

Article

A Convolutional Neural Network Approach for Assisting Avalanche Search and Rescue Operations with UAV Imagery

Mesay Belete Bejiga ¹, Abdallah Zeggada ¹, Abdelhamid Nouffidj ² and Farid Melgani ^{1,*}

¹ Department of Information Engineering and Computer Science University of Trento, 38123 Trento, Italy; mesaybelete.bejiga@unitn.it (M.B.B.); abdallah.zeggada@unitn.it (A.Z.)

² Département des Télécommunications, Faculté d'Electronique et d'Informatique, USTHB BP 32, El-Alia, Bab-Ezzouar, 16111 Algiers, Algeria; h.nouffidj@gmail.com

* Correspondence: melgani@disi.unitn.it; Tel.: +39-046-128-1573

Academic Editors: Francesco Nex, Xiaofeng Li and Prasad S. Thenkabail

Received: 11 November 2016; Accepted: 14 January 2017; Published: 24 January 2017

Abstract: Following an avalanche, one of the factors that affect victims' chance of survival is the speed with which they are located and dug out. Rescue teams use techniques like trained rescue dogs and electronic transceivers to locate victims. However, the resources and time required to deploy rescue teams are major bottlenecks that decrease a victim's chance of survival. Advances in the field of Unmanned Aerial Vehicles (UAVs) have enabled the use of flying robots equipped with sensors like optical cameras to assess the damage caused by natural or manmade disasters and locate victims in the debris. In this paper, we propose assisting avalanche search and rescue (SAR) operations with UAVs fitted with vision cameras. The sequence of images of the avalanche debris captured by the UAV is processed with a pre-trained Convolutional Neural Network (CNN) to extract discriminative features. A trained linear Support Vector Machine (SVM) is integrated at the top of the CNN to detect objects of interest. Moreover, we introduce a pre-processing method to increase the detection rate and a post-processing method based on a Hidden Markov Model to improve the prediction performance of the classifier. Experimental results conducted on two different datasets at different levels of resolution show that the detection performance increases with an increase in resolution, while the computation time increases. Additionally, they also suggest that a significant decrease in processing time can be achieved thanks to the pre-processing step.

Keywords: avalanche; convolutional neural network (CNN); deep learning; hidden Markov model (HMM); object detection; search and rescue operation; support vector machine (SVM); unmanned aerial vehicle (UAV)

1. Introduction

Avalanches, large masses of snow that detach from a mountain slope and slide suddenly downward, kill more than 150 people worldwide [1] every year. According to the Swiss Federal Institute for Snow and Avalanche Research, more than 90 percent of avalanche fatalities occur in uncontrolled terrain, like during off-piste skiing and snowboarding [2]. Backcountry avalanches are mostly triggered by skiers or snowmobilers. Though it is rare, they can also be triggered naturally due to an increased load from a snow fall, metamorphic changes in snow pack, rock fall, and icefall. The enormous amount of snow carried at a high speed can cause a significant destruction to life as well as property. In areas where avalanches pose a significant threat to people and infrastructure, preventive measures like snow fences, artificial barriers, and explosives to dispose of avalanche potential snow packs are used to prevent or lessen their obstructive power.

Several factors account for victims' survival. For example, victims can collide with obstacles while carried away by avalanches or fall over a cliff in the avalanche's path and get physically injured. Once the avalanche stops, it settles like a rock and body movement is nearly impossible. Victims' chance of survival depends on the degree of burial, presence of clear airway, and severity of physical injuries. Additionally, the duration of burial is also a factor in victims' survival. According to statistics, 93 percent of victims survive if dug out within 15 min of complete burial. Survival chance drops fast after the first 15 min of complete burial. A "complete burial" is defined as where snow covers a victim's head and chest; otherwise the term partial burial applies [3]. Therefore, avalanche SAR operations are time-critical.

Avalanche SAR teams use various ways to locate victims. For example, trained avalanche rescue dogs are used to locate victims by searching for pools of human scent rising up from the snow pack. Though dogs can be useful in locating victims not equipped with electronic transceivers, the number of dogs required and the time to deploy them are constraints. If victims are equipped with electronic transceivers like ARVA (Appareil de Recherche de Victime d'Avalanche), a party of skiers can immediately start searching for a missing member. However, such transceivers are powered by batteries and require experience to use. The RECCO rescue system is an alternative to transceivers where one or more passive reflectors are embedded into clothes, boots, helmets, etc. worn by skiers and a detector is used by rescuers to locate the victims. Once the area of burial is identified, a probe can be used to localize the victim and estimate the depth of snow to be shoveled. Additionally, an organized probe line can also be used to locate victims not equipped with electronic transceivers or if locating with the transceivers fails. However, such a technique requires significant man power and is a slow process. Recent advances in the field of UAVs have enabled the use of flying robots equipped with ARVA transceivers and other sensors to assist post-avalanche SAR operations [4–6]. This has reduced search time and allowed rescuers to search in areas that are difficult to reach and dangerous.

In the literature, there are active remote sensing methods proposed to assist with post-avalanche SAR operations. For example, the authors in [7] have shown that it is possible to detect victims buried under snow using a Ground Penetrating Radar (GPR). Since the human body has a high dielectric permittivity relative to snow, a GPR can uniquely image a human body buried under snow and differentiate it from other man-made and natural objects. With the advent of satellite navigational systems, Jan et al. [8] studied the degree to which a GPS signal can penetrate through the snow and be detected by a commercial receiver, making it a potential additional tool for quick and precise localization of buried victims. Following the work in [8], the authors in [9] also studied the performance of low-cost High Sensitivity GPS (HSGPS) receivers available in the market for use in post-avalanche SAR operation. In a more recent work, Victor et al. [10] studied the feasibility of 4G-LTE signals to assist SAR operations for avalanche-buried victims and presented a proof of concept that, using a small UAV equipped with sensors that can detect cellphone signals, it is possible to detect victim's cellphone buried up to seven feet deep.

Though there has been no research published documenting the use of vision-based methods, a type of passive remote sensing method specifically for post-avalanche SAR operation, it is possible to find papers that propose supporting SAR operations in general with image analysis techniques. Rudol et al. [11] proposed assisting wilderness SAR operations with videos collected using a UAV with an onboard thermal and color cameras. In their experiment, the thermal image is used to find regions with a possible human body and corresponding regions in the color image are further analyzed by an object detector that combines a Haar feature extractor with a cascade of boosted classifiers. Because of partial occlusion and the variable pose of victims, the authors in [12] demonstrated models that decompose the complex appearance of humans into multiple parts [13–15], making them more suited than monolithic models to detecting victims lying on the ground from aerial images captured by UAV. Furthermore, they have also shown that integrating prior scale information from inertial sensors of the UAV helps to reduce false positives and a better performance can be obtained by combining complementary outputs of multiple detectors.

In recent years, civilian remote sensing applications have greatly benefited from the development of smaller and more cost-effective UAVs. Some of the applications include: detecting and counting cars or other objects from aerial images captured by UAVs [16–18], assessing the impact of man-made or natural disasters for humanitarian action, and vegetation mapping and monitoring. In general, these are rapid, efficient, and effective systems to acquire extremely high-resolution (EHR) images. Additionally, their portability and easiness to deploy makes them well suited for applications like post-avalanche SAR operation. According to [19], out of 1886 people buried by avalanches in Switzerland between 1981 and 1998, 39% of the victims were buried with no visible parts while the rest were partially buried or stayed completely unburied on the surface. Moreover, the chance of complete burial can be reduced if avalanche balloons are used. Given this statistic, we present a method that utilizes UAVs equipped with vision sensors to scan the avalanche debris and further process the acquired data with image processing techniques to detect avalanche victims and objects related to the victims in near-real time.

The organization of this paper is as follows: the overall block diagram of the system along with the description of each block is presented in the next section. Datasets used and experimental setup are presented in Section 3. Experimental results are presented in Section 4 and the last section, Section 5, is dedicated to conclusions and further development.

2. Methodology

In this section we present a pre-processing method, partially based on the image segmentation technique, to filter areas of interest from a video frame followed by an image representation method based on Convolutional Neural Networks (CNNs or ConvNets) and train a Support Vector Machine (SVM) classifier to detect objects. Furthermore, we present a post-processing method based on Hidden Markov Models (HMMs) to take advantage of the correlation between successive video frames to improve the decision of the classifier. A block diagram of the overall system is shown in Figure 1.

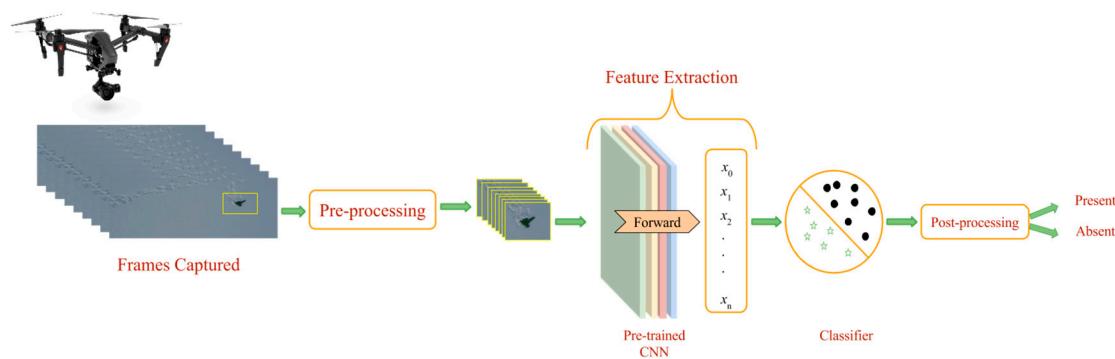


Figure 1. Block diagram of the overall system.

2.1. Pre-Processing

If we consider post-avalanche areas, they are covered by snow and hence mostly white. Assuming objects of interest will have a different color than snow, applying image segmentation methods will allow us to separate a frame into regions of snow and other objects. Then, these potential regions of objects are further processed by the next steps. This step allows us to process only regions of a frame and in some cases to skip or filter frames with no potential object regions, thereby providing a better localization of objects and a desirable reduced computation time. In the pre-processing step, a frame will be scanned with a sliding window and each window will be checked for a color different

than snow by thresholding the saturation component of the window in the HSV color space. We have adopted the thresholding scheme proposed in [20]:

$$th_{sat}(V) = 1.0 - \frac{0.8V}{255}, \quad (1)$$

where V represents the value of the intensity component. We decide that a pixel corresponds to an object if the value of the saturation component S is greater than or equal to $th_{sat}(V)$. In such a case, the window is said to contain an object.

2.2. Feature Extraction

Feature extraction is the process of mapping image pixels or groups of pixels into a suitable feature space. The choice of an appropriate feature extractor strongly affects the performance of the classifier. In the literature, one can find several feature extraction methods proposed for object detection in images or videos. Haar [21], Scale Invariant Feature Transform (SIFT) [22], and Histogram of Oriented Gradients (HOG) [23] are some of the methods most widely used to generate image descriptors.

In recent years, the availability of large real-world datasets like ImageNet [24] and high-performance computing devices has enabled researchers to train deep and improved neural network architectures like ConvNets. These classifiers have significantly improved object detection and classification performance. Besides training CNNs to learn features for a classification task, using pre-trained CNN architectures as a generic feature extractor and training classifiers like SVM has outperformed the performance results obtained by using ‘hand-designed’ features like SIFT and HOG [25,26].

CNNs are regular feedforward neural networks where each neuron accepts inputs from neurons in the previous layer and perform operations such as multiplication of the input with the network weights and nonlinear transformation. Unlike regular neural networks, a neuron in a CNN is only connected to a small number of neurons in the previous layer that are called local receptive fields. Moreover, neurons in a layer are arranged in three dimensions: width, height, and depth. CNNs are primarily designed to encode spatial information available in images and make the network more suited to image focused tasks [27]. Regular neural networks struggle from computational complexity and overfitting with an increase in the size of the input. In contrast, CNNs overcome this problem through weight sharing. Weight sharing is a mechanism by which neurons in a ConvNet are constrained in a depth slice and use the same learned weights and bias in the spatial dimension. These set of learned weights are called filters or kernels.

A typical CNN architecture (Figure 2) is a cascade of layers mainly made from three types of layers: convolutional, pooling, and fully connected layers.

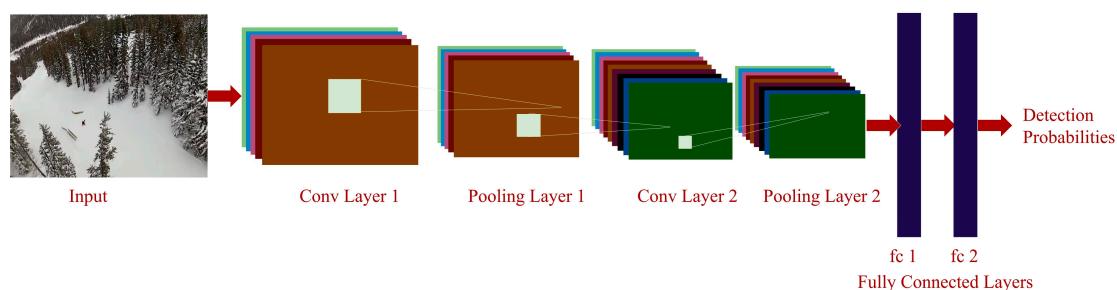


Figure 2. Example of CNN architecture for object recognition.

1. Convolutional layer

The convolutional layer is the main building block of a ConvNet that contains a set of learnable filters. These filters are small spatially (along the height and width dimension) and extend fully in the

depth dimension. Through training, the network learns these filters that activate neurons when they see a specific feature at a spatial position of the input. The convolution layer performs a 2D convolution of the input with a filter and produces a 2D output called an activation map (Figure 3). Several filters can be used in a single convolutional layer and the activation maps of each filter are stacked to form the output of this layer, which is an input to the next layer. The size of the output is controlled by three parameters: depth, stride, and zero padding. The depth parameter controls the number of filters in a convolutional layer. Stride is used to control the extent of overlap between adjacent receptive fields and has an impact on the spatial dimension of the output volume. Zero padding is used to specify the number of zeros that need to be padded on the border of the input, which allows us to preserve input spatial dimension at the output.

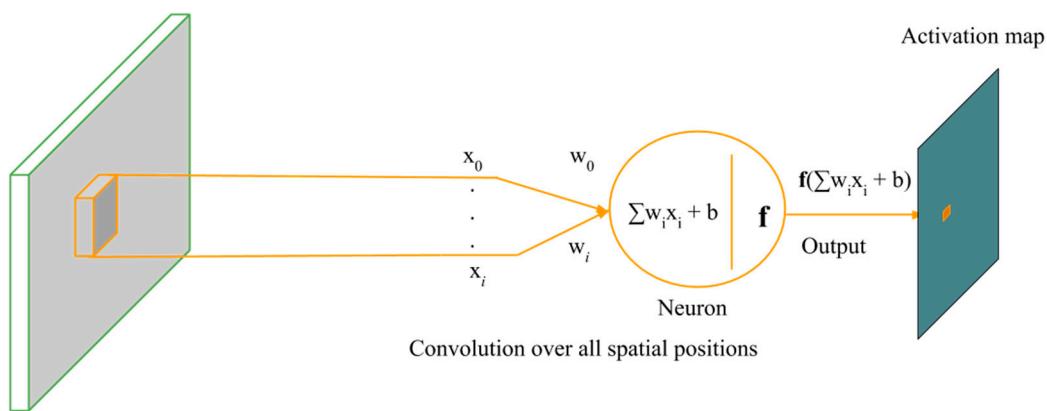


Figure 3. Illustration of the operation performed by a single neuron of the convolutional layer.

Although there are other types of nonlinear activation functions, such as the sigmoid and tanh, the most commonly used in ConvNets is the rectified linear unit (ReLU) [28] that thresholds the input at zero. ReLus are simple to implement and their non-saturating form accelerates the convergence of stochastic gradient descent [29].

2. Pooling layer

In addition to weight sharing, CNNs use pooling layers to mitigate overfitting risk. A pooling layer performs spatial resizing. Similar to convolutional layers, it also has stride and filter size parameters that control the spatial size of the output. Each element in the output activation map corresponds to the aggregate statistics of the input at the corresponding spatial position. In addition to control overfitting, pooling layers help to achieve spatial invariance [30]. The most commonly used pooling operations in CNNs are: (i) max pooling, which computes the maximum response of a given patch; (ii) average pooling, which computes the average response of a given patch; and (iii) subsampling, which computes the average over a patch of size $n \times n$, multiplies it by a trainable parameter β , adds a trainable bias b , and applies a nonlinear function (Equation (2)) [30]:

$$a_j = \tanh(\beta \sum_{i \in N \times N} a_i + b). \quad (2)$$

3. Fully connected layer

This layer is a regular multi-layer perceptron (MLP) used for classification, in which a neuron is connected to all neurons in the previous layer.

Once the network is set up, the weights and biases are learned by using variants of the gradient descent algorithm. The algorithm requires us to compute the derivative of a loss function with respect to the network parameters using the backpropagation algorithm. In the context of classification, the cross-entropy loss function is used in combination with the softmax classifier.

Training deep CNN architectures from scratch typically requires a very large training dataset, high computing power, and sometimes months of work. However, very powerful pre-trained models can be found, and they can be adapted to specific tasks either by fine tuning (using the network parameters as initialization and re-training with the new dataset) or as a simple feature extractor for the recognition task. Which type of transfer learning to use depends on the size of training dataset at hand and its affinity with the original dataset (exploited by the pre-trained model) [31]. In this work, we will make use of the publicly available trained CNN named GoogLeNet. It is trained for image classification tasks with the ImageNet ILSVRC2014 [32] challenge and ranked first. The challenge involved classifying images into one of 1000 leaf node categories in the ImageNet hierarchy. The ILSVRC dataset contains about 1.2 million images for training, 50,000 for validation, and 100,000 for testing. The network is 27 layers deep, including the pooling layers. Each convolutional layer contains 64 to 1024 filters of size 1×1 to 7×7 and they use a ReLu activation function. Max pooling kernels of size 3×3 and an average pooling kernel of size 7×7 are used in different layers of the network. The input layer takes a color image of size 224×224 . Besides the classification performance achieved by the network, the design of the deep architecture considers the power and memory usage of mobile and embedded platforms so that it can be put to real-world use at a reasonable cost. We refer readers to [33] for a detailed description of this model.

2.3. Classifier

The next step after feature extraction is to train a classifier suited for the task at hand. The choice of the classifier should take into account the dimensionality of the feature space, the number of training samples available, and any other requirements of the application. Motivated by their effectiveness in hyperdimensional classification problems, we will adopt the SVM classifier in this work. Introduced by Vapnik and Chervonenkis, SVMs are supervised learning models used to analyze data for classification and regression analysis. The main objective of such models is to find an optimal hyperplane or set of hyperplanes (in multiclass object discrimination problems) that separates a given dataset. They have been applied to a wide range of classification and regression tasks [34–36].

Consider a binary classification problem with N training samples in a d -dimensional feature space $x_i \in \Re^d$ ($i = 1, 2, 3, \dots, N$) with corresponding labels $y_i \in \{-1, +1\}$. There is an optimal hyperplane defined by a vector $\underline{w} \in \Re^d$ normal to the plane and a bias $b \in \Re$ that minimizes the cost function [37] given by:

$$\psi(\underline{w}, \xi) = \frac{1}{2} \|\underline{w}\|^2 + C \sum_{i=1}^N \xi_i, \quad (3)$$

subject to the following constraints:

$$\begin{cases} y_i(\underline{w} \cdot \phi(x_i) + b) \geq 1 - \xi_i, & i = 1, 2, 3, \dots, N \\ \xi_i \geq 0, & i = 1, 2, 3, \dots, N \end{cases}. \quad (4)$$

The cost function in Equation (3) combines both margin maximization (separation between the two classes) and error minimization (penalizing wrongly classified samples) in order to account for non-separability in real data. The slack variables (ξ_i 's) are used to take into account non-separable data, while C is a regularization parameter that allows us to control the penalty assigned to errors. Though initially designed for linearly separable data, SVMs were later extended to nonlinear patterns by using kernel tricks. A kernel function aims at transforming the original data into a new higher dimensional space using kernel functions ($\phi(\cdot)$'s) and classification (or regression) is performed in the transformed space. A membership decision is made based on the sign of a discriminant function $f(x)$ associated with the hyperplane. Mathematically,

$$\begin{aligned} \hat{y} &= \text{sign}\{f(x)\}, \text{ where} \\ f(x) &= \underline{w} \cdot \phi(x) + b. \end{aligned} \quad (5)$$

2.4. Post-Processing

In a video sequence, it can be reasonably expected that the change in content of successive frames is small. Therefore, it is highly likely for an object to appear in consecutive frames. With this in mind, we propose resorting to hidden Markov models to improve the decision of the classifier for a frame at time t based on the previous frame decisions. HMMs are statistical Markov models useful for characterizing systems where the unobserved internal state governs the external observations we make. They have been applied to a wide range of applications like human activity recognition from sequential images, bioinformatics, speech recognition, computational and molecular biology, etc., [38,39].

Consider a system with N distinct states, $S = \{s_1, s_2, s_3, \dots, s_N\}$, and M distinct observation symbols, $V = \{v_1, v_2, v_3, \dots, v_M\}$, per state. Given the following parameters,

1. State transition matrix, $A = [a_{ij}]$: probability that the system will be in state s_j at time t given the previous state is s_i .

$$a_{ij} = \Pr(q_t = s_j | q_{t-1} = s_i), \quad 1 \leq i, j \leq N; \quad (6)$$

where q_t is the state at time t .

2. Initial state probability, π : state of the system at time $t = 0$

$$\pi = \Pr(q_0 = s_i); \quad (7)$$

3. Observation symbol probability distribution in state s_j , $B = [b_j(k)]$

$$b_j(k) = \Pr(x_t = v_k | q_t = s_j), \quad 1 \leq j \leq N \\ 1 \leq k \leq M, \quad (8)$$

where x_t is the observation at time t , and given also the two main HMM assumptions, i.e., first-order Markov assumption (a state at time t only depends on a state at time $t - 1$) and the independence assumption (output observation at time t is only dependent on a state at time t), there are three basic problems that need to be solved in the development of a HMM methodology [39]. These are:

1. *Evaluation problem*: the objective of this problem is to calculate the probability of an observation sequence, $O = o_1, o_2, \dots, o_T$, given model parameters $\lambda = (A, B, \pi)$, i.e., $P(O|\lambda)$. In addition, it can be viewed as a way of evaluating how the model can predict the given observation sequence.
2. *Decoding problem*: it deals with finding the optimal state sequence, $S = s_1, s_2, \dots, s_T$, that best explains a given observation sequence, $O = o_1, o_2, \dots, o_T$, given model parameters λ .
3. *Learning problem*: it consists in estimating model parameters, $\lambda = (A, B, \pi)$, from a given training data (supervised or unsupervised) to maximize $P(O|\lambda)$.

For our detection problem, we have two hidden states, $S = \{s_1, s_2\}$, namely the presence and absence of an object in a frame (see Table 1). The observation variables, x , are image descriptors and our objective will be to maximize the instantaneous posterior probability (the probability that maximizes the decision of a frame at time t given all the previous observations). Mathematically,

$$q_t^* = \underset{1 \leq i \leq 2}{\operatorname{argmax}} P(q_t = s_i | o_1, o_2, \dots, o_t, \lambda). \quad (9)$$

Table 1. HMM notations in accordance to our detection problem.

s_1	$y = ' - 1'$
s_2	$y = ' + 1'$
o_t	x_t (image acquired at time t)
y_t	\hat{y} (Equation (4))

The state diagram is shown in Figure 4. There exists an efficient dynamic programming algorithm called the forward algorithm [39] to compute the probabilities. The algorithm consists of the following two steps:

1. *Prediction step*: predict the current state given all the previous observations:

$$P(q_t|x_{t-1}, x_{t-2}, \dots, x_1) = \sum_{s_{t-1}} P(q_t|q_{t-1})P(q_{t-1}|x_{t-1}, x_{t-2}, \dots, x_1); \quad (10)$$

2. *Update step*: update the prediction based on the current observation:

$$P(q_t|x_t, x_{t-1}, \dots, x_1) = \frac{P(x_t|q_t)P(q_t|x_{t-1}, x_{t-2}, \dots, x_1)}{\sum_{x_t} P(x_t|q_t)P(q_t|x_{t-1}, x_{t-2}, \dots, x_1)}, \quad (11)$$

using Bayes probability theorem

$$P(x_t|q_t) = \frac{P(q_t|x_t)P(x_t)}{P(q_t)}; \quad (12)$$

substituting Equation (12) into Equation (11), we obtain

$$P(q_t|x_t, x_{t-1}, \dots, x_1) = \frac{P(q_t|x_t)P(q_t|x_{t-1}, x_{t-2}, \dots, x_1)}{\sum_{x_t} P(q_t|x_t)P(q_t|x_{t-1}, x_{t-2}, \dots, x_1)}. \quad (13)$$

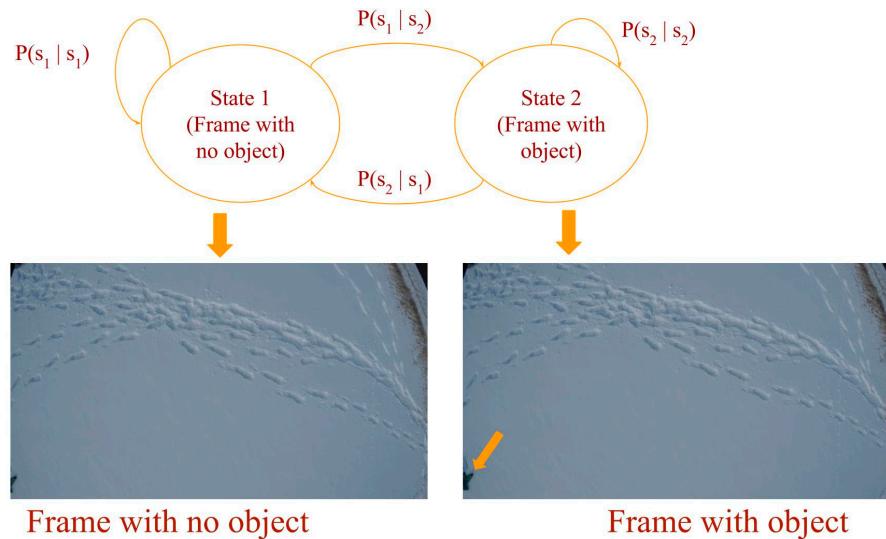


Figure 4. State transition diagram (the object pointed to by a yellow arrow is a jacket used to simulate the top half of a buried victim).

The posterior probability, $P(q_t|x_t)$, is obtained by converting the SVM classifier decision into a probability using the Platt scaling method [40]. Platt scaling is a way of transforming outputs of a discriminative classification model (like SVM) into a probability distribution over the classes. Given a

discriminant function, $f(x)$, of a classifier, the method works by fitting a logistic regression model to the classifier scores. Mathematically,

$$P(y = 1|x) = \frac{1}{1 + \exp(Af(x) + B)}, \quad (14)$$

where the parameters A and B are fitted using the maximum likelihood estimation method from a training set by minimizing the cross-entropy error function.

3. Data and Experimental Setup

3.1. Dataset Description

For this work, we have used two datasets. The first one was compiled by extracting successive frames from different videos of ski areas captured by UAVs freely available on the web. We edited the frames by placing objects of interest like body parts, backpacks, skis, etc. This dataset has a total of 270 frames, of which 165 were used for the training set and the rest for the test set. We have 59 and 52 positive samples in the training and test sets, respectively. The resolution of the images is 1280×720 . An example of positive and negative images is shown in Figure 5.

The second dataset is recorded on a mountain close to the city of Trento using a GoPro camera mounted on a CyberFed “Pinocchio” hexacopter. It consists of five videos of different durations recorded in 4K resolution (3840×2160) at a rate of 25 frames per second. For convenience, let us name them video 1, video 2, ..., up to video 5. Videos 1, 2, 3, and 4 are recorded at a height in the range of 2 to 10 m, while video 5 is recorded at a relatively higher height, between 20 and 40 m. The first two videos were recorded with the camera at 45° tip angle, while the others were captured with the camera pointing straight to the nadir. For this dataset, training set images are extracted from videos 1 and 2 and the rest are used for the test set. Sample frame snapshots are shown in Figure 6.



Figure 5. Example of positive (**top**) and negative (**bottom**) images from the first dataset. Objects of interest (partially buried skis (**left**) and top half-buried victim (**right**)) are marked with yellow circles.

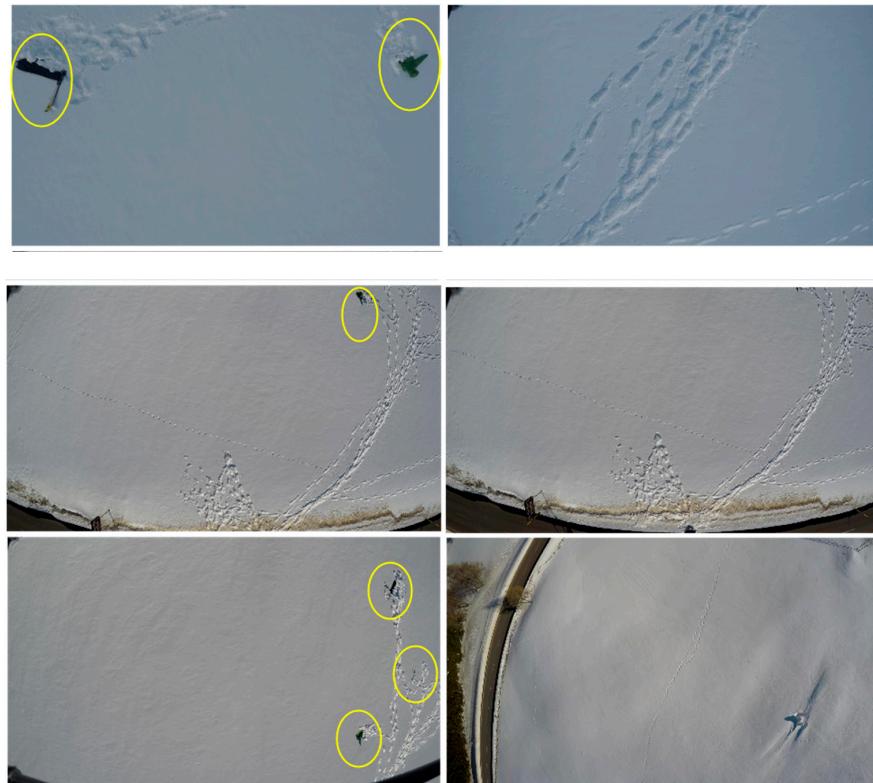


Figure 6. Positive (left) and negative (right) frame snapshots from the second dataset. Objects of interest (skis, jacket to simulate bottom half-buried victim, and ski pole) are marked by yellow circles.

3.2. Setup

As explained earlier, since our dataset is small and objects of interest are among the thousand classes onto which GoogLeNet is trained, we have used the network as a feature extractor. For this purpose, we removed the classification layer (layer 25) of the network. A forward propagation of zero center normalized image of size 224×224 through the network outputs a vector of image descriptor with 1024 elements. Moreover, since processing time is critical to our problem and the data are distributed in a high-dimensional space, we train a linear SVM for the task of classification. Both training and test features are scaled to have a unit length (Equation (14)) and the choice of best C (regularization factor) is performed with a grid search of values in the range of 2^{-15} to 2^5 using two-fold cross-validation.

$$x' = \frac{x}{\|x\|} \quad (15)$$

We have used the MatConvNet library [41] to operate on the pre-trained model and LibSVM library [42] to train SVM. All the experiments were conducted on a standard desktop computer with clock speed of 3 GHz and 8 GB RAM.

4. Results and Discussion

In this section, we report the experimental results obtained for both datasets. General information about all experiments can be found in Table 2. Accuracy, probability of true positives (P_{TP}), and probability of false alarm (P_{FA}) are the performance metrics used. P_{TP} and P_{FA} are calculated as follows:

$$P_{TP} = \frac{\sum \# \text{ of positive samples correctly classified}}{\sum \# \text{ of positive samples}} \quad (16)$$

$$P_{FA} = \frac{\sum \# \text{ of negative samples classified as positive}}{\sum \# \text{ of negative samples}}. \quad (17)$$

Table 2. General description of experiments conducted.

		Original Image Resolution	Resized to	Pre-Processing
First Dataset	Experiment 1	1280 × 720	224 × 224	No
	Experiment 2	1280 × 720	672 × 448, then 6 tiles (224 × 224)	No
	Experiment 3	1280 × 720	1120 × 672, then 15 tiles (224 × 224)	No
Second Dataset	Experiment 4	3840 × 2160	224 × 224	No
	Experiment 5	3840 × 2160	640 × 480	Yes
	Experiment 6	3840 × 2160	1280 × 720	Yes
	Experiment 7	3840 × 2160	1920 × 1080	Yes
	Experiment 8	3840 × 2160	No resizing	Yes

4.1. Experiments without Pre-Processing

For the first dataset, we conducted three separate experiments at different resolutions. The first experiment is conducted by resizing both training and test frames to an input size, 224×224 , of the pre-trained model and extracting the features. In the second experiment, each frame is divided into six tiles of 224×224 each after resizing to 672×448 (close to VGA). In the third experiment, 15 tiles of size 224×224 are generated from each frame after resizing to 1120×672 (close to the original resolution). The results are reported in Table 1.

From Table 3, it is clear that the overall accuracy increases and P_{FA} decreases with an increase in resolution. Contrarily, P_{TP} decreases for the second and third experiments with respect to the first and increases for the third experiment with respect to the second. We believe that the reason for having a high P_{TP} in the first experiment is because we are considering the whole frame, which contains unwanted objects like poles, trees, lift lines, etc. In the first experiment we have high P_{FA} because the whole frame is resized to 224×224 . The resizing makes objects of interest become insignificant with respect to the surrounding and thus forces the classifier to learn not only objects of interest but also the surroundings. On the other hand, the second and third experiments have small P_{FA} and increased P_{TP} due to tiling, which makes objects of interest in a tile more significant with respect to the surroundings and the classifier is able to better discriminate objects of interest from the background. Some qualitative results are shown in Figure 7.

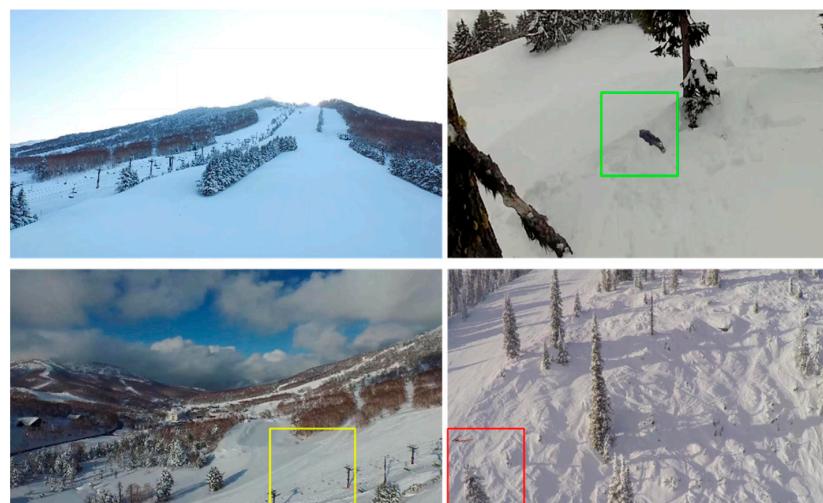


Figure 7. Example of correctly classified negative (**top left**) and positive (**top right**), false positive object marked in yellow rectangle (**bottom left**), and false negative object marked by red rectangle (**bottom right**).

Table 3. Classification results for the first dataset (Experiments 1–3).

	Accuracy (%)	P_{TP}	P_{FA}
Experiment 1	65.71	0.8462	0.5283
Experiment 2	94.29	0.6346	0.1095
Experiment 3	97.59	0.8065	0.0152

For the second dataset, the first experiment (Experiment 4 in Table 2) we conducted by downsampling each frame to a size of 224×224 . For this experiment, the training set is made up of 4000 frames, of which 2000 are positive samples, extracted from the first two videos. From the results in Table 4, video 3 has high accuracy and very low P_{FA} as compared to the other test videos. This is mainly due to the nature of the video. Almost all frames are either snow (white) or objects of interest on top of snow. So, downsampling the frames will not affect the visibility of objects of interest. On the other hand, frames from videos 4 and 5 contain background objects like cars, trees, etc. Additionally, video 5 is recorded at a higher height. For the reasons mentioned above, downsampling a frame to 224×224 results in higher insignificance of objects of interest with respect to the background and hence a high P_{FA} .

Table 4. Classification results for the second dataset at resolution of 224×224 (Experiment 4).

	Accuracy (%)	P_{TP}	P_{FA}
Video 3	84.34	0.8386	0.1470
Video 4	36.25	0.9405	0.7445
Video 5	44.13	0.4311	0.5472

4.2. Experiments with Pre-Processing

Next, we conducted four separate experiments at resolutions of 640×480 , 1280×720 , 1920×1080 , and 3840×2160 , respectively. Since the number of frames in this dataset is large, tiling each frame and labeling each tile is time-consuming. Alternatively, we composed a training set with 3000, of which 1500 are positive, image crops of size 224×224 from the first two videos at the original resolution and trained a linear SVM. During the test phase, each frame is scanned with a sliding window of size 80×80 and if a window passes the threshold, a crop of size 224×224 centered on the window is taken for further processing with the next steps. An example of this process is shown in Figure 8.

As seen from the results in Tables 5 and 6, for video 3 (experiments 5 to 8), the overall accuracy increases with an increase in resolution as compared to the results obtained in experiment 4. An exception is at the VGA resolution, where there is a decrease in accuracy due to loss of detail in downsampling. As expected, the probability of a false alarm (P_{FA}) drops significantly with an increase in resolution. On the other hand, P_{TP} has decreased with respect to the results obtained in experiment 4. However, it started to increase as resolution improved, yielding a significant increase at 4K resolution (experiment 8). We believe that the decrease is due to the difference in the training sets used for experiment 4 and experiments 5 to 8, while the increase is due to the more detailed information available with an increase in resolution.

Similarly, for video 4, the overall accuracy improves significantly as compared to the results obtained in experiment 4. However, it starts to drop, as compared to the result at VGA resolution (experiment 5), with an increase in resolution. In experiment 4 we have a high P_{FA} , but it decreases significantly as the resolution is improved. However, as we go from VGA (experiment 5) to 4K (experiment 8) resolution, there is an increase in P_{FA} . This is because of objects or part of objects in the background that have similarity with objects of interest, thus incurring the classifier in more wrong decisions. Moreover, the increase in P_{FA} has a negative impact on the overall accuracy. Though initially we have a decrease in P_{TP} at the VGA resolution with respect to the results obtained in experiment 4, there is an increase and stability in the rest of the experiments.

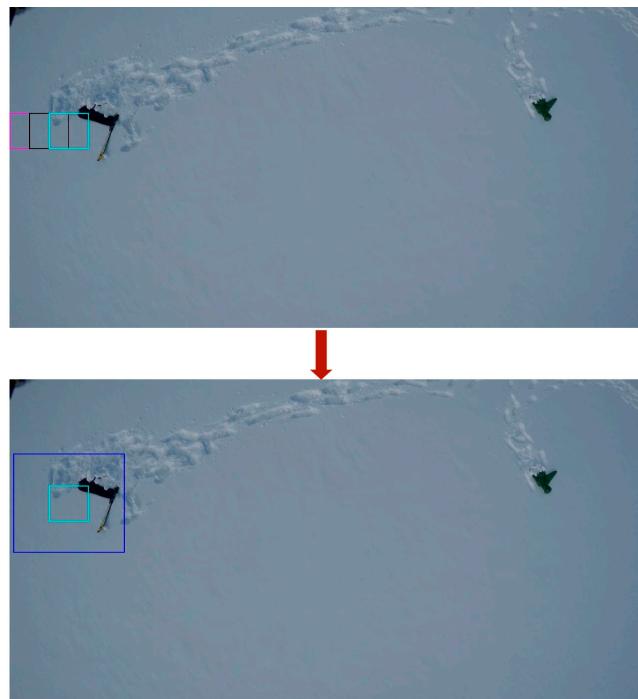


Figure 8. Example showing the pre-processing step. The image on top shows a frame being scanned by a sliding window, while the image on the bottom highlights a region (marked by blue rectangle), centered around a window (marked by cyan rectangle) selected for further processing.

Table 5. Classification results for the second dataset at 640×480 and 1280×720 resolutions (Experiments 5 and 6).

Experiment 5			Experiment 6		
	Accuracy (%)	P_{TP}		Accuracy (%)	P_{TP}
Video 3	78.95	0.6383	0.0020	88.40	0.8061
Video 4	96.93	0.8452	0.0080	93.31	0.9940
Video 5	62.72	0.3352	0.0409	67.72	0.4259

Table 6. Classification results for the second dataset at 1920×1080 and 3840×2160 resolutions (Experiments 7 and 8).

Experiment 7			Experiment 8		
	Accuracy (%)	P_{TP}		Accuracy (%)	P_{TP}
Video 3	90.01	0.8333	0.0080	94	0.9084
Video 4	77.32	0.9940	0.2687	70.63	0.9940
Video 5	74.34	0.5723	0.0620	78.93	0.7087

For video 5, we have a significant increase in the overall accuracy as the resolution increases. P_{TP} initially decreases at VGA resolution (experiment 5) with respect to the results obtained in experiment 4, but it starts to increase as the resolution increases. Moreover, we have less P_{TP} as compared to other videos because of the height at which the video is captured. Similar to the other videos, P_{FA} drops significantly with an increase in resolution. However, there is also a slight increase in experiments 5 to 8 due to reasons similar to those mentioned for video 4. Some qualitative results are shown in Figures 9 and 10.

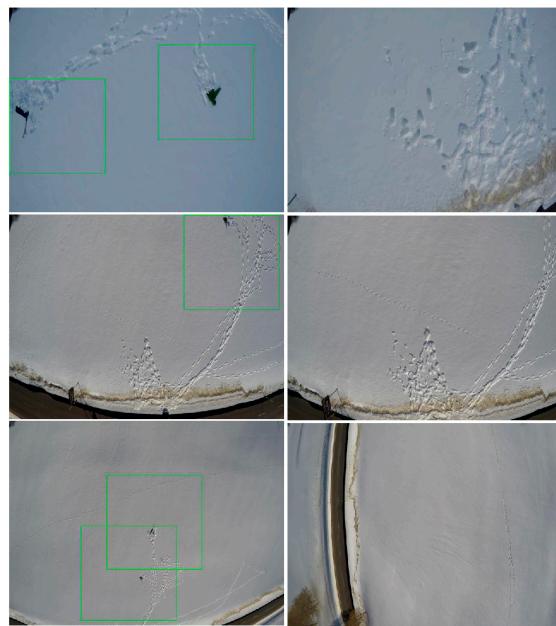


Figure 9. Snapshot of frames with correct positive (left) and negative (right) detection results at the VGA resolution from the second dataset. Regions of a frame containing an object are shown with a green rectangle.

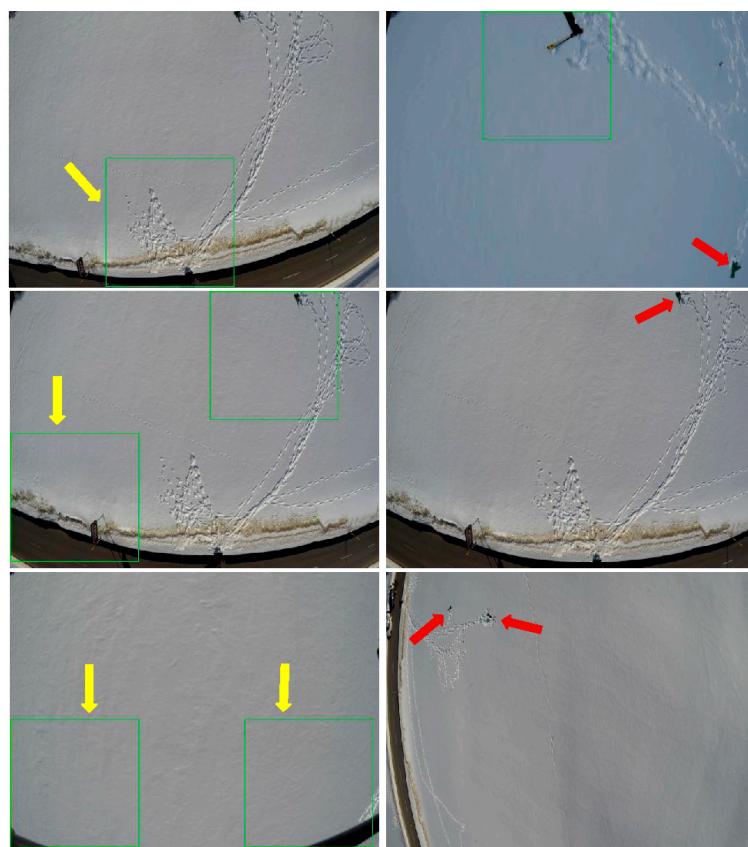


Figure 10. Examples of false positive (left) and false negative (right) frame snapshots at VGA resolution. Yellow arrows indicate false positive regions in a frame, whereas red arrows show missed objects in a frame.

4.3. Experiments with Markovian Post-Processing

In the previous experiments, decisions are made separately for each frame. However, in a video sequence, there is a correlation between successive frames and performance can be further improved by embedding this information in the decision-making process. As described in the previous methodological section, we have used HMMs to opportunely exploit this information. Model parameters, prior distribution, transition matrix, and observation probability distribution are calculated as follows:

- We have initialized prior distribution in such a way that the probability that there is no object in the initial frame is high. For such a purpose, we fixed this prior probability value to 0.9.
- The state transition matrix (Table 7) is calculated from the available labeled frames.
- Instead of the observation probability distribution, we use the posterior probability by converting SVM discriminant function value into a probability value using Platt's method and use it in the modified equation of the forward algorithm mentioned in Section 2.

The effect of post-processing on the prediction performance can be positive or negative. Indeed, it can correct wrong predictions made by the classifier (positive change) or change the correct prediction made by the classifier into a wrong prediction (negative change). Moreover, these positive or negative changes occur between successive frames where there is a transition from one state to the other in the prediction of the classifier. For example, consider two successive frames, at time t and $t - 1$. If the decision of the SVM at time t is different than the decision made by HMM for the frame at time $t - 1$, because of the small state transition probabilities it is highly likely for the HMM to remain in the same state for the current frame, thereby changing the decision of the SVM. Depending on the original label of the frame, this change can be either positive or negative. Therefore, the prediction performance of the system can either increase if there are more positive changes than negative changes or decrease if there are more negative changes than positive ones.

The results in Tables 8–10 show that for video 3 the impact of HMM is not that significant in improving P_{FA} . On the other hand, P_{TP} improves by more than 2% at the VGA resolution. For video 4, since the number of positive frames is very small an increase or decrease in P_{TP} does not affect the overall accuracy. For example, P_{TP} increases by 6% in the first experiment and decreases by approximately 10% at the VGA resolution, but the effect on the overall accuracy is very small. With an increase in resolution P_{FA} gets improved and accuracy increases by more than 5%. Though post-processing has a negative effect on the accuracy for video 5, we can see from the results that, as the resolution increases, P_{FA} drops and, consequently, the difference between the accuracies (achieved with and without post-processing) decreases. In general, it is possible to see that the gain of post-processing depends on the goodness of the classifier. When P_{TP} is high and P_{FA} is low, prediction performance gets improved or remains the same. In all other cases, the impact on prediction performance, especially on the overall accuracy, depends on the ratio of positive and negative frames. Examples of the positive and negative changes made by HMM are given in Figures 11 and 12.

Table 7. State transition matrix.

	Current Frame with No Object	Current Frame with Object
Previous frame with no object	0.9977	0.0023
Previous frame with object	0.0023	0.9977

Table 8. HMM detection result at resolution of 224×224 .

	Accuracy (%)	P_{TP}	P_{FA}
Video 3	84.95	0.8450	0.1440
Video 4	36.06	1	0.7580
Video 5	42.64	0.4120	0.5570

Table 9. HMM detection results at VGA and 720p resolutions.

	640×480			1280×720		
	Accuracy (%)	P_{TP}	P_{FA}	Accuracy (%)	P_{TP}	P_{FA}
Video 3	80.52	0.6642	0.0010	88.70	0.8090	0.0051
Video 4	96	0.7440	0.0010	95.26	0.9880	0.0517
Video 5	59.7	0.2768	0.0340	65.47	0.3712	0.0299

Table 10. HMM detection results at 1080p and 4K resolutions.

	1920×1080			3840×2160		
	Accuracy (%)	P_{TP}	P_{FA}	Accuracy (%)	P_{TP}	P_{FA}
Video 3	89.39	0.8211	0.0056	93.29	0.8910	0.0091
Video 4	82.89	0.99	0.2033	72.86	0.99	0.3226
Video 5	72.80	0.5178	0.0330	77.45	0.6179	0.0463

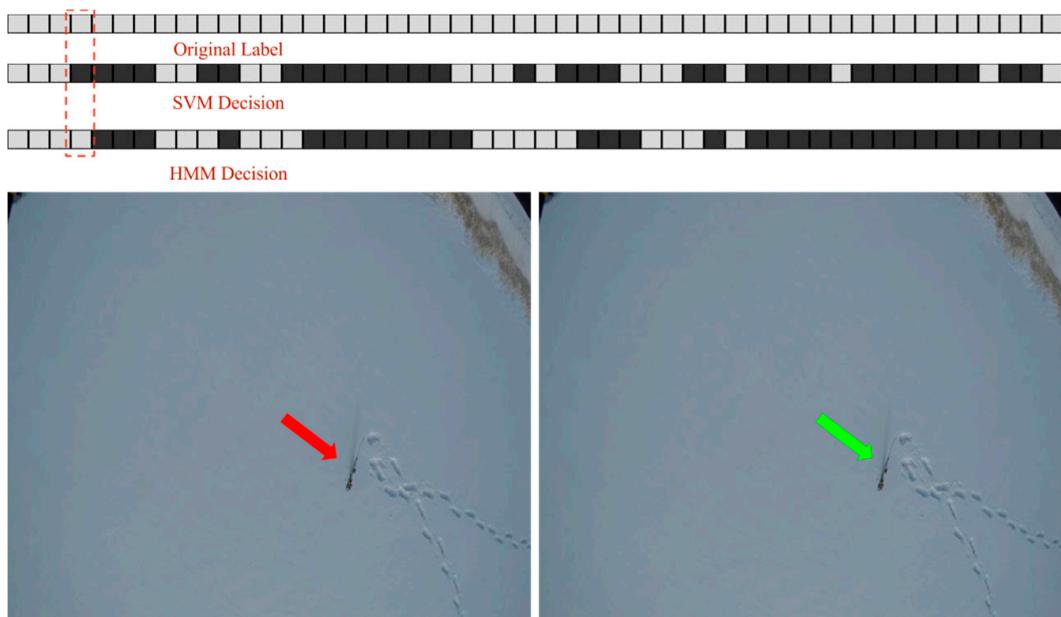


Figure 11. Example of positive change by HMM. The sequence of white and black squares on top indicates the label of successive frames. A white square indicates a frame has object of interest whereas a black square indicates the opposite. The frame where the change happened is outlined by a red dotted rectangle and the corresponding frame in the bottom. The frame for which SVM made a wrong decision is shown in the bottom left (the object in the frame, skis in this case, is indicated by a red arrow), whereas the same frame corrected by HMM is shown in the bottom right (the object in the frame is indicated by a green arrow). Note that the object is not localized since a post-processing decision is made at the frame level.

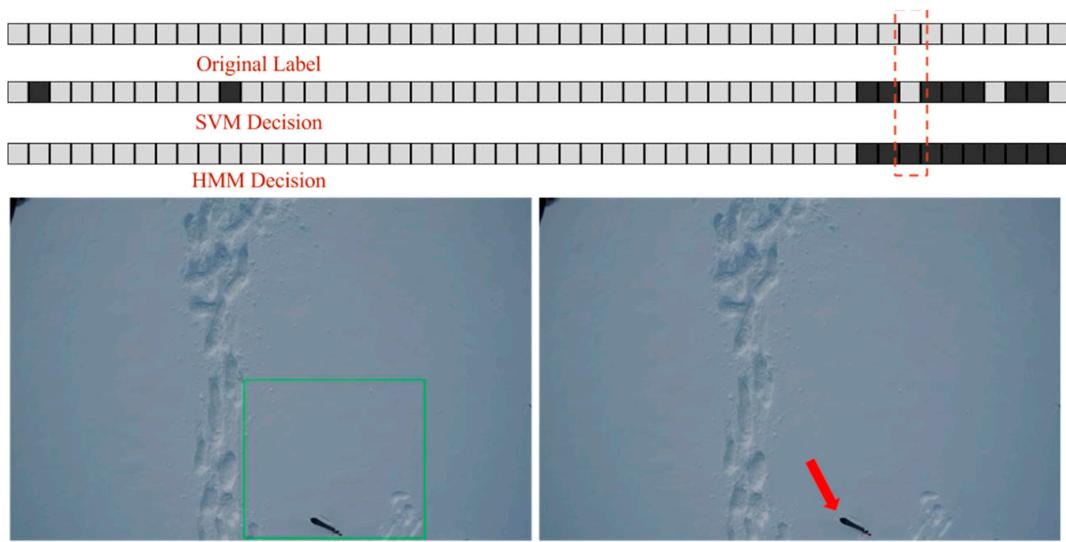


Figure 12. Example of negative change by HMM. The sequence of white and black squares on top indicates the label of successive frames. White squares indicate a frame has an object of interest, whereas black squares indicate the opposite. The frame where the change happened is outlined by a red dotted rectangle. The frame for which SVM made the right decision, with the object localized in a green rectangle, is shown in the bottom left. The same frame, for which HMM made a wrong decision, is shown in the bottom right.

4.4. Computation Time

The processing time required to extract CNN features and perform the prediction for an input image of size 224×224 is 0.185 s. For both the first and second datasets, detection at a resolution of 224×224 can be done at a rate of 5.4 frames per second. For the first dataset, since we used tiling to do detection at higher resolutions, the processing time is the product of the number of tiles per frame with the processing time required for a single tile (0.185 s). Therefore, at near VGA and full resolutions, the detection rates are 0.9 and 0.36 frames per second, respectively. For the second dataset, since we have the pre-processing step, we only extract features and perform prediction on frames that pass this step. Additionally, there can be more than one crop of size 224×224 from a single frame. The average processing time is reported in Table 11. The advantage of pre-processing as compared to the tiling approach is twofold. First, it reduces the processing time; second, it provides better localization of objects within a frame.

In general, from the experimental results obtained, it emerges that working at a higher resolution provides a significant improvement in prediction performance at a cost of increased processing time. The bar graph in Figure 13 shows the average accuracy and processing time for the second dataset.

Table 11. Detection speed (number of frames per second) for the second dataset.

	Video 3	Video 4	Video 5
224×224	5.4	5.4	5.4
640×480	3.63	1.8	2.88
1280×720	2.25	1.15	1.65
1920×1080	1.48	0.86	0.98
3840×2160	0.41	0.32	0.24

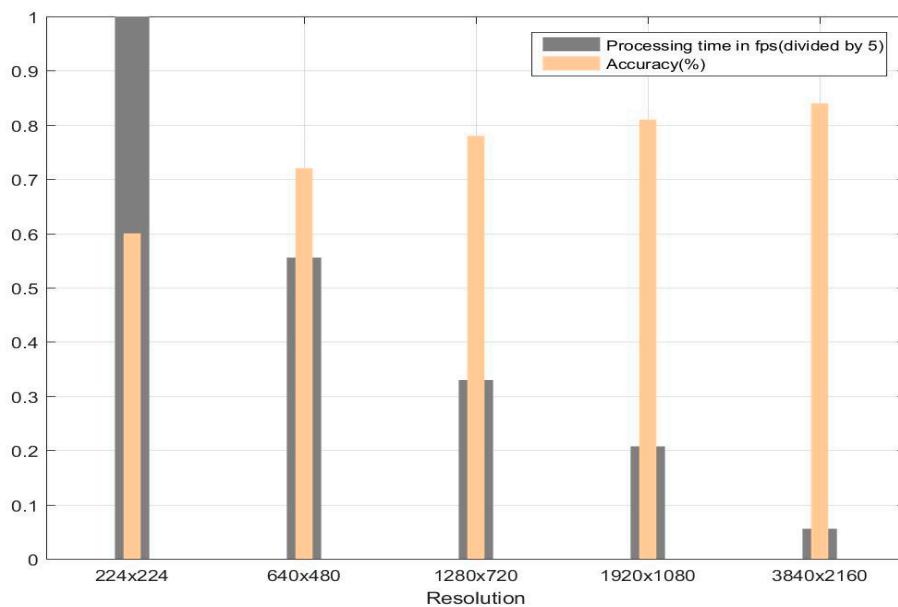


Figure 13. Bar graph showing the change in accuracy and detection rate as resolution increases.

4.5. Comparative Study

For the purpose of comparison, we conducted experiments at the higher resolutions available for both datasets using histograms of oriented gradients (HOG) feature extraction method. Histograms of oriented gradients (HOG) [40] is a method that is used to represent local object appearance and shape using local intensity gradients or edge directions. For a given image window, HOG features are computed as follows. First, the window is divided into small spatial areas called cells and each cell is represented by a 1-d histogram of gradients computed for each pixel. Next, cells are grouped spatially to form larger areas called blocks. Each block is then represented by a histogram of gradients, which is a concatenation of the normalized 1-d histogram of gradients of each cell within the block. The final HOG feature descriptor of the image window is formed by concatenating the aforementioned histograms of the blocks.

In our experiments, the parameters for HOG are set up as follows: the cell size is set to 32×32 pixels, the block size is set to 2×2 with 50% overlap, and a 9-bin histogram is used to represent the cell gradients. For both datasets, HOG descriptors are extracted from an image window of size 224×224 and a linear SVM is trained for the classification. The best regularization parameter (C) of the classifier is selected by using grid search and cross validation method. As the results in Tables 12 and 13 show, the overall accuracy of HOG-SVM classifier is significantly less than that of the CNN-SVM classifier. Additionally, the HOG-SVM classifier generates high false alarms (F_p) as compared to the CNN-SVM classifier. Our results thus confirm the idea that a generic classifier trained on deep features outperforms a classifier trained on features extracted with the traditional method [25,26].

Table 12. Comparison of HOG and CNN feature extraction methods for the first dataset.

	CNN-SVM	HOG-SVM
Accuracy (%)	97.59	61.91
P_{TP}	0.8065	0.8269
P_{FA}	0.0152	0.5849

Table 13. Comparison of HOG and CNN feature extraction methods for the second dataset.

	CNN-SVM			HOG-SVM		
	Video 3	Video 4	Video 5	Video 3	Video 4	Video 5
Accuracy (%)	94	70.63	78.93	85.82	62.45	63.24
P_{TP}	0.9084	0.994	0.7087	0.7596	1	0.8161
P_{FA}	0.0164	0.3480	0.1191	0.0056	0.4449	0.5763

5. Conclusions

In this work, we have presented a method to support avalanche SAR operations using UAVs equipped with vision cameras. The UAVs are used to acquire EHR images of an avalanche debris and the acquired image is processed by a system composed of a pre-processing method to select regions of interest within the image, a pre-trained CNN to extract suitable image descriptors, a trained linear SVM classifier for object detection and a post-processing method based on HMM to further improve detection results of the classifier.

From the experimental results, it is clear that improved resolution results in an increase in prediction performance. This is mainly due to the availability of more detailed information at a higher resolution, which enables the decision system to better discriminate objects of interest from the background. Contrarily, we have also seen an increase in false alarms because of background objects or parts of objects that exhibit similarity with the objects of interest. Though the computation time increases with an increase in resolution, it is possible to assert that, except at full resolution, the processing time is acceptable for such applications. Additionally, as seen from experimental results of video 5, the height at which frames are acquired is also an important factor that impacts on the prediction performance, and the results obtained with the other test videos suggest that scanning the debris at a lower altitude is preferable for better detection performance. Finally, the choice of resolution to perform detection should be done according to a tradeoff between accuracy and processing time.

Two main limitations can be observed in this study. The first is that the datasets used for training/testing are not yet fully representative. For example, the second dataset is characterized by very few objects. Although the task is not easy, it would be important to collect a more complete dataset by varying the context of the avalanche event and the conditions of partial burial of the victims, and by increasing the kinds of objects. The second limitation is that the thresholding mechanism used in the pre-processing depends on the single pixel intensities. Due to the loss of information incurred by image resizing, pixels associated with some of the objects fail to pass the threshold and hence objects are not detected. However, as the experimental results show, this problem is reduced with an increase in resolution. Since the main objective of the pre-processing is to reduce computational time by elaborating only a portion of a frame or skipping a frame, a method that is more robust at lower resolutions can be a topic of further research.

Operational scenarios of the proposed method are two. In the first one, the data are transmitted in real time to the ground station where the processing is performed in order to alert the operator when objects of interest are detected while the UAV (or a swarm of UAVs) performs the scans of the avalanche areas. In this scenario, problems of communication links between the drone and the ground station need to be resolved beforehand. In the second scenario, the processing is performed onboard the UAV. This allows us to reduce considerably the amount of information to be sent toward the ground station, which in this case can be reduced to simple flag information whenever a frame containing objects of interest is detected. The drawback is the processing capabilities, which are reduced with respect to those of a ground station. Work is in progress for an onboard implementation. Moreover, it is noteworthy that, although the first two videos used for training are acquired at 45°, we assume the acquisition to be performed at the nadir and the processing is performed on a frame-by-frame basis. A critical parameter that was thoroughly investigated in this study is the UAV height, which impacts

directly on the image resolution. There are other factors like illumination conditions and UAV stability in the presence of wind that deserve to be investigated in the future.

Acknowledgments: This research was carried out within the framework of a project entitled ‘Drones for Finding Avalanche-Buried’ funded by the University of Trento, Italy.

Author Contributions: All authors contributed to the conception and design of the methodology; M. Bejiga and F. Melgani dealt with the experimental validation and wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Society, N.G. Avalanche Facts, Avalanche Information, Avalanche Videos, Avalanche Photos—National Geographic. Available online: <http://environment.nationalgeographic.com/environment/natural-disasters/avalanche-profile/> (accessed on 26 December 2016).
2. Long-Term Statistics. Available online: http://www.slf.ch/praevention/lawinenunfaelle/lawinenstatistik/index_EN (accessed on 3 November 2016).
3. Brugger, H.; Etter, H.J.; Zweifel, B.; Mair, P.; Hohlrieder, M.; Ellerton, J.; Elsensohn, F.; Boyd, J.; Sumann, G.; Falk, M. The impact of avalanche rescue devices on survival. *Resuscitation* **2007**, *75*, 476–483. [[CrossRef](#)] [[PubMed](#)]
4. Drones can Save Lives in Search and Rescue Mission after Avalanche. Available online: <http://www.aerialtronics.com/2015/03/drones-can-save-lives-in-search-and-rescue-mission-after-avalanche/> (accessed on 26 October 2016).
5. European Satellite Navigation Competition. Available online: <http://www.esnc.info/index.php?anzeige=switzerland10.html> (accessed on 26 October 2016).
6. First Onsite Test for the UNIBO Drone Supporting Search and Rescue in Avalanches. Available online: <http://www.unibo.it/en/research/first-onsite-test-for-the-unibo-drone-supporting-search-and-rescue-in-avalanches> (accessed on 26 October 2016).
7. Fruehauf, F.; Heilig, A.; Schneebeli, M.; Fellin, W.; Scherzer, O. Experiments and Algorithms to Detect Snow Avalanche Victims Using Airborne Ground-Penetrating Radar. *IEEE Trans. Geosci. Remote Sens.* **2009**, *47*, 2240–2251. [[CrossRef](#)]
8. Stepanek, J.; Claypool, D.W. GPS signal reception under snow cover: A pilot study establishing the potential usefulness of GPS in avalanche search and rescue operations. *Wilderness Environ. Med.* **1997**, *8*, 101–104. [[CrossRef](#)]
9. Schleppe, J.B.; Lachapelle, G. GPS tracking performance under avalanche deposited snow. In Proceedings of the 19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2006), Fort Worth, TX, USA, 26–29 September 2006.
10. Wolfe, V.; Frobe, W.; Shrinivasan, V.; Hsieh, T.Y. Detecting and locating cell phone signals from avalanche victims using unmanned aerial vehicles. In Proceedings of the 2015 International Conference on Unmanned Aircraft Systems, Denver, CO, USA, 9–12 June 2015; pp. 704–713.
11. Rudol, P.; Doherty, P. Human Body Detection and Geolocalization for UAV Search and Rescue Missions Using Color and Thermal Imagery. In Proceedings of the 2008 IEEE Aerospace Conference, Manhattan, CA, USA, 1–8 March 2008; pp. 1–8.
12. Andriluka, M.; Schnitzspan, P.; Meyer, J.; Kohlbrecher, S.; Petersen, K.; Stryk, O.V.; Roth, S.; Schiele, B. Vision based victim detection from unmanned aerial vehicles. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Taipei, Taiwan, 18–22 October 2010; pp. 1740–1747.
13. Felzenszwalb, P.; McAllester, D.; Ramanan, D. A discriminatively trained, multiscale, deformable part model. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Anchorage, AK, USA, 23–28 June 2008; pp. 1–8.
14. Andriluka, M.; Roth, S.; Schiele, B. Pictorial structures revisited: People detection and articulated pose estimation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Miami Beach, FL, USA, 20–25 June 2009; pp. 1014–1021.

15. Bourdev, L.; Malik, J. Poselets: Body part detectors trained using 3d human pose annotations. In Proceedings of the 2009 IEEE Conference on 12th International Conference on Computer Vision, Koyoto, Japan, 29 September–2 October 2009; pp. 1365–1372.
16. Moranduzzo, T.; Melgani, F.; Bazi, Y.; Alajlan, N. A fast object detector based on high-order gradients and Gaussian process regression for UAV images. *Int. J. Remote Sens.* **2015**, *36*, 2713–2733.
17. Moranduzzo, T.; Melgani, F. Automatic Car Counting Method for Unmanned Aerial Vehicle Images. *IEEE Trans. Geosci. Remote Sens.* **2014**, *52*, 1635–1647. [CrossRef]
18. Moranduzzo, T.; Melgani, F. Detecting Cars in UAV Images with a Catalog-Based Approach. *IEEE Trans. Geosci. Remote Sens.* **2014**, *52*, 6356–6367. [CrossRef]
19. Brugger, H.; Falk, M. Analysis of Avalanche Safety Equipment for Backcountry Skiers. Available online: https://s3.amazonaws.com/BackcountryAccess/content/papers/brugger_falk_report_2002.pdf (accessed on 11 November 2016).
20. Sural, S.; Qian, G.; Pramanik, S. Segmentation and histogram generation using the HSV color space for image retrieval. In Proceedings of the 2002 International Conference on Image Processing, Rochester, NY, USA, 22–25 September 2002.
21. Viola, P.; Jones, M. Rapid object detection using a boosted cascade of simple features. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Kauai, HI, USA, 8–14 December 2001.
22. Lowe, D.G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
23. Dalal, N.; Triggs, B. Histograms of oriented gradients for human detection. In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05), San Diego, CA, USA, 20–25 June 2005; Volume 1, pp. 886–893.
24. Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Miami, FL, USA, 20–25 June 2009; pp. 248–255.
25. Sharif Razavian, A.; Azizpour, H.; Sullivan, J.; Carlsson, S. CNN features off-the-shelf: An astounding baseline for recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Columbus, OH, USA, 23–28 June 2014; pp. 806–813.
26. Donahue, J.; Jia, Y.; Vinyals, O.; Hoffman, J.; Zhang, N.; Tzeng, E.; Darrell, T. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In Proceedings of the ICML, Beijing, China, 21–16 June 2014; pp. 647–655.
27. O’Shea, K.; Nash, R. An Introduction to Convolutional Neural Networks. Available online: https://white.stanford.edu/teach/index.php/An_Introduction_to_Convolutional_Neural_Networks (accessed on 11 November 2016).
28. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
29. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2012; pp. 1097–1105.
30. Scherer, D.; Müller, A.; Behnke, S. Evaluation of pooling operations in convolutional architectures for object recognition. In Proceedings of the International Conference on Artificial Neural Networks, Thessaloniki, Greece, 15–18 September 2010; pp. 92–101.
31. CS231n Convolutional Neural Networks for Visual Recognition. Available online: <http://cs231n.github.io/transfer-learning/> (accessed on 27 December 2016).
32. ImageNet Large Scale Visual Recognition Competition (ILSVRC). Available online: <http://www.image-net.org/challenges/LSVRC/> (accessed on 27 December 2016).
33. Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; Rabinovich, A. Going deeper with convolutions. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 1–9.
34. Douha, L.; Benoudjit, N.; Douak, F.; Melgani, F. Support Vector Regression in Spectrophotometry: An Experimental Study. *Crit. Rev. Anal. Chem.* **2012**, *42*, 214–219. [CrossRef]
35. Pasolli, E.; Melgani, F.; Tuia, D.; Pacifici, F.; Emery, W.J. SVM Active Learning Approach for Image Classification Using Spatial Information. *IEEE Trans. Geosci. Remote Sens.* **2014**, *52*, 2217–2233. [CrossRef]

36. Segata, N.; Pasolli, E.; Melgani, F.; Blanzieri, E. Local SVM approaches for fast and accurate classification of remote-sensing images. *Int. J. Remote Sens.* **2012**, *33*, 6186–6201. [[CrossRef](#)]
37. Vapnik, V.N. *Statistical Learning Theory*, 1st ed.; Wiley-Interscience: New York, NY, USA, 1998.
38. Kim, E.; Helal, S.; Cook, D. Human Activity Recognition and Pattern Discovery. *IEEE Pervasive Comput.* **2010**, *9*, 48–53. [[CrossRef](#)] [[PubMed](#)]
39. Rabiner, L.R. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. IEEE* **1989**, *77*, 257–286. [[CrossRef](#)]
40. Platt, J. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Adv. Large Margin Classif.* **1999**, *10*, 61–74.
41. Vedaldi, A.; Lenc, K. MatConvNet: Convolutional Neural Networks for MATLAB. In Proceedings of the 23rd ACM International Conference on Multimedia, New York, NY, USA, 26–30 October 2015; pp. 689–692.
42. Chang, C.-C.; Lin, C.-J. LIBSVM: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.* **2011**, *2*, 27. [[CrossRef](#)]



© 2017 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).