A Motion-Compensated Filter for Ultrasound Image Sequences

Lee Markosian

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-96-14
May 1996

A Motion-Compensated Filter for Ultrasound Image Sequences

Lee Markosian

Abstract

Medical ultrasound images are an important diagnostic tool for physicians, but suffer from low image intensity, dropouts, and noise in the form of speckle. I propose a new method for filtering ultrasound image sequences which is based on a motion-compensated temporal filter. The key contribution is a new method for detecting and tracking image features which is fully automatic. Applying this algorithm in a motion-compensated temporal filter demonstrates its worth. The tracking and filtering algorithms together run at speeds approaching real-time on a standard scientific workstation.

1 Introduction

Medical ultrasound images play an important diagnostic role for physicians, and enjoy a number of advantages over other imaging techniques [1]. The advantages can be summed up with the remark that the acquisition of ultrasound images is fast, cheap, and safe. In particular, ultrasound image sequences can be acquired at video rates, allowing dynamic analysis of moving structures (such as heart walls). The equipment required is relatively inexpensive and portable. An ultrasound scan can be conducted in a doctor's office. The scan itself is noninvasive and does not entail exposing the patient to dangerous levels of radiation or toxic substances. As such, for some diagnostic tasks there may be no practical alternative to ultrasound imaging.

On the other hand, the quality of the images is relatively poor. The images are plagued by low image intensity contrast, dropouts in the image (in which structures exhibit apparent gaps, or disappear temporarily in some frames of a sequence), and "speckle" [2]. (See Figure 1).

Ultrasound images are formed by sending a pulse into the body from a transducer placed against the skin. The pulse is reflected by interfaces within the body between tissues with different properties. The transducer records the echoes returning from the pulse and a single scan line is formed from the information. A 2D image is formed by rapidly sending pulses along different rays within a plane. Speckle is caused by backscattering of the pulse as it encounters tissue microstructures, and by interference in the returning echoes. Since the echo is strongest when the pulse is perpendicular to the tissue interface, lateral parts of

tissue boundaries are poorly defined and may drop out. In image sequences, dropout can also occur when a tissue structure moves temporarily out of the imaging plane (due to its own motion, the motion of the transducer, or that of the patient).

The goal of this research is to develop an algorithm for filtering time sequences of medical ultrasound images in order to improve image contrast, preserve edge information, and reduce speckle. In order to achieve this goal according to the proposed method, a necessary sub-goal is to develop an effective algorithm to automatically locate and track features in the images. This sub-goal is really the primary challenge of this research, since filtering time sequences of noisy images is relatively straightforward given an automatic, accurate method for tracking motion in the images. What's more, locating and tracking features in noisy images such as these is recognized as a difficult problem in computer vision [1]. Since any effective filtering technique would be far more useful if it could be applied in real time as the images are acquired and viewed [4], an additional goal of this work is to achieve real-time speeds for an implementation of the algorithm running on a standard scientific workstation.

2 Previous work on speckle reduction

Previous attempts to reduce speckle in ultrasound images fall into two main categories: image filtering and compounding [3, 5]. Image filtering methods, also called post-formation methods, involve the application of image processing techniques to already acquired images. A straightforward approach would be to smooth the image with a low-pass filter, which would reduce speckle. Unfortunately, this would also have the side-effect of blurring the actual information content of the image. Bamber and Daft [6] proposed an adaptive mean-based (low pass) filter in which the filter is applied to each pixel in the image based on local image statistics near that pixel. Basically, the filter is not applied near object boundaries in the image, but is applied fully in areas with uniform background gray level. In this way it is hoped that edges will be preserved while speckle is still suppressed. Loupas et. al. [7] proposed a similar adaptive approach based on a median filter. Koo and Park [5] note that adaptive approaches based on local image statistics suffer from a sensitive dependence on the choice of statistical parameters to be used, which vary from one data set to the next. Also, the best choice for the size of the local region under consideration may vary from one region to the next in an image, and there are no systematic rules to adequately choose this value. Further, these methods tend to trade off speckle suppression for edge preservation.

Methods based on compounding involve the acquisition of multiple images which are averaged to produce an output with increased signal to noise ratio [3]. Spatial compounding involves the use of multiple images of the target structure, taken from slightly different viewpoints. This approach is fully applicable only to relatively few sites of clinical interest [4]. Another form of compounding is frequency compounding, which is based on the assumption that speckle produced at one imaging frequency is not correlated with that produced at other frequencies. This assumption is not always valid [4], and again the method has limited applicability.

Temporal filtering is an alternative approach which shares characteristics with both methods based on compounding and those based on image filtering [2, 8]. The method requires a time sequence of images as input. As in direct image filtering methods, pixels are averaged with their neighbors, but in this case the "neighbors" are chosen in the temporal dimension – from preceding or subsequent frames. In the case where no motion is present, this method is straightforward to apply: an output frame is produced from each input frame by taking, at each pixel, the average of the current pixel and pixels at the same position in previous (or subsequent) frames. Because speckle is largely uncorrelated with time – but image structures are correlated – this can be very effective. Applying this straightforward approach in the presence of motion (e.g. in echocardiograms) results in motion blur and loss of resolution. An alternative approach in that case (echocardiogram sequences) is to average frames taken from the same point in the cardiac cycle [2]. Aligning the frames correctly is difficult to implement in practice, since the clinician may not hold the transducer perfectly still, the patient may move (or breathe), the heart is moving in 3D and may not return exactly to the same position every time it completes a cycle, and the low temporal resolution of the ultrasound device may not allow frames from corresponding times in the cardiac cycle to be procured.

A more sophisticated idea is to use temporal filtering with motion compensation. This is the same idea as temporal filtering, except that motion in the image is taken into account when choosing the pixel in the previous frame that "corresponds" to a given pixel in the current frame. This strategy is the basic ingredient of most algorithms for restoration and enhancement of image sequences [8]. Shinya [9] applied this idea to the problem of anti-aliasing in computer-generated animations. MPEG video encoding uses a similar idea for image compression. As yet, efforts to apply this strategy in echocardiogram sequences have not been successful, mainly because of the difficulty of accurately estimating image motion in the presence of noise, signal dropout, and rapid motion of the heart combined with temporal undersampling [2].

3 A new approach

In this research I propose a new method for automatically locating and tracking features in echocardiogram image sequences. The key idea is to locate and track features at multiple scales in the image. This allows the feature location and tracking phases of the algorithm to work on simplified representations of the image, and it increases the speed and accuracy of the tracking by allowing information to be shared between scales. Next, motion estimates at identified feature points are extrapolated to derive a global motion field defined at every pixel using a scheme similiar to those used in image warping techniques. The global motion estimate is then used in a motion-compensated temporal filter applied to the input echocardiogram image sequence. The goal is to suppress speckle while preserving edges in the images. Achieving this goal will enhance both individual frames (as still images) and the entire sequence itself, since rapidly fluctuating speckle is particularly distracting when an ultrasound image sequence is played back as a cine loop.

Assuming that motion estimates for tracked features are accurate, the idea of using an image warping technique to assign a measure of image motion at every pixel suffers from some potential problems. In the case of a heart valve correctly detected to be moving rapidly away from a stationary heart wall, for example, it's likely that a technique based on image warping will assign some velocity to pixels in the narrow gap between the valve and the heart wall. Applying the motion-compensated temporal filter, these pixels will be deemed to "match" pixels along the heart wall in the previous frame. When the pixel values are averaged, the heart wall will appear to have bulged part-way into the empty cavity, chasing after the rapidly moving heart valve. This could give the illusion of pathological heart wall motion where none exists, rendering the filtering process worse than useless.

Another related problem occurs whenever any major image structure appears or disappears (drops in or out), or is temporarily occluded. Proceeding with temporal filtering in these cases could cause a ghost feature to appear where none should be.

The solution I propose for these cases (and in cases when the tracking algorithm fails in a region) is to modify the temporal filtering algorithm as follows: in areas where little or no motion is detected, the filtering can proceed at full strength. As motion increases, or in cases where the tracking fails, the filtering should operate at reduced levels or not at all. In this respect the filtering algorithm resembles adaptive algorithms such as [6] and [7].

4 Previous work on optical flow and feature tracking

Before proceeding to describe the implementation of my algorithm for estimating global image motion from tracked features, I will address existing methods for calculating image motion ("optical flow") and for tracking features in images.

A number of optical flow algorithms have been developed in the computer vision community. As a class, these algorithms are not suitable for applications such as ultrasound imaging which produce particularly noisy image sequences. Chu et al. [2] note that these algorithms are based on a number of assumptions, including the following:

- the image intensity at a fixed point on a structure does not vary with time or viewpoint.
- locally, all motion can be modelled as rigid body translation.
- velocities vary smoothly across the image

Chu et al. observe that these assumptions are not generally valid for ultrasound image sequences, and hence these types of algorithms do not perform well on that type of data. Camus [10] writes, "The current trend in optical flow research is to stress accuracy under ideal conditions and not to consider computational resource requirements or resistance to noise." In informal tests I conducted using implementations of a number of standard optical flow algorithms made available on the internet by ICCV '94, I found that none computed

the flow for a single frame of an echocardiogram sequence in less than 5 minutes, and all performed poorly.

Approaches which I considered for automatic feature tracking fall into two related categories: active contours, or "snakes," and deformable templates. Snakes were first proposed by Kass et al in 1987 [11]. In their formulation, a snake is a deformable contour which lies in an image and is subject both to image forces and to internal forces. The internal forces cause the contour to resist sharp bending or separation into pieces, and the image forces draw the contour to features of interest. Typically, snakes are designed to be "attracted" to edges, and are used to locate and track features which exhibit strongly defined edges along their boundary. Kass et al claim that snakes are somewhat resistant to noise and local tracking errors, and so are suitable for edge detection and feature tracking in real applications.

Amini et al [12] and Williams and Shah [13] discuss a number of disadvantages of snakes (and propose improvements). Among the disadvantages are instability, reliance on high order derivatives of the image data (which is problematic in the presence of noise), and the need for user intervention when tracking with snakes, both to provide a starting position and occasionally to guide the snake back to a feature when it loses its way.

In spite of these potential problems, Ayache et al [1] and Herlin and Ayache [14] describe a semi-automatic snake-based method for tracking features in echocardiograms. Their method is based on a pre-procesing edge detection step followed by a tracking phase in which a user positions the snake initially, then intervenes occasionally to guide the snake when it loses the feature.

Because any real-time temporal filtering algorithm requires a motion-estimation phase which is fully automatic, I chose to develop an alternative to snake-based feature tracking. I also considered that snakes would be ill suited for dealing with changing image topology, such as when a thin structure (e.g. a heart valve) in one frame appears to be joined to a heart wall, but subsequently separates from the wall and moves away from it. A snake which had identified the valve and wall as a single feature in the first frame would attempt to track the valve away from the wall while also remaining fixed to the wall. In effect, it would attempt to bridge a gap which should not be bridged.

Deformable templates have been proposed as an alternative method for detecting and tracking features in image sequences [15, 18]. A deformable template is a parametric representation (with a small number of parameters) of an idealized feature of the particular type one expects to track. Example applications include detecting or tracking a pedestrian [18], a hand [16], a vertebral trabecular bone [15], or a left ventricle [17]. With this approach, detecting the feature corresponds to searching the parameter space of the parameterized template to find the best match with the image data. Methods for conducting this search include simulated annealing [16] and iterative schemes based on Kalman filters [18] or energy minimization techniques (like snakes) [15].

At first, the idea of detecting and tracking features in echocardiogram image sequences seemed plausible, particularly since there are only a small number of "views" typically used by diagnosticians. It might be possible to tailor the tracking algorithm to "look for" a

particular set of features in a particular configuration in the data. This might be achieved using a deformable template. What's more, prior knowledge of the expected motion could be built into the model, as in [17]. After seeing actual echocardiogram data sets, though, I reconsidered. Even within image sequences taken with the same view, there is a great deal of variability (see Figure 2 and Figure 3). Designing a template which could automatically detect and track all features in any data set of a given type seems extremely hard.

5 Feature detection

Tracking features based on image intensity is preferable in noisy data sets to tracking based on detected edges, because the latter is more sensitive to noise. Many previous attempts to detect and track features in ultrasound image sequences have been based on detecting edges [1, 2, 14, 17, 19]. This may be due to specific objectives (precise measurement of tissue structure dimensions), or simply because much of the computer vision literature focuses on detecting edges (in photographic and video image data). Since in this application an accurate measurement of feature displacement between frames is needed rather a precise measure of feature dimensions, I chose to consider features to be localized regions of high grayscale intensity – usually either point-like "peaks" or long thin "ridge" structures.

The first step in my approach for automatic feature detection is to low-pass filter the images and to represent them explicitly at multiple scales with reduced resolution. (See Figure 4, Figure 5, and Figure 6). There are several advantages to this. For one, at coarse scales the image data is simpler to analyze because 1) the effects of noise are greatly reduced, and 2) at reduced resolution, large features are typically only several "pixels" across, rather than dozens or more at full resolution. This means that detecting (and tracking) a large-scale feature is easier, because it involves sifting through less data. This approach also takes into account the natural range of features occuring in the data, from large-scale chamber walls to valves to tiny thread-like papillary structures. In the tracking phase, the multi-scale representation allows large-scale features detected at low resolutions to pass on their motion information to associated small-scale features at higher resolutions.

Generating the reduced-resolution images for each frame is straightforward. First, the input image is low-pass filtered. Next, a half resolution image is created by forming one pixel from each block of four in the input image. This is done simply by averaging the four pixels. Next, a quarter-resolution image is formed from the half resolution image in the same way (recursively). Finally an eighth-resolution image is formed from the quarter resolution image [fig.]. Storing these additional images, together with the original, takes no more than 4/3 n, where n is the size of the original image. Thus, computing the reduced resolution images is economical in both space and time. It's also simple to implement.

The next step in detecting image features is to detect local maxima in image intensity. This is straightforward. An efficient way to accomplish this is to choose regularly spaced starting positions from which to initiate a "hill-climbing" search for nearby local maxima. The positions of these local maxima are then recorded.

The next step is to classify the local maxima according to their "shape." For example, a local maximum which is part of a broad, flat image region of nearly constant intensity should be classified differently than a local maximum which constitutes a localized bright point in the image. Such a classification is useful because in the latter case, a simple tracking strategy based on re-finding the local maximum is probably sufficient. In the former case such a simple strategy would be overly sensitive to noise. Similarly, a local maximum which is part of a long ridg-like shape should be treated differently (for tracking purposes) from one corresponding to a small round feature. Also, faint local maxima stand more chance of corresponding to noise and should not be tracked as persistently as strong local maxima.

Other authors have proposed methods for characterizing the shape of a gray-scale image at a local maximum [20, 22]. Selmaoui et al [20] propose a scheme based on classifying a pixel according to rules applied to the 3x3 window surrounding it. Gauch and Pizer [21] and Eberly et al [22] use concepts from differential geometry. In their approaches, the image intensity values can be thought of as describing a surface in 3D. Each local maximum can then be characterized by the principal curvatures and directions at that point. At a ridge, one principal curvature is large (and negative) and the other is nearly zero. The principal directions give the orientation of the ridge. At a round peak, both principal curvatures are are nearly equal (and negative). Methods exist from differential geometry for calculating curvature information at a point, given first and second order derivatives. See, for example, [23]. Gauch and Pizer estimate derivative information by first fitting a spline surface to the image data. Derivatives of the spline surface can be readily calculated. They apply this method to images with realively little noise, such as medical MRI images and fingerprints.

My approach is also based on deriving curvature information from the image "surface." Rather than explicitly calculate image derivatives, though, I use the following technique. At each local maximum p, the nearby surface is approximated with a degree 2 polynomial. This polynomial has the form

$$z = Au^2 + Buv + Cv^2 + Du + Ev + F$$

where u and v are the coordinates of p. Since p represents a local maximum, this can be simplified to:

$$z = Au^2 + Buv + Cv^2 + F$$

If u and v are expressed in local coordinates, then F is the grey-scale value at p.

The next step is to find the coefficients A, B, and C. To find the best values for these a least squares approach is used. In a 5x5 window around p one can write 24 equations in the 3 unknowns. In matrix notation this has the form:

$$M\mathbf{a} = \mathbf{z}$$

where M is a 24x3 matrix, $\mathbf{a}$ is a 3x1 vector of the unknown coefficients, and $\mathbf{z}$ is a 24x1

vector of image grey-scale values (with F subtracted from each). The best value for **a** is found in the least squares sense by the following:

$$\mathbf{a} = (M^T M)^{-1} M^T \mathbf{z}.$$

An important point to be made is that the matrix M is constant. As a result, the matrix $(M^T M)^{-1} M^T$ is constant and can be precomputed and stored. This means that solving for the coefficients A, B, and C is as easy as ... I mean, is efficient. (It involves the multiplication of a 3x24 matrix and a 24x1 vector).

Given the degree 2 polynomial approximating the surface locally at p, the principal curvatures and and directions can be found without resort to explicit derivatives, using techniques from analytic geometry [24]. The basic idea is that the iso-lines of the degree 2 polynomial surface around p form concentric ellipses (since p is a local maximum). The major and minor axes of any one of the ellipses are the principal directions of curvature, and the principal curvatures can be derived directly from the equation for the ellipse once it is expressed in the coordinate system of its major and minor axes.

Once curvature information has been found for a local maximum, the corresponding feature can be classified as a "peak" or a "ridge" (or discarded if it is too weak). Ridge points are then extended by iteratively adding points at either end along the 2nd principal direction (direction of minimum curvature), and recalculating the curvature information, until newly added points fail to satisfy some ridge criteria. The criterion I use is typically that the first principal curvature should be sufficiently big, and that the ratio of the first principal curvature to the second should exceed some value such as 1.5 (which can be adjusted). This ensures that the surface at each point along the "ridge" is sufficiently ridge-like. Figure [] shows features found with this technique.

6 Feature tracking

For several reasons I decided to pursue a strategy of detecting features from scratch at each frame, then tracking them only through the subsequent frame. I chose this approach in order to be able to deal with changing feature topology, as when a new featue "drops in" or an apparently single feature separates into two. Another reason is that the feature detection algorithm is fast and effective. The drawback to this approach is that it does not take advantage of temporal coherence in feature motion, which could possibly help avoid the effects of noise.

The feature tracking methods I use are ad hoc and could no doubt be improved. Nonetheless, they give usable results. Tracking begins at the coarsest resolution, and proceeds to finer resolutions. At each stage, motion information at the previous resolution is used to guide features within the current resolution. This speeds tracking by allowing features to use a smaller search space. It also means, though, that mistakes made at a coarse resolution are

propagated to finer resolutions. For this reason it is important that at coarse resolutions, feature tracking provides sub-pixel motion information.

The basic idea is to conduct the tracking on a case by case basis, according to the type of feature (peak or ridge) and current image resolution.

The simplest case is for peak features at half resolution (the highest, since I don't bother detecting or tracking features at full resolution). In that case the feature simply recenters itself at the pixel with maximum grey value in its vicinity.

At quarter-resolution and especially at eighth-resolution this is not a good strategy because it is subject to noise and does not provide sub-pixel information. To get sub-pixel information, I used the idea of "center of mass," where grey levels in the image are thought of as slots bearing mass proportional to their grey level. The feature computes the center of mass in a local region around it, first in the current frame, then in the subsequent frame. The difference between these center of mass measurements gives an indication of the image motion near the feature (provided measurements are made near a well-defined intensity maximum and the motion is not too great).

For ridge features, the basic idea is to use the curvature information near the ridge to make the ridge points first climb "sideways" to re-center themselves on the ridge, then to shift longways (if necessary) to account for motion along the ridge's length. Climbing sideways is straightforward and can be based on the same strategies as that applied to peaks (but in a restriceted direction). Shifting longways seems to be a good idea for relatively short ridges; it can be implemented by using a representative point (e.g. the point with maximum grey level intensity, usually near the center of the ridge). This point can use the technique for tracking peak points to detect its motion, then project that motion along the second principal direction of curvature (along the ridge), and pass this displacement to all the ridge points equally. An additional step of regularization is useful to prevent the ridge feature from falling apart or reacting too sensitively to noise. The drawback to this is that there are cases where the ridge feature should separate into pieces (discussed earlier). Additional tests can be made to check for this case (though I didn't implement that). At any rate, an advantage to starting the tracking from scratch at each frame is that tracking errors are not propagated beyond a given frame.

7 The temporal filter

Implementing a motion-compensated temporal filter is relatively straightforward, given tracking information at detected features. I'll briefly address several of the issues.

First, it's necessary to derive a global motion field which is defined at every pixel in the image. One way to to this is to take, at each pixel, a weighted sum of the motions of nearby feature points. The weighting typically is based on distance, so that nearby feature points contribute more heavily than distant points.

Secondly, in order to make the first step efficient, the operation of finding nearby feature

points at a given pixel must be fast. I used a scheme based on uniform spatial subdivision [25], in which feature points record their id's in "buckets" of dimension 4x4 pixels in a radius R around the feature point. When two feature points from the same feature record their id's in the same bucket, the one closest to the bucket overwrites the other one. This operation is bounded by fR^2, where f is the number of feature points and R is their radius of influence. During filtering, each pixel can check its corresponding bucket to get the list of feature points within its range. This takes time proportional to the number of nearby features (generally a small number).

Lastly, there are (at least) two ways to compute the temporal filter. One is to use a box filter, in which the output grey value at a pixel p is a weighted average of grey values of pixels corresponding to p (taking motion into account) over a window of k previous frames, where k is a small number (e.g. 3 or 4).

An alternative is to use a geometric filter in which the output at a pixel p is the weighted sum of the grey value at location p in the current frame and at the corresponding pixel in the previous *output frame*. Since the previous output frame contains averages of frames prior to it, this effectively calculates a weighted sum of grey values over all previous frames (in which the weights decrease geometrically).

One advantage to this approach is that it is easy to implement and allows the weighting factor at each pixel to be adjusted adaptively according to the output of the tracking phase. For example, to turn off filtering at a pixel, weights of 1 and 0 can be used for averaging grey-values in the current input and prior output frames, respectively. Originally, I tried an approach in which I turned off filtering at any pixel at which the estimated motion exceeded a certain threshhold. This had the side effect of causing distracting sudden changes in the output image sequences in some cases, when filtering abruptly turned on and off in local regions. I modified my approach to turn the filtering down as the estimated motion approaches the threshhold. This eliminated the distracting effects of the filter turning on and off abruptly. Implementing this kind of adaptivity would be more difficult using a box filter approach.

8 Results

Figures 10, 11, and 12 show one frame from an echocardiogram sequence. The first is the original frame. The second shows the results of applying temporal filtering without motion compensation. The image contains significant motion blur. The last figure shows the results of applying motion-compensated temporal filtering. Note the relative absence of motion blur. This is due in part to the motion compensation itself (i.e., matching a pixel in the current frame with the corresponding pixel in the previous frame according to the estimated motion vector at the current pixel), but is also due to the absence of filtering: the filter turned off near the rapidly moving thin heart valve. In both cases, this demonstrates the worth of the motion estimates provided by the tracking phase of the algorithm. This data set is somewhat unusual in that the image quality is particularly clear; nonetheless the image

produced by the motion-compensated temporal filtering algorithm does exhibit less speckle. This is more clearly seen when the image is in motion, since speckle is more noticeable when it flickers distractingly during playback of a cine loop.

Figures 13, 14, and 15 show a single frame from another (noisier) image sequence. The same observations hold: the image produced by temporal filtering without motion compensation exhibits clear signs of motion blur. The image produced with motion-compensated filtering has much less blur, but maintains a good degree of speckle reduction. Again, this is particularly apparent when the images are viewed in motion.

The tracking and filtering phases of the motion-compensated temporal filtering algorithm run at 4 to 5 frames per second on a Sun Sparcstation10/50. This does not count the time to convert the input images to their lower resolution representations. This processing rate, while not real-time, is at least in the ballpark and could be improved with a more efficient implementation, a faster computer, or by taking advantage of multiple processors.

9 Future work

I plan to develop a more rigorous (and reliable) tracking algorithm, wich takes advantage of temporal coherence. Some version of "snakes" may turn out to be applicable here. Also, more testing needs to be done, and more rigorous evaluation of the results of the temporal filter. A pipelined, multi-processor implementation may turn out to give the needed performance boost to make the filtering algorithm run in real time.

10 Acknowledgments

Thanks in particular to John Hughes, who suggested the idea for applying a motion-compensated temporal filter to ultrasound image sequences, and who gave me lots of ideas during the course of this work. Also thanks to Andy van Dam and the Brown Computer Graphics group for their support. And to Gillis Kallem!

References

[1] N. Ayache, I. Cohen, and I. Herlin. Medical image tracking. In A. Blake and A. Yuille, editor, Active Vision, pp. 285-301. MIT Press, 1992.

[2] C.H. Chu, E.J. Delp, and A.J. Buda. Detecting left ventricular endocardial and epicardial boundaries by digital two-dimensional echocardiography. IEEE Transactions on Medical Imaging, 7(2):81-90, 1988.

[3] A.N. Evans and M.S. Nixon. Temporal speckle reduction for feature extraction in ultrasound images. Computer analysis of images and patterns : 5th international conference, CAIP '93.

[4] F. Forsberg, A.J. Healy, S. Leeman and J.A. Jensen. Assessment of hybrid speckle reduction algorithms. Phys. Med. Biol. 36(11):1539-1549. 1991.

[5] J.I. Koo and S.B. Park. Speckle reduction with edge preservation in medical ultrasonic images using a homogeneous region growing mean filter (HRGMF). Ultrasonic Imaging 13:211-237. 1991.

[6] J.C. Bamber and C. Daft. Adaptive filtering for reduction of speckle in ultrasonic pulse-echo images. Ultrasonics 24:41-44. 1986.

[7] T. Loupas, W.N. McDicken and P.L. Allan. An adaptive weighted median filter for speckle suppression in medical ultrasound images. IEEE Transactions on Circuits and Systems, 36(1):129-135. 1989.

[8] S. Geman, D.E. McClure and D. Geman. A nonlinear filter for film restoration and other problems in image processing. CVGIP: Graphicsl Models and Image Processing 54(4):281-289. 1992.

[9] M. Shinya. Spatial anti-aliasing for animation sequences with spatio-temporal filtering. Proceedings of SIGGRAPH, '93 (Anaheim, California, August 1-6).

[10] T. Camus. Real-time optical flow. Ph.D. thesis, Department of Computer Science, Brown University, 1994.

[11] M. Kass, A. Witken and D. Terzopolous. Snakes: active contour models. Proceedings of First International Conference on Computer Vision, London, 1987, pp. 259-269.

[12] A.A. Amini, S. Tehrani and T.E. Weymouth. Using dynamic programming for minimizing the energy of active contours in the presence of hard constraints. Proceedings, Second International Conference on Computer Vision, 1988, pp. 95-99.

[13] D.J. Williams and M. Shah. A fast algorithm for active contours and curvature estimation. CVGIP: Image Understanding 55(1):14-26. 1992.

[14] I.L. Herlin and N. Ayache. Features extraction and analysis methods for sequences of ultrasound images. Computer vision–ECCV '92 : Second European Conference on Computer Vision, May 1992. pp. 43-57.

[15] P. Lispon, A.L. Yuille, D. O'Keefe, J. Cavanaugh, J. Taaffe and D. Rosenthal. Deformable templates for feature extraction from medical images. Computer vision–ECCV 90 : First European Conference on Computer Vision, April 23-27, 1990. pp. 413-417.

[16] A. Bennett and I. Craw. Finding image features using deformable templates and detailed prior statistical knowledge. BMVC 91 : proceedings of the British Machine Vision Conference, September 24-26, 1991. pp. 233-239.

[17] J.C. McEachen II, F.G. Meyer, R.T. Constable, A. Nehorai and J.S. Duncan. A recursive filter for phase velocity assisted shape-based tracking of cardiac non-rigid motion. Online publication: http://noodle.med.yale.edu/mceachen/iccv/iccv.ps.gz

[18] A.M. Baumberg and D.C. Hogg. An efficient method for contour tracking using active shape models. University of Leeds, School of Computer Studies, Tech Report 94.11. April 1994.

[19] L.D. Cohen. Note on active contour models and balloons. CVGIP: Image Understanding 53(2):211-218. 1991.

[20] N. Selmaoui, C. Leschi and H. Emptoz. Crest lines detection in grey level images: studies of different approaches and proposition of a new one. Computer analysis of images and patterns : 5th international conference, CAIP '93.

[21] J.M. Gauch and S.M. Pizer. Multiresolution analysis of ridges and valleys in grey-scale images. IEEE Transactions on Pattern Analysis and Machine Intelligence 15(6):635-646. 1993.

[22] D. Eberly, R. Gardner, B. Morse, S. Pizer and C. Scharlach. Ridges for image analysis. University of North Carolina, Department of Computer Science, Tech Report 93-055. 1993.

[23] M.P. do Carmo. Differential geometry of curves and surfaces. Prentice-Hall, Inc. 1976.

[24] P. Foerster. Pre-calculus with trigonometry. Addison-Wesley. 1991.

[25] G. Turk. Interactive collision detection for molecular graphics. M.Sc. thesis, Department of Computer Science, University of North Carolina. 1989.

11 Figures

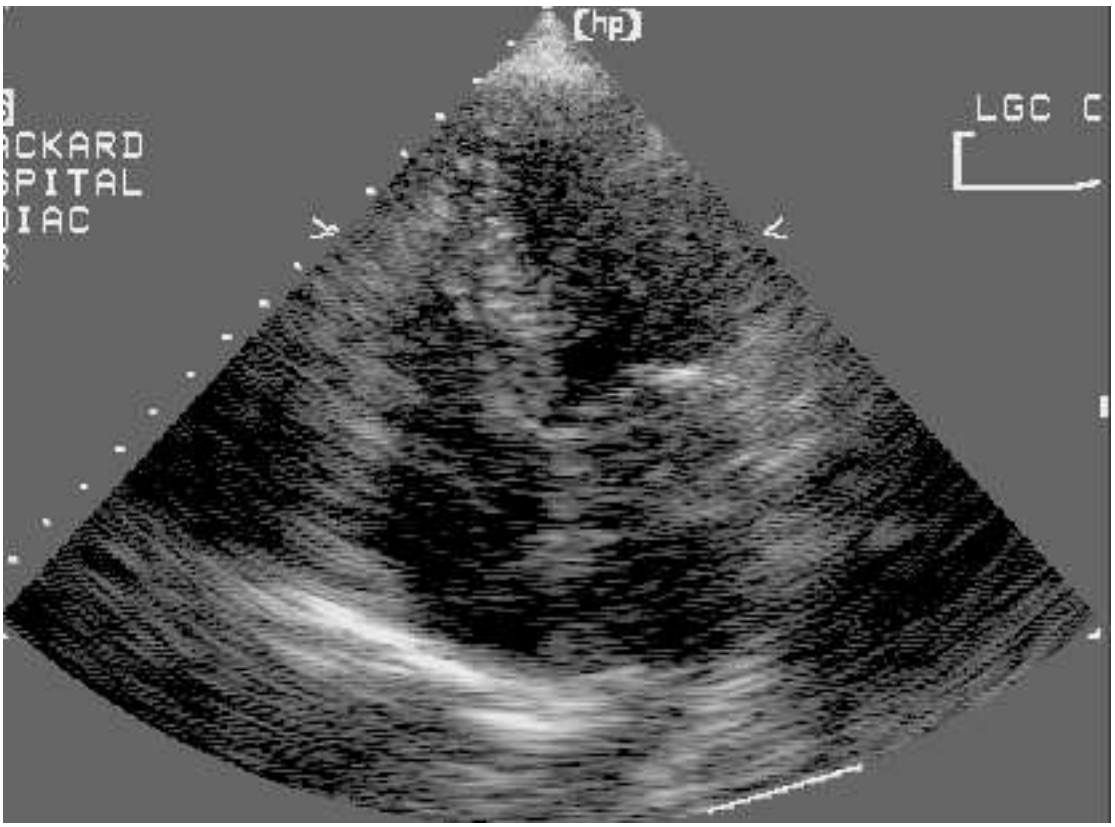

Figure 1: Noisy echocardiogram (apical four chamber view)

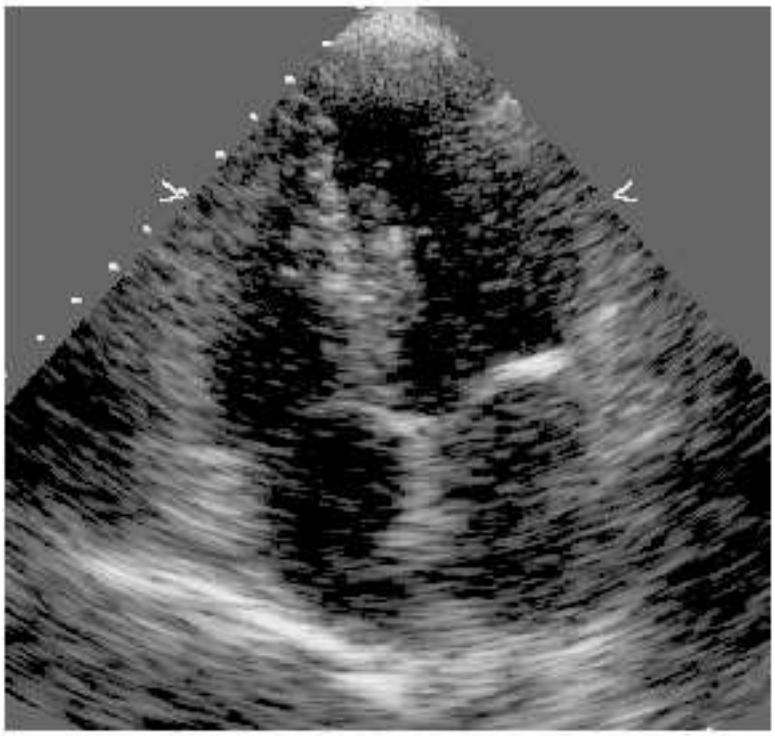

Figure 2: Apical four chamber view.

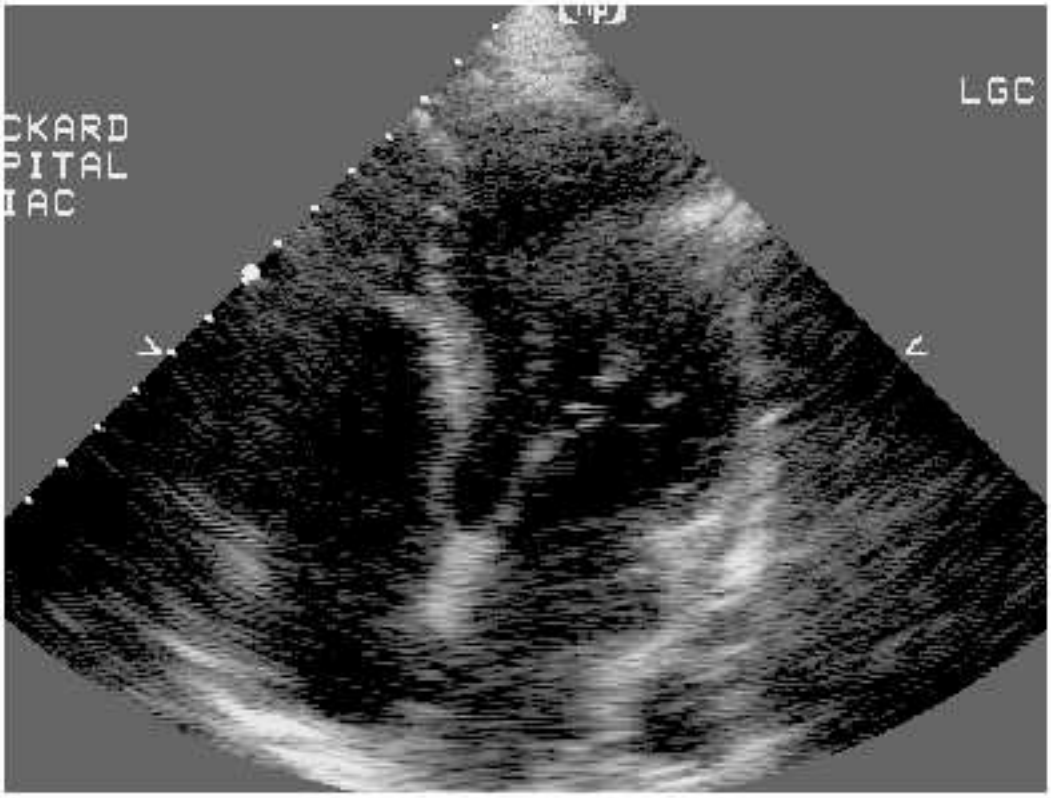

Figure 3: Apical four chamber view.

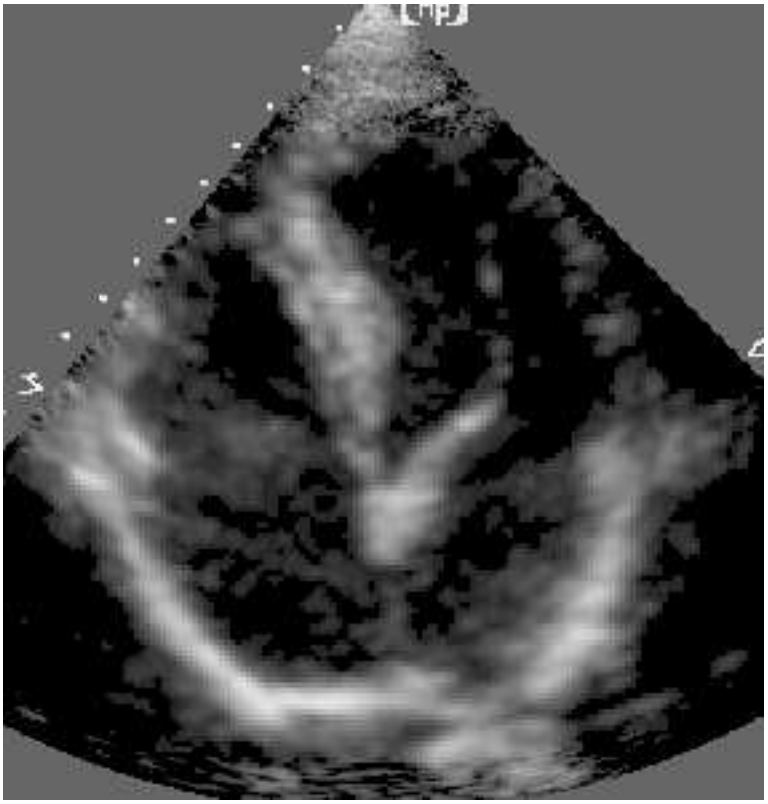

Figure 4: Half resolution.

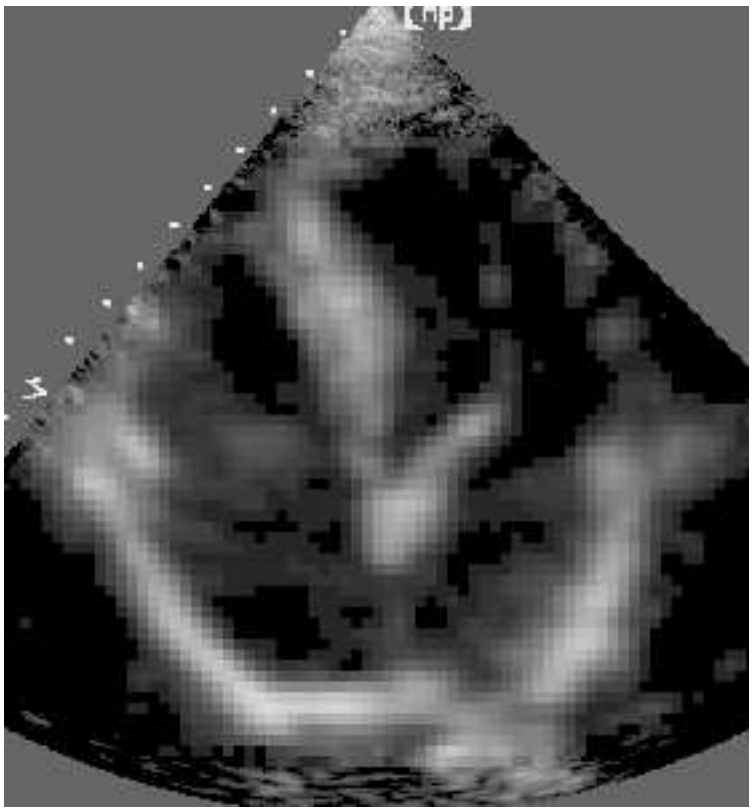

Figure 5: Quarter resolution.

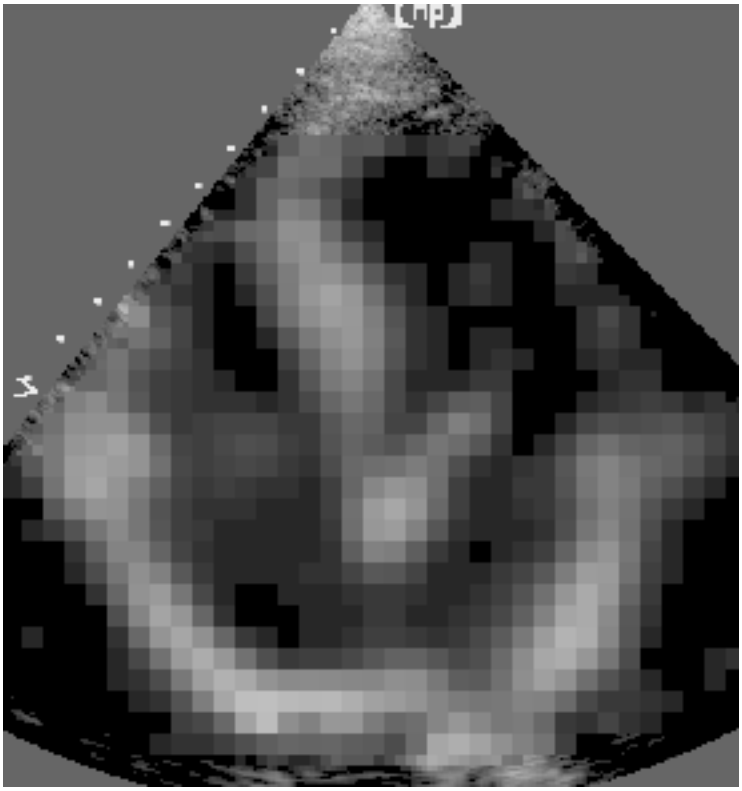

Figure 6: Eighth resolution.

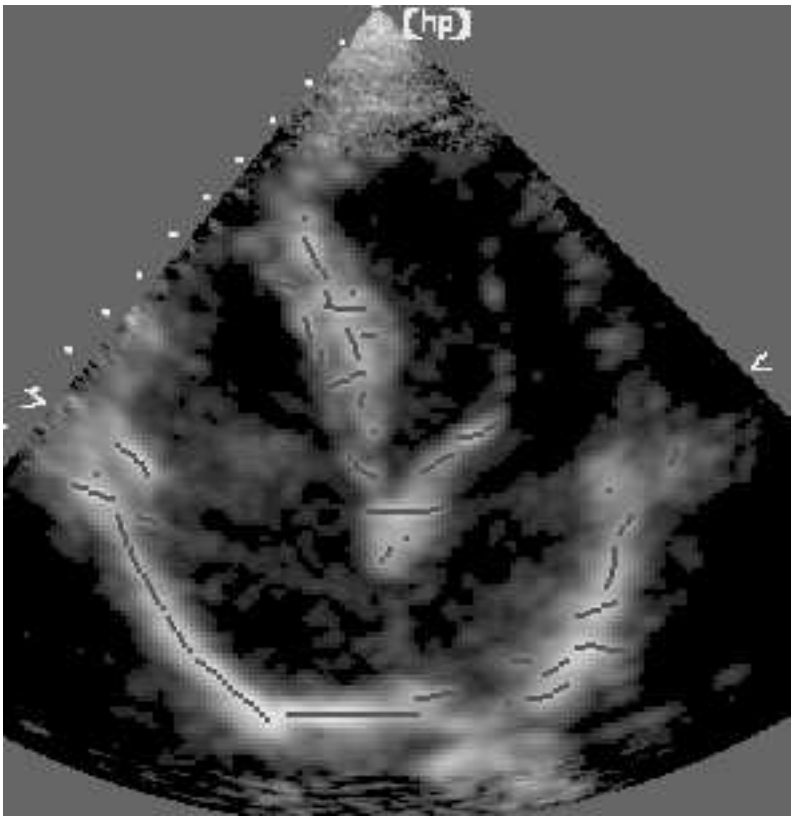

Figure 7: Half resolution with features.

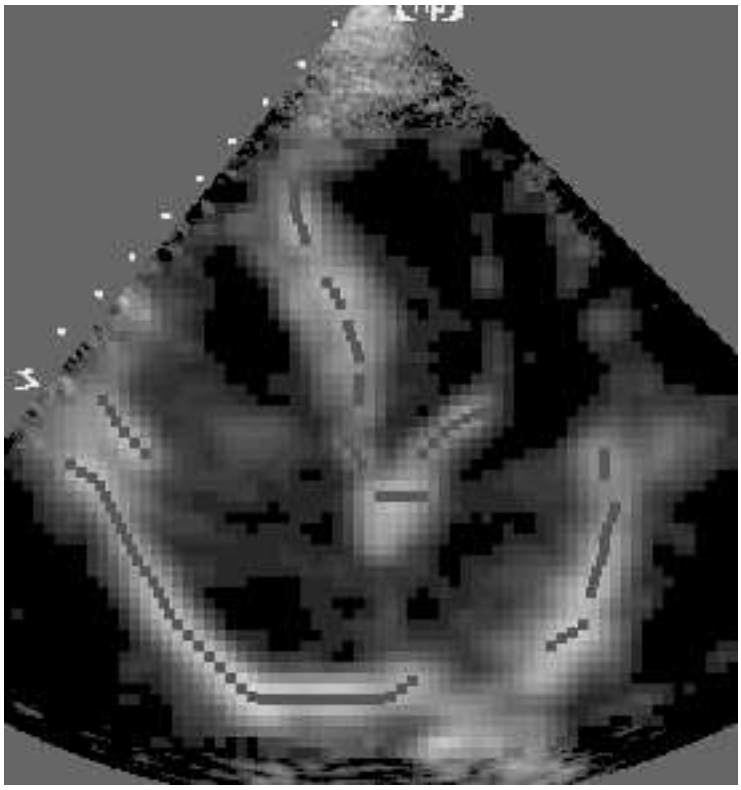

Figure 8: Quarter resolution with features.

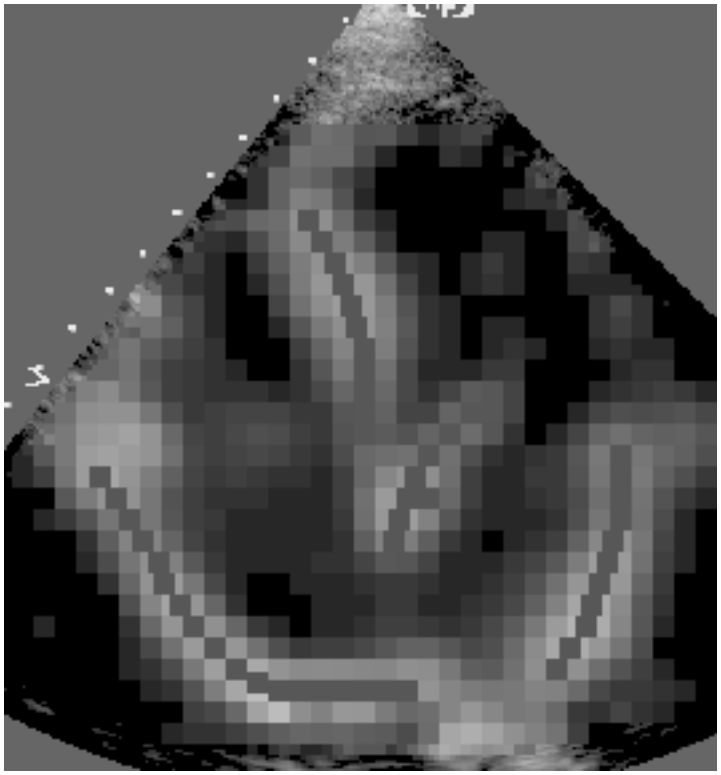

Figure 9: Eighth resolution with features.

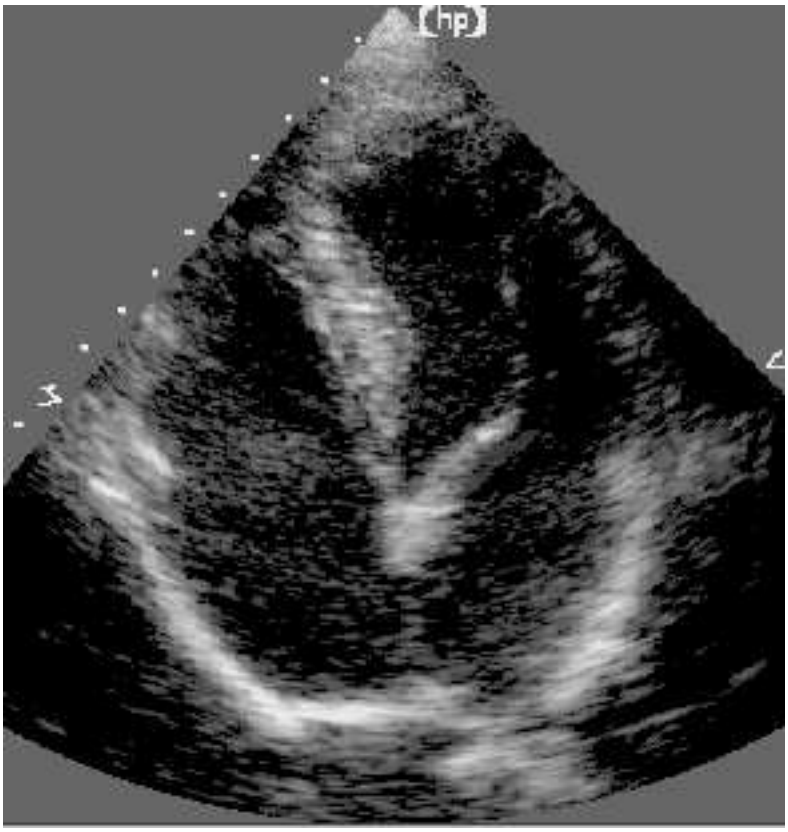

Figure 10: Original frame.

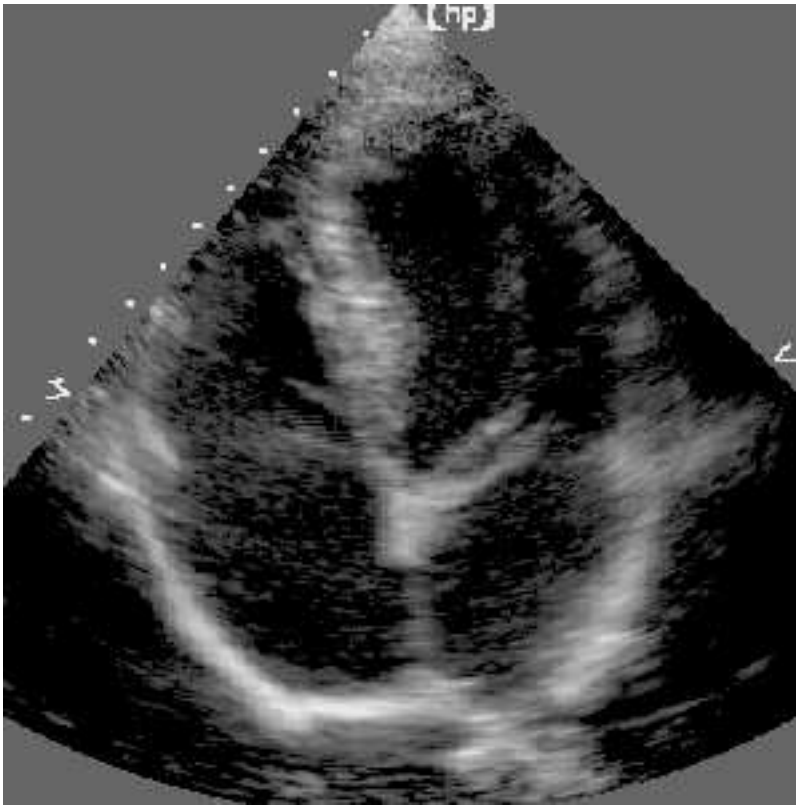

Figure 11: Temporal filter without motion compensation.

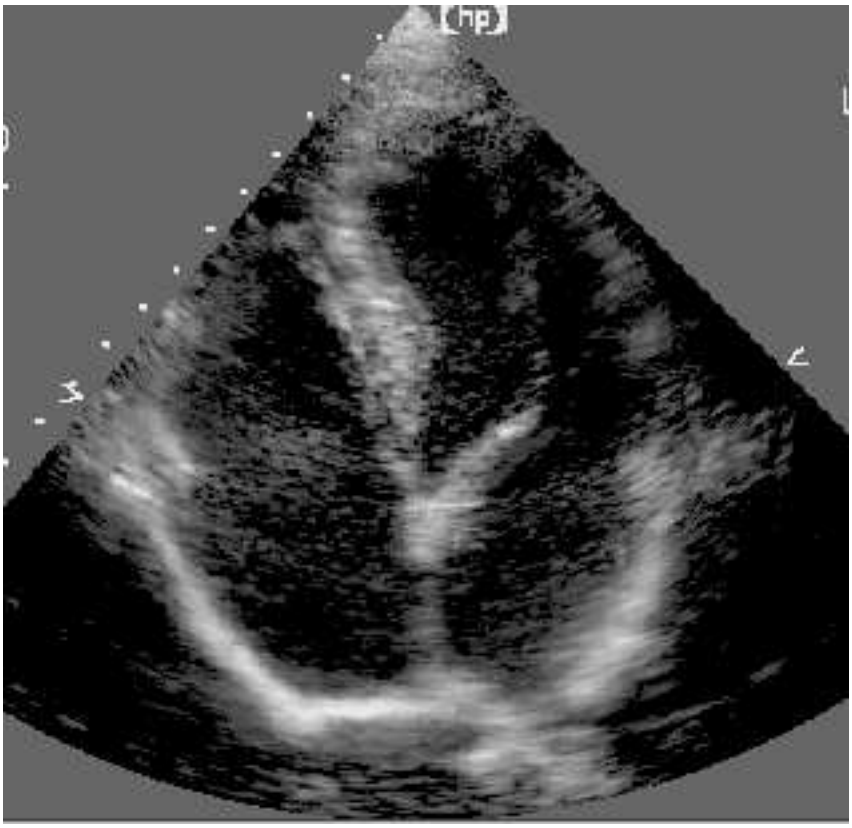

Figure 12: Temporal filter with motion compensation.

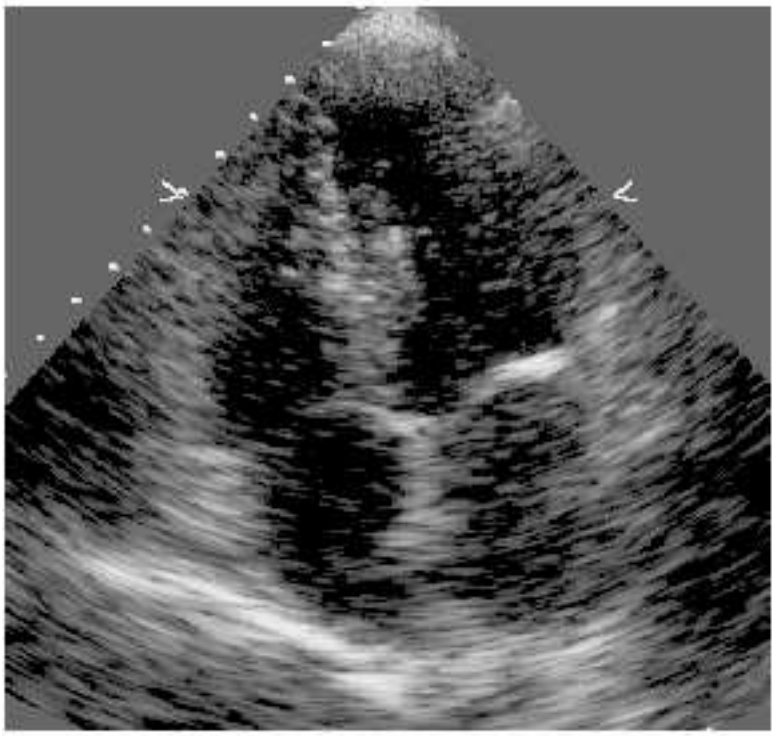

Figure 13: Original frame.

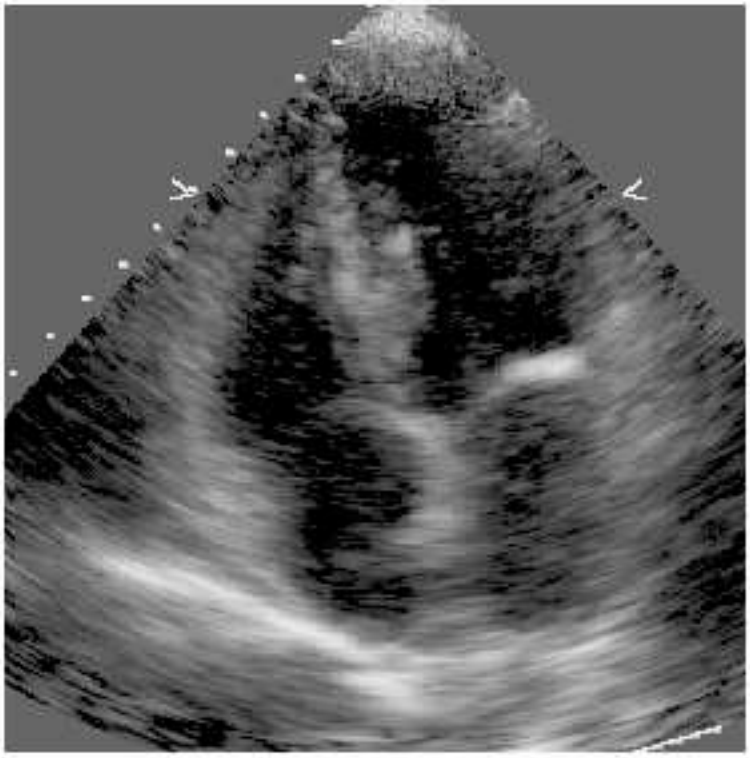

Figure 14: Temporal filter without motion compensation.

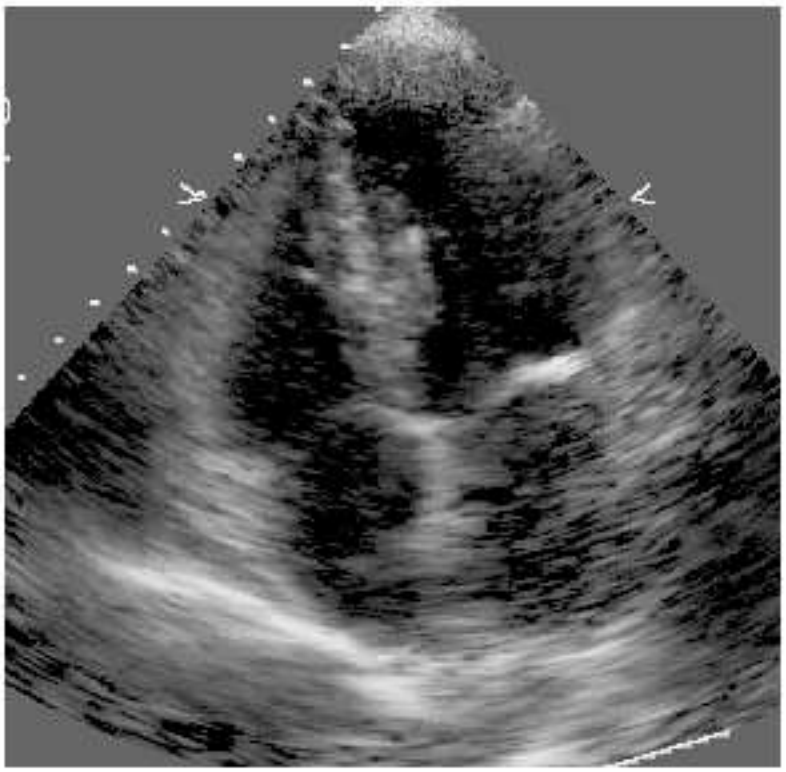

Figure 15: Temporal filter with motion compensation.