches reviewed above commonly learn a classifier to differentiate the foreground object from the background. However, such a classifier gives no account of temporal information essential to motion estimation, which performs pairwise comparison along the temporal dimension. In other words, the input to the classifier is always a single image patch not a pair of images. The proposed BoostMotion approach explores the possibility of using image pairs as inputs and embeds the temporal statistics into a learned similarity function. In [23], the temporal difference images are used in boosting a pedestrian detector (again not a similarity function). Incidently, learning the spatial statistics of optical flow is addressed in [18].

Two-class LogitBoost (positive – y=1, negative – y=0)

0. Input: (i) Training data $\{x_i;\ i = 1, 2, \ldots, N\}$ and their corresponding class labels $\{y_i;\ i = 1, 2, \ldots, N\}$. (ii) The structural space $\mathcal{F}$.
1. Start with weights $w_i = 1/N$, $i = 1, 2, \ldots, N$, $F(x) = 0$, and probability estimates $p(x_i) = 1/2$.
2. Repeat for $m = 1, 2, ..., M$:
   - (a) Compute working responses and weights:
   $$z_i = \frac{y_i - p(x_i)}{p(x_i)(1 - p(x_i))}; \qquad (7)$$
   $$w_i = p(x_i)(1 - p(x_i)). \qquad (8)$$
   - (b) Fit the function $f_m(x)$ by a weighted least-squares (LS) regression of z_i to x_i with weights w_i.
   $$f_m(x) = \arg\min_{f \in F}\{\epsilon(f) = \sum_{i=1}^{N} w_i(z_i - f(x_i))^2\}. \qquad (9)$$
   - (c) Update $F(x) \leftarrow F(x) + \frac{1}{2}f_m(x)$ and $p(x)$ via
   $$p(x) = \frac{\exp(F(x))}{\exp(F(x)) + \exp(-F(x))}. \qquad (10)$$
3. Output the classifier $sign[F(x)]$.

Figure 2. The two-class LogitBoost algorithm [10].

3. Boosting

We invoke the framework of boosting to learn the similarity function. Boosting iteratively selects weak learn-

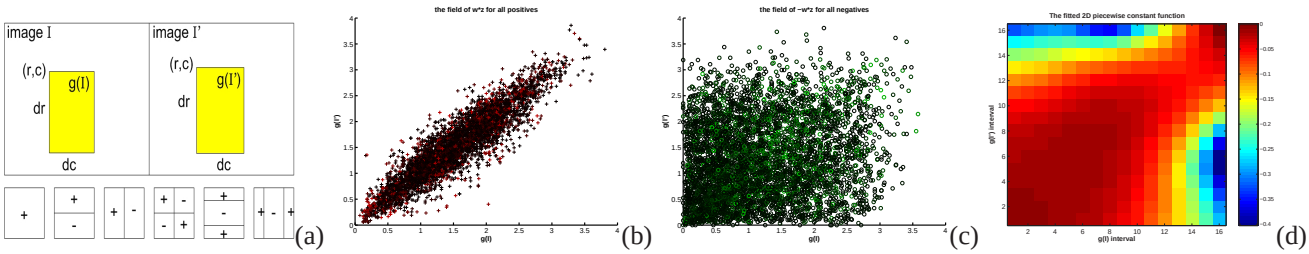

Figure 3. (a) A weak similarity function compares two local rectangle regions belonging to images $\mathtt{I}$ and $\mathtt{I}'$, respectively. Panels from (b) to (d) illustrate the process of fitting a 2D PWC function: (b) the field of $w * z$ for all positives; (c) the field of $-w * z$ for all negatives; and (d) the fitted 2D PWC function.

ers to form a strong learner using an additive form: $F(x) = \sum_{f_m(x)\in\mathcal{F}} f_m(x)$, where $F(x)$ is the strong learner, $f_m(x)$'s are the weak learners, and $\mathcal{F}$ is the structural space where the weak learners reside.

Boosting has three key components: (i) structural space; (ii) noise model or cost function; and (iii) selection algorithm. Different variants of boosting are proposed in the literature depending on different choices of the key components. We decide to use the LogitBoost algorithm [10] summarized in Figure 2. The LogitBoost algorithm differs from the commonly used AdaBoost algorithm [9] in the following two aspects. First, they optimize different cost functions. The AdaBoost algorithm minimizes an upper bound of the target misclassification error; the LogitBoost algorithm directly minimizes a negative binomial log-likelihood, which is a natural choice for a binary classification problem. Second, the weak learner in the AdaBoost is a hard classifier while that in the LogitBoost is not: experimental evidence seems to favor the latter.

The crucial step in the LogitBoost algorithm is step $2(b)$ in Figure 2: fitting a weighted LS regression of z_i to x_i with weights w_i. It operates as a feature selection oracle: picking up from the structural space $\mathcal{F}$ the weak learner (or feature function) that minimizes its weighted LS cost $\epsilon(f)$.

In our context, a data point x is an image pair $x = (\mathtt{I}, \mathtt{I}')$. One obvious choice for the boosted similarity function $s(\mathtt{I}, \mathtt{I}')$ is the probability of the class label $y(\mathtt{I}, \mathtt{I}')$ being 1, that is $s(\mathtt{I}, \mathtt{I}') = p(\mathtt{I}, \mathtt{I}')$.

3.1. Weak learner

We now define the weak leaner $f(\mathtt{I}, \mathtt{I}')$. Given the fact that different similarity functions are effective for different local regions, we construct the weak learners based on Haar-like local rectangle features [17, 22], whose rapid evaluation is enabled by the means of integral image.

A weak similarity function compares two local rectangle regions belonging to the two images $\mathtt{I}$ and $\mathtt{I}'$, respectively. As illustrated in Figure 3(a), we parameterize the rectangle feature g by (r, c, dr, dc, t) where (r, c) is the starting point of the rectangle, (dr, dc) is the height and width, and t is the feature type. There are six feature types as shown in Figure 3(a). Given a rectangle feature g and an image pair $(\mathtt{I}, \mathtt{I}')$, we compute two feature responses $g(\mathtt{I})$ and $g(\mathtt{I}')$ from the two integral images associated with $\mathtt{I}$ and $\mathtt{I}'$, respectively. In principle, we can allow that two local rectangles have different parameters; however we refrain from doing this because empirically this shows no clear advantage but significantly increases training complexity.

We focus on the 2D feature space of the two feature responses $g(\mathtt{I})$ and $g(\mathtt{I}')$ and model the weak learner $f(\mathtt{I}, \mathtt{I}')$ as a 2D piecewise constant (PWC) function of $g(\mathtt{I})$ and $g(\mathtt{I}')$, which has the following form:

$$f(\mathtt{I}, \mathtt{I}') = \sum_{j=1}^{J}\sum_{k=1}^{K} \alpha_{jk}[g(\mathtt{I}) \in T_j] \wedge [g(\mathtt{I}') \in T'_k], \quad (11)$$

where $[\pi]$ is an indicator function of the predicate π and α_{jk} is the constant associated with the region R_{jk}. In the above, we use a tessellation of the 2D feature space into non-overlapping regions $\{R_{jk} = T_j \wedge T'_k\}_{j,k=1}^{J,K}$, where $\{T_j\}_{j=1}^{J}$ and $\{T'_k\}_{k=1}^{K}$ are the J and K non-overlapping intervals for the feature response $g(\mathtt{I})$ and $g(\mathtt{I}')$, respectively. We empirically determine the interval boundary points by uniformly dividing the feature responses.

It is easy to show that, given a weak learner f that is associated with a feature g, the optimal weight α_{jk} that minimizes the weighted LS cost $\epsilon(f)$ in (9) is the weighted response z of all data points falling into the region R_{jk}.

$$\alpha_{jk} = \frac{\sum_{i=1}^{N} w_i z_i [g(\mathtt{I}_i) \in T_j] \wedge [g(\mathtt{I}'_i) \in T'_k]}{\sum_{i=1}^{N} w_i [g(\mathtt{I}_i) \in T_j] \wedge [g(\mathtt{I}'_i) \in T'_k]}, \quad (12)$$

where $(\mathtt{I}_i, \mathtt{I}'_i)$ is the i^{th} training image pair. Figure 3(b,c,d) illustrates the fitting process. Figure 3(b) visualizes the field of $w_i * z_i = y_i - p(x_i) = 1 - p(x_i)$ for all positives, where the color intensity corresponds to the value of $w * z$: the redder the plus sign is, the less likely the data point x is positive. The diagonal structure in Figure 3(b) shows that the two feature responses of the positives are roughly same. Figure 3(c) visualizes the field of $-w_i * z_i = p(x_i)$ for all negatives: the greener the circle sign is, the less likely the data point x is negative. As shown in Figure 3(c), the negatives are characterized by a widely-dispersed nature. Figure

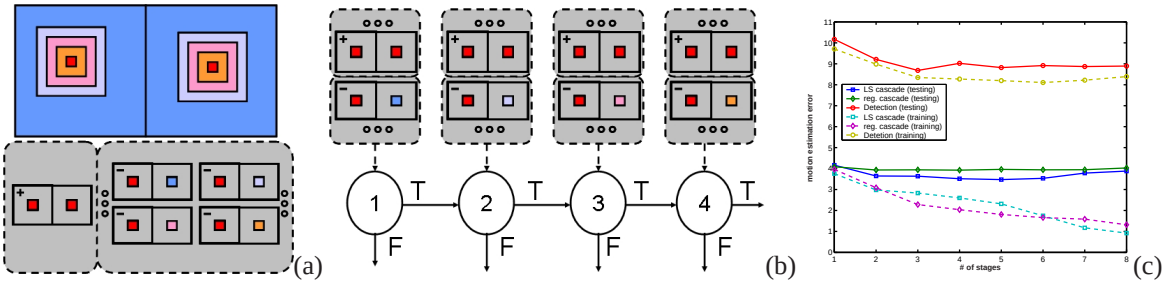

Figure 4. (a) Two successive frames and their corresponding positives and negatives. (b) Location-sensitive cascade training. (c) Performance comparison among location-sensitive cascade, regular cascade, and detection.

3(d) shows the fitted 2D PWC function: the constant coefficients α_{jk} along the diagonal lines are high, while off-diagonal ones are low. For the step 2(a) in Figure 2, the weak function f with the smallest weighted LS cost $\epsilon(f)$ is selected.

The use of nonparametric 2D PWC functions as weak learners is beneficial. Take the 1D case for example; 1D simple regression stumps [21, 22] that 'binarize' the feature response are often used as weak learners in the literature. It is easy to verify that any 1D PWC function can be constructed by combining multiple 1D simple regression stumps. The similar holds for the 2D case. Such a combination strengthens the modeling power of weak learners and consequently accelerates the training process. Our empirical evidence shows that the learning time is almost inversely proportional to the number of thresholds used in the weak learner. One may argue that it brings the risk of overfitting. But boosting has the incredible capability of combating the overfitting (in terms of classification though) even when the weak learner overfits. Further, in practice we smooth the fields of $w * z$ and w before taking the division in (12) to ameliorate the overfitting of the weak learner itself.

Boosting training requires huge memory space because one has to evaluate a huge matrix, whose row corresponds to the local rectangle feature and whose column to the training image pair. It is desired to store such a matrix in the memory in order to speedup the training process. Typically, the number of rectangle features is huge (e.g. more than $150K$ for a 24×24 image by an exhaustive construction [22]). In one of our experiments, we keep about 40K rectangle features and the number of the training image pair is about $10K$, storing the above matrix in a *float* precision requires about $40K \times 10K \times 4 \times 2 = 3.2GB$ memory space, exceeding the 2GB limit, which is the maximum contiguous block of memory of 32-bit operating systems. However, in order to learn the PWC function in our setting, we only need to store the interval index in the memory. In practice, we use 16 intervals, which implies that an *unsigned char* is enough to store two indices, leading to a moderate memory requirement of about $400MB$.

4. Location-sensitive cascade training

Generating positives and negatives, which are pairs of images, from annotated videos is illustrated in Figure 4(a). Given a pair of two successive frames (the left and right images in Figure 4(a)), it contributes one positive by cropping two image patches centered at the target pixel (denoted by the red color) from the left and right frames, respectively.

To generate negatives, we maintain the same image patch cropped from the left frame, i.e., centered at the target pixel, but force the center of the image patch cropped from the right frame away from the target pixel. Therefore, the negative possesses an additional location parameter that measures its distance to the target. Obviously, the number of negatives is theoretically infinite if a non-integer pixel grid is used. To cover as many negatives as possible, we follow [22] to train a cascade of strong classifiers, which is a degenerate decision tree. To train the strong classifier at a later stage, we maintain the same set of positives but bootstrap a new set of negatives that pass all previous strong classifiers (i.e., false positives). During scanning all test subwindows, the cascade structure is able to eliminate the negatives quickly.

The motion estimation accuracy is directly related to the selection of the negatives. On one hand, if the negatives are far away from the positives, it is easy to learn a perfect classifier but the accuracy is not guaranteed. On the other hand, if the negatives are too close to the positives, the accuracy is improved but it is hard to train a flawless classifier and might step into the zone of overfitting because the training positives and negatives are too confusing. Often in medical applications, different experts disagree with each other about the ground truth; thus, motion estimation only needs to be addressed in a pre-specified precision. To this end, we design a location-sensitive cascade training procedure that takes into account the location factor of the negatives.

The pixels of a video frame are divided into several regions according to their distances to the target pixel as illustrated in Figure 4(a), where the target pixel is denoted by the red color and the regions are color-banded. While preserving the features of the regular cascade training, the location-

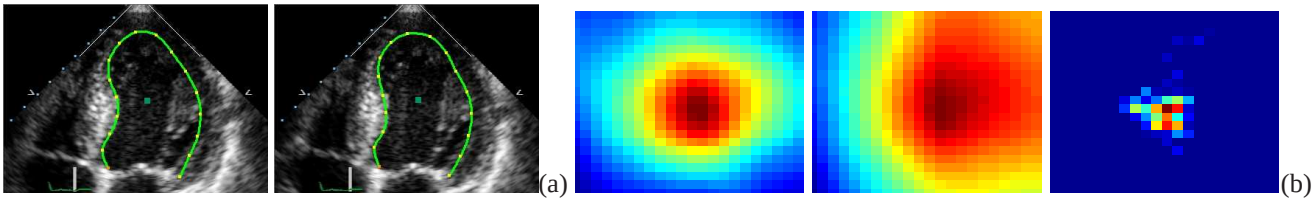

Figure 5. (a) Two successive original frames and their ground truth landmarks and contours. (b) From left to right: the response maps of the similarity functions of NCC, CD_2 and BoostMotion.

sensitive cascade training imposes an additional constraint: the negatives for several consecutive stages of the cascade are restricted to be from the same region. Further, the later stages use negatives closer to the positives; however, negative bootstrapping is still applied even across the boundary of the stages using negatives from different regions. This procedure is graphically illustrated in Figure 4(b). Refer to section 5.1 for a comparison between the location-sensitive cascade training and regular one, where we show that the location-sensitive cascade yields lower training and test errors (also see Figure 4(c)).

In practice, we need to convert the cascade output into the similarity function that measures the confidence of being positive. Suppose that the cascade consists of L stages and stage l has a strong classifier $F_l(\mathtt{I}, \mathtt{I}')$ that can be converted to posterior probability $s_l(\mathtt{I}, \mathtt{I}')$ using Eq. (10). Given the degenerate nature of the cascade, we approximate the final similarity function as:

$$s(\mathtt{I}, \mathtt{I}') \approx \prod_{l=1}^{L} s_l(\mathtt{I}, \mathtt{I}') = \prod_{l=1}^{L} \frac{\exp(2F_l(\mathtt{I}, \mathtt{I}'))}{\exp(2F_l(\mathtt{I}, \mathtt{I}')) + 1}. \quad (13)$$

For the negatives rejected at an early stage $L' < L$, we stop evaluating them at later stages and simply set a dummy probability $s_l(\mathtt{I}, \mathtt{I}') = \epsilon;\ l > L'$, where $\epsilon < 0.5$ is a small amount (we set $\epsilon = 0.1$ in our experiments).

5. Experimental results

We conducted extensive experiments using echocardiographic sequences that are ultrasound images of human hearts. As shown in Figure 5(a), we parameterized the LV endocardial wall by 17 landmark points along the contour and then interpolated the whole contour using a cubic spline.

5.1. Motion estimation

In the first experiment, we used the apical four chamber (A4C) view of stress echo sequences. The A4C view is a standard cardiac view used in clinical practices. We have 339 sequences that provides 3162 frame pairs. We randomly divided 339 sequences into two sets: the training set contains 270 sequences with 2543 pairs of frames and the test set 69 sequences with 619 pairs of frames. To reduce appearance variation, we aligned each video frame with respect to a mean shape using a rigid similarity transform and conducted experiments on the aligned domain.

We reported the results of estimating the motion vector of the left annulus point, i.e., the left end point of the LV endocardium, that is characterized by drastic appearance changes mainly due to the valve movement. Given the correct left image patch (see Figure 4 for illustration), we exhaustively searched within a searching neighborhood the best right image patch that maximizes the similarity function (e.g., Eq. (1)). We estimated the motion vector for all test image pairs and measured the estimation error in terms of absolute displacement. For an image pair, we set the size of the left and right images as 35×35 and the searching window $\mathcal{N}$ as $[-21, 21] \times [-21, 21]$.

We used the location-sensitive cascade training to learn a cascade of eight stages. We divided the whole search neighborhood into eight regions depending on their closeness to the center: specifically, they are $R_1 = \{21, 20, 19\}$, $R_2 = \{18, 17, 16\}$, $R_3 = \{15, 14, 13\}$, $R_4 = \{12, 11\}$, $R_5 = \{10, 9\}$, $R_6 = \{8, 7\}$, $R_7 = \{6, 5\}$, $R_8 = \{4, 3\}$ pixels away from the center. To train the i^{th} stage of strong classifier, we selected out negatives from the region R_i. For comparison, we also trained a regular cascade by randomly selecting out negatives at least three pixels away from the ground truth. Figure 4(c) plots the curves of the training and test errors against the number of cascade stages. The location-sensitive cascade training consistently reduces the test error till overfitting is reached; while the regular cascade training saturates the performance even at the second stage. Apart from that it yields lower training and test errors, the location-sensitive cascade training provides a steerable way to control the training process.

We also contrasted the BoostMotion approach, which uses pairs of images as inputs, with a detection algorithm, which uses single image patches as inputs. For the detection algorithm, we also trained a cascade of eight stages. From Figure 4(c), even the training error of the detector is higher than that of the BoostMotion, which implies that the positives and negatives, which are single image patches, are very confusing even they are far apart. Using the paired inputs significantly reduces the training and test errors. This is expected because motion estimation compares two images and thus temporal information is its key. Motion estimation

Approach	SSD	NCC	BHA	CD_2	BoostMotion	Detection
(a) Motion estimation training error	4.62 ± 2.48	4.59 ± 2.38	11.15 ± 6.66	4.39 ± 2.28	2.32 ± 1.72	8.34 ± 4.99
Motion estimation test error	4.54 ± 2.57	4.49 ± 2.45	11.40 ± 6.75	4.38 ± 2.17	3.47 ± 2.14	8.68 ± 5.07
(b) Tracking contour distance	10.56 ± 2.37	11.14 ± 2.45	14.31 ± 3.19	7.32 ± 2.45	4.28 ± 1.24	na

Table 1. (a) Motion estimation training and test errors and (b) tracking performance based on the contour distance obtained using different similarity functions and a detector approach.

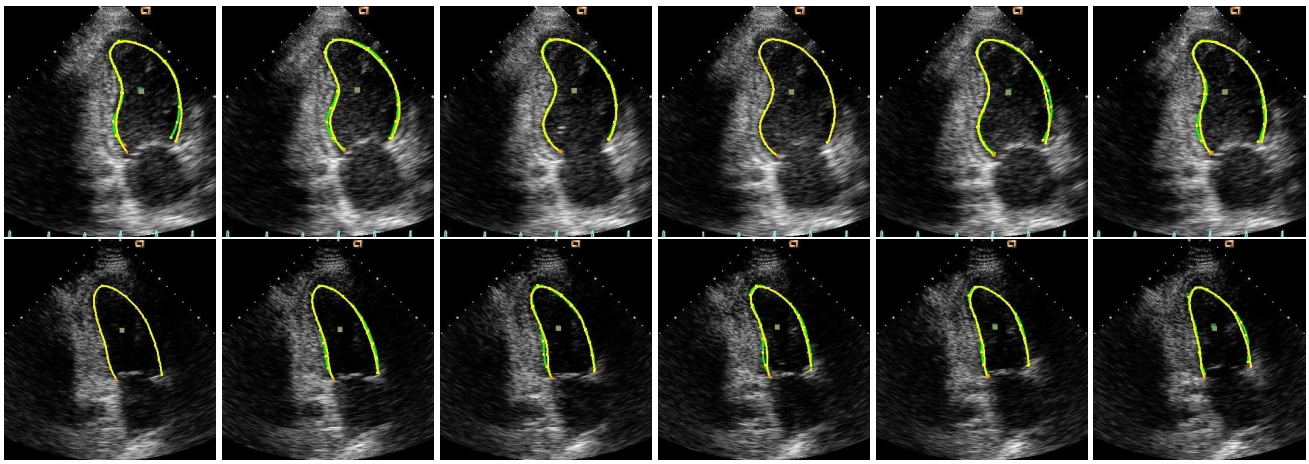

Figure 6. Sample frames of two echocardiographic sequences with the ground truth (yellow) and tracking result (green) overlaid. Top row: frames 1, 4, 6, 8, 11, and 14 of the first sequence. Bottom row: frames 1, 3, 5, 7, 8, and 11 of the second sequence.

based on one image is insufficient.

We compared the learned similarity function with the four conventional similarity functions reviewed in section 2.1. Table 1(a) reports the mean and standard deviation of the estimation error. For the BoostMotion and detection approaches, we reported the minimum test error. The BoostMotion approach yields the lowest training and test errors. In terms of test error, on average, it is only 3.47 pixels away from the ground truth, while the best among the others is the CD_2 similarity function whose estimation error is 4.38 pixels. The poor performance of the Bhattacharyya similarity function is probably due to the highly noisy nature of the ultrasound image and that only gray image is used. Figure 5 displays the response maps of different similarity functions for the sample pair of frames in Figure 5(a). The response map of the BoostMotion is peaked around the ground truth with a compact support region. Most of the off-center pixels are black because they are rejected by early stages of the cascade.

5.2. Echocardiography tracking

In the second experiment, we invoked the naive tracking algorithm that estimates motion vector frame by frame to perform echocardiography tracking [3, 5, 12, 15, 25]. We used 445 regular echocardiographic sequences of apical two chamber (A2C) view where appearance changes are less pronounced than the stress echo. The A2C view is another canonical echo view used in clinical practices. We randomly divided the 445 sequences into a training set with 356 sequences and a test set with 89 sequences. The alignment is conducted in a recursive fashion.

To calibrate the contour tracking accuracy, we need to measure the proximity between two contours. We simply used the average distance of the landmark displacement defined as $\sum_{i=1}^{17} |p_i - g_i|^2/17$, where p_i and g_i are i^{th} landmark point on the probe contour and the ground truth contour, respectively.

The tracking performances in terms of the above distance are listed in Table 1(b). Since each test sequence yields a contour distance, Table 1(b) documents its median and standard deviation. Using the BoostMotion similarity function substantially reduces the tracking error, with its corresponding error being 4.28 pixels; while the best similarity function (CD_2) other than the proposed one yields a tracking error of 7.32 pixels. Therefore, utilizing the boosted similarity function greatly reduces drifting and produces a temporally smooth contour. Some tracking examples are presented in Figure 6.

6. Summary

We presented an approach to learning a similarity function for motion estimation for applications with complex appearance changes, which are exemplified by an annotated video database. We used the LogitBoost algorithm to selectively combine weak learners into a strong similarity function. The weak learners were constructed as nonparametric

2D piecewise constant functions of the feature responses, which enhanced modeling power and brought savings in training time and storage requirement. Because the motion estimation accuracy is tied with the selection of negatives, whose have an additional location parameter measuring their distance to the positives, we proposed to train a cascade structure in a steerable manner using a location-sensitive negative bootstrapping. Compared with the regular cascade, the location-sensitive cascade achieved lower training and test error. Finally, we experimented the learned similarity function on echocardiographic sequences. It outperformed by a large margin conventional similarity functions and the detector algorithm, which ignores the temporal information. It also reduced drifting. Future works include combining spatial statistics [18] with the learned similarity function in motion estimation. For more robust tracking, we will investigate incorporating appearance update and temporal smoothing.

Acknowledgement

We thank Dr. Sriram Krishnan of CAD, Siemens Medical Solutions for providing the data.

References

[1] S. Avidan. Support vector tracking. In *CVPR*, volume 1, pages 184–191, 2001.

[2] S. Avidan. Ensemble tracking. In *CVPR*, volume 2, pages 494–501, 2005.

[3] D. Boukerroui, J. Alison, and M. Brady. Velocity estimation in ultrasound images: A block matching approach. In *IPMI*, pages 586–598, 2003.

[4] T. Brox, A. Bruhn, N. Papenberg, and J. Wiecker. High accuracy optical flow estimation based on a theory of warping. In *ECCV*, volume 4, pages 25–36, 2004.

[5] B. Cohen and I. Dinstein. New maximum likelihood motion estimation schemes for noisy ultrasound images, 2002.

[6] R. Collins and Y. Liu. On-line selection of discriminative tracking features. In *ICCV*, 2003.

[7] D. Comaniciu. Nonparametric information fusion for motion estimation. In *CVPR*, pages 59–66, 2003.

[8] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, volume 2, pages 142–149, 2000.

[9] Y. Freund and R. Schapire. A decision-theoretic generalization of online leaning and an application to boosting. *Journal of Computer and System Sciences*, 5(1):119.

[10] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Ann. Statist.*, 28(2):337–407, 2000.

[11] B. Horn and B. Schunk. Determining optical flow. *Artificial Intelligence*, pages 185–204, 1981.

[12] G. Jacob, A. Noble, and A. Blake. Robust contour tracking in echocardiographic sequence. In *Proc. Intl. Conf. on Computer Vision*, pages 408–413, 1998.

[13] V. Lepetit, J. Pilet, and P. Fua. Point matching as a classification problem for fast and robust object pose estimation. In *CVPR*, 2004.

[14] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. DARPA IU Workshop*, pages 121–130, 1981.

[15] I. Mikic, S. Krucinski, and J. Thomas. Segmentation and tracking in echocardiographic sequences: Active contours guided by optical flow estimates. *IEEE Trans. Medical Imaging*, 17:274–284, 1998.

[16] K. Mikolajczyk, R. Choudhury, and C. Schmid. Face detection in a video sequence - a temporal approach. In *CVPR*, 2001.

[17] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In *ICCV*, 1998.

[18] S. Roth and M. Black. On the spatial statistics of optical flow. In *ICCV*, 2005.

[19] J. Shi and C. Tomasi. Good features to track. In *CVPR*, pages 593–600, 1994.

[20] A. Singh and P. Allen. Image-flow computation: An estimation-theoretic framework and a unified perspective. *CVGIP: Image Understanding*, 56:152–177, 1992.

[21] A. Torralba, K. Murphy, and W. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR*, 2004.

[22] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, pages 511–518, 2001.

[23] P. Viola, M. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *ICCV*, pages 734–741, 2003.

[24] O. Williams, A. Blake, and R. Cipolla. A sparse probabilistic learning algorithm for real-time tracking. In *ICCV*, pages 353–360, 2003.

[25] X. S. Zhou, D. Comaniciu, and A. Gupta. An information fusion framework for robust shape tracking. *PAMI*, 27(1):115–129, January 2005.