

# Delineation of Rock Fragments by Classification of Image Patches using Compressed Random Features

Geoff Bull, Junbin Gao and Michael Antolovich

*School of Computing and Mathematics, Charles Sturt University, Bathurst, Australia*

**Keywords:** Compressed Sensing, Random Projections, Sparse Representation, Image Patches, Feature Extraction, Image Segmentation, Classification.

**Abstract:** Monitoring of rock fragmentation is a commercially important problem for the mining industry. Existing analysis methods either resort to physically sieving rock samples, or using image analysis software. The currently available software systems for this problem typically work with 2D images and often require a significant amount of time by skilled human operators, particularly to accurately delineate rock fragments. Recent research into 3D image processing promises to overcome many of the issues with analysis of 2D images of rock fragments. However, for many mines it is not feasible to replace their existing image collection systems and there is still a need to improve on methods used for analysing 2D images. This paper proposes a method for delineation of rock fragments using compressed Haar-like features extracted from small image patches, with classification by a support vector machine. The optimum size of image patches and the numbers of compressed features have been determined empirically. Delineation results for images of rocks were superior to those obtained using the watershed algorithm with manually assigned markers. Using compressed features is demonstrated to improve the computational efficiency such that a machine learning solution is viable.

## 1 INTRODUCTION

Monitoring rock fragmentation is a very important process in the mining industry. Knowledge of fragmentation can improve the economics of operating a mine through optimizing the operation of crushing equipment, and can be used to improve estimates of the volume of ore remaining in a mine. Most importantly, changes in fragmentation can alert the operators of a mine to potentially fatal conditions developing in the mine.

A standard technique to perform fragmentation analysis has been to sieve particles through progressively finer sieves and then weigh the contents of each sieve. However, increasingly image processing and statistical techniques are being applied to this problem to reduce costs and to make the collection of more data in a timely manner possible.

Most algorithms for performing fragmentation analysis on rock images rely on finding the boundaries that delineate individual rock fragments. Unfortunately simple edge detectors perform poorly at this task and more sophisticated approaches are necessary. There are a number of commercial software packages that have been widely used for analyzing frag-

mentation of rock particles on conveyor belts. One of these is based on a combination of edge detection techniques and watershed segmentation (Girdner et al., 1996). Unfortunately when used for less constrained images, e.g. of broken rock in an underground draw-point, many systems often require significant manual editing to correct false delineation of fragment boundaries, often taking more than 30 minutes per image (Demenegas, 2008). An underground mine with 300 draw-points would require a total processing time of less than 5 minutes per image if 1 image per draw-point was to be analysed per 24 hour period. Recent research (Noy, 2013; Thurley, 2013) has focussed on using 3D imaging, with promising results, to overcome the limitations of 2D imaging used in traditional analysis systems. However, upgrading the imaging systems already deployed in existing underground mines would be expensive and disrupt operations, and so there remains a need to improve the results of fragmentation analysis using 2D images.

The watershed algorithm (Beucher and Lantuejoul, 1979) treats an image like a topographic surface, and simulates flooding with water from local minima. The lines where waters from different sources meet are called watershed lines, and water is not permitted

to cross these. The watershed lines form the boundaries between image segments. A common problem with the watershed algorithm is that it is susceptible to noise and tends to over-segment images. One solution to this uses markers (Meyer and Beucher, 1990), but this solution needs the markers to be manually specified. The mean shift algorithm (Comaniciu and Meer, 1999) has been proposed a suitable method for generating markers (Amankwah and Aldrich, 2011), however that approach is computationally expensive.

Segmentation of grey scale rock images is often very challenging due to poor lighting and shadowing, and colour and texture variation, overlapped rocks, fine material and determination of scale (Thurley, 2009). Fragmentation analysis is difficult with 2D images (Thurley, 2009) and typically a human operator often must be involved in the analysis. It has been argued that 2D images contain insufficient information to differentiate between overlapping and non-overlapping rocks (Thurley, 2013). However, the fact that a human operator is able to manually edit the results (Siddiqui et al., 2009) suggests that improved algorithms may still yield better results with 2D images. Watershed segmentation has been extended by incorporating 3D surface data (Thurley and Ng, 2005) and this approach overcomes some of the issues observed with 2D images. Despite ongoing improvements to this technique (Thurley, 2009; Thurley, 2013), the fact remains that it is not always practical to collect the necessary 3D data.

A fundamental task for image processing and machine learning is the selection of appropriate features that generalize well and have a low computational overhead. Recent advances have seen Compressed Sensing (CS) (Donoho, 2006; Candes and Tao, 2005) used to learn features for image analysis and computer vision. For sparse signals, CS allows the sampling rate to be reduced well below the usual Nyquist rate while still allowing almost perfect reconstruction. Storage requirements and computational overhead are reduced accordingly. For CS, signals are “measured” by compressing them as they are acquired.

Classification in compressed space can achieve accuracies close to those achieved by classification in the original signal space (Calderbank et al., 2009). A recent example of CS in machine learning that confirms this is the use of compressed sensing features to assist in data dimensionality reduction (Gao et al., 2012). Randomly projecting data onto lower dimension subspaces has been found (Bingham and Manilla, 2001) to be as effective as conventional dimensionality reduction methods such as principal components analysis (PCA). Moreover, while random projections are significantly less computationally expen-

sive than PCA, they also do not introduce significant distortions to the data. An issue with PCA is that if the data contains outliers the projected subspace can lie an arbitrarily large distance from the true subspace (Wright et al., 2009).

CS can be used to design projections that increase the level of compression leading to individual features that are more informative than components of the original signal. For example, random feature selection has been used to get more accurate texture classification than with features that had been specifically designed for the task (Liu and Fieguth, 2012).

A recently demonstrated algorithm for tracking objects in video achieves real-time performance by using compressed features (Zhang et al., 2012). These features have been shown to be also useful for image segmentation (Bull et al., 2013). Although the generalized Haar-like feature used has a very high dimensionality, the computational burden is actually very low because features are randomly projected into a low dimensional subspace. The feature is derived from the generalized Haar-like wavelet of (Dollár et al., 2007) which is in turn derived from the Haar-like wavelet popularized by (Viola and Jones, 2002). These features are very useful because they are very sparse (Zhang et al., 2012), enabling dimensionality reduction by random projection, and they are very efficient to compute via the method of integral images (Viola and Jones, 2002).

In this paper, the use of machine learning together with compressed random features to solve the problem of finding fragment boundaries is proposed. Manually delineated images of rocks are used to train a support vector machine (SVM) (Cortes and Vapnik, 1995), and the model produced is used to predict boundary regions in test images. A compressed Haar-like feature vector is used and compared to using simple brightness patches as features. The minimum length of feature vector needed for good classification is empirically determined, and the optimum size of image patches to use for feature extraction is also found. The use of these techniques is justified by comparing the results with those achieved with the watershed algorithm.

The remainder of this paper is structured as follows: The proposed classification algorithm and the compressed Haar-like feature are described in Section 2. The experimental investigation is then presented, along with a discussion of the results, in Section 3. The results are summarized and conclusions drawn in Section 4.

## 2 THE PROPOSED APPROACH

### 2.1 Compressed Features

In this section the construction of random compressed features is described.

Compressed sensing (CS) (Donoho, 2006; Candes and Tao, 2005) is based on the idea that a signal can be reconstructed from a very limited number of measurements if the signal has a sparse representation in some basis. When such a sparse representation exists, the signal is said to be compressible. The Restricted Isometry Property (RIP) and the related Johnson–Lindenstrauss (JL) lemma (Baraniuk et al., 2008) are two well-known theorems of CS. RIP determines the conditions under which a compressed signal can be efficiently reconstructed. From JL it can be shown that if a signal is sampled using a properly designed measurement matrix the distances between signals after compression will be very close to the distances between the same signals before compression (Baraniuk et al., 2008). The proposed method relies on this observation.

The properties of compressed sensing only hold for signals that are sparse (i.e. most values being zero). However, many images are not sparse in their original domain and they must be represented in some basis where they are sparse. Consider a signal consisting of a  $n \times n$  square patch surrounding a pixel in an input image, that can be represented as an  $n^2$ -dimensional vector  $\mathbf{x} \in \mathbb{R}^{n^2}$ . If the patch  $\mathbf{x}$  is compressible, it can be transformed to a sparse vector  $\mathbf{f} = \Psi\mathbf{x} \in \mathbb{R}^D$  by a sparsifying matrix  $\Psi$ . The sparse vector will have most of the coefficients close to zero. Usually  $D \gg n^2$ , in our case  $D = n^4$ , and it would be very inefficient to perform computations directly on the sparsified vector. To make the calculations tractable, a dimensionality reducing transform  $\Phi$ , is applied to reduce the signal down to a  $k$ -dimensional vector

$$\mathbf{y} = \Phi\mathbf{f} = \Phi\Psi\mathbf{x} \in \mathbb{R}^k. \quad (1)$$

$\mathbf{y}$  is called a Compressed Sensing Feature and  $\mathbf{f}$  the Sparse Coding Feature (Gao et al., 2012).  $\Phi$  is known as the measurement matrix.

For the measurement matrix to preserve projected distances between feature vectors, the Johnson–Lindenstrauss lemma must be satisfied and Achlioptas (Achlioptas, 2001) found that is the case if the entries in  $\Phi$  are

$$\phi_{ij} = \sqrt{s} \begin{cases} 1 & \text{with probability } \frac{1}{2s} \\ 0 & \text{with probability } 1 - \frac{1}{2s} \\ -1 & \text{with probability } \frac{1}{2s} \end{cases}, \quad (2)$$

for values of  $s = 2$  and 3. In fact, according to Li et al. (Li et al., 2006),  $s \gg 3$  will still satisfy JL with a limit of  $s = D/\log D$  for approximately normal data. A value of  $s = D/4$  has been demonstrated to give good results (Zhang et al., 2012).

To generate the sparsifying matrix  $\Psi$ , the same method as Zhang et al. (Zhang et al., 2012) is used. The  $n \times n$  square around each pixel is convolved with all possible box filters and the responses of the filters are concatenated into a single feature vector  $\mathbf{f} \in \mathbb{R}^{n^4}$ . When this vector is multiplied by the sparse measurement matrix  $\Phi$ , most of the entries in  $\mathbf{f}$  are discarded, and this would be very wasteful if the calculations were performed explicitly. To avoid this, only the non-zero entries of the matrix  $\Phi\Psi \in \mathbb{R}^{k \times n^2}$  are stored, and only the locations, sizes and weights of the rectangles for the box filters that ultimately contribute to a compressed feature are calculated and stored. This approach for calculating a representation of  $\Phi\Psi$  is shown Algorithm 1. Zhang et al. (Zhang et al., 2012) observed that the box filter outputs randomly multiplied by  $\pm 1$  results in features very similar to the Haar-like features (Viola and Jones, 2002), and are very efficient to calculate by the method of integral images (Viola and Jones, 2002).

For an image with  $N$  pixels, the  $n \times n$  square patch around pixel  $i$  is the signal  $x_i$ . If all  $x_i$  patches in the image are gathered together, a matrix  $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_N] \in \mathbb{R}^{n^2 \times N}$  is formed. Then the compressed features for the entire image are  $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_N] \in \mathbb{R}^{k \times N}$ , where  $y_i$  is given by Equation 1.  $\mathbf{Y}$  is calculated according the method in Algorithm 2. The number of compressed features  $k$  and the size of the patches  $n^2$  are determined empirically, as discussed in Section 3.

### 2.2 Classification

In this section, the approach for classifying features extracted from rock images is described.

The objective is to identify boundaries between rock fragments in a grey-scale image. A supervised classification method is used to label small image patches as either “boundary” or “non-boundary” according to the following steps. Different images of rocks were selected for training, validation of parameters and final testing. The pixels in these images were manually designated as being on a boundary between rocks, or as not being on a boundary. An area around each boundary was masked so it would not be used for training and classification, as there is some uncertainty about the exact location of the boundary and what classification should be assigned to pixels close to the boundary.

There are many more non-boundary pixels than boundary pixels in an image. Ten percent of boundary pixels in each image were randomly selected for processing, and a similar number of non-boundary pixels were also selected for the training set. This equalization is to avoid the following scenario. If the dataset is very unbalanced, with say 90% of pixels in one class and 10% in the other, 90% accuracy can be achieved simply by classifying *all* pixels as being in the dominant class. It is also possible to handle this situation by weighting training samples; however sampling the data also reduces the processing time so is the preferred method when sufficient data is available.

---

**Algorithm 1:** Calculation of  $\Phi\Psi$  compression matrix representation.

---

**Input:**

patch size  $n$   
 number of features  $k$   
 minimum number of box filters  $n_{f_{min}}$   
 maximum number of box filters  $n_{f_{max}}$

**Output:**  $\Phi\Psi$  representation

---

```

for all feature  $j \in \{1 \dots k\}$  do
    Generate random number of box filters  $n_f$ 
     $n_f \in \{n_{f_{min}} \dots n_{f_{max}}\}$ 
     $weight = 1/\sqrt{n_f}$ 
    for all box filter  $bf \in \{1 \dots n_f\}$  do
        filter location = random within patch
        filter size = random
        assign weight with random sign to filter
    end for
end for
return set of box filters representing  $\Phi\Psi$ 
    
```

---



---

**Algorithm 2:** Calculation of  $\mathbf{Y} = \Phi\Psi\mathbf{X}$  for an image.

---

**Input:**

Image  
 $\Phi\Psi$  representation

**Output:**  $\mathbf{Y}$  compressed features for image

---

```

Calculate integral image to be used for filter response
calculations.
for all pixel patch  $x_i$  do
    for all feature  $j \in \{1 \dots k\}$  do
         $y_{ji} =$  weighted filter response for feature  $j$ 
    end for
end for
return  $\mathbf{Y}$  compressed feature vectors
    
```

---

At each selected pixel location both square patches of brightness features, and compressed Haar-like features, were extracted. A support vector machine (SVM) was trained using a radial basis function (RBF) kernel with the extracted features on the

training set. An SVM with RBF has two parameters:  $C$  and  $\gamma$ . A grid search was performed to find the best combination of parameters, with the validation dataset being used to measure which was best. A separate validation image was used rather than cross-validation on the training data because the training data, being spatially dependent, are not independent. Cross-validation on the training data would give a biased model that may not generalize well to other data. The method for training an SVM model is shown in Figure 1, and the method for performing a test using the trained model is shown in Figure 2.

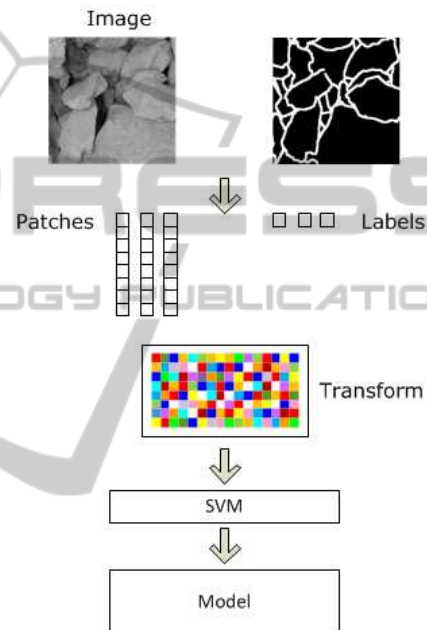


Figure 1: SVM training consists of creating a model using a training image and associated pixel labels. Features for training are created by taking small image patches and applying the transforms in Equation (1). When the raw intensity of the image is used as features, the transform is omitted.

### 3 EXPERIMENTAL RESULTS

The aims of the experiments were to establish the overall accuracy of identifying the boundaries of the rock fragments; to compare the accuracy of classification using compressed features with the accuracy using raw intensity patches; to find the optimum sized image patch; to determine the number of compressed features needed to get good accuracy; to understand the variability of results due to random generation of features; and compare the time taken for classification using compressed features with the time taken using raw intensity patches.

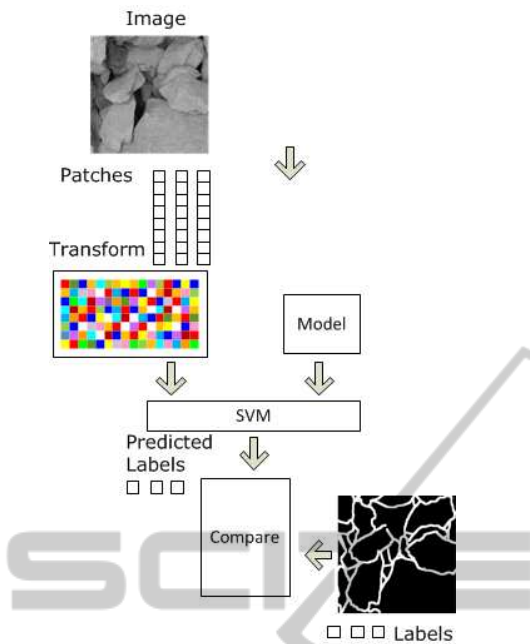


Figure 2: SVM validation and testing involves taking a test image and extracting features according to Equation (1). The previously trained SVM model is used to predict labels for the pixels in the test image and these labels are compared against a ground truth reference.

### 3.1 Training, Validation and Testing Data

Unfortunately, to the best of our knowledge, there is not a generally available dataset for this application. For training data, a grey-scale image of some rocks was selected as shown in Figure 3(a). To create training labels, the edges of rocks were manually delineated using a four pixel wide line, as shown in Figure 3(d). An  $n \times n$  square patch was taken around each pixel and the  $n^2$  pixel intensities in that square used as a feature. Pixels within 11 pixels of designated edges were excluded from the training dataset, because of the uncertainty as to whether they should be classified as edge or non-edge. The mask used for this purpose was generated by a morphological dilation of the edges in Figure 3(d). Pixels from edges were labelled +1 and pixels from non-edge regions were labelled -1. Each image in Figure 3 is 4 Mega-pixels.

Validation and testing data were generated using the same method as for training data, and this is also shown in Figure 3.

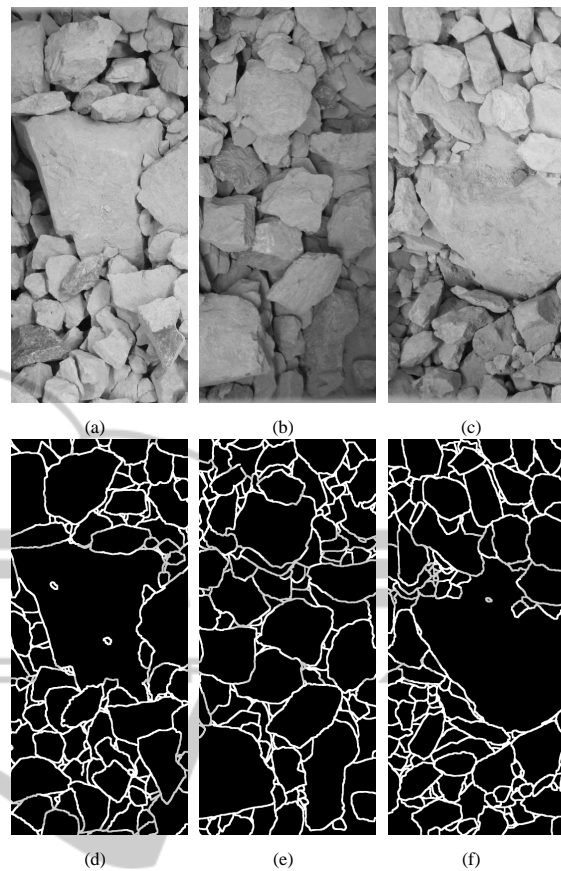


Figure 3: Data used for the experiments. The image data used for training, validation and testing is shown in (a), (b) and (c) respectively. The pixel labels used are shown below each image, in (d), (e), and (f).

### 3.2 SVM Training and Parameter Selection

For the experiments, the LIBSVM (Chang and Lin, 2011) support vector machine was used with a radial basis function (RBF) kernel. For this, two parameters are needed:  $C$  and  $\gamma$ . To choose the best values for the parameters, a grid search was performed using the validation image. The parameters that achieved the best accuracy for classifying the validation image were selected for use in testing.

To balance the training set, all pixels on edges were included, pixels not on edges were randomly sampled so that the training set includes features for edge and non-edge pixels in approximately equal proportions. Then to reduce the size of the training set, 10% of the data were randomly selected to be actually used for training.

### 3.3 Results and Discussion

To establish the overall accuracy of identifying the boundaries of the rock fragments and to compare the accuracy of classification using compressed features with the accuracy using raw intensity patches two SVMs were trained and validated using the appropriate images from Figure 3, using  $15 \times 15$  image patches. The first SVM used compressed Haar-like features of dimension 20, while the second used the raw intensity patch (dimension 225). Pixels from the test image in Figure 3 were sampled and classified by both SVMs. The resulting classifications are displayed in Figure 4. In each classified image, correctly classified pixels are displayed in green and incorrect pixels are red. The pixels that were not sampled are shown in their original colour.

The test accuracy for the compressed features in Figure 4(a) is 86.7% and for the the raw intensity features in Figure 4(b) is 82.5%. The result images for compressed features and for the raw intensity patch classifications have a different appearance partly because the two types of features having different precision/recall characteristics. With the raw intensity patches, more rock boundaries are recalled correctly but this is offset by poorer precision causing solid rock to misclassified as boundaries.

The performance of the proposed method was compared with the watershed algorithm on the test image. A satisfactory algorithm for generating watershed markers was not available, so markers were placed at the centre of each human identified fragment. This means that the watershed results were unrealistically good. The boundaries found by the watershed algorithm compared to human drawn boundaries are shown in Figure 4(c). If a watershed boundary was within 11 pixels of a “true” edge, the edge was deemed as having been recalled. Using this method, recall was calculated as 79.9%. However there were many watershed lines that did not correspond to edges, resulting in a precision (the % of predicted edges that are actually edges) being only 67.8%. For classification of compressed features in Figure 4(a), the recall is 80.7% and the precision is 89.8%. As a proxy for the accuracy of watershed, the F score is 82.5%. The F score for classification is 85%, which is very close to the measured accuracy. The two approaches are comparable (assuming proper markers can be generated for watershed), except that the watershed tends to generate more edges that are well away from the true edges.

The accuracy of classification using compressed features are compared with the accuracy using raw intensity patches are compared in Table 1. Similar accu-

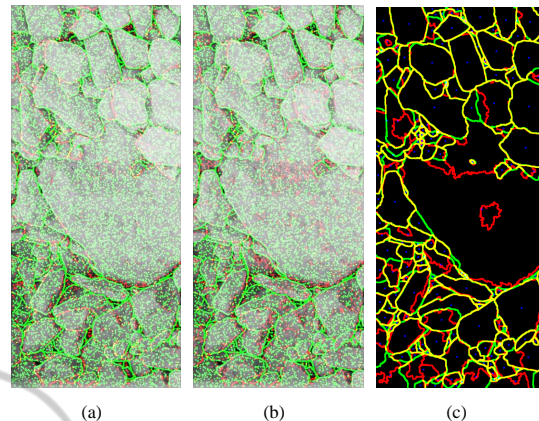


Figure 4: (a) Result for classifying sampled  $15 \times 15$  patches from the test image using compressed Haar-like features (20D), (correctly classified samples are marked with a green square, and incorrect samples with a red square); (b) result for classifying raw intensity patches (225D); and (c) comparison of watershed boundaries (red) with human drawn boundaries (green) in test image, yellow indicates the two coincide, and blue dots are watershed markers.

Table 1: Validation accuracy for three sizes of image patches using 20 compressed features per patch, compared to using raw image intensity features. The classification accuracies for the compressed and raw patches are comparable, as expected.

	CS (20)	Raw
$7 \times 7$ Accuracy	80.4%	83.5%
$15 \times 15$ Accuracy	85.3%	85.3%
$21 \times 21$ Accuracy	85.3%	83.8%

racies were achieved, identical for a  $15 \times 15$  patch. This is expected as the accuracy of classification of compressed signals should be close to that achieved for uncompressed signals (Calderbank et al., 2009).

To determine the optimum size of the image patches, compressed features of length 50 were generated for patch sizes from  $5 \times 5$  through to  $23 \times 23$ . The validation accuracy is plotted in Figure 5, together with the test accuracy. The validation accuracy was used to select the optimum size image patch:  $15 \times 15$ . The choice of image patch size is confirmed using the test accuracy. The  $13 \times 13$  patch actually had a slightly better test accuracy than the chosen patch size of  $15 \times 15$ .

To determine how many features were needed in the compressed feature vector, tests were performed with many different patch sizes and feature vector lengths. The testing and validation accuracies for  $7 \times 7$ ,  $15 \times 15$  and  $21 \times 21$  are plotted against the feature vector length in Figure 6. It was found for all patch sizes that accuracy was very poor with only a few features, where underfitting would be occurring. The accuracy increased rapidly with the number of

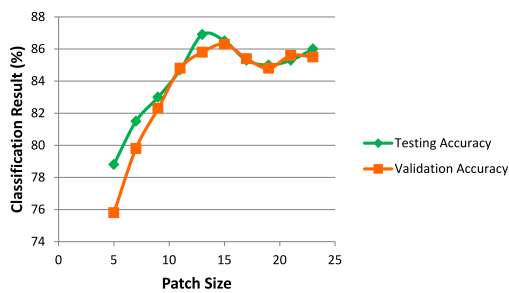


Figure 5: Validation and test accuracy for patch sizes from  $5 \times 5$  to  $23 \times 23$ , with the intensity patch compressed to 50 features. The optimum size patch is selected by finding the patch size with the highest validation accuracy, which is  $15 \times 15$ .

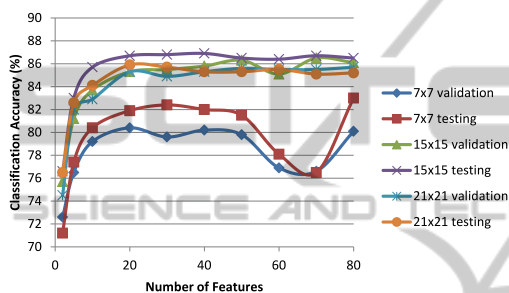


Figure 6: Number of compressed features versus validation and test accuracy for patch sizes of  $7 \times 7$ ,  $15 \times 15$  and  $21 \times 21$ . For all patch sizes, the number of features can be compressed to as few as 20 before accuracy drops off.

features. However, increasing feature lengths above 20 did not result in significant further increases in accuracy, regardless of the patch size. The accuracy for the  $7 \times 7$  patch, which of course contains 49 pixels, dropped off sharply, and rose again, when the compressed feature length increased above 50. A possible cause of this is overfitting.

Since the features are randomly generated, and the dimensionality reduced by using random projections, a test was performed to determine whether results varied greatly due to the random variations inherent in the method. Tests with varying random number generator seeds were run for images with compressed features of length 20 generated from  $15 \times 15$  patches. The accuracy for the test images, across five tests, varied between 85.6% and 86.7%. This small variation is expected with a properly constructed measurement matrix and sparse input signal.

A benefit of the compressed feature is that there is a much smaller computational burden. For a  $15 \times 15$  patch with 20 compressed features the grid search and evaluation needed to train the SVM model (requiring training 90 models and performing 270 classifications) could be completed in approximately 6.74 hours. For the same patch, using the raw image in-

tensity the grid search took in 74.5 hours. While the SVM only needs to be trained once for a particular mine site, the shortened training time due to the compressed features allows many more design options to be explored. The time to predict labels is a more important issue, as it is incurred for processing every image. The times taken to predict labels for the test image using  $15 \times 15$  patches are given in Table 2, where the entry for 225 features is for the raw intensity patch and the other entries are for compressed features. The use of compressed features reduces the processing time by over  $25\times$ , making the overall approach feasible.

Table 2: Time in seconds for classifying the test image using  $15 \times 15$  patches for different numbers of features. The entry with 225 is the raw patch with no compression. This demonstrates that feature compression is important for good performance, achieving more than  $25\times$  speed up for 20 features.

No. Features	20	50	80	225 (raw)
Prediction time (s)	21	30	120	579

## 4 CONCLUSIONS

A method has been proposed to delineate rock fragments with a possible application for the mining industry. The method overcomes some disadvantages of existing techniques by using a machine learning approach, combined with random compressed features, to classify small image patches. For the data set used, the optimum patch size was found to be  $15 \times 15$ . Just 20 compressed features were sufficient to give as good, or better, classification accuracy than the full 225 pixels in a raw image patch. Approximately 85% of rock fragments are classified correctly. The method compares favourably to the watershed algorithm and has the advantage of not needing markers. The small size of the compressed feature improved training times by a factor of 10 and classification times by more than 25, over using uncompressed image patches. Future work will look at enhancing the feature to improve accuracy, particularly for the case of overlapped rocks, and differentiating fine particles from solid rock.

## ACKNOWLEDGEMENTS

The research is sponsored by the Newcrest Mining Project at CSU and the Compact Grant from the Faculty of Business at CSU.

## REFERENCES

- Achlioptas, D. (2001). Database-friendly random projections. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '01, pages 274–281, New York, NY, USA. ACM.
- Amankwah, A. and Aldrich, C. (2011). Automatic ore image segmentation using mean shift and watershed transform. In *Radioelektronika (RADIOELEKTRONIKA), 2011 21st International Conference*, pages 1–4.
- Baraniuk, R., Davenport, M., DeVore, R., and Wakin, M. (2008). A simple proof of the restricted isometry property for random matrices. *Constructive Approximation*, 28(3):253–263.
- Beucher, S. and Lantuejoul, C. (1979). Use of watersheds in contour detection. In *International Workshop on image processing, real-time edge and motion detection/estimation*.
- Bingham, E. and Mannila, H. (2001). Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '01, pages 245–250, New York, NY, USA. ACM.
- Bull, G., Gao, J., and Antolovich, M. (2013). Image segmentation using random features. In *The 2013 5th International Conference on Graphic and Image Processing (ICGIP 2013)*.
- Calderbank, R., Jafarpour, S., and Schapire, R. (2009). Compressed learning: Universal sparse dimensionality reduction and learning in the measurement domain. *Manuscript*.
- Candes, E. and Tao, T. (2005). Decoding by linear programming. *Information Theory, IEEE Transactions on*, 51(12):4203–4215.
- Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- Comaniciu, D. and Meer, P. (1999). Mean shift analysis and applications. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1197–1203.
- Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20:273–297.
- Demenegas, V. (2008). Fragmentation analysis of optimized blasting rounds in the aitik mine: effect of specific charge. Master's thesis, Luleå tekniska universitet.
- Dollar, P., Tu, Z., Tao, H., and Belongie, S. (2007). Feature mining for image classification. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8.
- Donoho, D. (2006). Compressed sensing. *Information Theory, IEEE Transactions on*, 52(4):1289–1306.
- Gao, J., Shi, Q., and Caetano, T. S. (2012). Dimensionality reduction via compressive sensing. *Pattern Recognition Letters*, 33(9):1163 – 1170.
- Girdner, K., Kemeny, J., Srikant, A., and McGill, R. (1996). The split system for analyzing the size distribution of fragmented rock. In Franklin and Katsabanis, editors, *Measurement of Blast Fragmentation – Proceedings of the FRAGBLAST 5 Workshop.*, pages 101–108, Balkema.
- Li, P., Hastie, T. J., and Church, K. W. (2006). Very sparse random projections. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pages 287–296, New York, NY, USA. ACM.
- Liu, L. and Fieguth, P. (2012). Texture classification from random features. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(3):574–586.
- Meyer, F. and Beucher, S. (1990). Morphological segmentation. *Journal of Visual Communication and Image Representation*, 1(1):21 – 46.
- Noy, M. J. (2013). Automated rock fragmentation measurement with close range digital photogrammetry. In Sanchidrian Blanco, J. A. and Singh, A. K., editor, *Measurement and Analysis of Blast Fragmentation: Workshop FRAGBLAST 10 - The 10th International Symposium on Rock Fragmentation by Blasting.*, pages 13–21, Boca Raton, Fla. CRC Press/Balkema.
- Siddiqui, F., Shah, S. A., and Behan, M. (2009). Measurement of size distribution of blasted rock using digital image processing. *Engineering Sciences*, 20(2):81–93.
- Thurley, M. (2009). Fragmentation size measurement using 3d surface imaging. In Blanco, J. A. S., editor, *Fragblast 9 : Rock Fragmentation By Blasting. Proceedings of the 9th International Symposium On Rock Fragmentation By Blasting*, pages 229–237, Boca Raton, Fla. CRC Press/Balkema.
- Thurley, M. (2013). Automated, on-line, calibration-free, particle size measurement using 3d profile data. In Sanchidrian Blanco, J. A. and Singh, A. K., editor, *Measurement and Analysis of Blast Fragmentation: FRAGBLAST 10 - The 10th International Symposium on Rock Fragmentation by Blasting.*, pages 23–32, Boca Raton, Fla. CRC Press/Balkema.
- Thurley, M. J. and Ng, K. C. (2005). Identifying, visualizing, and comparing regions in irregularly spaced 3d surface data. *Computer Vision and Image Understanding*, 98(2):239–270.
- Viola, P. and Jones, M. (2002). Robust real-time object detection. *International Journal of Computer Vision*, 57(2):137–154.
- Wright, J., Ganesh, A., Rao, S., Peng, Y., and Ma, Y. (2009). Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C. K. I., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 2080–2088.
- Zhang, K., Zhang, L., and Yang, M.-H. (2012). Real-time compressive tracking. In Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., and Schmid, C., editors, *Computer Vision - ECCV 2012*, volume 7574 of *Lecture Notes in Computer Science*, pages 864–877. Springer Berlin Heidelberg.