AUTOMATIC DETERMINATION OF VISION LOCK ON THE SEAFLOOR IN THE PRESENCE OF DUST

Kristof Richmond, David Black-Shaffer, Stephen M. Rock
Aerospace Robotics Laboratory, Stanford University
Stanford, CA 94305
{kristof,dbs,rock}@arl.stanford.edu

Abstract

The Stanford Aerospace Robotics Lab (ARL), in cooperation with the Monterey Bay Aquarium Research Institute (MBARI), has developed a visual benthic station-keeping system and deployed it on an underwater ROV. The system indicates successful vision lock on the seafloor to the pilot. In order to improve the robustness of this system, we have added a DVL to provide velocity measurements to be fused with the position measurement from the vision system. The DVL measurements can be used to maintain an estimate of vehicle state during periods of vision loss-of-lock. This fusion requires the signal indicating visual lock be accurate, so that the estimator knows how much to trust each sensor. However, the current vision-lock indication can sometimes be confused, e.g. when a cloud of dust is blown into the field of view. In this case, the system may lock on to the dust cloud instead of the sea floor and lose its station. Thus, in order to provide a more accurate measure of the presence of dust in our field of view and thus of seafloor vision lock, we have developed a recognition algorithm for dust in these underwater scenes. After extracting moving edges in and the dynamic texture of regions in the view, the recognition is performed by a support vector machine trained on instances of stock video of the seafloor including and without dust in view. We have achieved recognition of dust in the video at an error rate comparable to that of a trained human user. The recognition algorithm returns a measure of confidence in the visual system measurement. This confidence can then be combined with that from the existing system as an input to a state estimator fusing vision and DVL measurements to arrive at a robust estimate of the vehicle state.

Figure 1: View of the floor of Monterey Bay from the ROV *Ventana* with dust cloud appearing on the right.

1 Introduction

The Stanford Aerospace Robotics Lab (ARL), in cooperation with the Monterey Bay Aquarium Research Institute (MBARI), has contributed significantly to the automation of tasks for underwater ROVs. In particular, we have developed and field-tested a system for automatically holding station over and maneuvering relative to the seafloor using a video camera mounted on the ROV [9]. Additionally, video mosaics of the seafloor are produced online as the vehicle maneuvers [5].

We first describe the current system and its limitations. The general sensing and control framework is presented in Section 1.1 and the primary sensors are described in more detail in Sections 1.2 and 1.3. We then enumerate some of the problems currently hampering autonomy of the system in Section 1.4, and identify the need for more accurate determination of visual lock on the seafloor in the presence of dust clouds in order to enable this system.

In the remaining sections, we present a method for

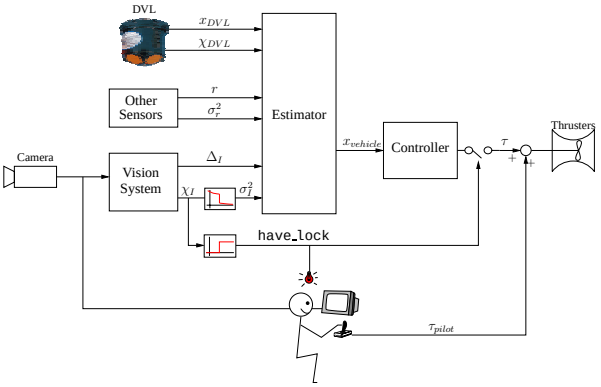


Figure 2: Overview of current automated station-keeping system implemented on board the ROV *Ventana*.

determining lock on the seafloor in the presence of dust. This consists of a machine learning approach for distinguishing between video sequences obscured by dust clouds and other sequences with clear visibility. In Section 2.1, we explain the features we extract from the video sequences for identification of dust clouds: a moving edge detector and a dynamic texture. In Section 2.2, we present the recognition system for classifying the sequences as to the presence or absence of dust: a machine learning framework using support vector machines. Finally, in Section 3, we present results of the learned models detecting dust and discuss some possible extensions.

1.1 System Overview

The current benthic station-keeping system is shown schematically in Figure 2. We use vision as the primary sensor, as it gives a local, drift-free measure of terrain-relative position. We have recently incorporated a DVL into the system, to provide a complementary measurement of vehicle state. Both sensors also have the advantage of being standard features on underwater ROVs.

The raw video feed is captured and processed by our vision system. This system is described in more detail in Section 1.2. The vision system extracts an image disparity, Δ_I, and a confidence in its measurement, χ_I. An experimentally-derived function [4] is then used to map the χ_I given by the vision processing into actual measurement variances, σ_I^2. The mapping contains a sudden transition from high variances to low ones. The location of this transition is used as a threshold on χ_I to produce the `have_lock` signal, indicating whether the vision system has lock on the seafloor or is unable to find a good correlation.

The DVL provides complementary measurements to the vision system. The integration of this instrument is

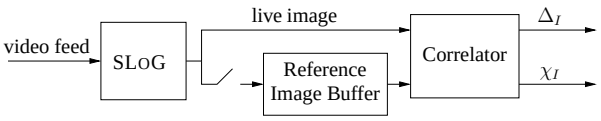


Figure 3: The vision system.

described further in Section 1.3. The DVL provides a direct measurement of the vehicle velocities which had to be estimated when using the vision system alone, and provides an alternative, albeit not drift-free, position estimate which can cover periods of vision loss-of-lock.

The image disparities and DVL velocities are merged in an estimator with information coming from other sensors, namely the vehicle altimeter, compass, inclinometers and camera position encoders, in order to produce the total vehicle state. This state estimate is passed to the controller, which calculates thruster outputs necessary to bring the vehicle to the desired state. If the system was deemed to have lock, the calculated thrusts are then fed to the thrusters to control the vehicle.

As the vehicle is remotely piloted, we have the possibility for human oversight and direct intervention in the system to protect against failures and unanticipated occurrences. Pilot intervention is enabled by summing joystick commands with those output by our system. Thus, if the vision system has lost lock, we turn off the controller output, and signal the pilot to take over control. If the system is having difficulties, such as upon initialization when large transients are present, the pilot may also act in concert with the control system until conditions have stabilized.

1.2 Vision System

The vision system for determining position relative to an initial location is shown in Figure 3. The pilot video stream from the ROV is captured by a frame grabber in the vision processing and automatic control computer. Video frames are filtered with a signum of Laplacian of Gaussian (SLoG) filter [11] to remove the effects of marine snow and uneven lighting [8]. The signum operation returns a binary image, which is then correlated to a reference image (also SLoG-filtered) snapped at initialization of the system. The peak location of the resulting correlation surface defines the image disparity, Δ_I, in pixels, between the current image and reference image.

Correlation of the binary images obtained from the SLoG filter is a fast average-of-XOR operation, returning a correlation surface constrained to peak between 0.0 and 1.0. A peak value of 1.0 indicates perfect correlation, 0.0 indicates perfect anti-correlation and 0.5 indicates the correlation of two Gaussian random images. Thus, a direct measure of confidence, χ_I, in the image

match can be determined from the peak magnitude of the correlation surface.

The vision system also contains a mosaicking capability which allows new reference images to be snapped, while maintaining the disparity in pixels to the original reference.

1.3 DVL Integration

To improve the vehicle state estimate and provide the capability to cover vision drop-outs, a DVL has been installed on board the ROV and integrated into our control system. The DVL provides a direct measurement of the vehicle velocities, complementing the position measurements of the vision system. Adding the DVL velocity measurements improves the estimate of vehicle state thus enabling better controller performance.

In addition, the DVL velocity measurements can be integrated over time to arrive at a complementary estimate of vehicle position. The errors in this measurement will grow without bound, and can only be used for limited periods of time.

The DVL also returns a measure of confidence, χ_{DVL}, in its measurements, so that the DVL and vision system measurements can be fused in an estimator in order to derive an improved estimate of vehicle state. This estimate can be accurately maintained during dropouts of either sensor, as long as the confidences χ_I and χ_{DVL} of each truly reflect the accuracy of the measurement. Thus the DVL position measurement allows the system to maintain knowledge of its position during loss of visual lock.

1.4 Limitations

While the system is consistently able to hold station and produce mosaics of the seafloor, the vision sensor has some limitations which hamper the autonomy of the system. A few of the factors which cause degradation of performance and/or failure of the system are

- poor lighting, especially if the vehicle has moved too high off the seafloor,
- areas of low visual texture, such as mud flats,
- occlusions in the field of view, usually in the form of dust clouds.

The first two limitations are related manifestations of the fact that the SLoG filter finds edges in the image, so that areas without sharp changes in image intensity will show up as areas of uncorrelatable noise. In such regions the vision system will come up with a low confidence value and signal a loss of lock, giving up control to the pilot. Under these conditions, the `have_lock` signal could be used directly in the estimator to switch over to using the DVL measurements.

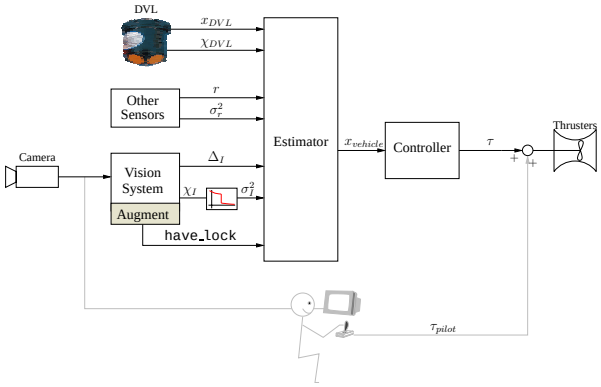


Figure 4: A more autonomous station-keeping system. Addition of the DVL provides a complementary measurement of the vehicle state. To reduce false-lock signals from the vision system, an augment specifically recognizing the presence of dust clouds is added to improve the accuracy of the `have_lock` signal. The system becomes more robust, and the pilot becomes a supplemental, but not required, component of the system.

The final limitation is due to environmental factors, and to the fact that the system requires a clear field of view to the seafloor. If a dust cloud is diffuse or smooth, there will be no texture to correlate on, and the system will appropriately signal a loss of lock. However, areas in the image with high visual texture are assumed to be part of the seafloor. Some dust clouds present a similar degree of texture as the seafloor, confusing the system and leading to an erroneous value of χ_I. In this case `have_lock` will be inaccurate, and the pilot must recognize the false-lock state and intervene independently. This issue severely restricts field applications of the visual positioning system by imposing a burden on the pilot and preventing autonomous deployment.

However, a new capability to recognize dust clouds in a video sequence can address the shortcomings of χ_I. Though χ_I often indicates a high confidence level during a dust occlusion event, we have developed a recognition capability which can accurately detect dust within three seconds of its appearance. Augmenting the existing vision system with this recognition capability, as shown in Figure 4, thus enables a reliable, rapid switch to DVL position sensing, which is unaffected by the presence of dust. The remainder of this paper develops the analytical foundation for a dust detection capability intended for use with a real-time control system for an unmanned underwater vehicle.

2 Dust Detection

In order to provide a more accurate indication of confidence in the vision measurement, we have developed a system which specifically recognizes the presence of dust in the camera field of view. The implementation of the recognition breaks down into two broad tasks: extracting features representing dust from the video data and performing the recognition on those features. Early attempts at an *ad-hoc* vision processing solution were marginally successful, but did not produce the degree of accuracy required to significantly improve the current vision lock signal. Thus, to achieve higher recognition accuracy, we implemented a supervised machine learning solution to determine appropriate models for video sequences containing dust.

In keeping with the objective of real-time capability, we have eschewed features and models requiring excessive computation. As the learning takes place offline, no such constraints are placed on the learning algorithm.

2.1 Feature Selection

In the context of many machine learning frameworks, a feature f is a vector representing a data instance. Thus the vector of pixel intensities constructed by stacking all columns of all frames of a video image sequence can be considered a feature f_{raw}, though its dimension (or number of elements), which we will denote by $|f_{raw}|$, is prohibitively large for computation and would require too many examples to be able to accurately fit a model to.

Thus, to make tractable the machine learning process, features of smaller size are required, i.e. we wish to find a mapping

$$f_d = \phi(f_{raw})$$

such that

$$|f_d| \ll |f_{raw}|.$$

The basic trade off in feature selection for a real-time application such as ours is the feature size versus the computation time. If $\phi(f)$ requires too much computation or if $|\phi(f)|$ is too large, it will not be suitable for use in a real-time system. In addition, given both of these constraints, $\phi(f)$ must retain the essential components of f so that recognition remains possible.

In this application, we have selected two primary features of video sequences to track based on the computational complexity and the amount of information they appear to carry. The first is a detector of moving edges, which highlights moving dust fronts. The second is a dynamic texture, which extracts a model of the dynamics of the image sequence.

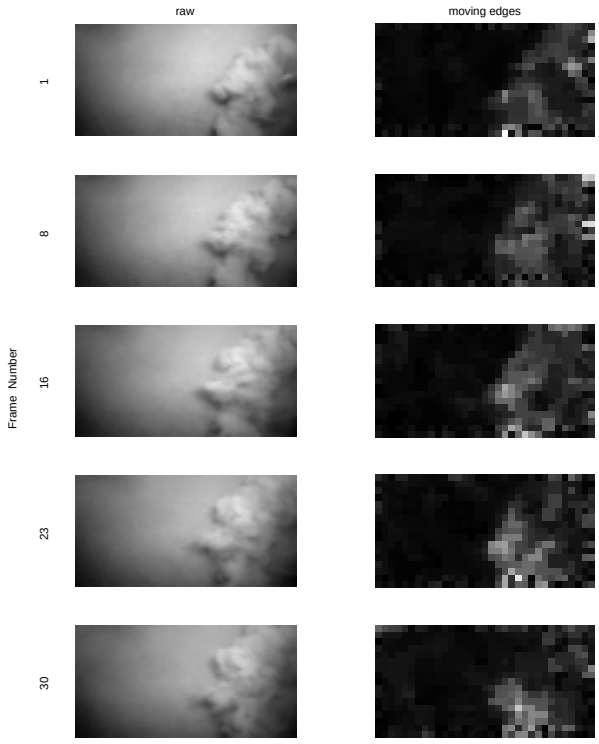


Figure 5: Moving edge filter applied to the sequence terminating with the image in Figure 1.

2.1.1 Moving Edges Feature

The first feature we use is a detector of moving edges (Figure 5). This filter is based on an optical flow magnitude filter. A temporally low-pass-filtered sequence is constructed from the raw video sequence and downsampled. Optical flow magnitude is then calculated by differencing this low-passed version of the sequence. Median filtering is applied to further reduce noise. For a sequence of τ images of size $n \times n$ and with downsample factor d, this feature f_{ME} contains $|f_{ME}| = \tau\frac{n^2}{d^2}$ elements.

2.1.2 Dynamic Texture Feature

The second feature is a *dynamic texture* [16] which fits a linear dynamical system (A, C), driven by Gaussian noise, to an image sequence. [16] shows that the system model $M = (A, C)$ can be easily extracted from an image sequence via a singular value decomposition (SVD). This extracts the principal components of the image sequence and reduces the data size. In the following discussion, we assume a sequence composed of τ frames of size $n \times n$ and k states in the dynamic texture model.

Following [16], we assume M is a simple autoregres-

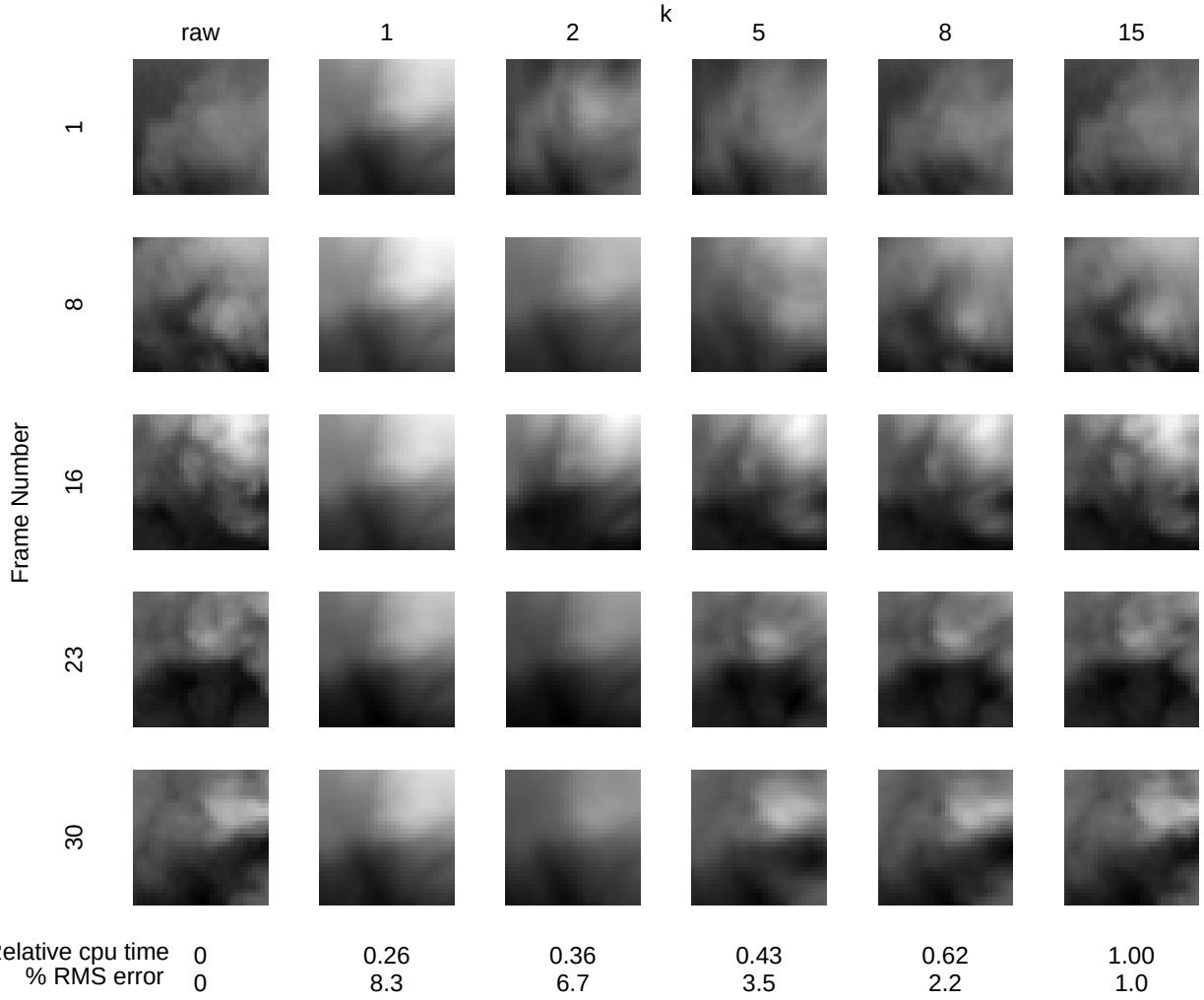


Figure 6: Dynamic textures extracted from a 30-frame sequence of a dust cloud over the seafloor. The original sequence is shown on the left. The number of states assumed for each texture increases toward the right. The first singular value extracts background illumination through the sequence. An increasing number of states (singular values) results in a better reconstruction of the original, but at the cost of more computation.

sive model with a Gaussian input distribution:

$$\begin{array}{ll} x(t+1) = Ax(t) + v(t) & x(0) = x_0,\ v(t) \sim \mathcal{N}(0, Q) \\ y(t) = Cx(t) + w(t) & w(t) \sim \mathcal{N}(0, R) \end{array}$$

where $A \in \mathbb{R}^{k \times k}$ and $C \in \mathbb{R}^{n^2 \times k}$. The captured images $y(t) \in \mathbb{R}^{n^2}$ are assumed to be the output of a linear dynamical system with k states and driven by white Gaussian noise. The maximum likelihood solution for the system parameters A, C, Q given a sequence of images $y(1), \ldots, y(\tau)$ is given by the SVD of the sequence matrix $Y \triangleq [y(1) \quad y(2) \ldots y(\tau)]$. Decomposing Y into $Y = U\Sigma V^T$, the system model is then

$$A = \Sigma V^T D_0 V (V^T D_1 V)^{-1} \Sigma^{-1}, \qquad C = U,$$

where $D_0 = \left[\begin{smallmatrix} 0 & 0 \\ I & 0 \end{smallmatrix}\right]$, $D_1 = \left[\begin{smallmatrix} I & 0 \\ 0 & 0 \end{smallmatrix}\right]$ are shift matrices. The input noise covariance Q can also be derived from the decomposition.

After having completed the SVD, calculating the model parameters A, C, Q from U, Σ, V would require $\mathcal{O}(k^2\tau)$ additional operations. Taking the independent elements of A, C, and Q and stacking them in a vector results in a feature f_{ACQ} of size $|f_{ACQ}| = k(n^2 + \frac{1}{2}(3k+1))$. However, simply retaining the elements of U, Σ, V as the feature $f_{U\Sigma V}$ saves additional computational cost and results in a feature of size $|f_{U\Sigma V}| = k(n^2 + \tau + 1)$. Generally, $n^2 \gg \tau \gg k$, so that the increase in feature size is negligible, and both features have size $\approx kn^2$. Thus we choose the dynamic texture feature f_{DT} to be

$$f_{DT} = f_{U\Sigma V}.$$

The raw image sequence contains $|f_{raw}| = \tau n^2$ elements, so that the feature size reduction achieved by extracting the dynamic texture is in the ratio $\frac{k}{\tau}$.

The number of states k to use in the model may be empirically determined by thresholding the singular values [1]. In choosing the threshold value, there are the competing objectives of capturing enough detail to define the dust texture, and reducing the feature size and thus computation time. Figure 6 shows the effect of thresholding the number of singular values retained. Ultimately, however, the best model is not the one that provides the most recognizable reconstruction, but the one that represents the phenomenon in minimal size and in a manner that enables high accuracy recognition.

2.1.3 Implementation

Having extracted from the raw data two features, f_{ME} and f_{DT}, which highlight dust clouds in the video stream, we combine them into a single vector f so that

$$f = \begin{bmatrix} f_{ME} \\ f_{DT} \end{bmatrix}.$$

This feature has size

$$|f| \approx \left(\frac{\tau}{d^2} + k\right) n^2.$$

Choosing values for the parameters τ, d, k, and n required weighing the fidelity of the information in f against its computation time and size, as well as other operational constraints. Since, given the vehicle bandwidth, the appearance of dust needs to be recognized within about 3 s, and since we normally sample the video stream at 10 Hz, we used the sequence length $\tau = 30$. We found that for typical video, a downsample factor of $d = 8$ accurately picked up dust fronts while removing noise. To reduce n and to localize the appearance of dust, we divided the area of interest of the video sequence into neighborhoods. We chose each neighborhood to be a block with side length $n = 32$, and determined that about $k = 8$ states suffice to adequately model the dust. These two parameters were also chosen to give adequate computation time for the real-time requirement. With these parameters, we achieve $|f| \approx 8 \times 10^3$, as contrasted with $|f_{raw}| \approx 3 \times 10^4$.

The average computation time for the SVD on a sequence of seafloor video, given these values of τ, n, and k, is 0.40 s on an Intel® XEON™ 2.00GHz processor using the ARPACK libraries via MATLAB®. A MATLAB® implementation of the moving edge filter requires 0.1 s to compute, so that extracting features from 4–5 neighborhoods in the 3 s sequence sample period is feasible with the current configuration.

2.2 Learning

After extracting the moving edges in the dynamic texture of neighborhood in the view, the recognition is performed by a model learned from instances of stock video of the seafloor including and without dust in view. To perform the learning, we use a support vector machine (SVM) framework, as it is well suited to this type of application.

2.2.1 SVM Learning

SVM learning [18][3], first introduced by Boser et al. [2], has become a fairly mature field, with numerous libraries and implementations available. A support vector machine is a *maximum margin classifier* [15]. Given a set of k training data $\mathcal{F} = \{f_1, \ldots, f_k\}$, with corresponding target labels $y_1 \ldots y_k, \quad y_i \in \{-1, +1\}$, an SVM classifies a vector space V into two regions, one denoted V^+, the other V^-. The division occurs at a *decision boundary*, which in the case of SVM is a separating hyperplane

defined by a normal vector w and intercept b. That is

$$V^+ \triangleq \{f : \langle w, f\rangle + b > 0\},$$
$$V^- \triangleq \{f : \langle w, f\rangle + b < 0\}.$$

The function $\gamma(f) = \langle w, f\rangle + b$ is the *functional margin*. The maximum margin classifier seeks to maximize the smallest distance from the training data vectors to the learned decision boundary, i.e. to solve—under appropriate scaling constraints on w and b—

$$\max \min_{f_i \in \mathcal{F}} |\gamma(f_i)|$$

subject to

$$y_i \gamma(f_i) \geq 1 \quad \forall i.$$

Thus it achieves optimal separation between $+1$ training examples and -1 training examples.

In the simplest case, the features are linearly separable and a simple hyperplane can be found as the decision boundary. However, in the more general case where features of a classification are not clumped in a linearly-separable manner, SVM employs regularization to deal with statistical outliers and kernel methods [10][14] to achieve linear separation in a mapped and/or higher-dimensional space.

Regularization involves modifying the objective function to allow outliers. The constraints are relaxed so that

$$y_i \gamma(f_i) \geq 1 - \xi_i \quad \forall i$$

and the optimization becomes

$$\max \left(\min_{f_i \in \mathcal{F}} |\gamma(f_i)| - C \sum_i \xi_i \right).$$

C is the regularization parameter which controls the weighting between maximizing the margin and penalizing outliers. Thus noisy datasets with indistinct boundaries can be accommodated.

To deal with non-linear decision boundaries, SVM employs *kernel functions*. A kernel K on two feature vectors f_1, f_2 is an inner product on mappings ϕ of the features:

$$K(f_1, f_2) = \langle \phi(f_1), \phi(f_2)\rangle.$$

The great advantage of kernels is that the mapping generally need not be explicitly computed, and functions can be proven to be kernel functions without any knowledge of the mapping they represent. For example, the polynomial kernel

$$K_p(f_1, f_2) \triangleq (f_1^T f_2 + c)^d$$

represents a mapping $\phi(f)$ into the space with elements consisting of all monomials of elements of f up to degree d—an $\mathcal{O}(|f|^d)$-dimensional space. However, computation of the kernel requires only $\mathcal{O}(|f|)$ time. The Gaussian kernel

$$K_G(f_1, f_2) \triangleq e^{-\frac{\|f_1 - f_2\|^2}{\sigma^2}}$$

requires the same computation time as K_p, and represents an ∞-dimensional mapping.

The learning procedure for SVM is an optimization in the dual space of the maximum-margin problem. A Lagrange multiplier α_i, associated with the constraint on the ith vector, will only be non-zero for those vectors closest to the boundary. These vectors are the *support vectors*. If an SVM contains m support vectors f_i and uses kernel $K(f_1, f_2)$, the functional margin becomes

$$\gamma(f) = \sum_{i=1}^{m} \alpha_i y_i K(f_i, f) + b.$$

Thus to permit real-time evaluation, m should be made as small as possible while keeping $K(f_1, f_2)$ rapidly computable. These are generally competing criteria, as simple kernels often result in less separable spaces with many support vectors (in the regularized version of SVM, the vectors on the wrong side of the decision boundary also become support vectors).

A feature of SVMs which makes them particularly attractive for our application is that they intrinsically calculate a confidence in the classification. The greater the margin between a feature and the decision boundary, the more confident the classification. Thus we can define a confidence χ_{SVM} in the classification as

$$\chi_{\text{SVM}} \triangleq g(|\gamma(f)|)$$

where g is a monotonically increasing function.

2.2.2 Cross-Validation

When using any supervised machine learning method with flexibility in the model parameters, such as the choice of kernel in SVM, care must be taken not to overfit the model to the training data, but to allow it to generalize well to samples outside the training set. A simple example of this is polynomial regression for two-dimensional data. A straight-line fit will likely have a large error on a given data set, while fitting with a very high order polynomial will eliminate error relative to the given data, but will not generalize well to new data points. The fit with the best generalization is of a degree somewhere in between.

To evaluate the model accuracy while avoiding overfitting, we employed k-fold cross validation [17]. This involves splitting the data set into k subsets. A set of model parameters is chosen for evaluation. Then training is performed on the union of the last $k-1$ subsets and the resulting model is tested on the first subset. Thus the

model is tested on different data than those from which it was derived. Using the same model parameters, this procedure is repeated for the remaining subsets until all have been exercised as both training and test data. The error rate for the particular choice of model parameters is then the combined rate for all the test sets.

2.2.3 Implementation

From the point of view of a real-time recognition capability, the primary question when implementing SVM learning is the choice of kernel $K(f_1, f_2)$. The kernel should be rapidly computable while giving large separation between dissimilar features. The two features we consider could require different kernels.

For the dynamic texture feature f_{DT}, [13] suggests a distance metric, the Martin distance $d_M(M_1, M_2)$, between dynamic texture models based on the principal angles between their observability matrices $O(A, C) = \begin{bmatrix} C^T & A^T C^T & (A^T)^2 C^T & \dots & (A^T)^k C^T \end{bmatrix}$. They also show experimentally that nearest-neighbor grouping based on d_M achieves successful grouping of similar image sequences 90% of the time. Thus, an appropriate kernel might be the Gaussian of the Martin distance: $K(M_1, M_2) = e^{-\frac{d_M(M_1, M_2)}{\sigma^2}}$. However, computation of the principal angles requires

1. computing A from the SVD: $\mathcal{O}(2k\tau^2)$ operations,
2. computing O_1, O_2: $4k^3n^2$ operations,
3. computing principal angles: $5n^2k^2 + 7k^3$ operations [7].

d_M is computed from the principal angles in $\mathcal{O}(k)$, so that this kernel, while mapping our features to a very separable space, does so at a computational cost of $\mathcal{O}((4k^3 + 5k^2)n^2) \approx 10^7$ operations. It may be possible to compute this kernel more efficiently by making certain assumptions, or by refining the algorithm. However, we have not pursued this line of research.

Instead, for computational efficiency, we used the polynomial kernel $K(f_1, f_2) = (f_1^T f_2 + c)^d$, where f_i contains both the dynamic texture feature and the edge detection feature. This kernel requires only $\mathcal{O}(|f|) \approx 8 \times 10^3$ operations to compute. However, the parameters c and d are not analytically determined. We would expect the polynomial order d to be at least 3, since [13] finds a simple geodesic distance to be a very poor indicator of similarity of dynamic textures, suggesting that the features are not simply clustered in the basic space. However, there is no other specific information on these parameters and they required empirical determination via cross-validation of learned models.

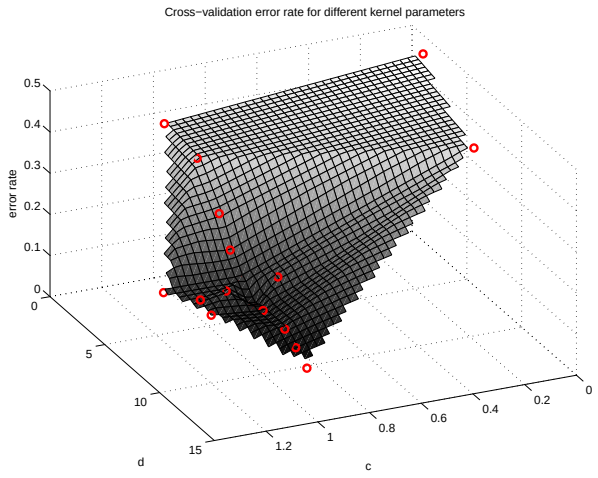


Figure 7: Cross-validation error obtained by performing 8-fold cross-validation of SVMs trained on our test data using various kernel parameters. Circles indicate particular model parameters tried. The lowest error rate obtained so far is 10.7%, though the graph indicates there may be a lower minimum.

3 Results

To create a training set for the SVM, we generated a library of video clips of the seafloor containing a total of 5:45 min of video. We set up an 8×4 grid of 32×32 pixel neighborhoods in the field of view. We then had human users go through the video in 1 s increments and mark the status of each neighborhood as always dust-covered, sometimes or somewhat dust-covered, or never dust-covered. The human labels were averaged over 3 s sequence lengths and thresholded to determine a label of $+1$ for dust, -1 for no dust, or the sequence was ignored if it did not clearly fall into the two categories. The resulting training set contains 1343 vectors, 712 containing dust and 631 without dust.

To carry out the actual learning process we used Ge's implementation [6] of the sequential minimal optimization for SVMs developed by Platt [12]. Training was performed on a variety of processors and architectures, with training times in the range of 6-52 hours for a single model. 8-fold cross-validation was used to find the best parameters c and d for our kernel, as well as for the regularization parameter which penalizes outliers. Thus determining the error rate for a single set of parameters required multiple weeks of run time in some cases. Figure 7 shows the resulting cross-validation error as a function of our kernel parameters. The lowest error rate achieved so far is 10.7% for a kernel of order $d = 14$, which compares favorably with the 90% recognition rate for the more complex decision metric in [13], though it

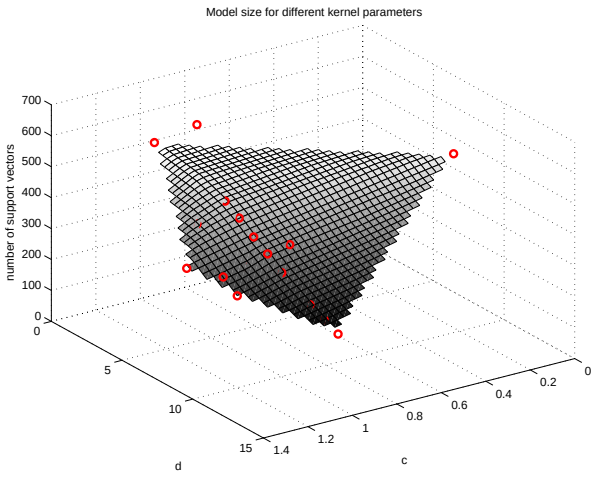


Figure 8: Average model size for 8-fold cross-validation of SVMs trained on our test data using various kernel parameters. Circles indicate particular model parameters tried. The fact that model size correlates well with error rate indicates that the models are too restrictive and not over-fitted. The smallest model so far contains 233 support vectors.

must be noted that this is a restricted, binary classification.

We found that when looking at the same data, untrained human users disagreed about 33% of the time on the classification dust/no dust. Users more experienced with looking at this type of seafloor video disagreed on 10.5% of the examples. Thus, purely in terms of the error rate, we have achieved similar performance to trained humans. Section 3.1 briefly discusses ways in which the performance is *not* up to the level of trained humans.

For the model parameters we tried, we also found that models with lower error rate contain fewer support vectors, making them more rapidly computable (see Figure 8). This indicates that the models are not over-fitted, as it shows there is a large number of outliers on the wrong side of the decision boundary. If the cross-validation error were to increase as model size decreased, an over-fitted decision boundary with few outliers but bad generalization would be indicated. Thus the models may benefit from some more degrees of freedom, e.g. by increasing the polynomial order d of the kernel.

Figure 9 shows the result of applying the best learned model to a set of neighborhoods not in the training database. To calculate confidence values, we used $g(x) = a\,\mathrm{sat}(x, b)$ where the gain a and saturation threshold b were empirically determined to scale the hypothesis values to percent confidence for most examples.

For this example, we covered a large portion of the field of view, though this would be impractical in a real-time implementation, as each frame required almost 4 times the sample period to compute. Tracking fewer neighborhoods in the view would render real-time computation feasible.

The SVM shows high confidence in well-lit areas and areas with good texture, and is less sure of the classification where lighting is poor. Thus it is a good complement to the correlation confidence χ_I of Section 1.2, as false lock due to the presence of dust clouds can be signaled with high confidence, overriding the vision correlation measurement.

3.1 Future Work

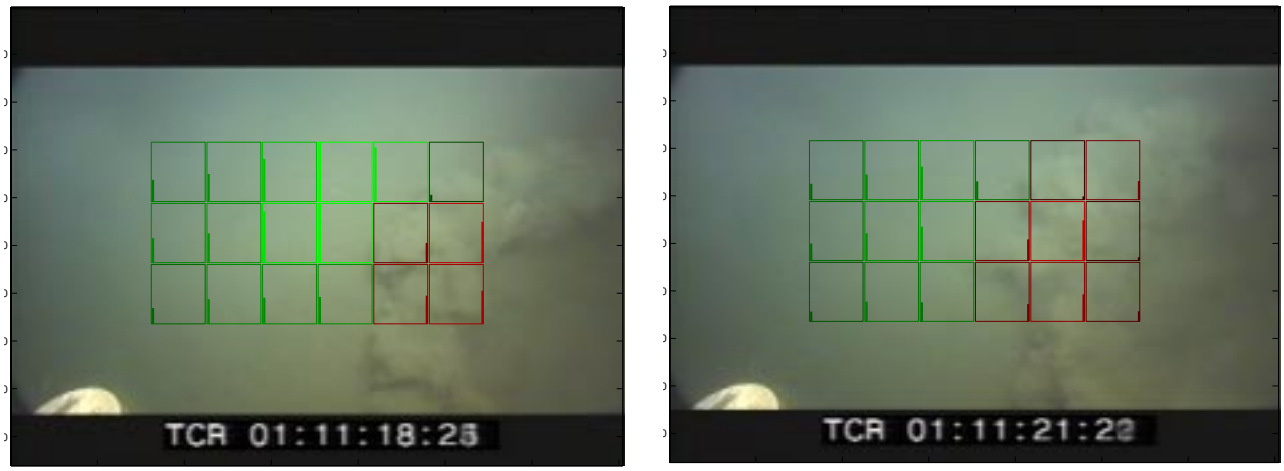
Though the basic machine learning framework and algorithms has produced acceptable results, improvements can be made to both feature selection and training set composition. Figure 10 shows that the learned model can have trouble distinguishing between seafloor features and dust clouds if the vehicle, and thus camera image, is moving. The current training set contained a preponderance of smooth, sandy sea bed without many outcroppings or seafloor features. Thus the sort of data of Figure 10, with much rigid texture, falls outside the primary range of experience of the model. However, increasing the training set size to encompass a more general range of experience will lead to impractically long training times. This in turn is due primarily to the still large feature size.

Reducing the feature lengths would greatly speed both learning and computation time. Moving edges and dynamic textures produce acceptable results, but the vector size restricts the number of training examples that can be computed upon in reasonable time. Thus, in our case, we have many fewer training vectors than the dimensionality of the search space, leaving large regions unexplored. Principal component analysis of the training set may reduce its dimensionality and highlight its most important components.

Finally, the estimator framework to fully merge the vision and DVL measurements using the more accurate `have_lock` signal must be developed, along with the capability to reinitialize the vision pixel disparities with the DVL position measurement after a period of loss-of-lock.

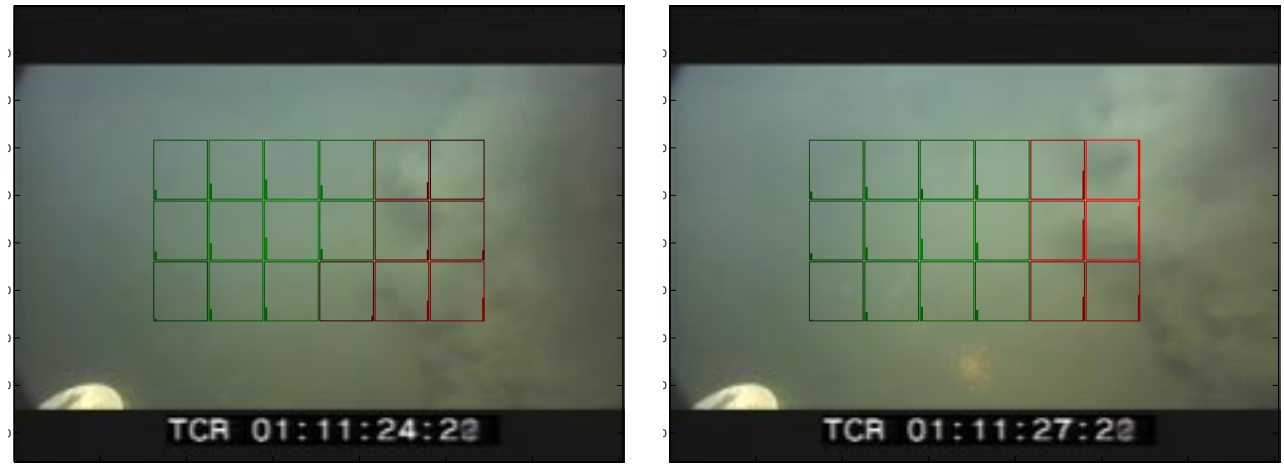
4 Conclusion

In order to increase the autonomy of our benthic station-keeping system, we have developed a capability to recognize clouds of dust obscuring the camera's view of the seafloor, with computation times fast enough for real-



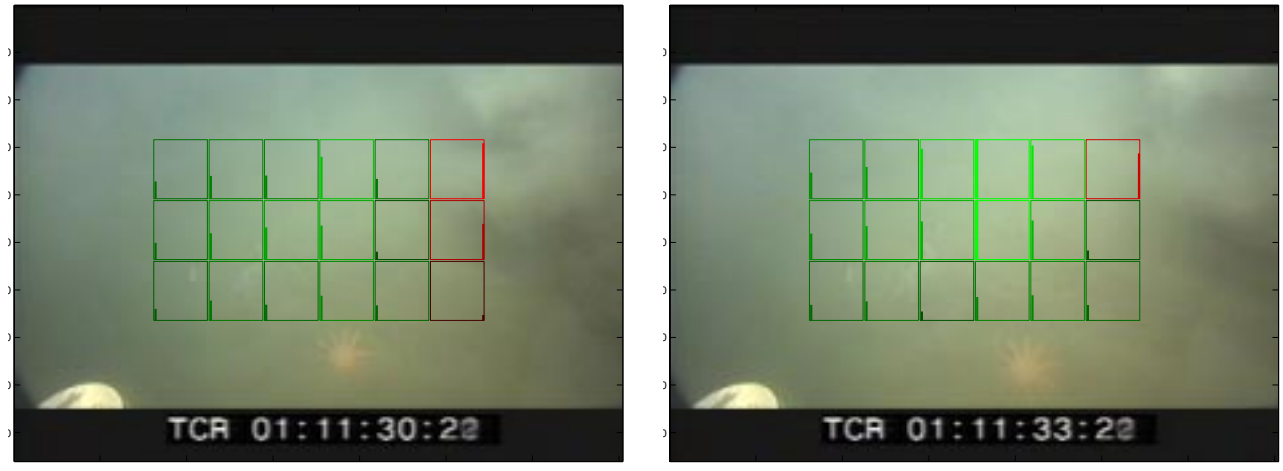
(a) Frame 1

(b) Frame 10



(c) Frame 20

(d) Frame 30



(e) Frame 40

(f) Frame 50

Figure 9: Learned model applied to sequence of Figure 1. A neighborhood with a red frame and confidence bar on the right indicates dust, a green frame with confidence bar on the left indicates no dust. Height of the bar and color intensity indicate relative confidence. Each frame shown required an average of 11.1 s to compute in our MATLAB® implementation on an Intel® XEON™ 2.00GHz processor. The bright object in the lower left is a mounting plate for the DVL.

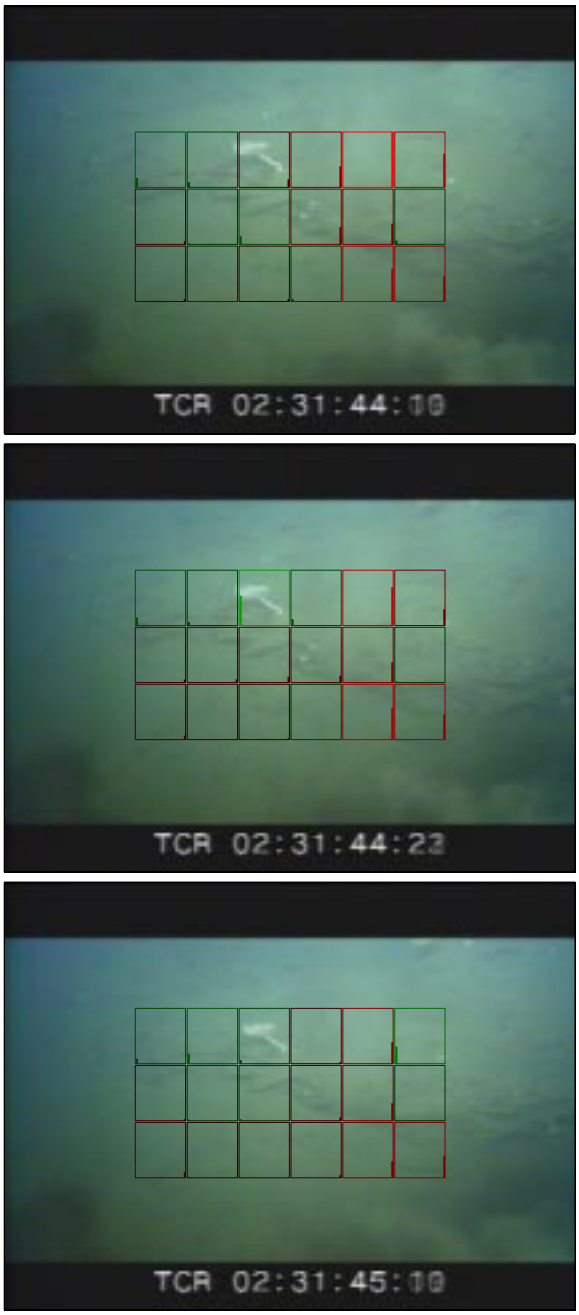


Figure 10: Learned model applied to a scene with a rock shelf bearing some underwater sea life. A faint dust cloud is correctly detected in the upper right. However, the vehicle was moving during the sequence, to that a portion of the rock shelf in the lower right is also detected as dust and falsely marked.

time use. By indicating how much confidence to place in the vision system measurements, this capability allows us to fully integrate our current visual position sensing system with the capabilities of a complementary sensor such as the DVL. The reliability of the vision measurement can be better judged and automatically merged with the complementary measurements in an estimator. This enables greater autonomy for the positioning system.

In order to find an improved vision confidence measure, we employed machine learning techniques to determine the presence of dust clouds in the view which might confuse the vision system. On the training data we gathered, these techniques resulted in a 10.7% error rate in detecting neighborhoods of dust in the view. This is comparable to the error rate achieved by trained human users. However, as shown in Figure 10, the learned models did not perform as well on some sample data somewhat outside the range of experience of the training set. Future work will aim at refining the learned models to be more generally applicable, as well as incorporating the deduced signal into a more robust state estimator.

References

[1] K. S. Arun and S.-Y. Kung. Balanced approximation of stochastic systems. *SIAM Journal on Matrix Analysis and Applications*, pages 42–68, Jan. 1990.

[2] B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual ACM Conference on Computational Learning Theory (COLT 1992)*, pages 144–152, Pittsburgh, PA, July 27–29 1992. ACM.

[3] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[4] S. D. Fleischer. *Bounded-Error Vision-Based Navigation of Autonomous Underwater Vehicles*. PhD thesis, Stanford University, Stanford, CA 94305, May 2000.

[5] S. D. Fleischer and S. M. Rock. Experimental Validation of a Real-Time Vision Sensor and Navigation System for Intelligent Underwater Vehicles. In *1998 IEEE Conference on Intelligent Vehicles*, Stuttgart, Germany, October 1998. IEEE.

[6] X. Ge. Sequential minimal optimization for SVM, 2001. Software available at http://www.datalab.uci.edu/people/xge/svm/.

[7] G. H. Golub and C. F. V. Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, Maryland, 1983.

[8] R. L. Marks. *Experiments in Visual Sensing for Automatic Control of an Underwater Robot*. PhD thesis, Stanford University, Stanford, CA 94305, June 1995. Also published as SUDAAR 681.

[9] R. L. Marks, M. J. Lee, and S. M. Rock. Visual sensing for control of an underwater robotic vehicle. In *Proceedings of IARP Second Workshop on Mobile Robots for Subsea Environments*, Monterey, May 1994. IARP.

[10] J. Mercer. Functions of positive and negative type and their connection with the theory of integral equations. *Philos. Trans. Roy. Soc. London, Series A*, 209:415–446, 1909.

[11] H. K. Nishihara. Practical realtime imaging stereo matcher. *Optical Engineering*, 23(5):536–545, 1984.

[12] J. Platt. *Fast Training of Support Vector Machines using Sequential Minimal Optimization*. In Schölkopf et al. [14], 1999.

[13] P. Saisan, G. Doretto, Y. N. We, and S. Soatto. Dynamic texture recognition. In *IEEE Computer Society Conference of Computer Vision and Pattern Recognition, CVPR 2001*, volume 2, pages II–52–57. IEEE, 2001.

[14] B. Schölkopf, C. J. C. Burges, and S. Mika, editors. *Advances in Kernel Methods: Support Vector Learning*. MIT Press, 1999.

[15] A. J. Smola, P. J. Bartlett, B. Schölkopf, and D. Schuurmans, editors. *Advances in Large Margin Classifiers*. MIT Press, 2000.

[16] S. Soatto, G. Doretto, and Y. N. Wu. Dynamic textures. In *IEEE International Conference on Computer Vision, ICCV 2001*, volume 2, pages 439–446. IEEE, 2001.

[17] M. Stone. Cross-validation choice and assessment of statistical predictions. *Journal of the Royal Statistical Society, Series B*, 36(1):111–1147, 1974.

[18] V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.