AFRL-RI-RS-TR-2015-055

PROTOTYPE FOR META-ALGORITHMIC, CONTENT-AWARE IMAGE ANALYSIS

UNIVERSITY OF VIRGINIA

MARCH 2015

FINAL TECHNICAL REPORT

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

AFRL-RI-RS-TR-2015-055 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/ S /
ERIC C. JONES
Work Unit Manager

/ S /
MICHAEL J. WESSING
Deputy Chief, Information Intelligence
Systems & Analysis Division
Information Directorate

REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)* MARCH 2015	**2. REPORT TYPE** FINAL TECHNICAL REPORT	**3. DATES COVERED** *(From - To)* JUN 2012 – SEP 2014

4. TITLE AND SUBTITLE PROTOTYPE FOR META-ALGORITHMIC, CONTENT-AWARE IMAGE ANALYSIS	**5a. CONTRACT NUMBER** FA8750-12-C-0181
	5b. GRANT NUMBER N/A
	5c. PROGRAM ELEMENT NUMBER 62305E
6. AUTHOR(S) S. Acton, K. Skadron, S. Ozer, R. Sarkar, D. Newell	**5d. PROJECT NUMBER** VMRG
	5e. TASK NUMBER 00
	5f. WORK UNIT NUMBER 08
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Rector & Visitors of the University of Virginia, University of Virginia, 1001 N Emmet St, Charlottesville VA 22903-4833	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RIEA, 525 Brooks Road, Rome NY 13441-4505	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/RI
	11. SPONSOR/MONITOR'S REPORT NUMBER AFRL-RI-RS-TR-2015-055

12. DISTRIBUTION AVAILABILITY STATEMENT

Approved for Public Release; Distribution Unlimited. PA# 88ABW-2015-1026
Date Cleared: 12 MAR 2015

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Report developed under the Defense Advanced Research Projects Agency (DARPA) Visual Media Reasoning (VMR) program. In this effort, several techniques were evaluated, including image segmentation and classification, and feature (algorithm) ranking, within a Content-Based Image Retrieval (CBIR) framework. The effort also examined CBIR performance in object recognition and classification. In this context, automated segmentation algorithms were developed, in particular of active contour-based segmentation techniques, and applied to the extraction of specific objects including weapons, humans, and planes. Self-nomination is the process by which an algorithm (feature-types), "optimal" for a given specific object type, is selected within a pool of available ones. The selection process is carried out by assigning higher weights based on the level of performance of each algorithm. In this effort, two approaches were proposed: the first was based on dictionary learning whereas the second used a multiple kernel learning technique. Both approaches were studied in detail and their results on a sample dataset are presented.

15. SUBJECT TERMS

Image Analysis, Computer Vision, Content-Based Image Retrieval (CBIR)

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** ERIC C. JONES
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	56	**19b. TELEPHONE NUMBER** *(Include area code)* N/A

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 SUMMARY

In this work we study evaluate several techniques, including image segmentation and classification, and feature (algorithm) ranking within the content-based image retrieval (CBIR) framework to evaluate the CBIR performance in object recognition and classification. A recent survey of CBIR techniques can be found in [1].

We also analyze the performance of various segmentation algorithms, in particular of active contour-based segmentation techniques, when applied to the extraction of specific object including weapons, humans, and planes.

Self-nomination is the process by which an algorithm and feature-types, "optimal" for a given specific object type, are selected within a pool of available ones. The selection process is carried out by assigning higher weights based on the level of performance of each algorithm. In this work we proposed two approaches: the first is based on dictionary learning whereas the second uses a multiple kernel learning technique. Both approaches are studied in details and their results on a sample dataset are presented.

2 INTRODUCTION

Recent efforts in learning from image data sets include selecting the salient features and combining them for the purpose of content-based image retrieval [1], [2], [3]. In such approaches a set (pool) of feature-types is first selected manually and then a specific technique is applied to choose the most salient (individual) features from each feature-types in the set. Once the salient features are selected, they are combined and used in the retrieval process via a classifier. Two important questions, that still remain unanswered by these approaches, are: "*which feature-types do we need to compute for retrieving a specific object type?*" and "*is one feature-type sufficient and generic enough to retrieve images accurately from each object class?*" In this report we present a technique that answer these questions that are related to how well a feature-type can discriminate between different object categories.

In the last three decades, numerous techniques have been proposed to localize and summarize the salient features of an image. Recent examples include the scale-invariant feature transform (SIFT) [4], dense-SIFT [5], histograms of oriented gradients (HOG) [6], Gabor features [7], D-nets [8] and local symmetry features [9]. Different studies have proposed different combinations of feature-types to gain higher accuracies. For example in [10] a combination of HOG and local binary patterns is used to increase the detection of humans as opposed to using exclusively HOG features whereas in [11] is reported that a combination of SIFT, HOG and color information yields an increased accuracy in flower classification respect to the use of each feature-type individually.

These examples illustrate that different feature-types will perform optimally on different class of objects and that the choice of the salient feature-type, or combination there of, becomes critical. The automatic selection of the salient feature-types for a specific category not only helps increasing the accuracy, as opposed to using single feature-type, but also reduces the computational and storage requirements since, in many cases, before selecting individual features, all the features for each feature-type need to be computed and stored.

This report presents a prototype self-nomination approach that, for each category, can select the most salient feature-types from a given pool of feature-types. In particular we propose two different approaches: the first is based on a dictionary learning technique whereas the second utilizes a multiple kernel learning (MKL [12]) method. In addition to our prototype self-nomination approach, as a part of this work, we developed and implemented a graphical user interface (GUI) that allows the user to apply several different segmentation algorithms to a given image and provides visualization of the segmented regions.

We used a subset of the ImageNet [13] database in our experiments.

Part of the contract items are addressed in the main text and the rest are addressed in the appendix.

3 METHODS, ASSUMPTIONS, AND PROCEDURES

In this section, we provide a theoretical background for the segmentation techniques, feature types and algorithms used in this report.

3.1 Segmentation techniques

Active Contour based segmentation using manual initialization: Implements an active contour based segmentation technique with Vector field convolution (VFC) as external field. The initialization of the segmentation is done manually. Here the initialization is achieved by selecting a contour encompassing the object of interest in the image. The final segmentation is obtained by further refining the initialized contour by active contour models, constrained by external forces and image forces that pull it towards image features like lines and edges. More information on this method can be obtained in [14].

Active Contour based segmentation using automatic initialization with intensities: Implements Poisson inverse gradient algorithm [15] for automated segmentation, and an active contour method for elastic delineation of objects in the analyzed images. The main thrust of the approach is an automatic initialization of the active contour by estimating the underlying external energy field. In this method, the external energy is influenced by the image intensities.

Active Contour based texture segmentation using automatic initialization with intensities and texture: Another implementation of Active Contour based segmentation with automatic initialization by Poisson Inverse gradient method [15]. The main difference, introduced by this method, is that the external force field is generated as a weighted combination of intensity and texture force fields. The intensity part is computed as the vector field convolution as described in [14]. The textural field is computed at each location s and angular orientation θ as:

$$\mathrm{E}_{\mathrm{texture}}(s,\theta) = \sum_{i\leq 1}^{N} |\boldsymbol{J}_i(x,y) * G'_{\sigma,\theta}(x,y)| \tag{1}$$

where $G'_{\sigma,\theta}(x,y)$ is the derivative of the Gaussian function, $G_\sigma = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{x^2+y^2}{2\sigma^2}\right]$, with scale parameter σ and angular direction θ, $J_i(x,y)$ is the magnitude response of the

Gabor filter [16], for $1 \leq i \leq N$ filter banks. Equation (1) gives the edge energy in an image. The texture field is obtained by evaluating the probability of finding a boundary nearest to the location s. This is done by assessing the prediction error of the edge energy along direction θ and $\theta + \pi$ at that location. The texture prediction error $e(s,\theta)$ and the probability of edge detection $p(s,\theta)$ are given respectively by $\sum_{i\leq 1}^{N} |\boldsymbol{J}_i(x,y) * DoG_{\sigma,\theta}(x,y)|$ and $\frac{e(s,\theta)}{e(s,\theta)+e(s,\theta+\pi)}$. Here $DoG_{\sigma,\theta}(x,y)$ is difference of Gaussians function evaluated along the angular direction θ. Thus the texture force field is obtained as:

$$\boldsymbol{F}_{texture}(s) = \sum_{\Theta(\mathrm{s})\leq\theta\leq\Theta(\mathrm{s})+\pi} \mathrm{E}_{\mathrm{texture}}(s,\theta)\, e^{j\theta} \tag{2}$$

where $\Theta(\mathrm{s}) = \mathrm{argmin}_{\theta} \sum_{\theta\leq\theta'\leq\theta+\pi} p(s,\theta')$.
The details of texture field formulation can be found in [17].

Area morphological operator based segmentation: This is an automated segmentation method. This segmentation technique is based on area morphological operators, which manipulate the connected components. These operators are used to remove image segments based on their area while retaining larger segments or objects. Detailed description of this method can be found in [18].

3.2 Features

Scale-invariant feature transform (SIFT): SIFT algorithm first looks for important descriptor pixels to summarize the entire image. Once the algorithm determines the important pixels (i.e., the descriptors), it computes local gradient histograms around each descriptor to summarize each descriptor locally by forming a 128-dimensional features for each descriptor pixel (point). More details can be found in [4].

Histograms of oriented gradients (HOG): HOG algorithm forms small overlapping image patches from a given image and then for each patch, it computes local gradient-based histograms. More details on HOG can be found in [6].

Color histograms: A color histogram forms bins in the red-green-blue (RGB) space and maps each pixel of a given image into one of these bins. The resulting histogram is called color histogram of the given image. Color histogram for hue-saturation-value

(HSV) is obtained in the same way by first converting the RGB to HSV space and then by applying the binning process in the HSV space.

Local binary pattern: Local binary pattern (LBP) is another feature type that captures the texture information on the image. This information is obtained by comparing the intensity value of a pixel with its neighboring pixel values. Further details can be found in [19].

Gabor features: Gabor features are mainly used as texture descriptors. The feature is obtained by convolving an image with a sinusoid modulated Gaussian of varying orientations and scale and then extracting the mean and variance of the response. Further details are available in [20],[7].

Fourier shape descriptor: Fourier shape descriptor is used to encode the shape information of a 2-D object. The main idea is to compute the frequency response of the contour of the object. Each point on the contour taken sequentially can be viewed as a signal. The Fourier transform of this signal provides the Fourier shape descriptor. Further details for this feature type can be found in [21], [22].

3.3 Compact feature representation

Bag-of-Features approaches: Bag of visual features approaches [23] (also known as bag-of-words, BoW) are used to map a set of multiple image features into a single histogram. In these approaches, the entire data space is quantized into a fixed number of bins and a histogram is generated by mapping each feature into one of the bins. While there have been many different approaches proposed to form such histograms, all such techniques fundamentally focus on how to represent and compute the histogram bins. K-means, Fisher encoding, Vlad and dictionary learning based approaches have been widely used in the literature.

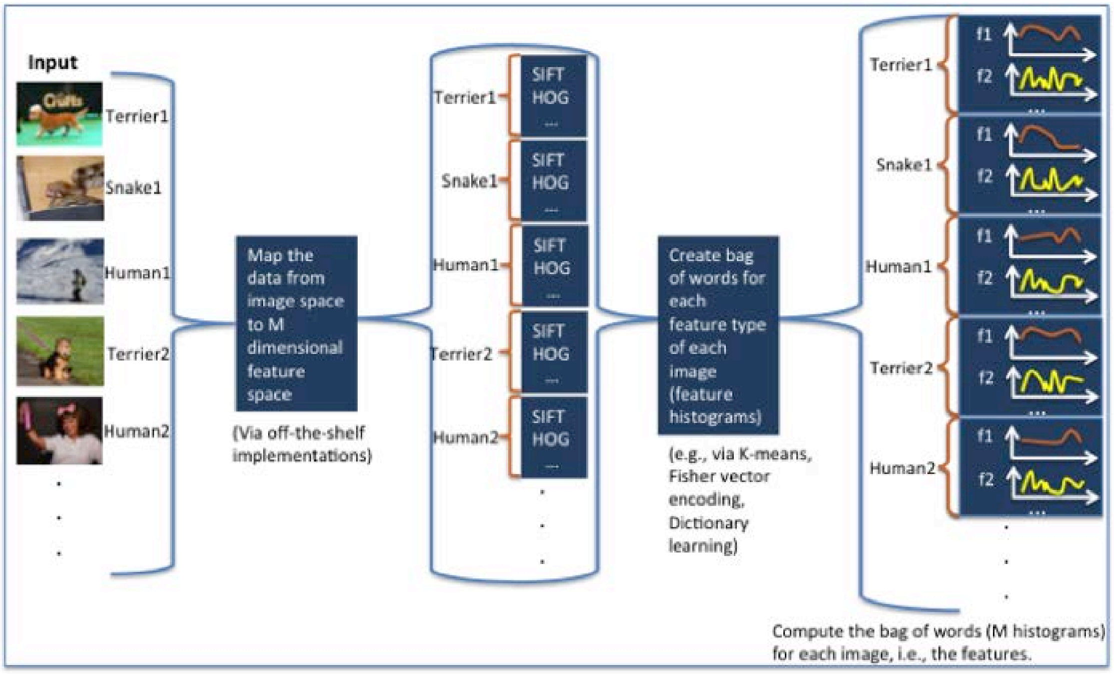

Input
Terrier1
Snake1
Human1
Terrier2
Human2
Map the data from image space to M dimensional feature space
(Via off-the-shelf implementations)
SIFT
HOG
Create bag of words for each feature type of each image (feature histograms)
(e.g., via K-means, Fisher vector encoding, Dictionary learning)
f1
f2
Compute the bag of words (M histograms) for each image, i.e., the features.

Figure 1: Flow of process creating the BoW histogram.

Figure 1 illustrates the flow of processes of creating the BoW histogram f_i for image *i*. For each image, first various feature types such as SIFT and HOG, and then each set of feature-types (i.e., the set of SIFT features and the set of HOG features) are mapped into a fixed length histogram. Thus, while the total number of computed SIFT descriptors may chance from one image to the next, the length of BoW histograms for SIFT features will be the same for each image. Below, we briefly describe different ways to create BoW histograms.

K-Means: *K* – means algorithm is a clustering algorithm that first partitions the data space into *K* number of clusters (sets), and then it assigns (labels) any given vector $\mathbf{x}_j$ into one of those *K* cluster centers based on their Euclidian distance to the cluster centers $\mathbf{c}_i$ such that the sum of the distances (the cost function *Q*) between the assigned data vectors $\mathbf{x}_j^i$ and their cluster centers $\mathbf{c}_i$ is the minimum [24]. The value of *K* is determined by the user.

$$Q = \sum_{i=1}^{K} \sum_{x_j \in c_i} \left\| \mathbf{x}_j^i - \mathbf{c}_i \right\| \tag{3}$$

Fisher Vector: Fisher vector encoding is another method for mapping the computed set of features into a histogram. In order to form a Fisher vector, the data set is first divided into K number of Gaussian distributions (as a Gaussian mixture model), and then both mean and variance information are decoded into the Fisher vector. Therefore the size of a Fisher vector histogram is 2KN (where N is the feature vector dimension and the value of K is predetermined by the user). More information on Fisher vectors can be found in [25].

Spatial Pyramid: The spatial pyramid method is used to partition the image into smaller patches and compute feature histograms for each of those patches (sub-regions). The final feature is obtained by concatenating the feature histogram of the entire image along with the histograms of the sub-regions.

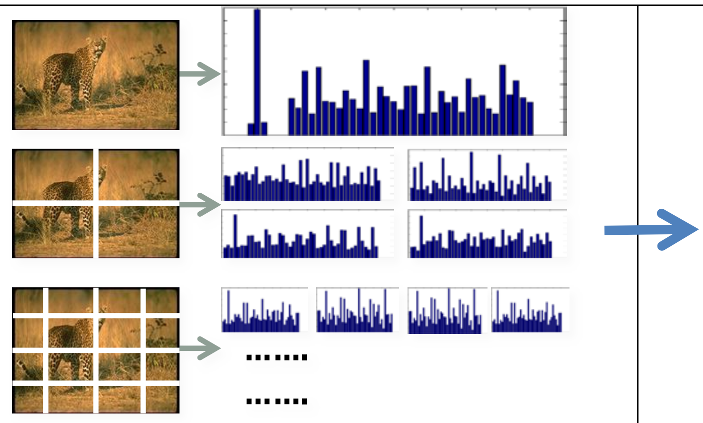
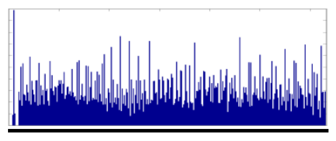
	
(a)	(b)

Figure 2: Spatial pyramid computation for a sample image. (a) shows how an image is divided into sub-regions and the feature histogram for each of the sub-region is computed. (b) The histograms computed in (a) is then concatenated to form one single histogram which now serves as the feature of the image.

This feature type incorporates spatial information into the histogram. The K-means algorithm as described in the previous section is also utilized here for computation of the feature histogram. This method is described in details in [26]. Figure 2 shows the basic steps of computing the spatial pyramid histogram for a given sample image.

3.4 Classifiers

In this work, due to their widely accepted generalization performance, we used kernel-based machine learning algorithms including support vector machines (SVM) and their variant: multiple kernel learning (MKL). In addition to those two, a dictionary learning based classifier was also implemented which exploits the sparsity in the dataset.

Support Vector Machines: For a given test (feature) vector $\mathbf{x}$ and a set of training data TD={($\mathbf{x}_1$,y_1), ($\mathbf{x}_2$,y_2),…, ($\mathbf{x}_n$,y_n)}, where y_i is the label of $\mathbf{x}_i$ and n is the total number of training samples, SVM [28] uses the following formulae to estimate its label y_i:

$$y = \operatorname{sgn}(f(\boldsymbol{x})) \tag{4}$$

and

$$f(\boldsymbol{x}) = \sum_{i=1}^{K} \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) - b \tag{5}$$

where α_i is a nonzero Lagrange multiplier for each SV $\mathbf{x}_i$, $y_i \in$ {-1,+1} is the class label, k is the number of support vectors, and b is the bias value. *K(.)* is the kernel function that gives a measure of the similarity in a reproducing kernel Hilbert space [27], [28], [29]. The α_i values in (5) are *learned* by maximizing the dual optimization problem $Q(\alpha)$*:*

$$Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x_i}, \mathbf{x_j}) \tag{6}$$

$$\text{subject to:} \qquad \sum_{i=1}^{n} \alpha_i y_i = 0 \qquad \text{and} \quad C \geq \alpha_i \geq 0, \tag{7}$$

where C is a pre-specified constant. Eq. (6) can be written in a matrix form:

$$Q(\boldsymbol{\alpha}) = \mathbf{c}^T \boldsymbol{\alpha} - \frac{1}{2}\boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha} \tag{8}$$

where $\mathbf{c}$= [1 1 1 ..1]^T, $\boldsymbol{\alpha}$= [α_1, α_2,…, α_n]^T. $\mathbf{H}$ is the Hessian matrix where $\mathbf{H}_{i,j} = \mathbf{K}_{i,j}\mathbf{R}_{i,j}$ and where:

$$\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j), \qquad \mathbf{R} = [y_1, y_2, ..., y_n]^T [y_1, y_2, ..., y_n] \tag{9}$$

In this work, we use Gaussian kernel function defined as $K(\mathbf{x}_i, \mathbf{x}_j) = e^{\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2}$ where the σ value is a user specified scalar parameter.

Multiple Kernel Learning: MKL model is proposed to estimate the optimal kernel [30]. It models the kernel function as a linear combination of t different kernel functions such that:

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_{m=1}^{t} \beta_m K_m(\boldsymbol{x}_1, \boldsymbol{x}_2) \qquad \text{subject to } \sum_{m=1}^{t} \beta_m = 1 \tag{10}$$

where β_m is the weight for the m^{th} kernel. Applying (10) in (6) yields a new cost function for MKL:

$$Q(\alpha,\beta)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j\sum_{m=1}^{t}\beta_m\big(\boldsymbol{x}_i,\boldsymbol{x}_j\big) \tag{11}$$

The survey paper in [31] studies the performance of various related MKL techniques. In this paper, we use *simpleMKL* [32] to compute the weights. *simple*MKL employs a two-step optimization process in which the kernel weights β_m are optimized by fixing the α_i values, and then α_i values are optimized by fixing the β_m values. The final values of β_m reveal the importance (the saliency) of the m^{th} kernel function.

Dictionary learning based classifier: Sparse coding can be efficiently utilized by representing a feature vector Y as a linear combination of some basis vectors. This can be written as $Y = DX$, where D is a matrix in which columns represent the basis vectors, which we call dictionary, and X contains the representative sparse codes. The motivation for sparse coding for classification is that projects the data on smaller dimensional subspaces while maximizing the separation between data i.e., similar type of data will share similar subspaces. The purpose is to build class representative dictionary, so that sparse codes generate for features belonging to the same class, share similar dictionary atoms. We solve the following optimization to obtain the desired dictionary.

$$\begin{gathered}\underset{X,D,A,W}{\operatorname{argmin}}\,\mathcal{C}(X,D,A,W)\\ \mathcal{C}(X,D,A,W)=\quad \|Y-DX\|_2^2+\gamma\big\|\dot{\boldsymbol{\mathcal{X}}}-\boldsymbol{I}X\big\|_2^2+\alpha\|Q-AX\|_2^2+\\ \beta\|H-WX\|_2^2\\ \text{s.t}\ \|x_v\|_0\le t\ \ \forall v\end{gathered} \tag{12}$$

where $\dot{\boldsymbol{\mathcal{X}}}=\begin{bmatrix}\boldsymbol{\mathcal{X}}_1 & \dots & \mathbf{0}\\ \vdots & \ddots & \vdots\\ \mathbf{0} & \dots & \boldsymbol{\mathcal{X}}_C\end{bmatrix}$**,** and $\boldsymbol{\mathcal{X}}_i$ is the sparse code generated for class i determined by solving the following

$$\underset{\boldsymbol{\mathcal{X}}_i,D_i}{\operatorname{argmin}}\ \|Y_i-D_i\boldsymbol{\mathcal{X}}_i\|_2^2\ s.t\ \forall k=\{1\dots N_i\},\|x_k\|_0\le t \tag{13}$$

Then, $\dot{\boldsymbol{\mathcal{X}}},\mathbf{0}\in\mathbb{R}^{K\times N_i}$ **.** $\boldsymbol{I}\in\mathbb{R}^{M\times M}$, is an identity matrix. $Q=[Q_1,Q_2,\dots\dots,Q_C]$, as defined in [36], is the label determining the pair of dictionary atom and signal sharing the same class. $Q_i(a,b)=1$ if $\boldsymbol{d}_a$ and $\boldsymbol{y}_b$ are the dictionary atom and training data represents class i. A is a transformation matrix that would regularize the sparse codes of the same class to share similar dictionary atoms. H is the matrix containing the class labels i.e., $H(i,b) = 1$ if $\boldsymbol{y}_b$ is a member of class i. Here we assume a linear classifier model; the label of an input signal is given as:

$$(\ell(\boldsymbol{y}_v) = i) = \underset{i}{\operatorname{argmax}}(W^T \boldsymbol{x}_v) \tag{14}$$

W is the classifier determinant parameter, which regularizes the sparse codes from same class to share similar dictionary atoms. The details of the method described here can be found in [33], [34], [35], [36]. The following is used in conjunction with this method.

Mutual Information: Mutual information between two random variables provides a measure of how much dependent they are on one another. Higher the mutual information greater is the dependency. A relevance measure between features and the class they belong to can be obtained by maximizing the mutual information [37], [38], [39], [40], [42]. For a given feature $\boldsymbol{x}$ the mutual information between the feature and its class $\ell(\boldsymbol{x}) = i$ is given by (15).

$$\mathbb{I}(\boldsymbol{x}, \ell(\boldsymbol{x}) = i) = H(i) - H(i|\boldsymbol{x}) \tag{15}$$

where $H(i)$ is the entropy given by:

$$H(\boldsymbol{x}) = p(\boldsymbol{x}) \log\left(\frac{1}{p(\boldsymbol{x})}\right) \tag{16}$$

3.5 Data sets

Along this project, we have used several datasets including ETH Zurich building dataset, Caltech 101 [43], Caltech 256[44], PASCAL [45]and ImageNet [13] datasets. We include a brief description for each dataset below:

ETH Zurich buildings dataset: This dataset contains over 1000 building images. The images contain mainly Zurich city buildings. The database is created and maintained by ETH-Zurich and can be downloaded from the URL:

http://www.vision.ee.ethz.ch/datasets/index.en.html

Caltech 101 Dataset: This dataset contains101 object categories with a total of 9,144 images. The number of images in a class varies from 31 to 800 [43]. The dataset is available from URL:

http://www.vision.caltech.edu/Image_Datasets/Caltech101/

Caltech 256 Dataset: There are 257 (formed of 256 object categories and one “clutter” category) image classes in this dataset. Each class contains at least 81 images. In this data set there are 30607 images total. Since the number of available images in each class is relatively small (about 100 images per class on average). The dataset is available from the following URL:
http://www.vision.caltech.edu/Image_Datasets/Caltech256/.

PASCAL Dataset: The Pascal dataset [45] has been widely used as a benchmark dataset in many image-based recognition and classification systems. The data set and more information is available from the URL:
http://pascallin.ecs.soton.ac.uk/challenges/VOC/

ImageNet Dataset: This is the dataset used for the “ImageNet large scale visual recognition challenge” (ILSVR) 2013 [13]. In the data set, some categories are quite similar in appearance (such as categories 5, 6, 7 and 8 in Figure 7 and in Figure 8); the objects in some images are occluded and some images may include more than one object type. The entire dataset contains two overlapping datasets (for two different tasks): The detection and classification datasets. More information on ImageNet data set can be found from the following URL: http://www.image-net.org/challenges/LSVRC/2013/.

Classification category: The classification category includes images for training, validation and for testing. It also includes the bounding box information for the training images. The training set includes 1000 categories and each category includes about 1300 images.

Detection category: The detection category (and the folder) includes images for training, validation and testing. The bounding box information is also provided in xml format for validation and training images. The xml format for the bounding boxes can be read by the development kit that is also included under the detection category.

3.6 Assumptions

In this work, we have the following assumptions:

a) There is an image database available.
b) Various feature-types are pre-computed and available for each image in the database.
c) There is a full running version of Matlab.
d) Third party libraries are already available within the computing environment.

4 RESULTS AND DISCUSSION

In this section, we present the results obtained by our presented approaches, and the documentation for the attached software that is used to generate our results.

4.1 Segmentation and User Interface (sec. 4.1 from contract)

4.1.1 Develop and implement a VMR approach that applies segmentation and classification in order to identify which subsequent object recognition algorithms to use

(sec. 4.1.1. from contract) Automatically identifying objects from a query image based on features like shape, color and texture using a set of algorithms first requires the object to be isolated from its background clutter so that the contour of the object as well as the region (that accounts for the color and texture of the object) within the contour is also available. The objective is, given a query image as an input to a segmentation algorithm, the output would be the isolated object of interest. The segmentation algorithm forms the basis of a prototype system that would demonstrate the ability to dynamically select the appropriate image analysis algorithm based on query type and image contents. A robust segmentation algorithm is desired as inclusion of significant amount of background clutter increases the probability of detecting false positives thus implying that the overall performance depends on the accuracy of segmentation.

4.1.2 Develop working software including plug-and-play interfaces

(sec. 4.1.2 from contract) The segmentation algorithms have been developed in Matlab environment. Each of the algorithms is implemented as separate functions for ease of use, which takes an image as an input and output the coordinates of the contour. A simple yet effective guided user interface is developed to choose the type of segmentation and display the contour.

4.1.3 Demonstrate a VMR approach that applies segmentation and classification. Document algorithm, software, and results of demonstration

(sec. 4.1.3-4.1.4 from contract) We developed our software in Matlab environment. The various functions and GUI (user interface) are available in the attached disk.

List of functions for implementing segmentation algorithms:

1) **VFC_segmentation.m** is the main function for executing Active contour based segmentation using Vector Field convolution. This algorithm uses manual initialization. The user is prompted to initialize the snake by selecting points around the object of interest.

   vert = VFC_segmentation(Imagefile)

   Input: Imagefile; provide the full path of the image

   Output: vert; The co-ordinates for the contour is stored sequentially in the variable

2) **PIG_segmentation.m** is the main function for executing the algorithm Active contour using Vector Field convolution and with automatic initialization by Poisson inverse gradient.

   [numseg vert]= PIG_Segmentation(Imagefile)

   Input: Imagefile: provide the full path of the image

   Output:Numseg: This variable stores the number of contours obtained

   vert: The co-ordinates for the contours is stored sequentially in the variable

3) **TexturePIG_Segmentation.m** is the main function for executing the algorithm that uses VFC and texture fields to compute the external force field and automatic initialization by Poisson inverse gradient. To use this algorithm add *PIG_texture* to Matlab path

   [numseg vert] = TexturePIG_Segmentation(Imagefile)

   Input: Imagefile: provide the full path of the image

   Output:Numseg: This variable stores the number of contours obtained

   vert: The co-ordinates for the contours is stored sequntially in the variable

For the above three methods add the Active Model Toolbox (AMT), Version 2.0, toolbox to Matlab path. This is also available from http://viva.ee.virginia.edu/

Type 'help amt' or 'doc amt' for function list and help. Please keep the folder name AMT unchanged.

4) **AOC_Segmentation.m** is the main file for demonstrating segmentation using area morphologicaloperators.

[numseg vert] = AOC_Segmentation(Imagefile)

Input: Imagefile: provide the full path of the image

Output:Numseg: This variable stores the number of contours obtained

vert: The co-ordinates for the contours is stored sequentially in the variable

Segmentation results using the three different algorithms with automated initialization is shown in Figure 3.

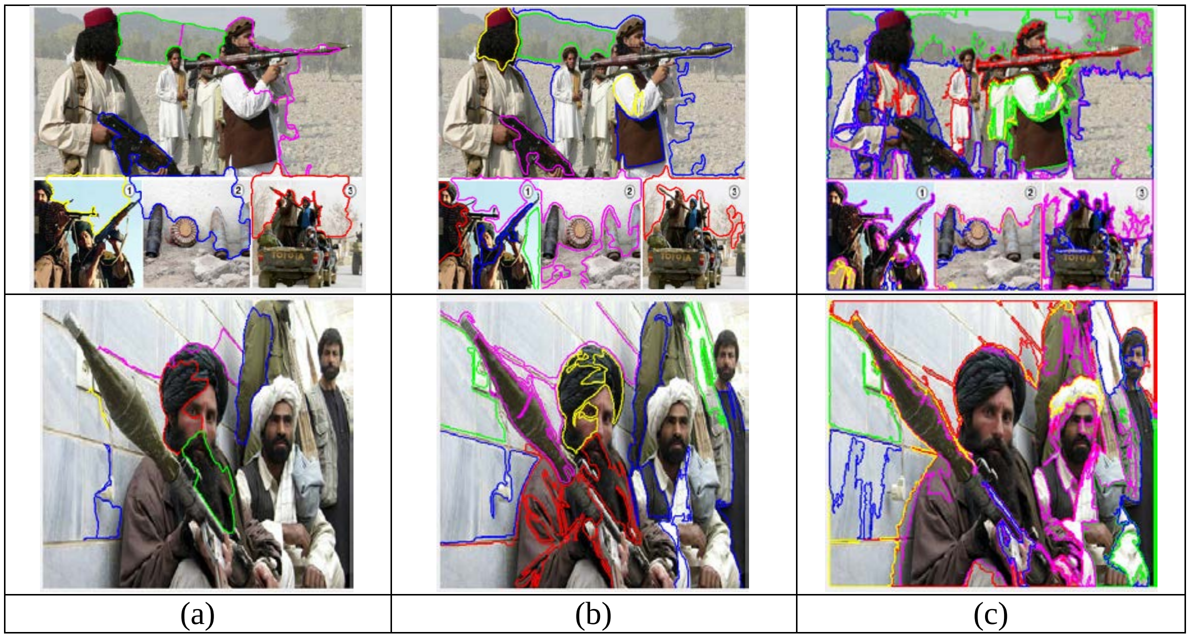


Figure 3: Segmentation results using the different segmentation algorithms. (a) Shows results of active contour model with image intensity driven external force field and automated initialization by Poisson inverse gradient, (b) shows segmentation results for image intensity and texture driven external force field with PIG based automated initialization. (c) Shows segmentation results using area morphological operators.

<ins>User Interface for segmentation</ins>

VMR_segmentation runs the GUI for the automated segmentation algorithms listed above. The GUI has options for selecting the algorithm, selecting the image and running the segmentation process. The left display panel shows the image selected. The main display panel (on right) shows the different segments (the contours of the segments are plotted in different colors) of the image. A sample of the GUI screen is shown in Figure 4.

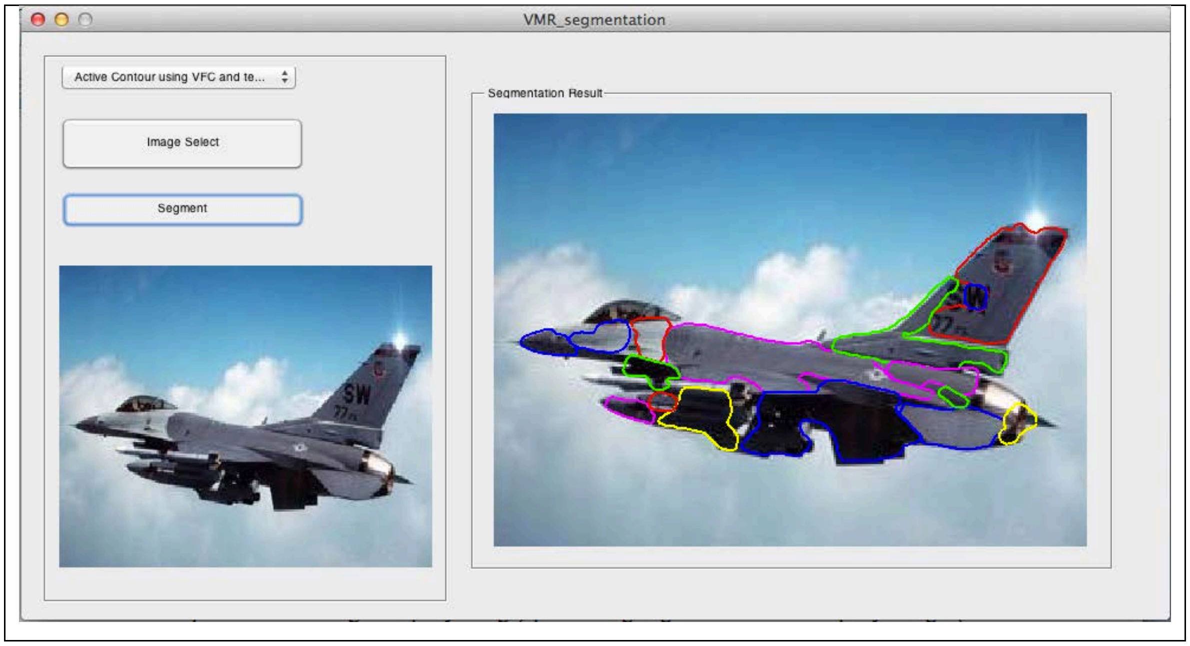


Figure 4: The GUI developed to compare the different automated segmentation algorithms. The GUI has the option to select the algorithms and also the image to segment. The left hand panel displays the image selected and the right panel displays the image with the segments.

<u>User Interface for testing classification performance using various features:</u>

To enhance the ease of using and testing all the algorithms implemented to this point, a simple yet efficient graphical user interface (GUI) has been developed in Matlab. The GUI would provide options for selecting the methods the user wants to test. Also the option for selecting the query image, performing segmentation of the query image (if required) needs to be provided. The output for segmentation would be the segmented object and respective outputs for the different methods used to date. A screenshot for the GUI is shown in Figure 5.

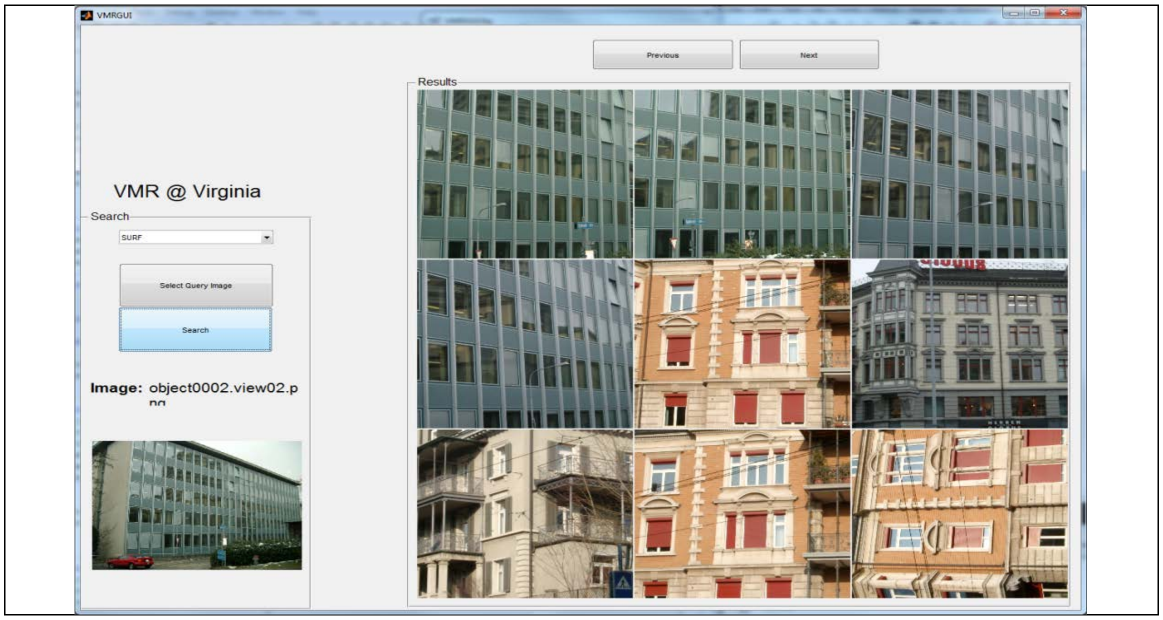


Figure 5: Shows a sample screen of the designed user interface. The method to be used can be selected from the drop down button. An option to select the query image is present and the query image is shown at the bottom left hand corner. On the right the retrieved images are shown in order of the best possible match. The 'previous and next' buttons are used to scroll through the pages to view the next set of retrieved images.

4.1.4 CBIR as an Ingredient in Recognition (sec. 4.2 from contract)

Content-based image retrieval (CBIR) has been used as a promising approach in many applications including classification, recognition and categorization [41], [43], [3]. CBIR techniques focus on the content of an image as opposed to focusing on semantically segmented image parts only. In this section, we specifically focus on the use and effectiveness of CBIR for recognition.

A typical CBIR system contains two major steps including the feature computation step and the (classifier) training step. The paper [1] includes a detailed survey on recent CBIR techniques. The performance of a given CBIR system depends on the selected feature-type (such as SIFT, HOG or color histogram) to summarize the image content and on the selected classifier's type (such as SVM, a Neural Network, a Bayesian classifier, etc.). The final retrieval process is typically performed by applying the trained classifier on the new input image to "predict" its category (i.e., the label) from a range of pre-determined candidate image categories.

The content of an image can be represented in different ways including the texture, color and shape information. And based on the image category and image content, using different types of information to characterize and discriminate different image categories typically provide better results. Therefore in this section, we design and use a CBIR system that can compute more than one feature-types characterizing different properties of a given image, and then select the most discriminative feature-type for each image category.

4.1.5 Develop and implement software to identify faces via a generalized CBIR method

(sec. 4.2.1 from contract) In the back off meeting with the project director, we were suggested to evaluate weapons instead of faces. Therefore, in this section we focus on weapon identification via CBIR.

For weapon classification purpose, we used CalTech-256 data set. CalTech256 dataset includes ak47 images under one of its 257 categories. For each image in the data set, first corresponding SIFT features are computed. Then, these set of SIFT features are mapped into BoW histograms by using a k-means algorithm. As a classifier, in this section, we used the support vector machines (SVM) algorithm. For training we selected 30 samples from each class and trained the SVM. The training data is formed such that the ak47 images form (+1) class and the images from the remaining 256 categories form the (-1) class. Figure 6 shows the results obtained at different values of the Gaussian kernel parameter. The best accuracy is obtained at the minimum kernel value 0.1. Gaussian kernel is used. The highest achieved accuracy was %99.61. In our experiments, we used Matlab’s built-in SVM implementation where the optimization technique was selected as “sequential minimal optimization (SMO). The error percentage is defined as:

$$error\ percentage = \frac{100}{N}\sum_{i=1}^{N} abs(y_i - y_i') \tag{17}$$

where y' is the predicted label and N is the total number of test samples. The function *abs*(.) is the absolute value.

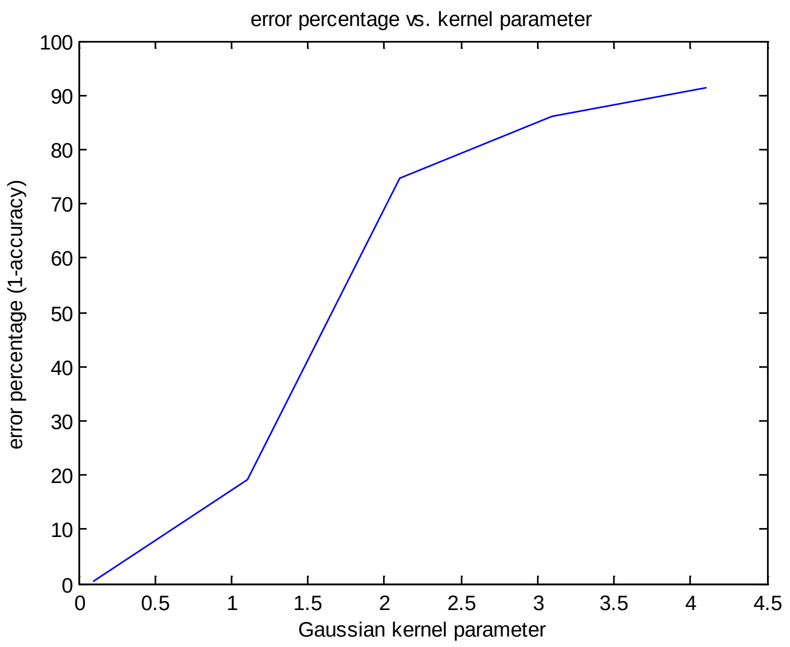


Figure 6: This figure illustrates how the performance of the SVM classifier changes with respect to the change in the kernel parameter. Performance drops drastically, as the value of the chosen Gaussian kernel parameter increases.

4.1.6 The value of CBIR as an ingredient in classification

(sec. 4.2.2 from contract): A typical CBIR technique utilizes a two-step approach in which first a set of features computed for an image and then a collection of all the computed features is used to represent the entire image in a classifier. A classifier is trained with these features for retrieval purposes.

In this section to determine the value of CBIR as an ingredient in classification, we used different feature-types including SIFT, HOG and color histograms to represent the image content. Our experiment data is a subset of the ImageNet dataset. The subset includes 20 classes and each class includes 200 images. As classifier, we used both SVM and MKL implementations. The results are presented in the next section.

4.1.7 Documentation and results

(sec. 4.2.3 from contract): Our software runs on Matlab and uses third party libraries. The software is available in the attached CD-ROM.

A) Documentation:

A list of feature computing functions and their descriptions are given below:

1) computeSIFTFeatures.m : This function computed SIFT features for the set of input images. This function takes two parameters: the first one is *trainingdata* containing the entire image data set, and the second one is *imagepatchdimension* used as an internal threshold.

Example usage:

SIFTFeatures = computeSIFTFeatures(trainingdata,imagepatchdimension);

The input data is saved in standard two dimensional Matlab cell format. The first dimension of the cell gives the total number of classes and the second dimension gives the total number of images in each cell (this code assumes that each class contains the same number of images as in the ImageNet data set). This a particular cell such as *trainingdata*{*i,j*} contains the *j*th image information from the *i*th class. Since our main image source is from ImageNet dataset, we save both the image and the bounding box information in each cell. Thus, the command *trainingdata{i,j}.image* gives the actual image (as a 2D matrix), the command *trainingdata{i,j}.bbox* would give the bounding box coordinates for the object in that image as a vector. Its format is [*xmin, xmax, ymin, ymax*].

The second parameter *imagepatchdimension* is used to checks for a potential error given in the bounding box information. If the bounding box information is too narrow along either axis, then it will be considered as error and instead of the bounding box information, the entire image information is used for that particular corresponding image. *Imagepatchdimension* is the minimum acceptable bounding box thickness along either axis. (In our experiments, it is set to 6).

It returns all the computed SIFT features for the entire image dataset. The output is also in the cell format having the same dimensions as the input data. Since SIFT features yields both descriptor information and the features for each descriptor, we saved both descriptor information and features separately for each image. For instance, SIFTFeatures{i,j}.siftdata.f1 yields SIFT features (in the format as they are computed in VLFeat), and SIFTFeatures{i,j}.siftdata.d1 yields SIFT descriptor information in the same format as they are computed in VLFeat.

2) computeRGBFeatures.m*:* This function computes color histograms for the given set of input images. Example usage:

RGB_features = computeRGBFeatures(trainingdata,imagepatchdimension, binnumberalongeachaxis);

It takes a set of input images and their corresponding bounding box informations as input (*trainingdata*) and saves the color histograms in the same cell format (see the *computeSIFTFeatures.m* description for details on the input data format). The second threshold (*imagepatchdimension*) is used to ignore the bounding box information if its any dimension is less than the specified pixels along either axis. The third parameter *binnumberalongeachaxis* sets the total number of bins along each color (Red, Green and Blue) channels. Therefore, the length of final color histogram is *binnumberalongeachaxis^3.*

4) computeHOGFeatures.m*:* This function computes the HOG features for the given set of input images and corresponding bounding boxes in *trainingdata*. Example usage:

HOG_features = computeHOGFeatures(trainingdata,imagepatchdimension);

The input images are divided into small square blocks (the block size is given by the second input). Similar to previous functions, this function also uses the second input as being the threshold for checking the bounding box size and ignores the bounding box information if the bounding box size is smaller than this threshold (in pixels) in either dimension.

5) computeSYMFeatures.m**:** This function computes the symmetry features as described in [9] for the given set of input images and corresponding bounding boxes in *trainingdata*. Example usage:

SYM_features = computeSYMFeatures(trainingdata,imagepatchdimension);

<u>List of functions computing Bag-of-Words Histograms:</u>

Below functions compute BoW histograms for SIFT and HOG features.

1) computeBoWHistogramsForSIFT.m: This function computes the BoW histograms from the given set of SIFT features. Example usage:

[SiftHistograms,Centroids, covariances, priors] =
computeBoWHistogramsForSIFT(trainingSIFTdata,Binsize,Fisher_KMeans);

where *trainingSIFTdata* is the set of SIFT features (in a cell structure) computed by the *computeSIFTFeatures* function, *Binsize* is the parameter K (as defined in K-means and Fisher vectors). *Fisher_KMeans* is a flag that selects the method to compute the histogram bins. If it is set to 1, K-means algorithm is used to compute the histogram, otherwise Fisher vectors are computed as the BoW histograms.

It returns the computed histograms (*SiftHistograms*), Bin *Centroids* (mean values), and the covariance and prior values (*covariances*, *priors*) for Fisher vector.

2) computeBoWHistogramsForHOG.m: This function computes the BoW histograms from the given set of HOG features. Example usage:

```
[HOGHistograms,Centroids, covariances, priors] =
computeBoWHistogramsForHOG(HOG_features,Binsize,Fisher_KMeans);
```

Please run the *script CBIR_demo1.m* as a demonstration of all these functions.

B) Results:

The below experiments (Experiment 1, Experiment 2 and Experiment 3) computes the area under the curve (AUC) values for each category as a performance criterion. We computed the area under the curve (AUC) values from the precision and recall curves of the test data. Precision is defined as TP/(TP+FP) and recall is defined as TP/(TP+FN), where TP is the total number of true positives and FN is the total number of false negatives.

Experiment 1: In this experiment, we have used a subset of ImageNet data set. The subset includes 20 classes where each class contains 200 images. HOG, SIFT and color histograms are computed for each image in the data set. shows sample images from each class from the dataset.

Figure 7**: Sample images for each category from the** dataset **used in experiments 1 and 2.**

At this section we studied if using a single feature type (such as HOG) for all image categories is sufficient. For that, we trained the classifier (MKL) by using the three feature-types (SIFT, HOG, color histograms) for each class.

Table 1: Area under the curve (AUC) values computed from precision and recall values for the 20-200 dataset where K=256.

Class:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
C:	0.19	0.21	0.16	0.18	0.05	0.11	0.07	0.11	0.12	0.07	0.12	0.04	0.05	0.05	0.07	0.13	0.09	0.05	0.03	0.08
S:	0.18	0.24	0.21	0.05	0.09	0.30	0.14	0.11	0.14	0.26	0.11	0.08	0.05	0.07	0.08	0.13	0.05	0.15	0.04	0.14
H:	0.22	0.15	0.66	0.25	0.08	0.17	0.18	0.08	0.12	0.37	0.16	**0.10**	**0.12**	0.15	**0.27**	0.44	**0.43**	0.13	0.06	0.35
C+S:	0.17	0.26	0.15	0.19	0.05	0.13	0.16	0.15	0.22	0.13	0.11	0.07	0.05	0.05	0.09	0.21	0.07	**0.16**	0.04	0.14
S+H:	0.21	0.19	0.62	0.24	0.08	**0.34**	**0.23**	0.10	0.13	0.36	0.15	0.09	**0.12**	0.14	0.26	0.43	0.42	0.12	0.06	**0.38**
C+H:	0.29	0.25	**0.68**	0.26	0.07	0.12	0.21	0.13	0.18	**0.38**	**0.19**	0.06	0.11	**0.18**	**0.27**	**0.47**	0.43	0.13	**0.07**	0.34
C+S+H:	**0.27**	**0.27**	0.66	**0.30**	0.07	0.14	0.22	**0.15**	**0.28**	0.35	0.17	0.08	0.11	**0.18**	0.26	0.46	0.42	0.12	0.06	0.37

Table 1 is computed by training the MKL classifier with 190 samples. The first 95 samples are from the (+1) category and the remaining 95 samples are formed by taking 5 samples from each other class (5x19=95). All the remaining images are used for the testing case (3530 images). The images with no features are ignored in the data set (total of 14 images). For this test, the K value (for K-means algorithm) is set to 256 for both SIFT and HOG BoW histograms.

The highest value among all the feature combinations is highlighted for each class in the table. As it can be seen, the most of the strongest values are obtained by combining different feature types.

Table 2: CBIR results over 20 classes are presented. The CBIR system is tested for various combinations of SIFT (S), HOG (H) and color histograms (C). The average area under the curve values (AUC) are listed under each feature combination

Features:	**C**	**S**	**H**	**S+C**	**S+H**	**H+C**	**All-3**
Average AUC:	0.10	0.13	0.22	0.13	0.23	0.24	0.25

Experiment 2: This experiment is conducted to study the effect of the K value on the classification performance. We have used the same dataset and the approach used in Experiment 1. Therefore, in this experiment we only changed the K value to 1000 in K-means algorithm and created the BoW histograms accordingly.

As classifier, MKL algorithm is being used with the same settings as in Experiment 1. MKL is trained for each class separately using one vs. all approach (95 samples for positive class, and 95 samples for the negative class are used for training). Table 3 lists AUC values computed from the precision and recall figures.

Table 3: Area under the curve (AUC) values computed from precision and recall values for the 20-200 dataset where K=1000.

Class:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
C:	0.19	0.21	0.16	0.18	0.05	0.11	0.07	0.11	0.12	0.07	0.12	0.04	0.05	0.05	0.07	0.13	0.09	0.05	0.03	0.08
S:	0.15	0.26	0.08	0.03	0.07	0.27	0.14	0.10	0.15	0.26	0.08	0.05	0.05	0.05	0.07	0.14	0.03	**0.19**	0.04	0.15
H:	0.21	0.15	0.67	0.19	**0.08**	0.15	0.23	0.08	0.19	**0.32**	**0.19**	**0.08**	**0.11**	**0.13**	**0.27**	0.37	**0.40**	0.12	**0.06**	0.34
C+S:	0.16	0.28	0.15	0.19	0.05	0.12	0.12	0.14	0.14	0.12	0.10	0.07	0.05	0.05	0.08	0.21	0.08	**0.19**	0.04	0.18
S+H:	0.21	0.24	0.63	0.17	**0.08**	**0.33**	0.23	0.09	0.16	0.30	0.16	**0.08**	0.10	0.12	0.26	0.34	**0.40**	0.16	**0.06**	0.34
C+H:	**0.27**	0.23	**0.69**	0.23	**0.08**	0.13	**0.24**	**0.15**	**0.26**	**0.32**	0.16	0.06	0.10	0.12	**0.27**	**0.40**	**0.40**	0.12	**0.06**	0.35
C+S+H:	0.26	**0.31**	0.65	**0.30**	0.07	0.13	**0.24**	**0.15**	0.25	0.30	0.15	0.07	0.10	**0.13**	0.25	0.38	**0.40**	0.16	**0.06**	**0.36**

Table 4 shows the average value (over 20 classes shown in Table 3) for each feature combination.

Table 4: CBIR results over 20 classes are presented. The CBIR system is tested for various combinations of SIFT (S), HOG (H) and color histograms (C). The average area under the curve values (AUC) are listed under each feature combination.

Features:	C	S	H	S+C	S+H	H+C	All-3
Average AUC:	0.10	0.12	0.22	0.13	0.22	0.23	0.24

Experiment 3: In addition to Experiment 1 and Experiment 2, we have also created another data set containing 30 classes (i.e., image categories) where each class contains 300 images. This experiment is designed to see if the results are consistent as the number of images increase in the dataset. Similar to Experiment 1 and 2, in this experiment, one vs. all approach is used for each class and MKL algorithm is used with the same parameters.

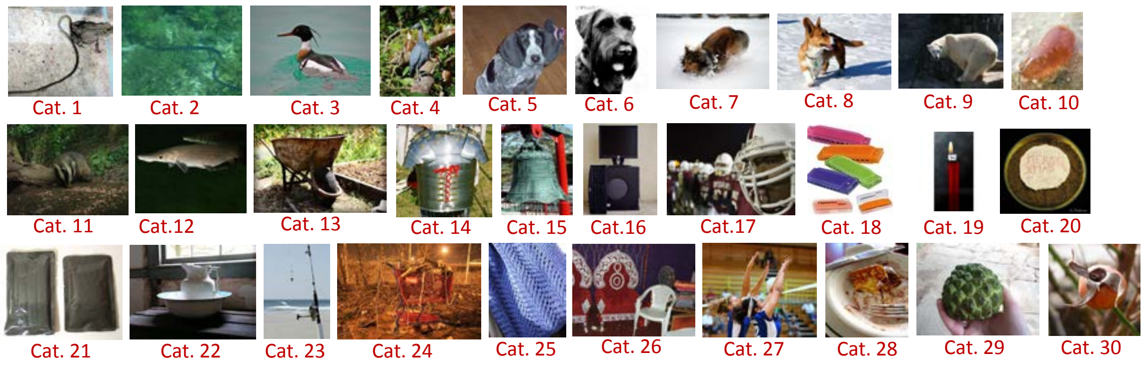
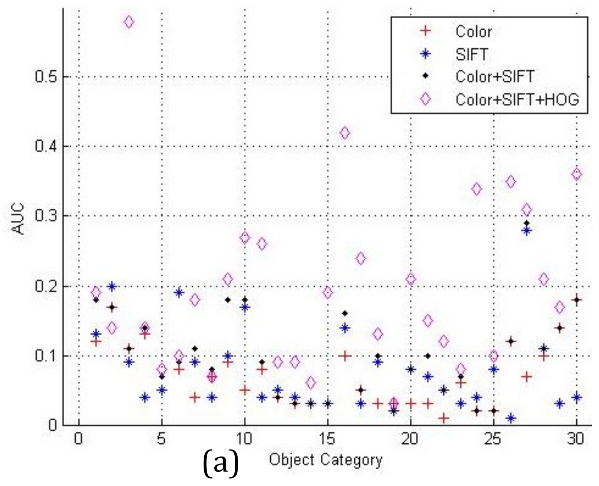


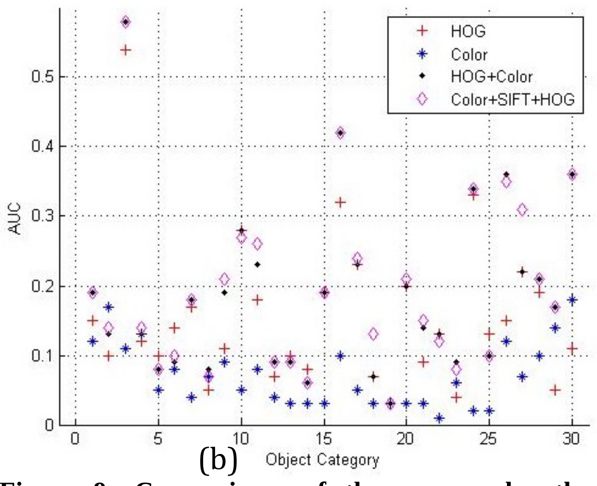
Figure 8: Sample images from each category are shown from the used data set [13]. Each image is down-sampled and resized to fit into the figure.

BoW histograms are created by using K-Means algorithm where K is set to 1000. The average AUC values for that data set are given in the below table. Figure 8 shows the AUC values in three different figures. Each graph in the figure compares the results of different feature-type combinations. Fig. 8a compares the AUC values for Color, SIFT, Color & SIFT, Color & SIFT & HOG combinations. Fig. 8b compares the AUC values for HOG, Color, HOG & Color, Color & SIFT & HOG combinations. Fig. 8c compares the AUC values for HOG, SIFT, HOG & SIFT, Color & SIFT & HOG combinations. Table 5 shows the average AUC values computed over 30 classes for each feature-type combination.

Table 5: CBIR results over 30 classes are presented. The CBIR system is tested for various combinations of SIFT (S), HOG (H) and color histograms (C). The average area under the curve values (AUC) are listed under each feature combination.

Features:	**C**	**S**	**H**	**S+C**	**S+H**	**H+C**	**All-3**
Average AUC:	0.07	0.08	0.16	0.10	0.17	0.19	0.20





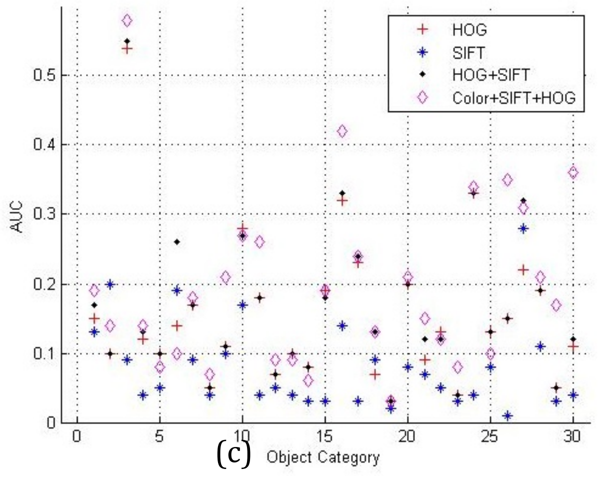


Figure 9: Comparisons of the area under the precision-recall curves for each object category are shown vertically: On each plot, combinations of a pair of feature-types are shown and those results are compared to the case where all three feature-types are used. (a) AUC values for each class are given for the Color, SIFT, Color&SIFT, Color&SIFT&HOG features. (b) AUC values for each class are given for the HOG, Color, Color&HOG, Color&SIFT&HOG features. (c) AUC values for each class are given for the HOG, SIFT, HOG&SIFT, Color&SIFT&HOG features.

A conclusive discussion for Experiment 1, 2 and 3:

A comparison of Table 2 and Table 4 shows that increasing K value from 256 to 1000 shows a slight performance reduction on average. The results in Table 5 shows lower than the ones shown in Table 4 This is mainly due to the uneven number of used data (for example 29x300= 8700 images for the negative class vs. 1x300=300 images for the positive class). However, both tables (Table 4 and Table 5) consistently show that, combining different feature-types increases the performance and accuracy in classification.

While HOG was the dominant single feature-type providing the highest AUC value among all three feature-types on average, Table 1, Table 3 and Figure 8 show that there is no single feature-type that can show the best classification performance for "all the classes" and that the combination of different-feature types can provide significant improvement in classification.

Table 1, 2, 3 and Table 4 show that the content of an image carries important information for the classifier. The feature-type used to represent that image content can make significant difference on the performance of a classifier. Thus, it remains important to select the best feature-type or the set of the best feature-types for each category separately.

4.2 Automatic Building Recognition (sec. 4.3 from contract)

Here, we implement a building recognition algorithm that is based on the algorithm presented in [46].

4.2.1 Techniques for building recognition

(sec. 4.3.1 from contract): Our building recognition approach includes following steps:

1. Detect all of the lines in the image using Canny edge detection and Hough transform.
2. Look at the entropy of the line orientations, and remove lines in regions of the image that have high entropy (this is to deal with random lines detected in trees, bushes, etc.)
3. Of the low-entropy lines, count the number of lines that run parallel to each other: *getParallelLines*()

4. Of the low-entropy lines, determine the lines that co-terminate (meaning two lines that come to an end, such as the frame in a window or roof or masonry on a building, stairs, etc.): *getCoterminations*()

5. Of the co-terminating lines, count the lines that form a U (3 lines co-terminate at roughly 90 degrees) or an L (2 lines co-terminate at roughly 90 degrees): *getUjunctions*() *getLJunctions*()

6. Normalize the num parallel lines, num U junctions, and num L junctions, by the total number of low entropy lines to form the descriptor array of 3 floats: *extractLineDescriptor*()

7. Train SVM (30/70 train/test) using only the building descriptors and all of the categories of images that we had in the VMR image set: *buildClassifiers*()

8. Generate a confusion matrix for all of the classes: *buildClassifiers*()

4.2.2 Documentation, Results and Assessment of the presented techniques on building recognition

(Sec. 4.3.2. – 4.3.3 from contract): This algorithm is implemented in C/C++. The corresponding files are *cbir.c* and *cbir.h*. Below is a list of the methods included in these files:

getParallelLines(): this method computes the parallel lines in the code.

getCoterminations(): this is the method that returns the lines that co-terminate.

getUjunctions(): This method computes the U junctions and returns the information for them.

getLJunctions(): This method computes L junctions and return them.

extractLineDescriptor(): This method computes the line descriptors and return them.

buildClassifiers(): This method builds an SVM classifier and trains the classifier by using 30 building descriptors. It also generates a confusion matrix.

Figure 10 shows both the original images and the detected L and U junctions.

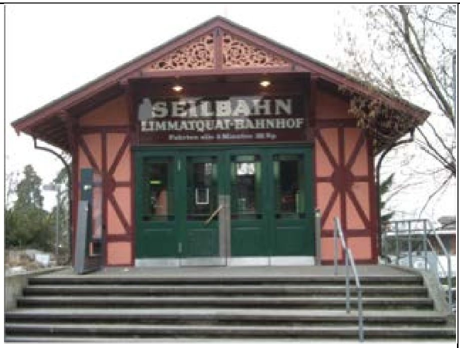
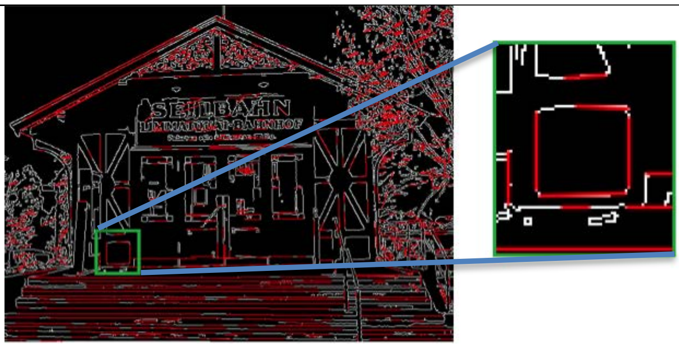
	
Original Building	Hough transform with Canny edge detection

Figure 10: A building image and the detected L and U junctions are shown in the image. The right most image shows a zoomed in portion of a building displaying the L-U junctions.

4.3 CBIR as a "backstop" algorithm for recognition (sec. 4.4 from contract)

The current state-of-the-art recognition techniques focus on using template matching based models such as deformable parts models, (DPM [47]). Thus, these techniques analyze certain parts of images individually yielding the capability of drawing a bounding box around the detected objects. On the other hand, CBIR techniques typically analyze an image based on the entire image content. Therefore, CBIR techniques can be used as a backstop, when the "template matching" based techniques cannot make a decision with confidence. This approach is illustrated in Figure 11.

In this section, we assume that for a given test image, the template-match based algorithms did not yield a result with a reliable confidence already. In such a scenario, we propose using a CBIR approach in which various image features are computed and used as input for a classifier. The classifier recognizes the image category accordingly. Please refer to Section 4.2 for the details of the used CBIR system.

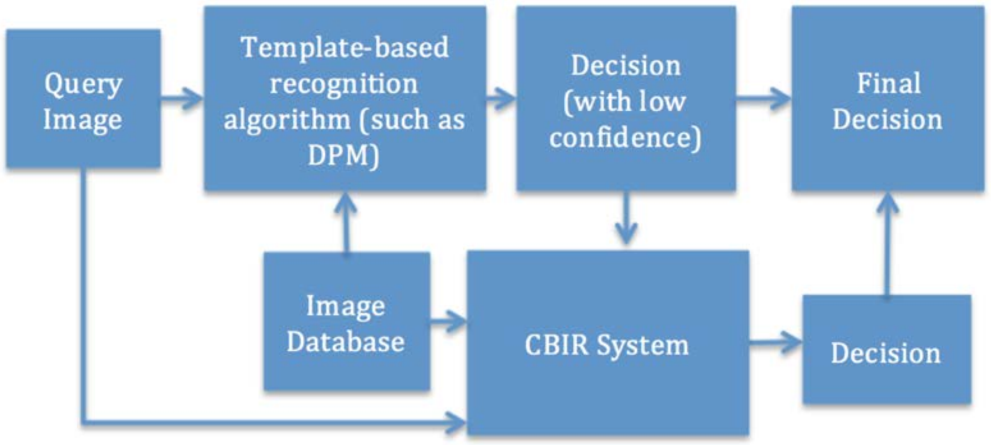

Query Image
Template-based recognition algorithm (such as DPM)
Decision (with low confidence)
Final Decision
Image Database
CBIR System
Decision

Figure 11: Use case scenario for CBIR as a backstop algorithm for recognition.

Please refer to section 2.5.2 for the quantitative analysis of our proposed system.

4.4 Viability of Self-nomination (sec. 4.5 from contract)

Self-nomination nominates (assigns weights) individual algorithms (such as SIFT, HOG and color histograms) for a given category. The proposed self-nomination approach and the use-case scenario are given in Figure 12.

In our first proposed approach, each algorithm has its own weight and the importance of the algorithm is reflected in the value of the weight such that the values of weights change between zero and one signifying the importance of the algorithm. The weights are computed through a multiple kernel learning framework.

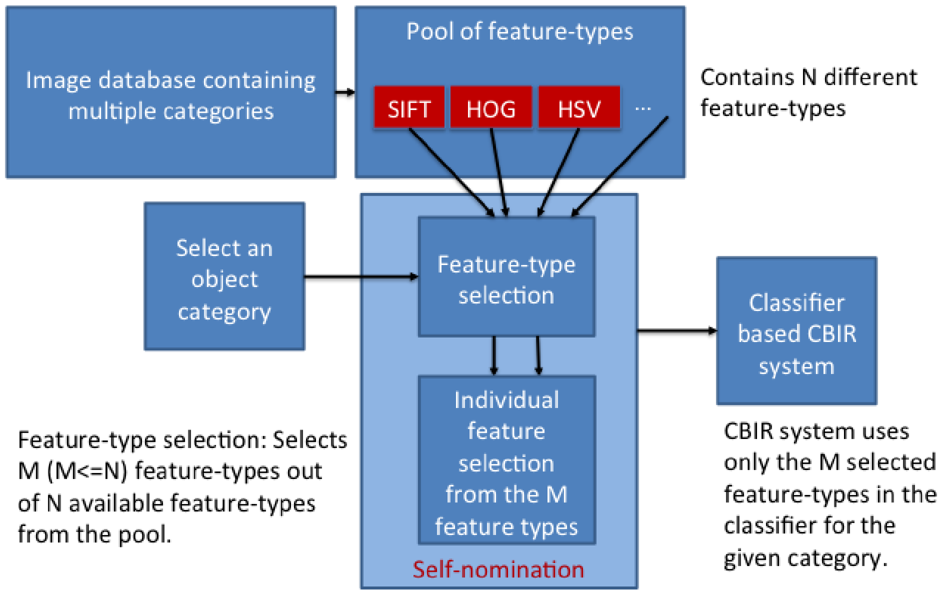


Image database containing multiple categories
Pool of feature-types
SIFT
HOG
HSV
...
Contains N different feature-types
Select an object category
Feature-type selection
Individual feature selection from the M feature types
Self-nomination
Classifier based CBIR system
Feature-type selection: Selects M (M<=N) feature-types out of N available feature-types from the pool.
CBIR system uses only the M selected feature-types in the classifier for the given category.

Figure 12: Self-nomination in VMR.

The second approach uses a dictionary learning technique to develop a class discriminative dictionary for the training dataset. This discriminative dictionary is then used to determine the class label for the test images. The self-nomination of the algorithms for a given test image is based on maximizing mutual information.

4.4.1 Investigate and develop methodology for a "self-nomination" paradigm in which a large pool of algorithms self-score their suitability and implement in software.

(sec. 4.5.1 from contract): In this section we discuss the self-nomination paradigm and two different approaches to address the problem.

A. In this section we propose a similarity-based approach for self-nomination. In our proposed approach, we compute the kernel matrices as given in Eq. (9) for each algorithm from a pool of the algorithms and then, weight each kernel matrix such that its weight will reflect its suitability. The problem, then, becomes finding an automated approach that will automatically compute the kernel weights based on their suitability.

A kernel function *K*(.) fundamentally is a measure of the similarity in a reproducing kernel Hilbert space [27], [49]. Therefore in this paper, we will define the similarity with *K*(.). For the rest of the paper, we will use the notation $\mathbf{x}_i^m$ to represent the BoW histogram of the i^{th} image from the m^{th} object category where m = 1,2.3,…,L (where L is the total number of object categories). Furthermore, we will assume that each category included the same number (*A)* of images and K^g(.) represents the similarity (i.e., the kernel) for the g^{th} feature-type where g = 1,2,3,…N (where N is the total number of feature-types). Then, ideally, for a given BoW histogram x_i we would expect that the similarity between x_i and any other image's histogram from its category should be greater than its similarity to any image from any other category for $\forall g$ such that:

$$K^g(\mathbf{x}_i^m, \mathbf{x}_i^m) \geq K^g(\mathbf{x}_i^m, \mathbf{x}_j^m) > K^g(\mathbf{x}_i^m, \mathbf{x}_j^{m+1}) \tag{18}$$

where *m+1* represents any other object category (*i*,*j*=1,2,…,*A*). Then, for all the images in the category *m,* we would expect:

$$\sum_{j=1}^{A} K^g(\mathbf{x}_i^m, \mathbf{x}_j^m) > \sum_{j=1}^{A} K^g(\mathbf{x}_i^m, \mathbf{x}_j^{m+1}) \tag{19}$$

However, for some feature-type *d* we obtain:

$$K^{d}(\mathbf{x}_i^{m}, \mathbf{x}_j^{m}) < K^{d}(\mathbf{x}_i^{m}, \mathbf{x}_j^{m+1}) \tag{20}$$

Assigning a different weight to each feature-type would yield the following inequality:

$$\sum_{g=1}^{N}\sum_{j=1}^{A} w_g K^{g}(\mathbf{x}_i^{m}, \mathbf{x}_j^{m}) > \sum_{g=1}^{N}\sum_{j=1}^{A} w_g K^{g}(\mathbf{x}_i^{m}, \mathbf{x}_j^{m+1}) \tag{21}$$

where w_g is zero if Eq. (19) holds. Notice that Eq. (20) is shown only for one class (the class *m*). In this work, our goal is finding an approach to learn the coefficients w_g. For that purpose a binary classifier (with a "one vs. all" approach) fits our methodology since such a classifier can learn weights based on their discriminative power for each object category where the object category represents the (+1) class and all other categories would represent the (-1) class collectively. We choose the MKL framework to find the coefficients w_g where $w = \beta$ and where each gram matrix [27] is formed of all the BoW histograms for the feature-type *g* such that:

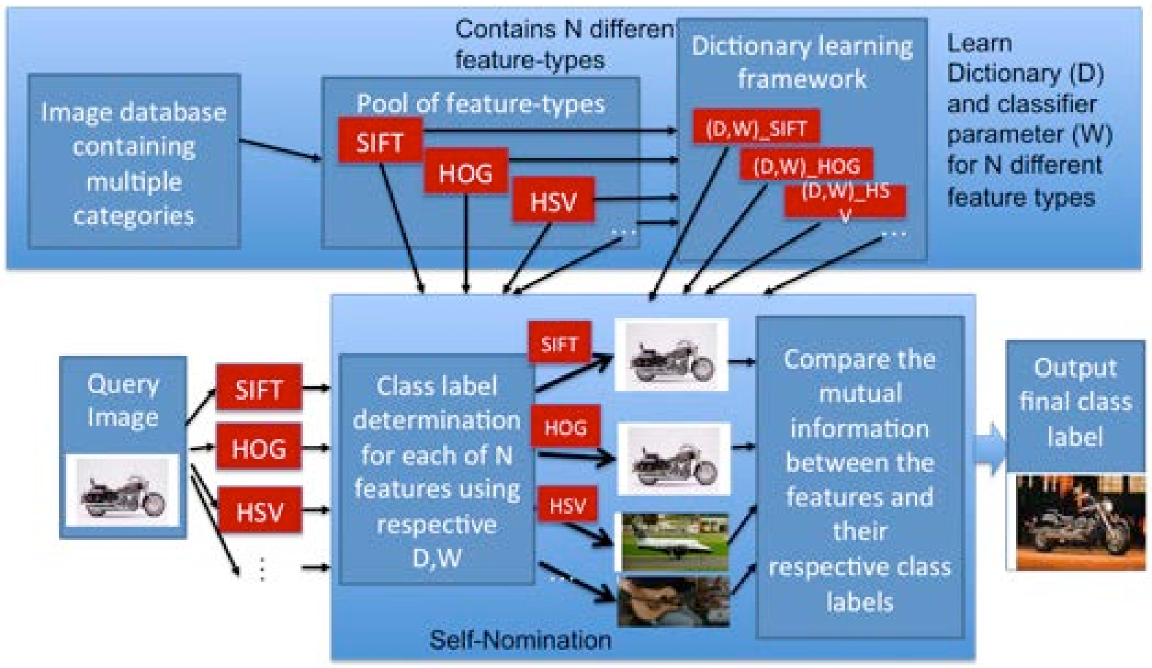
$$\mathbf{K}_{ij}^{g} = K^{g}(\mathbf{x}_i, \mathbf{x}_j) \tag{22}$$

and $\mathbf{K}_{i,j}$ in (11) is modelled such that:

$$\mathbf{K}_{ij} = \sum_{g=1}^{N} \beta_g \mathbf{K}_{ij}^{g} \tag{23}$$

In (22), each feature-type forms a Gram matrix and its weight signifies the saliency of that feature-type for the detection of the (+1) class. Our implementation uses SimpleMKL [32] as a solver.

B. In this approach we propose an information theoretic approach to dynamically choose the feature descriptor based on a given query type and the image contents. As mentioned earlier a relevance measure between features and the class they belong can be obtained by maximizing the mutual information. Hence we approach the problem from information theoretic viewpoint, that a particular feature is more accurate for classifying an image when the mutual information between the feature and the class is maximized [36].



Contains N different feature-types
Image database containing multiple categories
Pool of feature-types
SIFT
HOG
HSV
...
Dictionary learning framework
(D,W)_SIFT
(D,W)_HOG
(D,W)_HSV
...
Learn Dictionary (D) and classifier parameter (W) for N different feature types
Query Image
SIFT
HOG
HSV
Class label determination for each of N features using respective D,W
SIFT
HOG
HSV
Compare the mutual information between the features and their respective class labels
Self-Nomination
Output final class label

Figure 13: Detailed use case scenario for mutual information based feature nomination method.

We define a feature descriptor type F_l where $l = 1 \dots \dots L$ and L denotes the number of feature types being used for classification. For our experiments we use four features F_1: scale invariant feature transform (SIFT), F_2: histogram of oriented gradients (HOG), $F_{3:}$ local binary pattern (LBP), and F_4: HSV color histograms. We use our feature nomination algorithm to choose between these four features to provide the ultimate classification result. A detailed use case scenario of our method is shown in a block diagram in Figure 13.

The feature vector $Y^l = \left[Y_1^l, Y_2^l, \dots\dots, Y_C^l\right]$ corresponds to feature type l, for classes $1 \dots . C$. The respective sparse codes are $X^l = [X_1^l, X_2^l, \dots\dots, X_C^l]$. The sparse codes for a particular feature descriptor l is obtained by solving

$$\underset{X^l, D^l, A^l, W^l}{\operatorname{argmin}} \; \mathcal{C}(X^l, D^l, A^l, W^l)$$

$$\begin{aligned} \mathcal{C}(X^l, D^l, A^l, W^l) = {} & \|Y^l - D^l X^l\|_2^2 + \gamma \left\|\dot{\boldsymbol{X}}^l - \boldsymbol{I} X^l\right\|_2^2 + \alpha \|Q - A^l X^l\|_2^2 \\ & + \beta \|H - W^l X^l\|_2^2 \end{aligned} \tag{24}$$

As the number of features in the training set remains the same irrespective of the feature descriptor type, Q, H, which correlate between the features and their classes, remain same. For a given query image q, the feature descriptor y_q^l for feature type l is computed and the respective sparse code $\boldsymbol{x}_q^l$ is obtained by solving,

$$\underset{x_q^l}{\operatorname{argmin}} \left\| \boldsymbol{y}_q^l - D^l \boldsymbol{x}_q^l \right\|_2^2 \text{ s.t } \left\| \boldsymbol{x}_q^l \right\|_0 \leq t \tag{25}$$

The feature specific class label for the test image is given by

$$(\ell(x_q^l) = i) = \max_i((W^l)^T \boldsymbol{x}_q^l) \tag{26}$$

The class labels obtained from (26) may or may not be similar for all the feature types. So it is necessary to determine the most relevant class for the query image. Once the feature specific class labels are obtained, the next step is to determine which feature type is more relevant for classification and determine the class as determined by the nominated feature.

We keep the number of training features per class constant, which implies that the entropy of a class is also constant. Thus maximizing the mutual information between a feature and a class would mean minimizing the conditional entropy $H(i|\boldsymbol{x})$. The class conditional entropy can either be computed from the original feature or the sparse codes obtained by solving (25). To account for the any loss of information that may have incurred due to sparse coding of x_q^l, we compare $H(\ell(x_q^l)|y_q^l)H(\ell(x_q^l)|x_q^l)$ for all $\boldsymbol{l}$. Thus the final classification result is given by the nominated feature type:

$$\ell(q) = \min_l H(\ell(x_q^l)|y_q^l)H(\ell(x_q^l)|x_q^l) \tag{27}$$

4.4.2 Assessment of the developed methodologies

(sec. 4.5.2 from contract) The assessment for the two above mentioned methods are described here.

A. We have conducted two experiments in conjunction with experiments 1 and 3. The following experiments show the automatically computed weights for each algorithm (self-nomination) for different feature-type configurations.

Experiment 4: For our experiments, we choose 30 image categories (see Figure 8 for sample images) from the ImageNet data set [13]. For each category, we randomly select 250 images to form our experimental data set. The BoW histograms are only computed within the bounded boxes for each image. The bin centers of histograms are obtained by k-means algorithm (k=1000). For training, we employed a “one vs. all” scheme for each object category. From each target category, we select 145 images for the training and 5 images from all the other 29 categories yielding a 145 vs. 145 training set. This scheme is selected to avoid any bias in the learning towards the dominating class in the training data. All the remaining images from each category are used for testing. We trained the MKL algorithm individually for different combinations of the three types of features by using the Gaussian kernel. We used 10 Gaussian kernel parameters {0.00002, 0.0002, 0.04, 1, 4, 6, 7, 9, 10 12} yielding 10xN kernels within the MKL model. First we trained MKL by using the color histograms, the SIFT BoW histograms and the HOG BoW histograms individually. Then we used pairs of these features for the training. Finally, we used all three feature-types for training. Each trained MKL is applied on the test data.

The AUC values for each class were shown in Figure 8 for different combinations of the feature-types. Table 6 shows that, on average, there is a gain in combining feature-types as opposed to using them individually. The gain is maximal (%4) when the highest average single feature-type AUC value is compared to the one of all three feature-types. However, this is not necessarily true for each case (for example, see category 6 in Figure 9c). A comparison of all three plots in Figure 9 shows that the HOG (H) features contribute to the AUC values the most on average.

The weights for each feature-type are summed over the 10 kernel parameters and the results are shown in Table 2 for the case where N=3. These are the values signifying the saliency of each feature type. Among all three, the HOG features are weighted the most over the 30 categories (see the average values in Table 2).

Table 6: Kernel weights (βm, m=1,2,3) are shown for each category when the MKL is trained by using all three feature-types.

Category:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	Ave.
S:	0.00	0.00	0.00	0.00	0.25	0.02	0.03	0.29	0.00	0.00	0.00	0.00	0.00	0.00	0.04	0.00	0.00	0.28	0.07	0.00	0.00	0.04	0.09	0.00	0.00	0.00	0.80	0.00	0.00	0.00	0.06
H:	0.75	0.70	0.84	0.00	0.44	0.06	0.91	0.36	0.65	1.00	0.41	0.60	0.47	0.38	0.96	0.87	1.00	0.72	0.88	1.00	0.66	0.95	0.15	0.91	0.71	0.56	0.20	0.94	0.18	0.38	0.62
C:	0.25	0.30	0.16	1.00	0.31	0.92	0.06	0.35	0.35	0.00	0.59	0.40	0.53	0.62	0.00	0.13	0.00	0.00	0.05	0.00	0.34	0.02	0.76	0.09	0.29	0.44	0.00	0.06	0.82	0.62	0.31

Experiment 5: This experiment is conducted by using the same dataset and the same configuration of Experiment 1. In this experiment, we studied how the kernel weights are assigned as the number of feature-types changes. For that, we first trained the algorithm by using pairs of feature-types including Color&SIFT, Color&HOG and SIFT&HOG. Then we also trained the algorithm by using all three feature-types. The weights are given below (the highest values are highlighted).

Table 7: The classifier is trained by using Color (RGB) histograms and SIFT features.

Class:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Color:	**0.9**	**0.8**	**1**	**1**	**0.7**	**0.9**	0.3	**0.8**	**0.8**	**0.7**	**0.9**	**0.5**	**0.8**	**0.8**	0.2	**0.7**	**0.9**	0.1	0.3	0
SIFT:	0.1	0.2	0	0	0.3	0.1	**0.7**	0.2	0.2	0.3	0.1	**0.5**	0.2	0.2	**0.8**	0.3	0.1	**0.9**	**0.7**	**1**

Table 8: The classifier is trained by using Color (RGB) histograms and HOG features.

Class:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
Color:	0.4	**0.5**	0.0	**0.7**	0.5	**0.9**	0.2	**0.7**	**0.5**	0.0	**0.7**	0.3	0.5	0.2	0.0	0.1	0.0	0.0	0.0	0.0
HOG:	**0.6**	**0.5**	**1.0**	0.3	**0.5**	0.1	**0.8**	0.3	0.5	**1.0**	0.3	**0.7**	**0.5**	**0.8**	**1.0**	**0.9**	**1.0**	**1.0**	**1.0**	**1.0**

Table 9: The classifier is trained by using SIFT and HOG features.

Class:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
SIFT:	0.2	0.2	0.0	0.0	0.2	**0.7**	0.5	0.5	0.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2
HOG:	**0.8**	**0.8**	**1.0**	**1.0**	**0.8**	0.3	**0.5**	**0.5**	**0.9**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**0.8**

Table 10: The classifier is trained by using all three feature-types.

Class:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
SIFT:	0.1	0.1	0.0	0.0	0.2	0.1	**0.4**	0.1	0.1	0.1	0.0	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2
HOG:	**0.6**	0.4	**0.9**	0.3	**0.4**	0.1	0.4	0.2	0.3	**0.8**	0.3	**0.5**	**0.5**	**0.8**	**1.0**	**0.9**	**1.0**	**1.0**	**1.0**	**0.8**
Color:	0.3	**0.6**	0.1	**0.7**	0.4	**0.9**	0.2	**0.7**	**0.5**	0.0	**0.7**	0.3	0.5	0.2	0.0	0.1	0.0	0.0	0.0	0.0

The self-nominated weights for all feature-types are presented in Table 7, 8, 9 and in Table 10 for each class. Table 7 lists the weights the feature-types between Color histograms and SIFT features for each class (over 20 classes). As the highlighted values indicate, color histograms we selected as the most discriminative feature-type for 15 classes when compared to the SIFT features. SIFT was weighted as more discriminative for only the remaining 5 classes.

Table 8 compares color histograms to HOG features. In Table 8, the algorithm selected HOG features to be the most discriminative feature-type for 12 classes when

HOG features are compared to the color histograms. Color histograms were more discriminative for only 4 classes, when they are compared to HOG features. For the remaining 4 classes, the algorithm equally weighted both color histograms and HOG features.

Table 9 compares SIFT features to HOG features. In this comparison, HOG features were selected for 17 classes, while SIFT was selected only for one class. For the remaining two classes, both SIFT and HOG features were equally weighted.

Table 10 compares all three feature-types. In this case, SIFT features were selected for only one class, HOG features were selected for 12 classes and color histograms were selected for 5 classes as being the most discriminative feature-type. For the class 13, both HOG and color histograms were equally selected (weighted).

Comparing these results to the individual AUC values given in Table 1, we can conclude that, indeed the self-nomination algorithm can weight the most discriminative feature-types on average.

B. We performed experiments on Caltech 101 dataset to assess the performance of the mutual information based feature-nomination method. The results have been published [36] in an IEEE conference on image processing.

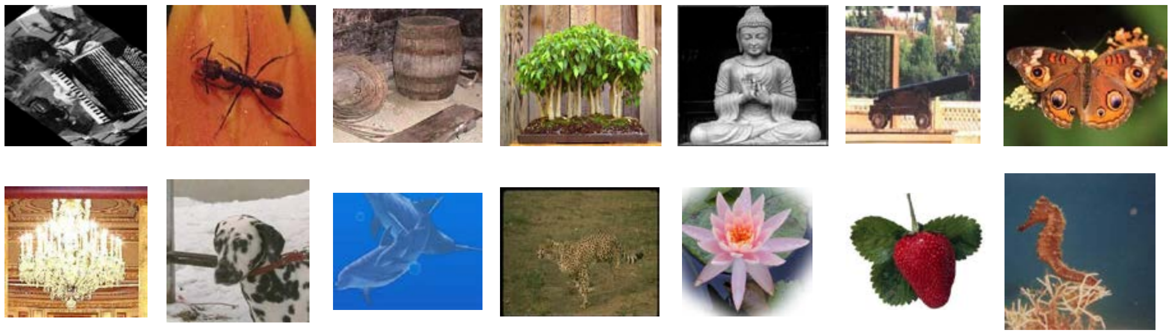
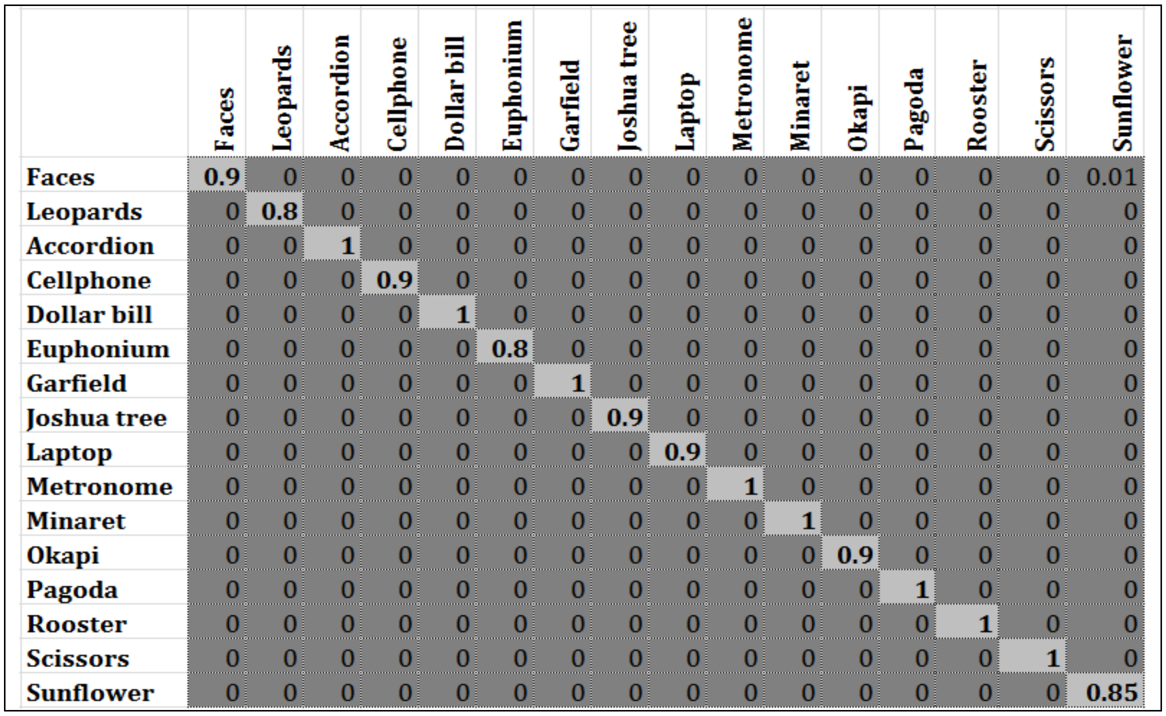


Figure 14: Sample images from Caltech 101 dataset

Figure 14 shows some sample images from Caltech 101 dataset.

Experiment 6: Experiments were performed using the Caltech101 dataset, which contains 101 different categories with 9,144 images. The number of images in a class varies from 31 to 800. We choose randomly selected 28 images per class to train the classifier for each of SIFT, HOG, LBP and HSV color histograms. The sparse codes, the training dictionary and the classifier parameter were obtained using these four features.



	Faces	Leopards	Accordion	Cellphone	Dollar bill	Euphonium	Garfield	Joshua tree	Laptop	Metronome	Minaret	Okapi	Pagoda	Rooster	Scissors	Sunflower
Faces	0.9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.01
Leopards	0	0.8	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Accordion	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Cellphone	0	0	0	0.9	0	0	0	0	0	0	0	0	0	0	0	0
Dollar bill	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
Euphonium	0	0	0	0	0	0.8	0	0	0	0	0	0	0	0	0	0
Garfield	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Joshua tree	0	0	0	0	0	0	0	0.9	0	0	0	0	0	0	0	0
Laptop	0	0	0	0	0	0	0	0	0.9	0	0	0	0	0	0	0
Metronome	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Minaret	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
Okapi	0	0	0	0	0	0	0	0	0	0	0	0.9	0	0	0	0
Pagoda	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Rooster	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
Scissors	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
Sunflower	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.85

Figure 15: The figure shows the confusion matrix (the diagonal entries show the classification accuracy when a test image from the classes along the row is classified correctly) for 16 sample classes which have classification accuracy over 80% using the feature the feature nomination scheme.

In Figure 15, we show accuracy percentage (number of correct class predictions/number of test images in that class) using feature descriptor voting scheme for 16 sample classes which have accuracy more that 80%. About 10% of the classes for the dataset have 100% accuracy and 12.7% classes have more than 90% accuracy. In this figure, the rows correspond to the category from which the test image belongs and each column corresponds to the training categories. There are total 101 categories in the dataset and we are comparing 16 of these classes, other classes remain hidden. The diagonal values in Figure 15 correspond to the accuracies to being correctly classified. The off-diagonal values correspond to the false positives i.e., a test image belongs to one class, but is mis-classified as another class.

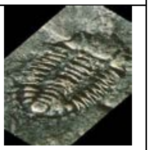
SN: **1**	SN: **0.19**	SN: **0.33**	SN: **1**	SN: **0.53**	SN: 0.25	SN: **1**	SN: **0.48**
S=**1**	S=0.12	S=**0.33**	S=**1**	S=0.4	S=0	S=1	S=0.41
H=0.8	H=0.18	H=**0.33**	H=0.91	H=0.27	H=0	H=0.88	H=0.40
L=0.92	L=0.18	L=0	L=0.92	L=0.33	L=**0.5**	L=0.76	L=0.20
C=0.08	C=0.12	C=0	C=0	C=0.14	C=0.25	C=0.52	C=0.17
S+H=0.8	S+H=0.14	S+H=0.3	S+H=0.86	S+H=0.33	S+H=0	S+H=0.84	S+H=0.4
H+L=0.86	H+L=0.18	H+L=0.167	H+L=91	H+L=0.30	H+L=0.25	H+L=0.82	H+L=0.31
L+S=0.96	L+S=0.15	L+S=0.167	L+S=0.95	L+S=0.4	L+S=0.25	L+S=0.8	L+S=0.34
S+L+H=0.9	S+L+H=0.2	S+L+H=0.2	S+L+H=0.9	S+L+H=0.4	S+L+H=0.2	S+L+H=0.8	S+L+H=0.4
							
SN: 0.20	SN: **0.25**	SN: 0.**46**	SN: **1**	SN**: 0.48**	SN: **0.4**	SN: 0.7	SN: 0.85
S=0.18	S=0.2	S=0.46	S=**1**	S=0.44	S=0.2	S=**0.78**	S=**1**
H=**0.25**	H=0.2	H=0.38	H=0.95	H=0.48	H=0.4	H=0.69	H=0.83
L=0.09	L=0.11	L=0.23	L=0.73	L=0.28	L=0.2	L=0.53	L=0.78
C=0.03	C=0.08	C=0	C=0.52	C=0.08	C=0.2	C=0.09	C=0.14
S+H=0.2	S+H=0.2	S+H=0.37	S+H=0.87	S+H=0.28	S+H=0.28	S+H=0.66	S+H=0.81
H+L=0.177	H+L=0.16	H+L=0.307	H+L=0.84	H+L=0.3	H+L=0.3	H+L=0.61	H+L=0.81
L+S=0.14	L+S=0.17	L+S=0.34	L+S=0.86	L+S=0.2	L+S=0.2	L+S=0.65	L+S=0.89
S+L+H=0.1	S+L+H=0.2	S+L+H=0.4	S+L+H=0.8	S+L+H=0.3	S+L+H=0.3	S+L+H=0.7	S+L+H=0.9

Figure 16: Figure shows the classification accuracies using SIFT (S), HOG (H), LBP (L), HSV color histogram (C) and mutual information based self-nomination (SN) algorithm. We also show comparison when a weighted combination of the features used is taken using the same dictionary learning based classifier.

Figure 16 shows the comparison of self-nomination algorithm where it chooses the appropriate feature for classification with the features when used as a single descriptor. For all the experiment we use the dictionary learning based classifier. A comparison is also shown when a weighted average of the features is computed for classification purpose. Here the weights are selected manually.

5 CONCLUSIONS

In this work, we presented two proto-type approaches for our above-mentioned self-nomination approach. We studied various feature types and compared their results in various experiments in the results section. Our attached code shows various

5.1 A Conclusive discussion and conclusion on Experiments 4 and 5

While MKL is successful in weighting the salient feature-types on average, for the categories 2 and 6, it failed on choosing the salient feature-types. For category 2, MKL weighted the HOG and color features as being the most salient feature-types. However, SIFT features yielded the highest AUC values for category 2 (See Figure 9a or Figure 9c). Similarly for the 6th category, MKL weighted the color histograms being the most salient type. However, color histograms based feature-type provides the least performance by itself in general (see Figure 9). We changed the order of the used features-types in (23) for these cases to see if ordering matters and observed that changing the order did not change the results.

A comparison of the Table 2 and Figure 9states that, although the MKL algorithm failed to assign a stronger weight to the most salient feature-type correctly for categories 2 and 6, in all other cases it found the salient features yielding the highest (or near highest) AUC values.

In our preliminary experiments we also noticed that using unequal numbers of samples from each class, yields biasing towards the class that includes the most samples. Avoiding such bias is possible by selecting equal number of samples from each class and we included the software (code) performing that task during the generation of the training data.

5.2 A Conclusive discussion and conclusion on Experiment 6

As can be seen from the Figure 16, while the proposed dictionary learning based algorithm may not give the maximum accuracy in some categories, for other categories, it either chooses the feature that gives the maximum accuracy or chooses different features to provide higher accuracy. For test images belonging to the same class, the self-nomination algorithm may not choose the same feature type, but nominates the feature, which is most suitable for that image based on its contents.

6 REFERENCES

[1] Singhai, Nidhi, and Shishir K. Shandilya. "A survey on: content based image retrieval systems*." International Journal of Computer Application*s 4, no. 4 (2010): 22-26.

[2] Zhang, He, and Sha Zijun. "Commerce Image Retrieval with Combination of Descriptors." *International Journal of Computer Science & Network Security* 13, no. 6 (2013).

[3] Yeh, Yi-Ren, Ting-Chu Lin, Yung-Yu Chung, and Y-CF Wang. "A novel multiple kernel learning framework for heterogeneous feature fusion and variable selection." *Multimedia*, IEEE Transactions on 14, no. 3 (2012): 563-574.

[4] Lowe, David G. "Distinctive image features from scale-invariant keypoints." *International journal of computer vision*, 60, no. 2: 91-110, 2004.

[5] Liu, Ce, Yuen Jenny, Torralba Antonio, Sivic Josef, and Freeman William T.. "SIFT flow: dense correspondence across different scenes." *In Computer Vision–ECCV* 2008, pp. 28-42. Springer Berlin Heidelberg, 2008.

[6] Dalal, Navneet, and Triggs Bill. “Histograms of oriented gradients for human detection.” *In Computer Vision and Pattern Recognition*, 2005. CVPR 2005. IEEE Computer Society Conference on, vol. 1, pp. 886-893. IEEE, 2005.

[7] Manjunath, Bangalore S., and Ma Wei-Ying. "Texture features for browsing and retrieval of image data." *Pattern Analysis and Machine Intelligence*, IEEE Transactions on 18, no. 8 (1996): 837-842.

[8] von Hundelshausen, Felix, and Sukthankar Rahul. "D-Nets: Beyond patch-based image descriptors*." In Computer Vision and Pattern Recognition* (CVPR), 2012 IEEE Conference on, pp. 2941-2948. IEEE, 2012.

[9] Hauagge, Daniel Cabrini, and Snavely Noah. "Image matching using local symmetry features*." In Computer Vision and Pattern Recognition* (CVPR), 2012 IEEE Conference on, pp. 206-213. IEEE, 2012.

[10] Wang, Xiaoyu, Han Tony X., and Yan Shuicheng, “An HOG-LBP human detector with partial occlusion handling.” *In Computer Vision*, 2009 IEEE 12th International Conference on, pp. 32-39. IEEE, 2009.

[11] Nilsback, M-E., and Zisserman Andrew. “Automated flower classification over a large number of classes.” *In Computer Vision, Graphics & Image Processing*, 2008. ICVGIP'08. Sixth Indian Conference on, pp. 722-729. IEEE, 2008.

[12] Lanckriet G. R. G., Cristianini N., Bartlett P., Ghaoui L. E., Jordan M. I., “Learning the Kernel Matrix with Semidefinite Programming”, *The Journal of Machine Learning Research*, 5, p.27-72, 12/1/2004.

[13] Deng, Jia, Dong W, Socher R, Li L.J., Li K., and Fei-Fei L. "Imagenet: A large-scale hierarchical image database." *In Computer Vision and Pattern Recognition*, 2009. CVPR 2009. IEEE Conference on, pp. 248-255. IEEE, 2009.

[14] Li Bing and Acton Scott, "Active Contour External Force Using Vector Field Convolution for Image Segmentation," *IEEE, Trans. on Image Procs.*, 2007.

[15] Li Bing and Acton Scott T., "Automatic Active Model Initialization via Poisson Inverse Gradient," IEEE,Trans. *on Image Procs.*, 2008.

[16] Dunn D., Higgins W.E., and Wakeley J, "Texture segmentation using 2-D Gabor elementary functions ," IEEE PAMI, vol. 16, pp. 130-149 , 1994.

[17] Ma Y. and Manjunath B.S., "Edge flow: a framework of boundary detection and image segmentation," in Computer Vision and Pattern Recognition, 1997.

[18] Acton Scott T., "Fast Algorithms for Area Morphology," Digital Signal Processing, vol. 11, no. 3, pp. 187-203(17), July 2001.

[19] Ojala T, Pietikainen M, and Maenpaa T, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns ," vol. 24, no. 7, pp. 971 - 987, 2002.

[20] Daugman John, "Complete Discrete 2-D Gabor Transforms by Neural Networks for Image Analysis and Compression ," IEEE Trans on Acoustics, Speech, and Signal Processing., vol. 36, no. 7, pp. 1169–1179, July 1988.

[21] Zahn C.T. and Roskies R.Z., "Fourier descriptors for plane close curves ," IEEE Trans. Computers, vol. C-21, pp. 269-281, March 1972.

[22] Persoon Eric and King-Sun Fu., "Shape discrimination using Fourier descriptors," Systems, Man and Cybernetics, IEEE Transactions on , vol. 7.3, pp. 170-179, 1977.

[23] Fei-Fei, L.; Perona, P., "A Bayesian hierarchical model for learning natural scene categories," Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on , vol.2, no., pp.524,531 vol. 2, 20-25 June 2005.

[24] Jain, Anil K., and Richard C. Dubes. Algorithms for clustering data. Prentice-Hall, Inc., 1988.

[25] Jaakkola, T., Haussler, D. “Exploiting generative models in discriminative classifiers”, In: NIPS, 1999.

[26] Lazebnik S., Schmid C., and Ponce J., "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in CVPR, 2006.

[27] Schölkopf, Bernhard, Ralf Herbrich, and Alex J. Smola. "A generalized representer theorem." *In Computational learning theory*, pp. 416-426. Springer Berlin Heidelberg, 2001.

[28] Vapnik V., “Statistical Learning Theory”, *John Wiley and Sons, Inc*., New York, 1998.

[29] Ozer S., Chen C. H., Cirpan H. A., "A set of new Chebyshev kernel functions for support vector machine pattern classification", *Pattern Recognition* 44, no. 7: 1435-1447, 2011.

[30] Lanckriet G. R. G., Cristianini N., Bartlett P., Ghaoui L. E., Jordan M. I., “Learning the Kernel Matrix with Semidefinite Programming”, *The Journal of Machine Learning Research*, 5, p.27-72, 12/1/2004.

[31] Bucak, S., Rong Jin, and A. Jain. "Multiple Kernel Learning for Visual Object Recognition: A Review." *IEEE Trans. on Pattern Analysis and Machine Analysis*, 2013.

[32] Rakotomamonjy A., Bach F., Canu S., Y. Grandvalet, "SimpleMKL", *Journal of Machine Learning Research* 9 (2008): 2491-2521.

[33] Jiang Z., Lin Z., and Davis L.S., "Label Consistent K-SVD: Learning a Discriminative Dictionary for Recognition," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 11, pp. 2651 - 2664, 2013.

[34] Zhang Q., and Li B., "Discriminative k-svd for dictionary learning in face recognition," 2010, IEEE Conference on Computer Vision and Pattern Recognition.

[35] Elad M. and Aharon M., "Image denoising via sparse and redundant representations over learned dictionaries," Image Processing, IEEE Transactions on, vol. 15(12), pp. 3736-3745., 2006.

[36] Sarkar R., Skadron K., and Acton S.T., "A Meta Algorithm for classification by feature nomination," in ICIP, 2014.

[37] Vasconcelos M. and Vasconcelos N., "Natural image statistics and low-complexity feature selection," Pattern Analysis and Machine Intelligence, vol. 31.2, pp. 228-244, 2009.

[38] Z. Wang, Q. Zhao, D. Chu, F. Zhao, and L. J. Guibas, "Select informative features for recognition," in ICIP, 2011.

[39] Kwak N. and Choi C.H., "Input feature selection by mutual information based on Parzen window," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24(12), pp. 1667-1671, 2002.

[40] Peng H., Long F., and Ding C., "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," IEEE Transactions on, PAMI, vol. 27(8), pp. 1226-1238., 2005.

[41] Yang, Jingjing, Yuanning Li, Yonghong Tian, Lingyu Duan, and Wen Gao. "Group-sensitive multiple kernel learning for object categorization." In Computer Vision, 2009 IEEE 12th International Conference on, pp. 436-443. IEEE, 2009.

[42] François Fleuret, "Fast binary feature selection with conditional mutual information," The Journal of Machine Learning Research , vol. 5, pp. 1531-1555., 2004.

[43] Fei-Fei L., Fergus R., and Perona P., "Learning generative visual models from few trainig samples an incremental Bayesian approach tested on 101 object categories," in CVPR, Workshop on Generative-Model based vision, 2004.

[44] Griffin Gregory, Alex Holub, and Pietro Perona. "Caltech-256 object category dataset." (2007).

[45] Everingham, Mark, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The pascal visual object classes (voc) challenge." *International journal of computer vision* 88, no. 2 (2010): 303-338.

[46] Iqbal, Qasim, and Jake K. Aggarwal. "Retrieval by classification of images containing large manmade objects using perceptual grouping." *Pattern recognition* 35, no. 7 (2002): 1463-1479.

[47] Felzenszwalb, Pedro F., Ross B. Girshick, David McAllester, and Deva Ramanan. “Object detection with discriminatively trained part-based models.” *Pattern Analysis and Machine Intelligence*, IEEE Transactions on 32, no. 9 (2010): 1627-1645.

[48] Vedaldi A. and Fulkerson B., “VLFeat: An Open and Portable Library of computer vision algorithms”, 2008, *www.vlfeat.org*/.

[49] Vedaldi, Andrea, Varun Gulshan, Manik Varma, and Andrew Zisserman. "Multiple kernels for object detection." *In Computer Vision*, 2009 IEEE 12th International Conference on, pp. 606-613. IEEE, 2009.

APPENDIX

In this appendix, we cover the rest of the contract items that are not addressed in the main text.

A.1 Documentation

(sec. 4.5.6 from contract): The documentation and details of the implementation of the two methods stated above are given here

A. In our implementation of multiple kernel learning based self-nomination, we include third party (off-the-shelf) libraries such as VLFeat [48] and SimpleMKL [32]. Most of the algorithms presented in this report are implemented on the Matlab platform.

FeatureNominationwithMKL: This Matlab function splits the input data into training and testing, it trains the classifier and returns the classification results and self-nominated weights. It computes accuracy for both positive (+1) and negative (-1) classes separately from the test data. It forms the training data from the inputdata *AllData* based on the second and third parameters *outclasssamplenumber* and *currentClass*. The data is split into positive and negative samples based on the *CurrentClass* value. Then from each negative class, a total of *outclasssamplenumber* samples are taken. The algorithm trains the classifier by taking equal number of training samples from both positive and negative classes. *FeatureSet* is the total number of feature-types included in *AllData*.

```
[ positive_classpercentage, negative_classpercentage, beta ,ypred, ytest
]=FeatureNominationwithMKL(AllData, outclasssamplenumber,currentClass,FeatureSet)
```

The returned values are *positive_classpercentage, negative_classpercentage, beta, ypred, ytest*. Among those, *beta* values are the self-nomination weights (between 0 and 1) signifying the relevant importance of each feature-type, *ypred* is the f(**x**) value estimated by using Eq. (5) and *ytest* is the true labels for the testing data; *positive_classpercentage* is the accuracy value for the positive samples in the test data (given as a percentage); *negative_classpercentage* is the accuracy value for the negative samples in the test data (given as percentage).

B. We implemented our algorithm in Matlab platform. Our implementation includes third party libraries like VLFeat [48] for feature computation, spatial pyramid calculation [26] and K-SVD algorithm for dictionary learning [35]. These algorithms are easily available from the respective websites.

Functions for feature nomination by maximizing mutual information:

1) **trainDictionary:** This Matlab function computes the different features, computes spatial pyramid features, learns the discriminative dictionary for each of the feature types and computes the respective sparse codes.

[*trainspatialpyramidFeature, trainDictionary, trainSparsecodes, classifierParam*] = *trainDictionary(traindata, classInfo, codeInfo)*

Input: *traindata*, input the training class filenames as a matfile

classInfo, input the parameter H as in (24)

codeInfo, parameter Q in equation (24)

Output: *spatialpyramidFeature,* contains the spatial pyramid feature of the training data.

trainDictionary, the learned dictionary for the training data.

trainSparsecode, the sparse codes for the training data

classifierParam, the classifier parameter W for all the features

An option to add more feature types is also available with the code.

2) **testMetaAlgorithm:** This algorithm computes the testing part where the input is one or a stream of test images and the output is the corresponding class for the self-nomination algorithm. Computing the features for the test data, computing the sparse code for the test data, evaluation of the conditional entropy for each of the features are incorporated within he following function

[*classlabel*] = *testMetaAlgorithm(testdata, trainspatialpyramidFeature, trainDictionary, trainSparsecodes, classifierParam)*

Input: *testdata,* contain the test image file names in a matfile

trainspatialpyramidFeature, the features computed from the training data

trainDictionary, the dictionary computed from the training data

trainSparsecodes, the sparse code computed from the training data

classifierParam, the classifier parameter W output from the *trainDictionary.m*

Output: *classlabel,* the final class label of the test data

In addition to the above-mentioned functions other functions are provided to assess the accuracy of the algorithm, to compute the *classInfo*, *codeInfo*. Details on each of these functions are available with the code.

A.2 Integrated System (sec. 4.6 from contract)

We have developed an integrated system that first reads all the images from a data sets, computes different feature types, computes bag of words histograms and then selects the most appropriate feature-type (algorithm) for a given category. At the same time, the system also makes a decision via the built-in SVM classifier. Our integrated system works on Matlab environment.

A.2.1 Assembled software system

(sec. 4.6.1 from contract): An assembled software system has been developed on Matlab platform and it is available on our website:
http://viva-lab.ece.virginia.edu/viva/doc/research_vmrdarpa.html .

A.2.2 Assess the integrated software system

(sec. 4.6.2 from contract): Our integrated system uses the individual functions used in the above-mentioned sections. Therefore it yields the same results as the results presented in sections 2.2 and 2.5. Please refer to those sections for the integrated system's performance.

A.2.3 Documentation

(sec. 4.6.3 from contract): Integrated system uses the same functions as the sections 2.5 and 2.2. Therefore, please refer to those sections for the documentation.

A.3 Reports (sec. 4.7 from contract)

A.3.1 Progress towards accomplishment

(sec. 4.7.1 from contract): We have reported the progress toward the accomplishment of the contract at VMR meetings. In particular we have attended the quarterly PI meetings and submitted interim reports.

A.3.2 Continually determine the status of funding

(sec. 4.7.2 from contract): We have provided the monthly invoicing.

A.3.3 Conduct presentations at such times and places designated in the contract schedule.

(sec. 4.7.3 from contract): We have regularly attended VMR meetings and presented the summary and accomplishments for the contract.

A.3.4 Document all technical work accomplished and information gained.

(sec. 4.7.4 from contract): In this work, we have presented and developed various algorithms and implementations. These include segmentation algorithms, an analysis to determine the value of various CBIR techniques, and an algorithm to select the important and suitable algorithms.

Our findings in this work are listed as follows:

1) While the manual initialization based segmentation can isolate the object of interest more accurately, the segmentation result is partially dependent on the initialization. A more proximal initialization yields an improved result. Since manual initialization of the images found in the field is not desirable, automatic initialization for segmentation is more appropriate for this application. The available (semantic) active contour and area morphology based segmentation algorithms work satisfactorily on most of the images. However their performance reduces in certain conditions like increase in background clutter, severe illumination variation on the object, smaller aspect ratio of the object in comparison to the size of the image etc.
2) U and L junctions help increasing the building recognition. However, since there are many other objects have U and L junctions frequently (such as tree), their performance reduces in large image datasets containing many different object types.
3) While the most accepted and commonly used feature-types are SIFT and HOG, in this work we noticed that while both SIFT and HOG features are alike, each captures different information and each works better for different classification purposes. In particular, SIFT features can capture the information of a scene better

than HOG features can. HOG features capture the pose (the structure and viewing angle of an object) of objects better than SIFT features can when they are both represented with K-means based BoW histograms in a classifier.

4) Multiple kernel learning and Dictionary learning with maximizing mutual information based approaches can be employed to select the salient feature-types (algorithms) among a pool of algorithms. While in some individual cases they may not select the most optimal feature-type, on average, these approaches perform well on selecting the salient feature-types.

A.3.5 Collaboration

(sec. 4.7.5 from contract): We have collaborated with different participants on planning, designing and testing the applications. In particular we have had collaborations with our colleagues from Univ. of Missouri, Mississippi State University and Michigan Tech. In addition, we have also collaborated with Mr. Harpreet Sawhney (from SRI) and with Dr. Reuven Meth (from Leidos).

A.4 Software (sec. 4.8 from contract)

All the developed software is available in the attached CDROM (media).

A.4.1 User's Guide

(sec. 4.8.1 from contract): The source code for our developed software is available in the attached CDROM (media).
Each individual file includes its own description in it to describe how to use the function and its purpose.