

# Detecting Particles in Cryo-EM Micrographs using Learned Features

Satya P. Mallick<sup>1</sup>      Yuanxin Zhu<sup>2</sup>      David Kriegman<sup>3</sup>

<sup>1</sup> Department of Electrical and Computer Engineering,  
University of California at San Diego, La Jolla, CA 92093, USA

<sup>2</sup>Center for Integrative Molecular Biosciences and Department of Cell Biology,  
The Scripps Research Institute, La Jolla, CA 92037, USA

<sup>3</sup> Department of Computer Science and Engineering,  
University of California at San Diego, La Jolla, CA 92093, USA

December 9, 2003

## Abstract

A new learning-based approach is presented for particle detection in cryo-electron micrographs using the Adaboost learning algorithm. The approach builds directly on the successful detectors developed for the domain of face detection. It is a discriminative algorithm which learns important features of the particle's appearance using a set of training examples of the particles and a set of images that do not contain particles. The algorithm is fast (10 seconds on a 1.3 GHz Pentium M processor), is generic, and is not limited to any particular shape or size of the particle to be detected. The method has been evaluated on a publicly available dataset of 82 cryo-EM images of keyhole lympet hemocyanin (KLH). From 998 automatically extracted particle images, the 3-D structure of KLH has been reconstructed at a resolution of 23.2Å which is the same resolution as obtained using particles manually selected by a trained user.

**Keywords:** Particle detection, Adaboost, learning, cryo-electron microscopy.

# 1 Introduction

As part of the process of reconstructing the 3-D electron density maps of particles (particularly protein macromolecules) from cryo-electron microscope (cryo-EM) images, there is the need to identify the image locations corresponding to the projection of copies of the same type of particle. Due to the low signal to noise ratio in cryo-EM images and the desire for atomic resolution reconstruction on the order of 4Å, it has been estimated that the images of a million particles will be required [7, 9]. In order for such reconstructions to be routine, it will be necessary for particles to be automatically detected (picked) in images. A wide variety of methods have been proposed for particle picking. Nicholson and Glaeser give a good review of different algorithms [14], and they have been broadly classified into the following:

1. Template-based methods.
2. Edge detection-based methods.
3. Intensity comparison methods.
4. Texture-based methods.
5. Neural network methods.

We would like to add to the above list a new class of techniques which have been successfully applied in other imaging domain, but have not yet been applied to the problem of detecting particles, namely *learning based-methods*. A supervised learning-based approach is one in which the algorithm is trained on a set of example images of particles and possibly images of non-particles. Once the learning-based algorithm is trained, it is then possible to scan a detection window over an input micrograph and classify each window as to whether or not it contains a particle. Broadly, classifiers can be divided into

1. Generative classifiers: A generative approach requires examples of particles only.
2. Discriminative classifiers: A discriminative approach requires examples of both particles and non-particles (See for example, Fig. 1).

An approach using a neural network is an example of a learning-based approach, but, as pointed out in [14], it may be computationally expensive. In this paper, we propose a fast learning-based discriminative approach inspired by the success of the algorithm by Viola and Jones [24][25] for human face detection. There are many other domains for which the problem of 3-D object detection in images arises, and face detection, which often serves a precursor to face recognition, has been very heavily studied; a recent survey [26] cites over 180 papers, and the most successful approaches (lowest error rates) are learning-based. Consequently, our aim in this paper is to explore the use for particle picking of one of the best learning-based face detection algorithms since it has the virtues

of having one of the lowest error rates on standard data sets and is the fastest of the accurate methods [25].

In our opinion an ideal particle detection algorithm should have the following characteristics:

1. It should be generic enough to allow detection of particles of any shape or size using a single algorithm, i.e., it should work well on spherical particles, filaments, asymmetric particles, as well those with other symmetries.
2. It should have high specificity for the image of the particles on which it is trained to detect and reject other “particle-like” objects. For example, if it is trained to detect the side view of KLH, it should detect only the side view and not the top view.
3. The parameters used in the algorithm should be automatically determined.
4. It should be insensitive to differences in the imaging conditions (contrast, bias, signal to noise ratio) between training and actual use.

We believe that the first characteristic is critical in the sense that it would be practically impossible and probably unnecessary to develop a different detection technique for each type of particle. Many particle detection algorithms are effective at detecting symmetric particles, spherical particles with circular images in particular [13, 23]. These methods do well on a particular class of particles, however, they do not present a generic solution. Another popular class of methods is the template-based approach [4, 19, 23]. Template-based methods present a single algorithm to detect particles of different shapes and sizes. Such approaches can be considered as generative learning-based approaches since the templates are based only on the particles to be detected. Often a threshold on the correlation result must be manually selected to determine a tradeoff between the false-positive rate and false negative rate. As there is no model for non-particles, one might expect a high false-positive rate when there are non-particles that happen to look somewhat similar actual particles. Another approach deals with denoising the image first with the hope that detection becomes much easier in denoised image [11]. We however use the noisy images themselves to train our classifier and expect that the classifier also learns a discriminator in spite of the noise. Other important approaches are by Bern, Hall, Ludtke, Roseman, Sigworth, Bajaj, Zhu, Penczek and Volkmann [27].

## 2 Overview of the Approach

In this section, we present a broad overview of the approach which can be divided into an off-line learning phase followed by on-line particle detection. The result of the learning phase is to produce a two-category classifier which takes as input a window of a digital micrograph (e.g., a 50 by 50 pixel sub-image) and classifies it as either containing a particle or not containing a particle. During on-line detection, a detection window is scanned over an input micrograph, and for each location

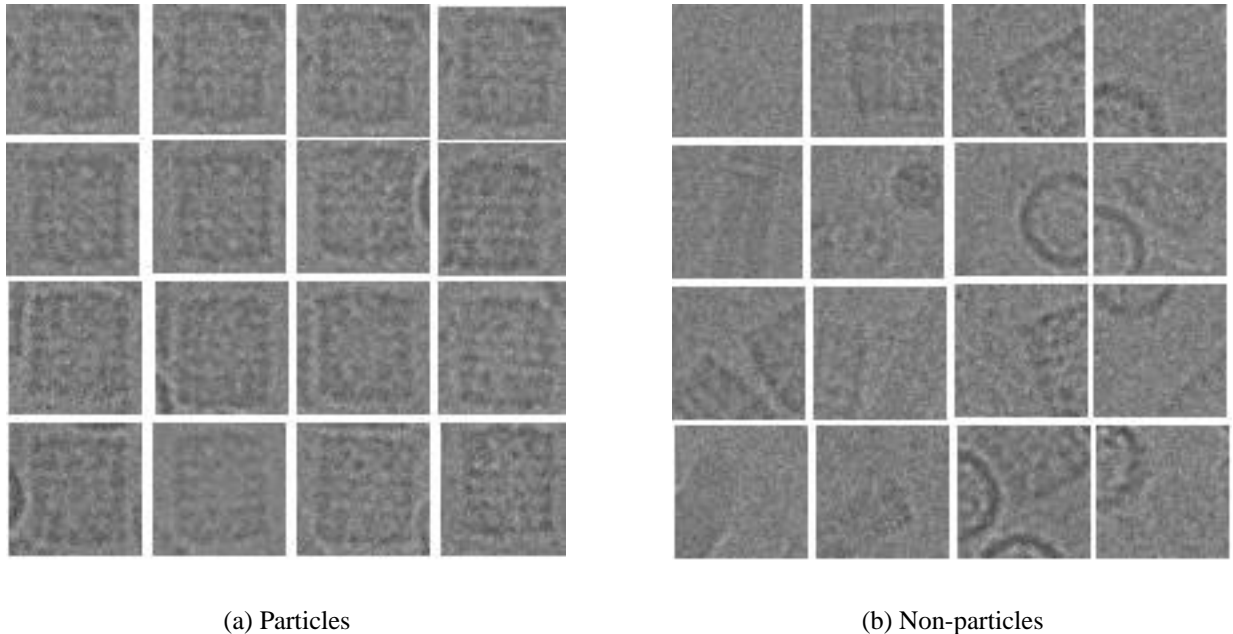


Figure 1: Cropped and aligned images of Keyhole Lympet Hemocyanin (KLH) particles and sample images of non-particles.

(pixel), the sub-image covered by the window centered at that location is classified as particle/non-particle. If the detector is trained to only detect particles in a particular 2-D orientation in the image plane (e.g., particles shown in Fig. 1.a) while particles in a micrograph may appear at any orientation, then the detector is scanned multiple times. During each scan, particles in a particular orientation are detected; either the detector is “rotated” with each scan, or else the detector is fixed, but the image is rotated.

While the on-line detection process is rather standard, the key is the learning phase which produces an effective classifier. The learning process is given a training set  $\mathcal{I}_p$  of images of particles and a set  $\mathcal{I}_{np}$  of images that are not particles. Each image in the training set is a window cropped from a micrograph (in our case, the windows are 50 by 50 pixel subimages of a 512 by 512 pixel subsampled micrograph). Figure 1 shows a set of 16 particle images and a set of 16 non-particle images. Within  $\mathcal{I}_p$ , the images are rotated so the particles assume a particular orientation. For KLH in Fig. 1, the rectangular particles are aligned with the sides of the window. The set of non-particle images should contain both a representative set of cropped subwindows (e.g., images just containing ice) as well as images that might be expected to confuse the classifier (e.g., deformed or broken particles, particles in undesirable 3-D orientations, overlapping particles, particles that are too close together to be useful for reconstruction, etc.).

Our detector and its training follow closely that of [25] in that the detector is composed of multiple stages, called a cascade, of two-category classifiers (See Sec. 5 for a description of the cascade). In turn, classifier  $F(t)$  at each stage is composed of a linear combination of  $K$  simple classifiers.

Let  $f_k(t)$  where  $k \in \{1, \dots, K\}$  denote such a simple classifier;  $f_k(t)$  is taken to be a function which takes a cropped window of an image  $t$  as input and produces either  $+1$  or  $-1$  as an output. The classifier  $F(t)$  for a stage of the cascade is then determined from a linear combination of these classifiers

$$F(t) = \begin{cases} 1 & \sum_{k=1}^K \alpha_k f_k(t) \geq \mathcal{T} \sum_{k=1}^K \alpha_k \\ -1 & \text{Otherwise} \end{cases} \quad (1)$$

where  $\alpha = (\alpha_1, \dots, \alpha_K)$  is a vector of weights determining the contribution (reliability) of each classifier to the final decision, and  $\mathcal{T}$  is a threshold that determines the tradeoff between false-negative and false-positive rates.

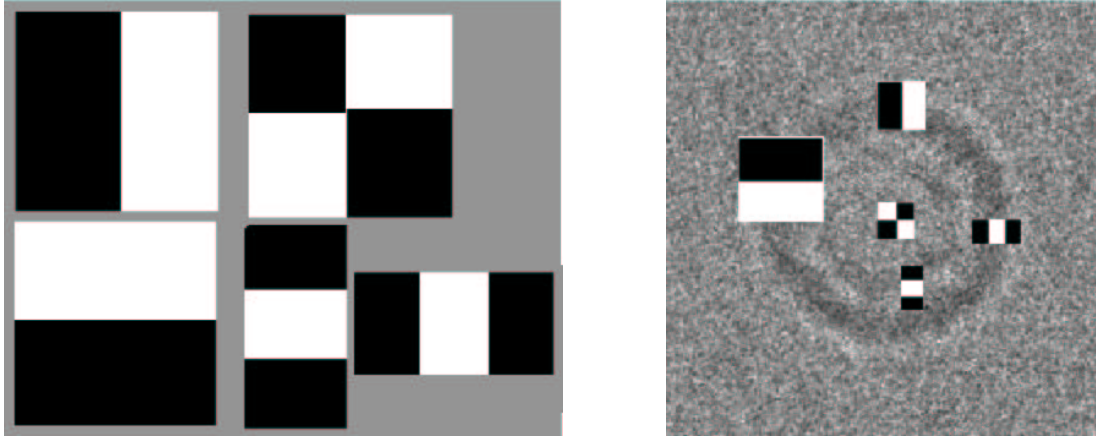
The training phase then amounts to determining the functions  $f_k(t)$  and the vector of weights  $\alpha$  using the training sets  $\mathcal{I}_p$  and  $\mathcal{I}_{np}$ . The approach taken here and also in [25] is to generate a very large number of potentially useful classifiers (also known as features of the image) and select from this candidate set the  $K$  most effective ones using the Adaboost algorithm. A feature could be any property of an image region which can potentially allow it to be classified as a particle or a non-particle. However, a classifier  $f_k(t)$  using only a single feature is likely to have high error rates. Such a classifier, which may perform only slightly better than random, is called a “weak classifier” [5, 6]. The idea of Adaboost is to linearly combine many such weak classifiers to form a “strong classifier”. A strong classifier is one which gets arbitrarily close to the ideal classifier as the number of images in the training set approaches infinity.

In this paper, the set of candidate features, denoted by  $\mathcal{F}$ , is derived from the so-called integral image of the original image, and this process is detailed in Section 3. The feature space  $\mathcal{F}$  can be extremely large; typically there are more features than the number of pixels in an image (e.g., in our implementation, there were twelve times as many features as pixels). However, only a modest subset of these features are really necessary for effective classification (e.g., in our experiments the final classifier only used 40 features). The process of selecting these relevant features using Adaboost is described in Sec. 4.

### 3 Feature Generation

In this section we elaborate on the particular types of weak classifiers (features) used for particle detection in our implementation. We have used five types of features shown by the rectangular templates in Figure 2. The value of a feature is determined by placing one of these templates at a location in the detection window, and taking the difference between the sum of all pixel intensities in the white regions and the sum of all pixel intensities in the black regions. Hence, the total number of features generated is given by the product of the number of types of templates and the number of pixel locations where a template is entirely contained within the detection window. In addition, these five types of templates are computed at multiple scales.

The reasons for using these rectangular features are [25]:



(a) Five different types of features.

(b) Features shown at different scales.

Figure 2: Five different types of rectangular features are used in detection, and these feature are considered over a range of scales and at all locations within the detection window.

1. The rectangular features provide an estimate of the first and second derivatives of the image at a particular scale.
2. Rectangular features can be calculated extremely fast using *integral images*.

Once the feature value is computed, the weak classifier takes that single number as input and makes a decision of particle vs. non-particle.

### 3.1 Integral Image and Feature Calculation

An integral image  $I_{int}$  of an image  $I$  is the image formed by the two dimensional integration of the original image, and it is given by:

$$I_{int}(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y') \quad (2)$$

The integral image can be calculated very quickly (i.e.,  $O(n)$  time where  $n$  is the number of pixels) using the following recurrence rule.

$$s(x, y) = s(x, y - 1) + I(x, y) \quad (3)$$

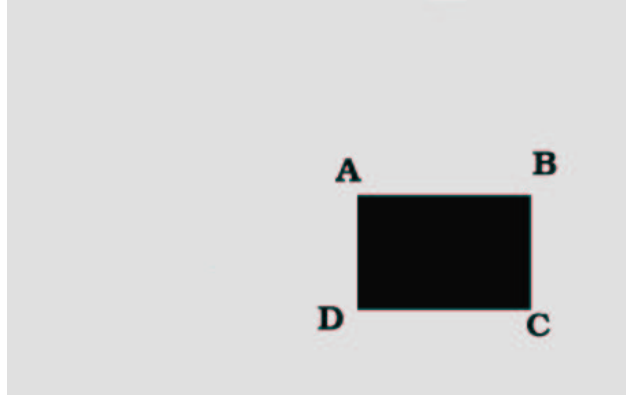


Figure 3: Calculating the sum of all pixels in the black rectangular area using the integral image  $I_{int}$ .

$$I_{int} = I_{int}(x - 1, y) + s(x, y) \quad (4)$$

Note that if an  $m$  by  $n$  image has  $b$  bits per pixel, then the  $n$  by  $m$  integral image will require  $b + \log_2 m + \log_2 n$  bits. E.g., the integral image of a 1024 by 1024 12-bit image would require 32 bits per pixel.

The calculation of rectangular features involves computing the sum of the pixel values in a rectangular region of the image, and the integral image gives a fast way to calculate this sum. Consider Fig. 3. The sum  $S$  of the pixel values of all pixels in the rectangle ABCD is given by:

$$S = I_{int}(C) + I_{int}(A) - I_{int}(B) - I_{int}(D) \quad (5)$$

Hence, the  $O(n)$  operation of calculating the sum of all pixel intensities in a given rectangle has been reduced to an  $O(0)$  operation, once the integral image has been computed. In other words, calculating a particular feature value requires only a small number of array accesses and arithmetic operations.

### 3.2 The Weak Classifier

Each weak classifier is based on univariate quadratic discriminant analysis [3, p. 41]; it takes the value of a feature as input and outputs whether or not the feature indicates the presence of a particle. We assume that the distribution of each feature on the set of particles  $I_p$  and non-particles  $I_{np}$  are respectively a Gaussian with different mean and variance.

As we will see in the next section and as part of the Adaboost algorithm, rather than considering all training examples to be equally important, a weight is assigned to each image in the training set so as to make the algorithm learn difficult examples. Let,

$w_p(i)$  = weight assigned to the  $i^{th}$  particle image in the training set.

$w_{np}(j)$  = weight assigned to the  $j^{th}$  non-particle image in the training set.

such that  $\sum_i w_p(i) + \sum_j w_{np}(j) = 1$  with  $w_p(i) > 0$  and  $w_{np}(j) > 0$ .

Now, consider a feature  $f$ , and let  $v^f(t)$  denote the value of the feature  $f$  in a detection window  $t$ . Over the training set, let us define  $\mu_p^f$  and  $\sigma_p^{f2}$  as the weighted mean and variance of the feature  $f$  over the set of particle images  $\mathcal{I}_p$ , such that

$$\mu_p^f = \frac{\sum_i w_p(i) v^f(i)}{\sum_i w_p(i)} \quad \sigma_p^{f2} = \frac{\sum_i w_p(i) (v^f(i) - \mu_p^f)^2}{\sum_i w_p(i)}$$

For the non-particle images,  $\mu_{np}^f$  and  $\sigma_{np}^{f2}$  are likewise defined as the weighted mean and variance of  $f$  over  $\mathcal{I}_{np}$ .

Using quadratic discriminant analysis, the discriminant functions corresponding to the set of particles  $P_p^f(t)$  and non-particles  $P_{np}^f(t)$  are

$$P_p^f(t) = -\frac{1}{2} \log(\sigma_p^{f2}) - \frac{(v_f(t) - \mu_p^f)^2}{2\sigma_p^{f2}} + \log(\lambda) \quad (6)$$

$$P_{np}^f(t) = -\frac{1}{2} \log(\sigma_{np}^{f2}) - \frac{(v_f(t) - \mu_{np}^f)^2}{2\sigma_{np}^{f2}} + \log(1 - \lambda) \quad (7)$$

where  $\lambda = \frac{\sum_i w_p(i)}{\sum_i w_p(i) + \sum_j w_{np}(j)}$ , and the weak classifier  $f(t)$  declares the detection window  $t$  to be a particle if  $P_p^f(t) \geq P_{np}^f(t)$ . As a function, the weak classifier  $f$  is given by:

$$f(t) = \begin{cases} 1 & P_p^f(t) \geq P_{np}^f(t) \\ -1 & P_p^f(t) < P_{np}^f(t) \end{cases} \quad (8)$$

## 4 Feature selection

As explained in the previous section, we generate five different types of features, each at a different scale, and each at all locations within a detection window. Hence, the number of potential features is typically many times the number of pixels in the detection window. However, all features are not useful for classification, and only a few features may be sufficient. The next task is to select a set of a few important features out of this huge feature space.

Feature selection is done using Adaboost (or Adaptive Boosting) proposed by Freund and Schapire [6], and its use for feature selection is described below:

1. Let  $(t_1, o_1), \dots, (t_n, o_n)$  be the training data where the  $t_i$ 's are the training images (i.e.,  $t_i \in \mathcal{I}_p \cup \mathcal{I}_{np}$ ), and the  $o_i$ 's are corresponding outputs (i.e.,  $o_i$  equals 1 or  $-1$  depending upon whether or not  $t_i$  is an image of a particle).
2.  $w_{1,i} = \frac{1}{n}$  is the weight associated with the  $i^{th}$  training image during the first round of boosting.
3. Let there be  $K$  rounds of boosting, in which one feature is selected per round for a total of  $K$  features.

For  $k = 1, \dots, K$  :

- Normalize:  $w_{k,i} \leftarrow \frac{w_{k,i}}{\sum_{j=1}^n w_{k,j}}$
- For each feature  $j$ , train a weak classifier  $f_j$  as in Sec 3.2 using  $\mathcal{I}_p$  and  $\mathcal{I}_{np}$ . Let  $f_j(t_i)$  denote the value of the  $j^{th}$  feature on the training image  $t_i$ . The error for feature  $j$  is given by:

$$\epsilon_j = \sum_i w_i |f_j(t_i) - o_i|$$

- Select  $f_k$  to be the classifier with the lowest error  $\epsilon_k$ .
- Update the weights on the training images as

$$w_{k+1,i} = w_{k,i} \beta_k^{e_i}$$

where  $\beta_k = \frac{\epsilon_k}{1-\epsilon_k}$ ;  $e_i = 1$  if example  $t_i$  is classified correctly, and  $e_i = 0$  otherwise.

4. The final strong classifier is

$$F(t) = \begin{cases} 1 & \sum_{k=1}^K \alpha_k f_k(t) \geq \mathcal{T} \sum_{k=1}^K \alpha_k \\ -1 & \text{Otherwise} \end{cases}$$

where  $\alpha_k = \log \frac{1}{\beta_k}$ , and  $0 \leq \mathcal{T} \leq 1$  controls the threshold. A higher value of  $\mathcal{T}$  would lead to higher false negative rate and lower false positive rate, whereas a lower value of  $\mathcal{T}$  would lead to a lower false negative rate and higher false positive rate.

A detailed analysis of Adaboost can be found in [5, 6, 8]. Here, we present an intuitive overview of the algorithm. In Adaboost, all images are initially weighted equally. Among all the features in the feature space  $\mathcal{F}$ , a feature is selected which gives the lowest error when classifying the images in the training set  $\mathcal{I}_p \cup \mathcal{I}_{np}$ . The weights of the training images which were misclassified are then increased, so that during the next round of boosting when the next feature is selected, the difficult images will be weighted more heavily. One feature (weak classifier) is selected at each round of boosting, and a parameter  $\alpha_i$  is calculated for the  $i^{th}$  classifier which indicates the relative importance of that feature. The most important contribution of boosting is the guarantee that a linear combination of “weak classifiers” with weights  $\alpha_i$  will produce a “strong classifier.” A strong classifier is by definition a classifier which has a performance arbitrarily close to the ideal classifier as the number of images in the training set goes to infinity.

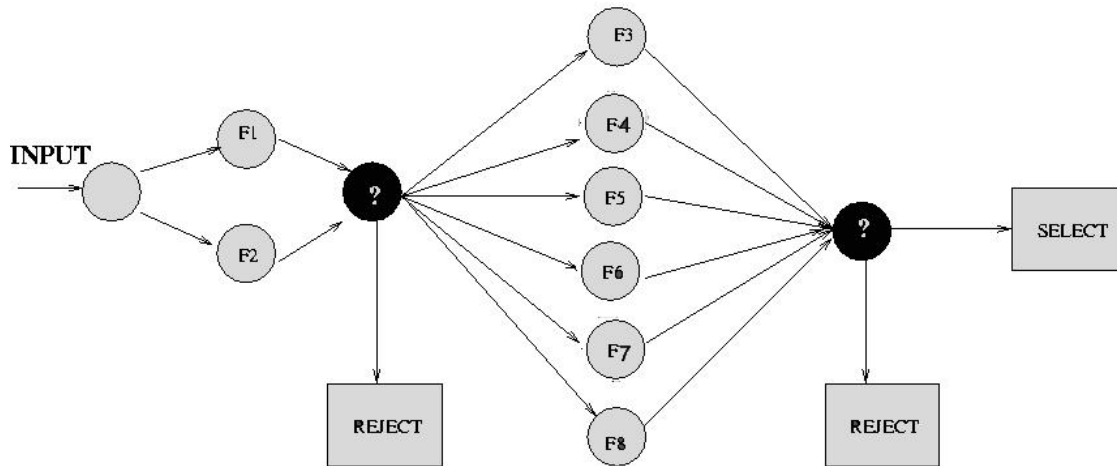


Figure 4: A two stage cascade of classifiers

## 5 Cascading Classifiers

Feature selection using Adaboost provides a modest number of relevant features from a huge feature space. Since the training set contains images of a particle in a single orientation, the detector will only detect particles in that same orientation. Hence, a test micrograph is rotated over of a range of orientations in order to detect particles at other orientations. This can be time consuming, and so further speedups to the detection process are useful.

Once a set of features (classifiers) is selected by Adaboost, a linear combination of all of them can be used to generate a single strong classifier. We note that most locations do not contain particles, and so it is possible to reject most locations in the micrograph as non-particles by only considering a few features out of the selected ones. Hence, a cascade of classifiers can be used instead of a single monolithic classifier[1, 17, 24, 25]. A cascade of classifiers performs classification with a number of features at each stage of the cascade. Some locations in the test micrograph are rejected at each stage. Only those locations which cross all stages are labeled as particles. The cascade of classifiers thus speeds up the detection process by eliminating the need to evaluate all of the selected features. To make the idea of a cascade of classifiers clear, we show a simple two stage cascade of classifiers in Figure 4. At each stage a decision is made based on the features at that stage. Some locations are rejected, and a few move on to the next stage. The locations which test positive at all stages of the cascade are labeled as particles.

The number of features increases as we go to higher stages and so does the threshold. This ensures that locations which are obviously non-particles are rejected in the early stages without eliminating actual particles, and we do not need to calculate all selected features for all locations. The final stage has the maximum number of features and has a high threshold. The previous stages can be thought of as “rejection stages” whose aim is to reject non-particles rather than to select particles. The final stage can be thought of as the “selection stage” in which we select from the remaining

locations, those that contain particles.

In our implementation, we used forty features, and the cascade consists of four stages with 3, 7, 12 and 18 features. For a monolithic 40 feature classifier, the amount of time required to detect particles on a down-sampled 512x512 pixel micrograph at 8 different orientations was roughly 50 seconds on a 1.3 GHz Pentium M processor. This time was brought down to around 10 seconds with a cascade of classifiers with four stages.

It should be noted that the early stages of the cascade make decisions using only a few features, but they reject a majority of the locations as non-particles. Typically 50-60% of the locations are rejected at the very first stage itself. A higher threshold at lower stages, would speed up the algorithm at the expense of missing a larger fraction of the actual particles (more false negatives).

## 6 Implementation and Results

We have implemented in C/C++ the presented method for detecting particles in Cryo-EM images using a four stage cascade with a total of forty features. The result of applying the detector is an image indicating those locations where a particle is said to be present. As there will usually be positive responses at multiple, neighboring locations for each particle, the results are post-processed using connected component analysis, and the mean of each component is reported as the location of a particle [10]. The running time on a down-sampled 512x512 pixel micrograph at 24 different orientations was 10 seconds on a 1.3 GHz Pentium M processor.

The algorithm was trained on a training set containing approximately 1200 particle images and 3100 non-particle images. As described earlier, the set of non-particle images should contain both a representative set of cropped subwindows (e.g., images just containing ice) as well as images that might be expected to confuse the classifier (e.g., deformed or broken particles, particles in undesirable 3-D orientations, overlapping particles, particles that are too close together to be useful for reconstruction, etc.). The training process took approximately 10 minutes.

The detection experiments described here were carried out on 82 digital micrographs of Keyhole Lympet Hemocyanin (KLH) acquired with a nominal magnification of 66,000x and a voltage of 120kV using a Philips CM200 transmission electronic microscope with a 2048 x 2048 CCD Tietz camera. For these experiments, detection was performed on images acquired in far from focus conditions of  $-2\mu m$ , and they were downsampled after averaging to 512 x 512 pixels. The dataset is available at [http://ami.scripps.edu/prtl\\_data/](http://ami.scripps.edu/prtl_data/), and a more detailed description can be found in [27]. The data set includes manually picked locations of particles by Frabrice Mouche, and these serve as “ground truth” when reporting error rates in this paper. It is important to note that, unlike many other domains for which objects are detected in images, in this case different experienced microscopists will label different image locations as particles, and on different days the same microscopist may give different labels. These challenges in evaluating and comparing particle detection algorithms are discussed in [27].

KLH particles assume only two preferred 3-D orientations in ice, and this results in the circular and rectangular shaped particle images shown in Fig. 5. As the dataset only includes ground truth labels for particles whose appearances are rectangular, we present both qualitative and quantitative results for the rectangular views, but only qualitative results for the circular views. Note that parts of the images may be cluttered with other structures including broken molecules, particles in intermediate views between the two preferred orientations, aggregates of two or more particles, references such as tobacco mosaic virus, and contaminants.

To evaluate our method using a limited quantity of data and avoid the problem of testing the method on images used to train the classifier, we use the standard “leaving-one-out” cross-validation strategy [3, 18]. Such a problem arises if one were to crop training examples from an available image set, and then evaluate the classifier on the same image set; in such a case the reported error rates will almost always be lower than what would have occurred on an unseen data set.

In the ‘leaving-one-out’ strategy, we train the algorithm using the particles and non-particle images selected manually from all but one micrograph. The micrograph which is not used for training becomes the test micrograph on which particle detection is done. This process is repeated for each of the images in the dataset. Cross validation ensures that testing will not occur on the training data, yet we can use a large number of samples for training. The price for this evaluation methodology is that the detector must be re-trained 82 times for this data set.

## 6.1 Qualitative Results

Qualitative results are shown in Figures 5 and 6 for detecting KLH particles in circular and rectangular views. Note that these are typical results and are not the images with the best results. In Figure 6, note that some detected locations do not correspond to clean, isolated particles (false positives) and that a small number of particles which look to be clean are not detected (false negatives). There are a few points worth mentioning:

1. Circular views are easier to detect than rectangular views.
2. With a suitable choice of threshold, we can ensure that no circular views are detected when we are trying to detect rectangular views.
3. Although the algorithm rejects most of the long rectangular particles (which are probably multiple particles stuck together), long particles still represent the most important source of false positives.

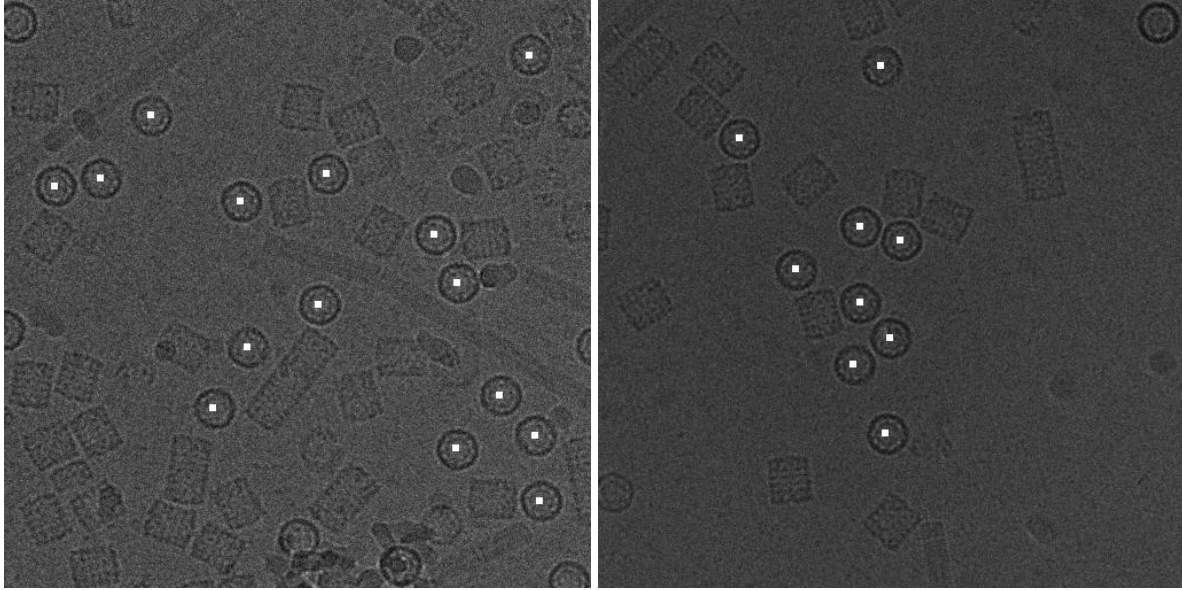
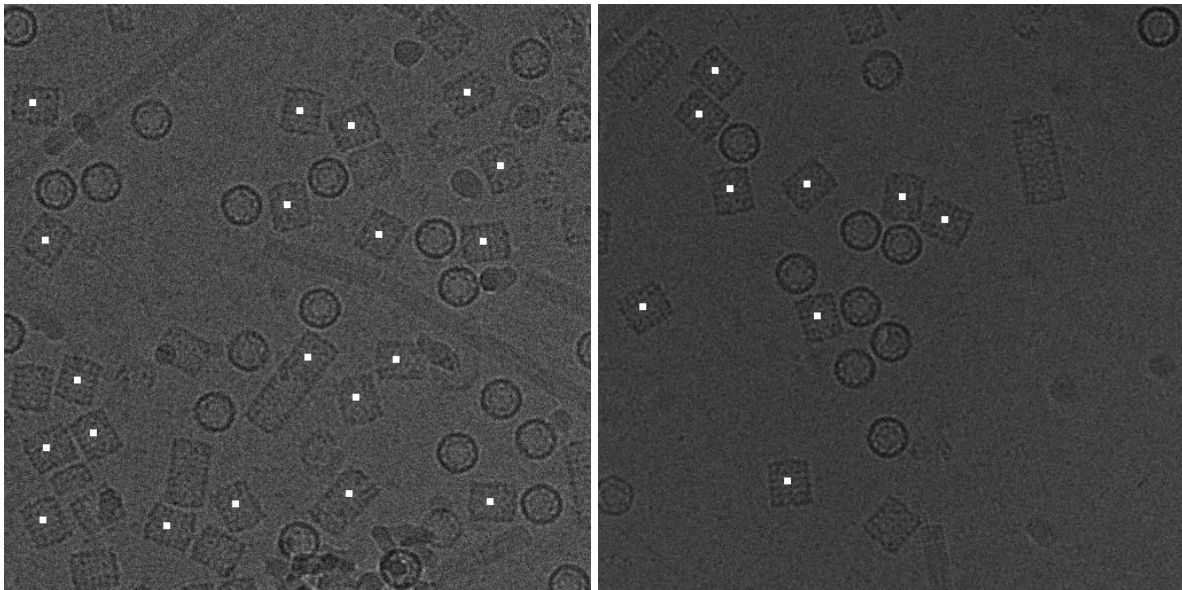


Figure 5: Circular views of KLH detected in two typical micrographs.



(a)

(b)

Figure 6: Rectangular views of KLH detected in two typical micrographs. In (a) 16 out of 18 particles present were detected and 4 out of the 20 regions labeled as particles were actually not particles. In (b) 9 out of the 10 particles present were detected, and none of the 9 regions labeled as particles were actually not particles.

## 6.2 Receiver Operating Characteristic

The performance of a classifier is given in terms of its False Positive Rate and False Negative Rate. The two rates are defined as follows.

$$\begin{aligned}\text{False Positive Rate} &= \frac{\text{Total number of false positives}}{\text{Total number of detections}} \\ \text{True Positive Rate} &= \frac{\text{Total number of true positives}}{\text{Actual number of particles present}}\end{aligned}\tag{9}$$

As one varies parameters in an algorithm (the threshold  $\mathcal{T}$  in our case), the false positive and false negative rates will vary. One common way to visualize this tradeoff is via the receiver operating characteristic (ROC), and the ROC curve of the final classifier is shown in Figure 7. This curve represents the performance of the classifier, using “leaving-one-out” on all 82 test micrographs. In addition, the error rates on this data set are also plotted for the algorithms by Bern, Hall, Ludtke, Roseman, Sigworth, Bajaj and Zhu. These error rates are based on the Mouche’s manual labelling, and the results were presented at the “*Multidisciplinary Workshop on Automatic Particle Selection for Cryo-electron Microscopy*.” See also [27]. The point at the “knee” of the ROC curve is usually chosen as the operating point because the false positive rate and false negative rate are both low there. An ROC is an excellent visual aid to compare performance of different classifiers. Points to the left of the ROC represent superior results, and the points to the right of the ROC represent inferior results.

The ROC shown using dashed lines shows a variant of our method in which the circular particles are first detected and removed before the rectangular particles are detected. The ROC shown using solid lines is the case when the rectangular particles are detected in the presence of circular particles. The close proximity of the two curves implies that we have nearly achieved the second property of an ideal classifier listed in the introduction. The point we want to emphasize here is that a classifier which tends to detect particles other than the intended one, does not have high specificity.

## 6.3 Performance under Additive Noise

In order to assess the performance of the classifier when the signal to noise ratio is lower, the test images of KLH were corrupted with additive noise. In particular, the pixel values of the micrographs were scaled to a range of 0 to 1, and zero-mean independent Gaussian noise with increasing variance was added to the pixel values. Note of course that the original images are very noisy themselves. Continuing to use the leaving-one-out evaluation paradigm, the classifiers were trained on “noise-free” images, and the standard deviation of the noise in the test images ranged from  $\sigma = 0$  to  $\sigma = 0.1$ . Picking a set of thresholds for the detector giving false positive and false

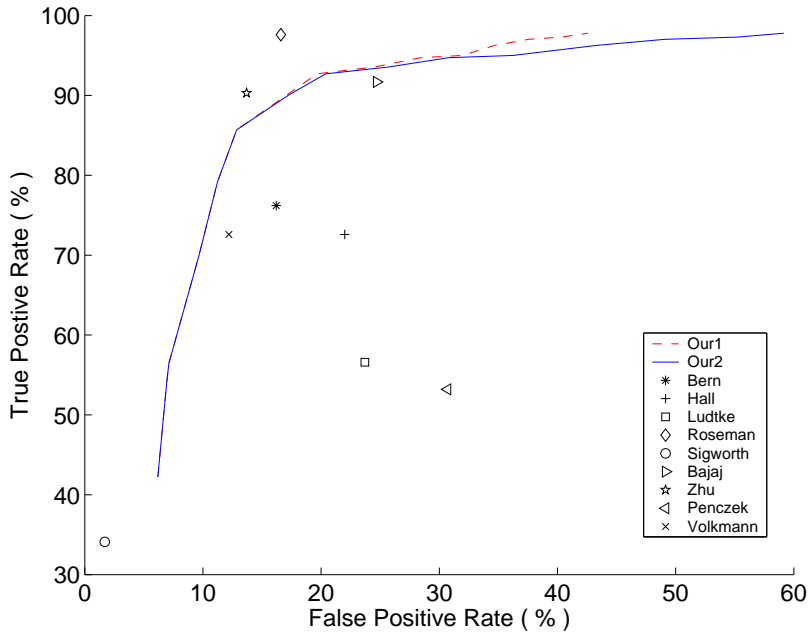


Figure 7: Receiver operating characteristic (ROC) of the final particle detector. Our1 is the ROC with circular particles removed and Our2 is the ROC obtained without removing the circular particles before detection of the rectangular particles.

negative rates of 13.69% and 13.72% without additional noise (a single point on the ROC curve in Fig. 7), the resulting absolute errors and error rates under noise are shown in Fig. 8.

As seen from Fig. 7, the total number of false positives and the total number of true positives fall monotonically because fewer particles are detected with increasing noise. As expected the true positive rate falls monotonically. When the images of particles are corrupted with increasing noise levels, they begin to look less and less like particles, and so there are fewer and fewer detects. On the other hand, the false positive rate decreases marginally and then rises abruptly with excessive noise. While this decrease may at first be surprising, it is attributable to the fact that nearly all false positives in the noise-free case arise for non-particles that have significant structure (broken fragments, multimers, etc.). Only very rarely is a seemingly random part of say the ice erroneously labeled as a particle. Consequently, with fewer detections, the error rate decreases. With excessive noise, the algorithm degenerates to give a high false positive rate. With excessive noise the number of detections falls much more rapidly than the total number of falls positives, giving a very high false positive rate.

## 6.4 Sparse Detection

In general, if a particular location is detected as a particle, we would expect that some of its neighbors will also respond positively to the same particle. Because of this, we perform post-

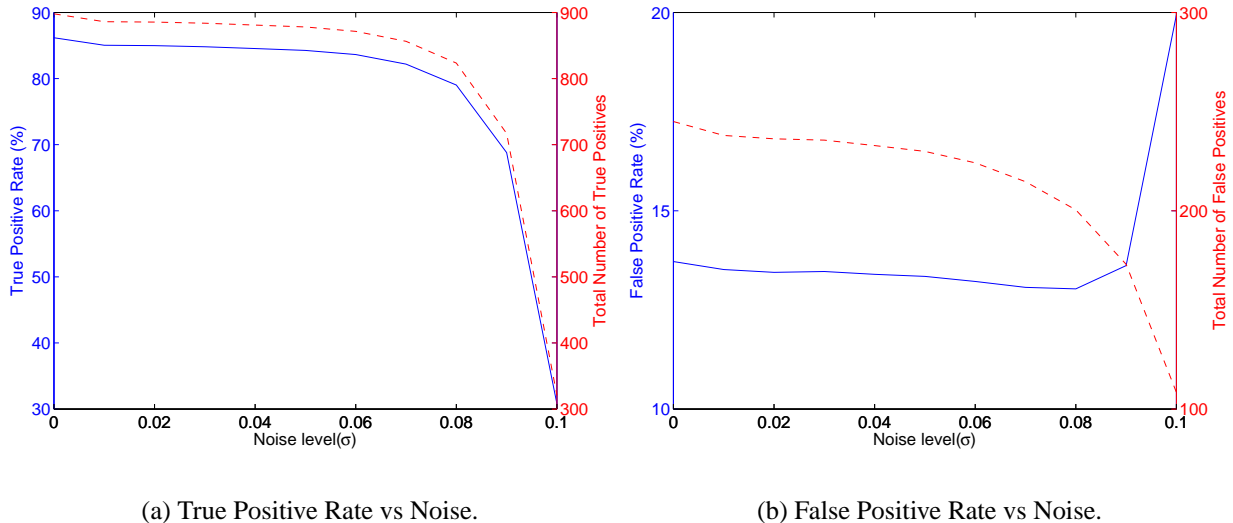


Figure 8: Performance under additive Gaussian noise. The left ordinates indicates the percent rates and correspond to the solid curve, while the right ordinate indicates the absolute values and correspond to the dashed curve. The abscissa gives the noise level in terms of standard deviation of the zero-mean Gaussian noise.

filtering to label a cluster of detected locations as coming from the same particle (See Sec. 6). Hence, we also expect that we can, without much loss of accuracy, evaluate the detector at every  $n^{\text{th}}$  pixel instead of every pixel, and therefore speed up the detection process. The error rates and timings are shown in Fig. 9. As the subsampling increases, the overall number of detected particles decreases. Hence, the true positive rate as well as the false positive rate decrease monotonically.

## 6.5 Reconstructed Three-Dimensional Density Maps of KLH

Given the coordinates of particles detected in the far-from-focus images, we were able to extract 998 KLH particles from the corresponding near-to-focus images in the set of 82 defocus pairs, by first aligning the defocus pairs using phase correlation and then shifting the coordinates according to the alignment results [28]. Afterwards, using the EMAN software package [12] and a previous reconstruction solved to a resolution of  $40\text{\AA}$  as the initial reference model, a standard iterative refinement procedure was applied to create a 3-D density map from the extracted set of particles. The refinement converged (the Fourier shell correlation curve from one iteration to the next is essentially unchanged) after five iterations. The final 3-D map, shown in Figure 10, was estimated at a resolution of  $23.2\text{\AA}$ , using the 0.5 Fourier shell correlation criteria [2, 20], which has the same resolution as the one reconstructed from the set of 1042 particles manually selected in the same set of images by an experienced user [27] and exhibited similar features to the structure already published [15].

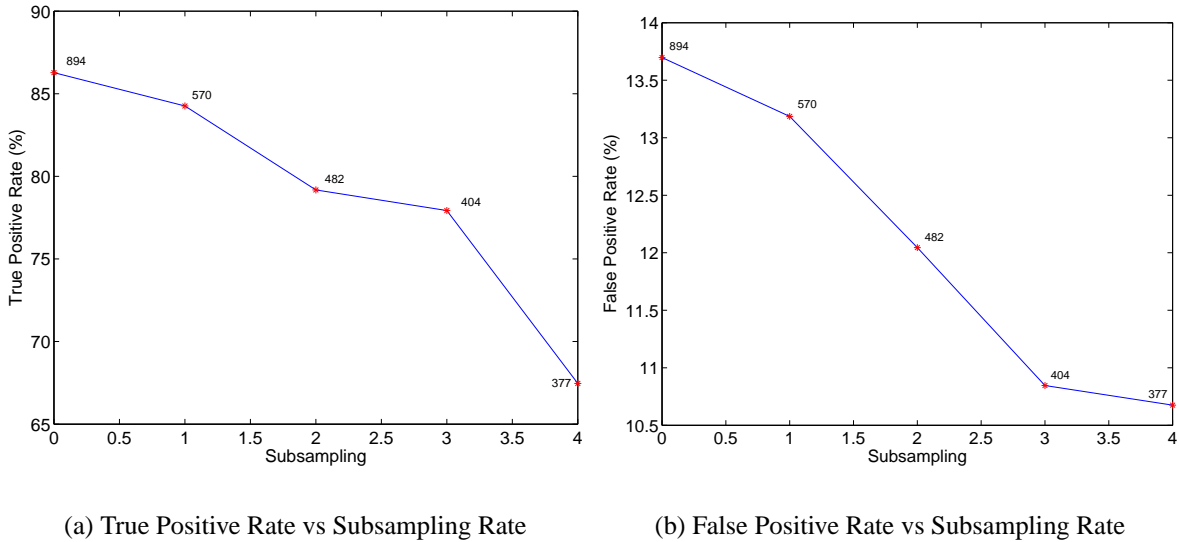


Figure 9: Error rate vs. subsampling rate: The ordinate subsampling is the number of pixels skipped when evaluating the detector. For each plotted point, the number indicates the time required to run the algorithm on all 82 micrographs.

## 7 Conclusion

In this paper, a fast learning-based approach is introduced for particle detection using a cascaded classifier that is trained using Adaboost. The effectiveness of the method was demonstrated on a dataset of 82 micrographs of Keyhole Lympet Hemocyanin (KLH). We have emphasized that learning-based approaches can be very fast and have the potential to give a generic solution to the problem of particle picking. There are many directions for future work based on the encouraging results presented here. First of all, this paper introduces a single learning approach using a very specific type of feature. Certainly, there are many other feature types to explore and different training methodologies besides Adaboost; while they may provide greater accuracy, it is unlikely that they will be as fast. In addition, there are other sources of information that can be used to create a more effective classifier. For example, a low resolution reconstruction, when available, can be used to provide additional positive training examples, and these may be important for particles that are equally likely to assume any 3-D orientation. For the negative training set, it may be possible to use a bootstrapping approach which was so successfully used for face detection in [22]. In this case, whenever a human identifies a detection as a mistake, the corresponding image becomes an example of a non-particle. There are also opportunities to improve the detection rate by pre-filtering/denoising the training and test images prior to detection. Finally, we would also like to test the current algorithm on more difficult examples of particles such as those which are fully asymmetric, those that assume arbitrary 3-D orientations, and those whose images have low contrast even under far-from-focus imaging conditions.

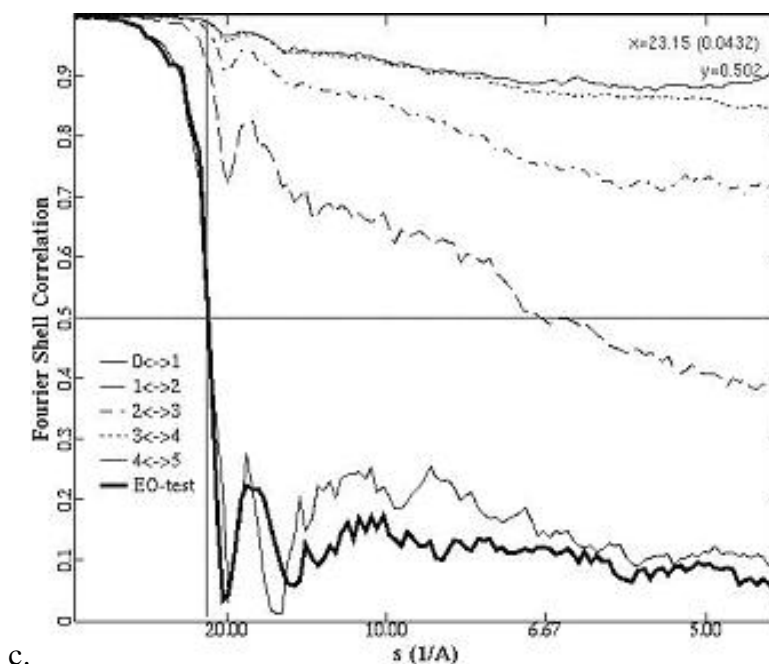
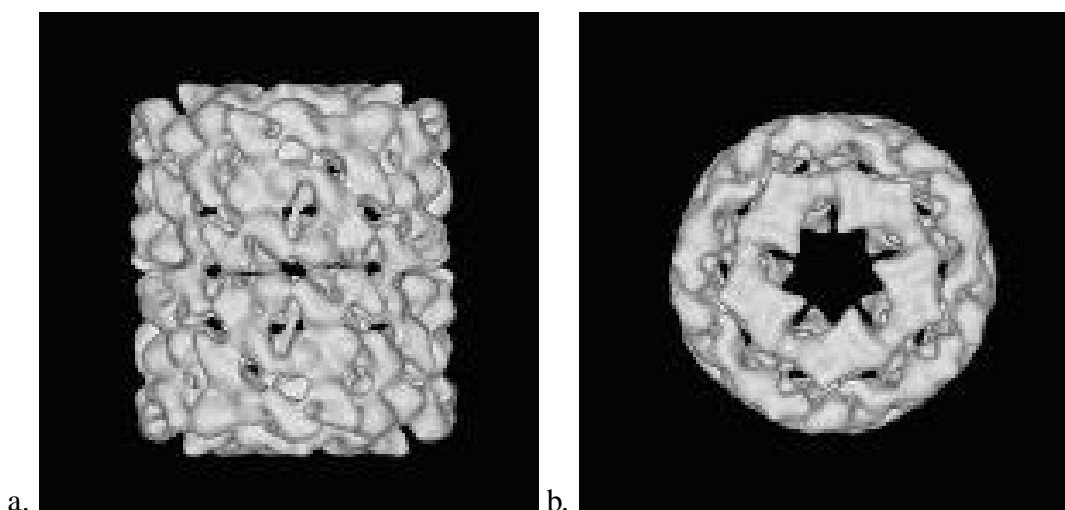


Figure 10: Results of three-dimensional reconstruction of KLH using automatically extracted particles as described in the text. The starting point was a dataset of 82 defocus pairs of images (2048 by 2048 pixels digital images) collected at a nominal magnification of 66,000 (2.2 Å/pixel) and a dose of 10 e/Å for a near to focus image. (a), (b) The KLH density maps (side- and top- view of isosurfaces) reconstructed from a set of 998 automatically extracted particle images. (c) Plot of FSC between each 3-D map and the map from previous iteration. The thick black line illustrates resolution determination using FSC (i.e., the FSC between maps generated with only the even-numbered particles versus one with only the odd-numbered particles).

## 8 Acknowledgments

We would like to thank Dr. Clint Potter and Dr. Carragher Bridget for their help, cooperation and insightful suggestions, as well as Francisco Guerra and Fabrice Mouche for their help in image acquisition and map reconstruction. D. Kriegman was supported in part by NIH GM61939, and Y. Zhu was supported by NIH (GM61939 and RR17573) and NSF (DBI-0296063). The work presented here used data supplied by the National Resource for Automated Molecular Microscopy which is supported by the National Institutes of Health through the National Center for Research Resources' P41 program (RR17573).

## References

- [1] Y. Amit, D. Geman, and K. Wilder. Joint induction of shape features and tree classifiers. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 19:1300–1306, 1997.
- [2] B. Bttcher, S. Wynne, , and R. Crowther. Determination of the fold of the core protein of hepatitis b virus by electron cryomicroscopy. *Nature*, 386:88–91, 1997.
- [3] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley and Sons, second edition, 2001.
- [4] J. Frank and T. Wagenknecht. Automatic selection of molecular images from electron micrography. *Ultramicroscopy*, 12:169–176, 1984.
- [5] Y. Freund and R. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence* 14(5), pages 771–780, 1999.
- [6] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.
- [7] R. Glaeser. Electron crystallography: Present excitement, a nod to the past, anticipating the future. *Journal of Structural Biology*, 128:3–14, 1999.
- [8] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, 2001.
- [9] R. Henderson. The potential and limitations of neutrons, electrons, and x-rays for atomic resolution microscopy of unstained biological molecules. *Q. Rev. Biophys*, 28:171–193, 1995.
- [10] B. Horn. *Robot Vision*. MIT Press, Cambridge, Mass., 1986.
- [11] R. Kimmel, R. Malladi, and N. Sochen. Image processing via the Beltrami Operator. *Asian Conference on Computer Vision, LNCS*, 1351:574–581, 1998.
- [12] S. J. Ludtke, P. R. Baldwin, , and W. Chiu. Eman: Semiautomated software for high-resolution single-particle reconstructions. *J. Struct. Biol.*, 128:82–97, 1999.
- [13] I. M. B. Martin, D. C. Martinescu, R. E. Martin, and T. S. Baker. Identification of spherical virus particles in digitized images of entire electron micrographs. *Journal of Structural Biology*, 120:146–157, 1997.
- [14] W. V. Nicholson and R. M. Glaeser. Review: Automatic particle detection in electron microscopy. *Journal of Structural Biology*, 133:90–101, 2001.
- [15] e. a. Orlova, E.V. Structure of keyhole limpet hemocyanin type 1 (klh1) at 15Å resolution by electron cryomicroscopy and angular reconstitution. *J. Mol. Biol.*, 271(3):417–437, 1997.
- [16] P. Penczek. Appendix: measures of resolution of resolution using fourier shell correlation. *J. Mol. Biol.*, 280:115–6, 1998.
- [17] J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

- [18] B. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.
- [19] A. Roseman. Particle finding in electron micrographs using a fast local correlation algorithm. *Ultra-microscopy*, 94:225–236, April 2003.
- [20] W. Saxton and W. Baumeister. The correlation of averaging of a regularly arranged bacterial cell envelope protein. *J. Microscopy*, 127:127–138, 1982.
- [21] A. Stoschek and R. Hegerl. Automated detection of macromolecules from electron micrographs using advanced filter techniques. *Journal of Microscopy*, 185:76–94, 1997.
- [22] K.-K. Sung and T. Poggio. Example-based learning for view-based human face detection. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 20(1):39–51, 1998.
- [23] P. Thuman-Commike and W. Chiu. Automated detection of spherical particles in spot-scan electron cryomicroscopy images. *Journal of the Microscopy Society of America*, 1:191–201, 1995.
- [24] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, 2001.
- [25] P. Viola and M. Jones. Robust real-time object detection. *Technical Report CRL*, 2002.
- [26] M. Yang, D. Kriegman, and N. Ahuja. Detecting faces in images: A survey. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 24(1):34–58, 2002.
- [27] Y. Zhu, B. Carragher, R. Glaeser, D. Fellmann, C. Bajaj, M. Bern, F. Mouche, F. Haas, R. Hall, D. Kriegman, S. Ludtke, S. Mallick, P. Penczek, A. Roseman, F. Sigworth, N. Volkman, and C. S. Potter. Automatic particle selection: Results of a comparative study. *Journal of Structural Biology*, 2003. Under review.
- [28] Y. Zhu, B. Carragher, D. Kriegman, R. Milligan, and C. Potter. Automated identification of filaments in cryoelectron microscopy images. *Journal of Structural Biology*, pages 302–312, 2001.