Intelligent Image Compression and Feature Extraction

Salem Cherenet

Advisor: David Wettergreen

Co-Adviser: Tai Sing Lee

ABSTRACT

In areas that are unsafe for humans to directly study, robotic exploration is an excellent method for collecting and interpreting data. Robots can be used for exploration; but, they require operator input about what tasks to accomplish. In order for operators to plan out optimal tasking, information about the state of the rover is required. Panoramic imaging of the rover's last position useful information; however, each panorama is made up of over a hundred high resolution images, and sending this information is hard due to bandwidth limits. In this paper, we explore a way to effectively decrease the size of panoramas without losing important information.

1. INTRODUCTION

In an area which is unsafe for humans to directly study, such as the Chernobyl nuclear reactors, the World Trade Center towers, and the surface of Mars, robotic exploration is an excellent method for collecting and studying data about that area [1]. The visual system of a robot provides operators with information about the environment for safe navigation, determine its location, and detect sites of interest. In addition, the mission goals and the behavior of the operator are also other factors that improve information acquisition.

In geological survey missions that are carried out by autonomous rovers, determining location of the rover is one of the top priorities [1]. One of the ways location is determined is via study of panoramas. Panoramas are images that are made by stitching high resolution images together. For the case of geological surveys, these panoramas tend to be made up of more than a hundred high resolution images, which make it difficult to analyze everything in the scene and/or transmit the whole data to the operators.

To solve this issue, we would like to first come up with a matrix of what we consider an important feature in the image. Once we have criteria set for what an important image feature is, we would need to determine relative importance of different image features. Next we would assign different resolution values based on relative importance. This results in an image with lower file size and a distributed resolution map (saliency map) that has high resolution in areas where important features exist, low resolution where there are no important features.

In this paper we use deep auto-encoders and TextureCam random forest classifier, compress images by over 90% and generate saliency map.

2. RELATED WORK

In 2008, Glasgow et al published an information optimization model that minimizes bandwidth needs while

maintaining information quality [1]. This model was based on two-year case study of an astrobiology field test.

The proposed model by Glasgow et al started by identifying targets; targets are areas which are of interest. Once targets were identified, they were classified into a smaller group of classes based of probability of detection. The main reason for this was that the number of targets per mission could be very huge and optimizing the problem for each target would be a very difficult [1].

Once the classification was done, the next step was to assign values for each class, A_k. The assignment of values was based on scientists' behavior while completing the specified mission. Once each target class has an assigned value, the next step was to compute the probability of detecting each target class in different regions. These regions under consideration were classified based on the robot's camera elevation angles. Finally, an information rate was computed using (2), which is found by taking the derivative of (1).

$$\max \frac{\sum_{j=0}^{m}\sum_{k=0}^{s} V(A_k)p(A_k|B(r_j))}{\sum_{j=0}^{m} B(r_j)} \qquad (1)$$

$$\dot{I} = \frac{dV}{dB}\frac{\sum_{j=0}^{m}\sum_{k=0}^{s} V(A_k)p(A_k|B(r_j))}{\sum_{j=0}^{m} B(r_j)} \qquad (2)$$

Where $V(A_k)$ is the value assigned to each target class as shown in Table 1, r_j is each region classified based on the robot's camera elevation (Table 2), and $B(r_j)$ is the number of bits used to acquire a data in region r_j.

TABLE 1: List of Target classes and their approximated values based on observations during the LITA filed tests.

Target Class (A_k) Value		(0-1)
0	Sun	1
1	Clouds	0.8
2	Local Topographic Highs (i.e. mountain hills)	0.9
3	Slopes/Drop-offs (Difficult to traverse regions)	0.5
4	Drainages/Channels	0.75
5	Rocks > 1m	0.3
6	Rocks < 1m	0.3
7	Sediment	0.3

TABLE 2: List of regions and their associated camera elevation angles [1]

Region (r_j)	Elevation
0	-90
1	-76
2	-62
3	-48
4	-34
5	-20
6	-6
7	8
8	22

3. BACKGROUND

3.1 Machine learning techniques for data classification and regression

There are several different approaches to effectively analyze and classify data. However, depending on the type of data, the amount of data, and the different conditional constraints one method might be found more efficient and/or suitable than other.

In this paper we are used a random forest classifier [2] and Hinton et al auto-encoder [3] for region classification and feature detection.

In the next few subsections I would like to introduce some machine learning methods so that the reader would have an understanding of the later sections.

3.1.1 Naïve Bayes Classifier

The Naïve Bayes algorithm is based on conditional probabilities by applying the Bayes' theorem (given by equation 4).

$$P(\mathrm{Y}|\mathrm{X}) = \frac{\mathrm{P(X|Y)}}{\mathrm{P(X)}} * \mathrm{P(Y)} \qquad (4)$$

The Bayes' theorem calculates a probability of an event occurring given a probability of a prior event. This model is naïve because it assumes the attributes are conditionally independent [5]. Since the denominator does not depend on Y, we usually are only interested in the numerator. Doing some mathematical manipulation on equation 4 results in the following relationship

$$P(\mathrm{Y}|\mathrm{x}_1, \mathrm{x}_2, \ldots \mathrm{x}_n) = P(Y) * \prod_{i=1}^{n} P(\mathrm{x}_i|Y) \qquad (5)$$

For the purpose of classification, we apply a decision rule based on (5). One of the most common rules used for classification in Naïve Bayesian method is called the maximum a posteriori (MAP) decision rule. The MAP decision rule takes the most probable hypothesis as the classification. Mathematically this can be expressed as:

$$Class(\mathrm{x}_1, \mathrm{x}_2, \ldots \mathrm{x}_n) = \underset{y}{\mathrm{argmax}} \; P(Y = y) * \prod_{i=1}^{n} P(\mathrm{x}_i|Y = y) \qquad (6)$$

3.1.2 Artificial Neural Network

Artificial Neural networks are computational network models that function in similar ways as natural neural networks. Natural neurons take in information through synapses and emit a signal through an axon to another neuron (see Figure 2). The transmission of the signal occurs only if the received signal is greater than the threshold for the activation of the neuron.

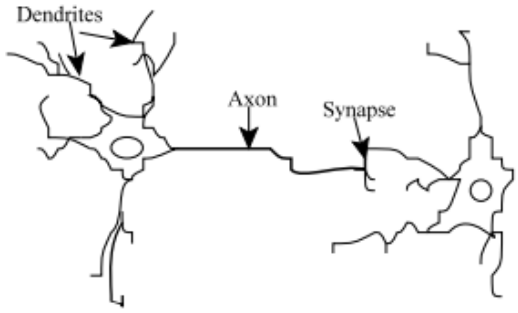

Figure 1 - Natural neurons (artist's conception) [4].

The first artificial neural model was introduced by McCulloch and Pitts (1943) [4]. Since then there have been several improvements to make it more robust. Figure 3 shows general schematics of how an artificial neural network (ANN) is set up. Figure 4 shows the same setup with a single neuron in the hidden layer. Each connection between the layer nodes is weighted. The weights can be positive or negative. A negative weight implies that the signal is inhibited whereas a positive weight implies the signal is amplified [4].

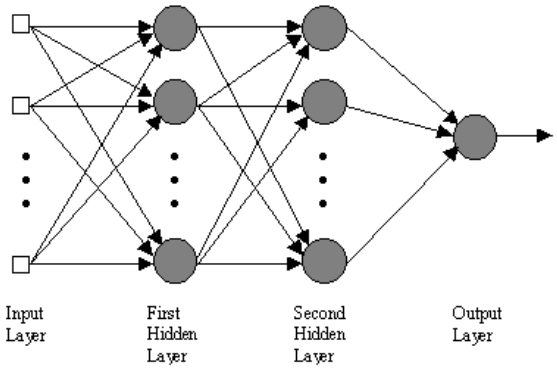

Figure 2 - ANN with two hidden layers [6].

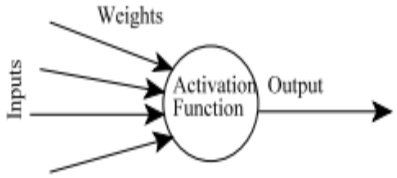

Figure 3 - An artificial neuron [6].

3.1.3 Decision tree learning

Decision tree learning is the simplest form of inductive learning algorithm. In decision tree learning, several inputs which are described by a set of attributes are given to the algorithm. The algorithm then performs a sequence of tests on the inputs and predicts an output for the given input [5]. The way decision tree works is very similar to how humans make decision in day to day basis.

In the decision tree algorithm, the examples, attributes and default values are given as a matrix input. Before running any recursion learning, a decision tree algorithm checks if all the default outputs are the same. If that is the case, then no matter what the value of each attribute is the output is going to be that output. If all the inputs are empty, the algorithm returns the default value. If the above two cases are not met then the algorithm runs recursively to determine what is the best attribute. The best attribute is defined as the attribute that gives the most information gain to the decision making. In other word, we want to choose an attribute that would minimize the steps taken to reach to a decision [5]. Mathematically this is given by (1).

$$IG(Y|X) = H(Y) - H(Y|X) \quad (1)$$

Where IG – Information Gain
H(Y) – the entropy of Y (equation 2)
H(Y|X) - the entropy of Y given X (equation 3)

$$H(Y) = -\sum_{j=1}^{m} P_j log_2^{P_j} \quad (2)$$

$$H(Y|X) = \sum_{j=1}^{m} P(X = v_j) * H(Y|X = v_j) \quad (3)$$

Where P is the probability, v_j is the individual attributes, and H(Y|X=v_j) is the conditional entropy of Y given a specific value of X.

Decision tree algorithm can deal with both discrete and continuous inputs and outputs. The learning process for discrete input and discrete output is called classification and for continuous case it is called regression [5].

3.1.4 TextureCam

In this paper we used a random forest, variation of Decision Tree method, based classification code, TextureCam.

TextureCam has been trained and tested on panoramic images from mars exploration rover (MER) for robustness. D. Thompson et al. published the Receiver Operating Characteristic (ROC) curve shown in Figure 3.

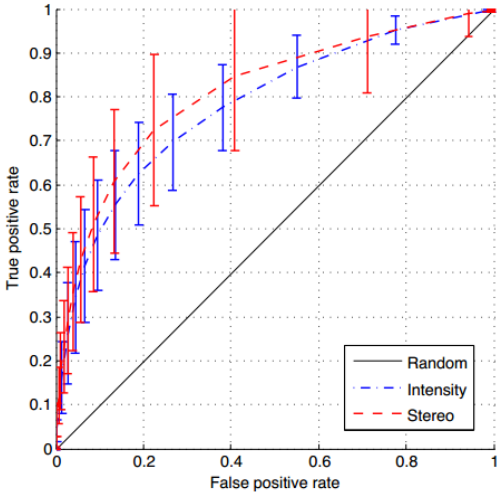

Figure 6: Performance ROC for the Legacy panorama. The classifier was trained on the Mission Success panorama. Both datasets consist of 23 images. [2]

3.1.5 Support Vector Machine

Support vector Machine is an algorithm that is based on Vapnik-Chervonenkis theory. SVM has a strong theoretical foundation [9]. SVM is a kernel-based algorithm (i.e. SVM takes an input that needs to be classified, and by applying a kernel function, it transforms the input into a higher dimension so that the problem is solvable). Just like all the aforementioned algorithms, Support Vector Machine can also do both classification and regression. It can also handle binary or multiclass targets [10].

3.2 Feature detection and extraction

In the past few decades several new methods have been used in computer vision applications. Most of these methods are variations of supervised learning that requires labeled data. For this paper we used an unsupervised learning method called deep auto-encoder [3]. A deep auto-encoder is made up for stacks of Restricted Boltzmann Machine (RBM) layers for pretraining, any it learns nonlinear principal component analysis (PCA). Pretraining is essential because for relatively large networks, backpropagation gives much worse results if no pretraining is used [11].

3.2.1 Restricted Boltzmann Machine (RBM)

A restricted Boltzmann machine is a stochastic neural network system with a visible and a hidden layer [7]. Each neuron on the visible layer will be weighted and pass through some function (usually a sigmoid) to create the neurons on the hidden layer. Figure 1 shows a simple setup of a restricted Boltzmann machine. Where red represents the visible layer and blue represents the hidden layer. Each circle is called a neuron.

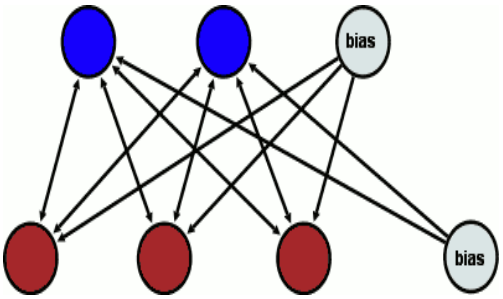

Figure 1: Restricted Boltzmann Machine [7]

3.2.2 Deep Belief Auto-Encoder

Deep Auto-Encoder [8] is also a form of an artificial neural network that tries to perform dimensional reduction via nonlinear PCA method. The input and the output of an auto-encoder have the same meaning, whereas the middle layer is composed of lower number of neurons. This method is a unsupervised learning where it reduces the number of neurons form layer to layer until it reaches a specified middle layer then tries to reconstruct the input image from the neurons in the middle layer ("code layer" in Figure 2). Figure 2 describes this process briefly.

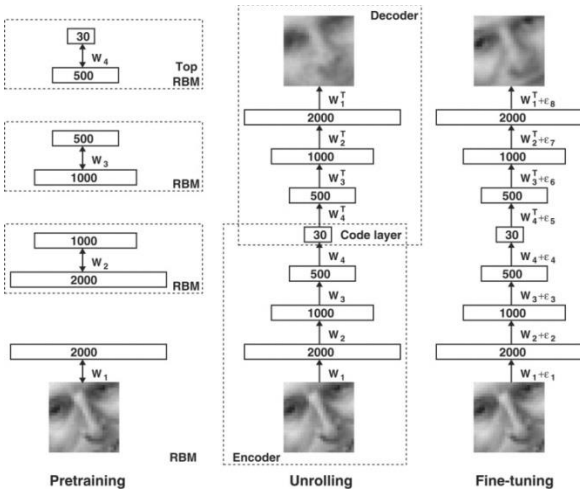

Figure 2: Pre-training consists of learning a stack of restricted Boltzmann machines (RBMs), each having only one layer of feature detectors. The learned feature activations of one RBM are used as the "data" for training the next RBM in the stack. After the pre-training, the RBMs are "unrolled" to create a deep auto-encoder, which is then fine-tuned using backpropagation of error derivatives [8]

3.3 Classifier Performance

Classifier's performance is measured by taking the ratio of correctly classified datasets to datasets trained or tested. An accuracy percentage can also be calculated from a confusion matrix by summing the diagonal values and dividing it by the number of cases. A confusion matrix is a table that visually shows a performance of an algorithm. Table 3 shows an example of such a table.

Table 2: Confusion Matrix

		Predicted class		
		Rainy	**Sunny**	**Windy**
Actual class	**Rainy**	5	3	0
	Sunny	2	3	1
	windy	0	2	11

4 METHOD

The approach taken in this paper combines local and global classification, and feature detection. Here global refers to classification done on each panorama, whereas local classification and feature detection is done on each image (each image, which make up a panorama, will be referred as an "Image Patch" from here on out).

4.1 Data Acquisition

The data for this project was acquired from Carnegie Mellon Field Robotics center's LITA group. These data were collected during the two year geological survey, between 2003-05, in the Atacama Desert, Chile. Figure 3 below shows a typical panorama looks like.

Figure 3: Panoramic image at 3% resolution

4.2 Local Classification and Feature Detection

A. Preprocessing

An auto-encoder network requires input images as row vectors where each pixel is treated as a single unit in the visual layer. Since each patch is high resolution (1280 by 920 px) generating a training and/or testing data would result in a huge matrix. Therefore, for each global class (see section 4.3), we converted each patch into a 32 by 32 grayscale image and rotated from -90 deg to 90 deg and scaled it from 0.4 to 1 creating about 427 images per patch. This results in an average of 8000 images per class.

B. Pre-training and Fine-tuning

To reduce computational cost, all datasets were subdivided into several mini-batches. For all datasets, each hidden layer was pre-trained for 200 passes through the entire training set. The weights were initialized randomly and were updated after each mini-batch [12]with a learning rate of 0.001, weight cost of 0.0002, initial momentum of 0.5 and final momentum of 0.9 (see hinton code for details http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html)

Once the network goes through several layers of RBM, it performs backpropagation via gradient decent method to adjust the weights.

C. Classification

For each global class (sec 4.3), the nine neurons at the code layer would be used to train Support Vector Machine (SVM). Once training is done, the SVM will be given each of the nine neurons of a specific global class we are interested in. This results in each of the code layer neurons being classified into different global classes.

D. Local Saliency Map

After the classification is done, we take three out of the nine neurons to construct a mask. The mask is a binary form of the selected neurons.

4.3 Global Classification

The global classification is done at a full panorama level. Each panorama would be color labeled following Glasgow's relative value matrix (Table 1). Table 3 shows the color label for each class.

Once few panoramas have been trained using TextureCam [2], then any new/unseen panorama could be classified with high accuracy. Figure 4 and Figure 5 show a typical label panorama and a TextureCam classification panorama respectively.

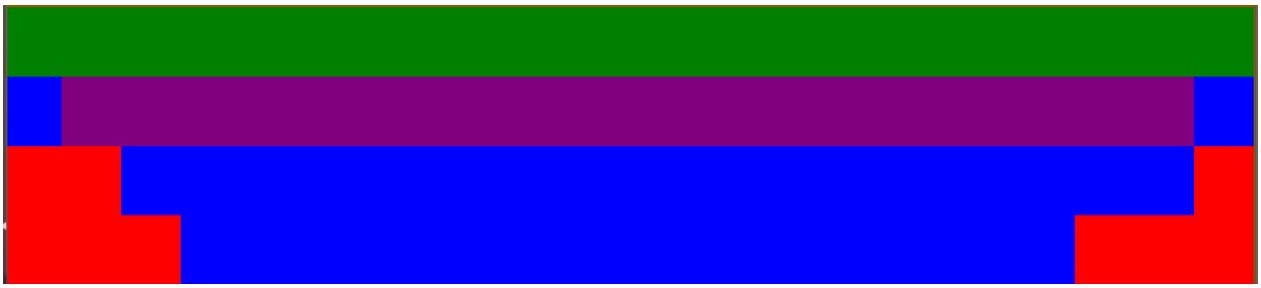

Figure 4: Label data for the panorama in Figure 3

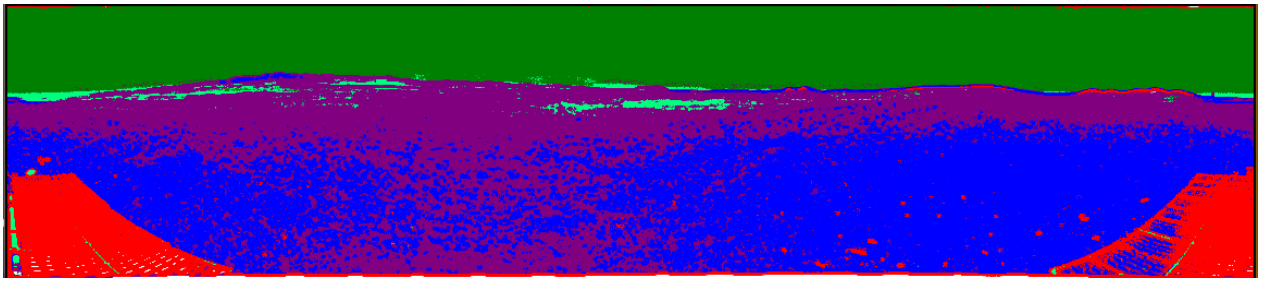

Figure 5: TextureCam classification for Figure 3 and Figure 4

Finally, the output label image will be used as a global saliency map for resolution adjustment (section 4.4).

Table 3: Global Classification Color Code

Class name	Class color	Class label
Black patches (artifact of Gigapan stitcher)	Black	0
Multiple classes in same patch	White	1
Rover parts	Red	2
Sky	Green	3
Rocks & Sediment	Blue	4
Slops & drop-offs	Yellow	5
Drainage/Channels	Brown	6
Clouds	Orange	7
Mountains	Purple	8
Sun	Spring Green	9

4.4 Resolution adjustment

The resolution adjustment begins with reducing the whole panorama to 50% resolution.

Next, based on the global classification, each class will be assigned different resolutions ranging from 6% to 50% based on its relative importance (Table 1).

Even though the resolution adjustment based on global classification does a good job reducing file size, it fails to discriminate possible informative features within class. Therefore, further resolution adjustment will be done on each patch for classes with high resolutions.

This is achieved by taking each mask, and using them to select only portions of the data from a Laplacian image while performing Gaussian pyramid expansion.

4.5 Stitching

The last stage of this process is to stitch the resolution adjusted image patches back together. Since the raw images from the rover have overlapping regions, we used a commercial stitching brand, GigaPan, for stitching each panorama. After the initial stitching, each panorama was broken into smaller patches in order to perform local and global classification.

5 RESULTS

5.1 TextureCam Performance

TextureCam was trained and tested on three panoramas. Each patch was given a single label based on Table 3. In cases where a patch contained one of the classes and part of the rover's body, it was given a label 1. Table 4 and Table 5 below show the confusion matrix, and the training squared error, and testing squared error.

Table 4: Summary of TextureCam Training performance on Panoramic Images

Confusion Matrix				Accuracy
8	0	1	0	88.09
0	21	0	0	
1	0	26	8	
0	0	0	19	

Table 5: Summary of TextureCam Test performance on Panoramic Images

Confusion Matrix					Accuracy
2	0	3	0	0	82.54%
0	19	0	0	0	
0	0	16	0	0	
0	0	7	14	0	
0	0	0	1	1	

5.1 Deep Auto-Encoder Performance

The deep auto-encoder was used to generate features for SVM. The robustness the auto-encoder was measured by its reconstruction error. The following is a summary of testing and training reconstruction squared errors for the network. Each data class had 8967 individual images of which 7000 were used for training and the rest was used for testing.

Table 6: Summary of Deep Auto-Encoder Network

Data class	Train Squared Error	Test Squared Error
Class 3	0.553	1.552
Class 8	2.113	3.950
Class 4	2.017	3.136
Class 2 & class 4	2.0972	11.705
Class 0	0	0
Class 0	0	0

5.2 SVM Performance

The datasets used for SVM were 63, 32 by 32 images, which were taken from 2 panoramas. The training and testing accuracies were 78% and 34% respectively.

5.3 Image File Size Reduction

Table 7: Panorama image file size for different levels of image compression

Type	File size (JPEG)-MB
Original	32
Gray, full resolution	8.5
Gray half resolution	3.03
Gray half resolution with global saliency map	1.8
Gray half res with local and global saliency map	< 1*

*work in progress

6 DISCUSSION

The main goal of this paper was to reduce panoramic image's file size, and create local and global saliency maps. This was achieved by generating a global saliency map via TextureCam, and local saliency map through a combination of SVM and deep auto-encoders.

The previous section shows promising results in different aspects. As shown in Figure 5 and Table 5 TextureCam does a good job classifying the panorama with 82% of accuracy. Even though the label image (Figure 4) had grids that have two classes, TextureCam was able to correctly classify those classes. For example, the ground between the left and right solar panels of the rover was classified as just solar panel due to the way labeling was done. However, texture cam was able to differentiate between the solar panel and the ground in those regions very well. On the other hand, there are several pixels on the panorama, especially on the ground consisting of only small rocks, where the classifier seems to have less accuracy.

Deep auto-encoder algorithm used in this paper has been used for face recognition task with 165000 images and hand written digit recognition task with 60000 images [3]. However, it has not been on panoramas. In this paper, we used close to 54000 individual images taken from one panorama. The highest test square error was observed for data class 4. Data class 4 is a class of image patches containing small (<1m) rocks, which has the least relative importance (Table 1). Therefore, the reconstruction error is acceptable.

There was perfect reconstruction for data class 5 and 6. These two classes consisted of image patches with uniform color no texture. Even though, these patches are not off interest, we can still use them as a baseline test for the algorithm.

In the regions we are most interested in, (i.e. data class 2 and 3), we have reconstruction squared error less than 4. Having low reconstruction error implies that the code layer features are the most important features required for the reconstruction of the dataset.

Local salience mapping has not been completed, due to the high error rate in the SVM classifier. Only 54 images taken from one panorama were used for training the SVM. This is not an optimal way of training the SVM because not all panoramas contain all classes. We are currently working on including images from several panoramas covering all different classes in order to increase the SVM accuracy.

Image reduction was one of the two goals of this project. As shown in Table 7, applying global saliency helped reduce the size of the panorama to less than 2 MB. Work in progress shows that applying local saliency would help reduce panoramic image's file size to sub-megabyte.

In summary, we were able to reduce panoramic image's file size by than 90%, and apply local and global saliency maps to help as visual cues.

7 ACKNOWLADGEMENTS

I would like to give special thanks to David Wettegreen, Tai Sing Lee, Michael Furlong, and David Thompson for being instrumental in giving me direction throughout this project.

I would also like to thank the National Consortium for Graduate Degrees for Minorities in Engineering and Science, Inc. (GEM), SLAC National Accelerator Lab, and Carnegie Mellon Mechanical Engineering department for sponsoring my graduate study.

Reference

[1] Glasgow, J., G. Thomas, E. Pudenz, N. Cabrol, D. Wettergreen, P. Coppin (2008), "Optimizing Information Value: Improving Rover Sensor Data Collection, IEEE Trans. Systems, Man and Cybernetics, Part A: Systems and Humans, 38(3).
[2] K. L. Wagstaff, D. R. Thompson, et al., "Smart Cameras for Remote Science Survey". Unpublished manuscript.
[3] Hinton, G. E., Osindero, S. and Teh, Y. (2006)
A fast learning algorithm for deep belief nets. Neural Computation, 18, pp 1527-1554.
[4] Bertin, Emanuel. *Sextractor 1.0a User's Guide*. Paris: Institut D'Astrophysique De Paris. Pdf.
[5] Stuart Russell, Peter Norvig Artificial Intelligence. A Modern Approach, Prentice Hall.
[6] NeuroDimention. "NeuroSolutions: What Is a Neural Network?" *NeuroDimension*. NeuroDimention,inc, 2011. Web. 03 Apr. 2012. <http://www.nd.com/welcome/whatisnn.htm>.
[7] "Restricted Boltzmann Machine - Short Tutorial." *IMonad Software RSS*. N.p., n.d. Web. 06 Mar. 2013.
[8] Hinton, G. E., Osindero, S. and The, Y. (2006) A fast learning algorithm for deep belief nets. Neural Computation, 18, pp 1527-1554.
[9] Gehring, W.J., Coles, M.G.H., Meyer, D.E., Donchin, E., 1995. A brain potential manifestation of
error-related processing. In: Karmos, G., Molnar, M., Csepe, V., Czigler, I., Desmedt, J.E. (Eds.), Perspectives of Event-Related Potentials Research (EEG Suppl. 44), pp. 261–272.
[10] Oracle. "Support Vector Machines." *Support Vector Machines*. Oracle Data Mining, 2008.Web. 11 Apr. 2012. <http://docs.oracle.com/cd/B28359_01/datamine.111/b28129/algo_svm.htm>.
[11] Hinton, G. E. and Salakhutdinov, R. R. (2006) Reducing the dimensionality of data with neural networks. Science, Vol. 313. no. 5786, pp. 504 - 507, 28 July 2006.