HUMAN-TOOL-INTERACTION-BASED ACTION RECOGNITION FRAMEWORK FOR AUTOMATIC CONSTRUCTION OPERATION MONITORING

A Dissertation

by

YANG LIU

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Julian Kang
Co-Chair of Committee,	Wei Yan
Committee Members,	Edelmiro Escamilla
	Zofia Rybkowski
Head of Department,	Robert B. Warden

August 2017

Major Subject: Architecture

ABSTRACT

Monitoring activities on a construction jobsite is one of the most important tasks that a construction management team performs every day. Construction management teams monitor activities to ensure that a construction project progresses as scheduled and that the construction crew works properly in a safe working environment. However, site monitoring is often time-consuming. Various automated or semi-automated tracking approaches such as radio frequency identification, Global Positioning System, ultrawide band, barcode, and laser scanning have been introduced to better monitor activities on the construction site. However, deploying and maintaining such techniques require a high level of involvement by very specific well-trained professionals and could be costly.

As an alternative way to monitor sites, object recognition and tracking have the advantage of requiring low human involvement and intervention. However, it is still a challenge to recognize construction crew activities with existing methods, which have a high false recognition rate. This research proposes a new approach for recognizing construction personnel activity from still images or video frames. The new approach mimics the human thinking process with the assumption that a construction worker performs a certain activity with a specific body pose using a specific tool. The new approach consists of two recognition tasks, construction worker pose recognition and

tool recognition. The two recognition tasks are connected in sequence with an interactive spatial relationship.

The proposed method was developed into a computer application using Matlab. It was compared against a benchmark method that only uses construction worker body pose for activity recognition. The benchmark method was also developed into a computer application with Matlab. The proposed method and the benchmark method were tested with the same sample set containing 500 images of over 10 different construction activities. The experimental results show that the proposed framework achieved a higher reliability (precision value), a lower sensitivity (recall value), and an overall better performance (F_1 score) than the benchmark method.

DEDICATION

This dissertation is dedicated to my parents and to my wife. I could not accomplish this without you. You are my hope and the reason for me to keep fighting.

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Kang, my co-chair, Dr. Yan, and my committee members, Dr. Escamilla and Dr. Rybkowski, for their guidance and support throughout the course of this research.

Thanks also go to my friends and colleagues and the department faculty and staff for making my time at Texas A&M University a great experience.

CONTRIBUTORS AND FUNDING SOURCES

This work was supervised by a dissertation committee consisting of Dr. Julian Kang (committee chair), Dr. Escamilla, and Dr. Rybkowski of the Department of Construction Science and Dr. Wei Yan (committee co-chair) of the Department of Architecture.

All work for the dissertation was completed independently by the student.

There are no outside funding contributions to acknowledge related to the research and compilation of this document.

NOMENCLATURE

PARS	Pose-Based Action Recognition System
PTARS	Pose-Tool Action Recognition System
TPARS	Tool-Pose Action Recognition System
SVM	Support Vector Machine
k-NN	k-Nearest Neighbors
HOG	Histogram of Oriented Gradients
TP	Ture Positive
FP	False Positive
FN	False Negative
P	Precision
R	Recall

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

It is a daily job for every construction management team to monitor personnel, equipment, and materials on the construction jobsite. Various technologies have been adopted for the monitoring process to improve the efficiency of human observation and to obtain data with minimal manual collection. Data can be collected in various formats, such as count, location, and trajectory of construction personnel, certain equipment, and/or construction materials. Such data can be used to analyze current practices on the jobsite and to come up with suggestions for future improvement for many diverse purposes, such as project progress monitoring (Golparvar-Fard, Peña-Mora, Arboleda, & Lee, 2009; Maalek, Lichti, & Ruwanpura, 2015), activity sequence analysis (Liu & Golparvar-Fard, 2015), productivity measurement (Teizer, Cheng, & Fang, 2013; Yi & Chan, 2013), resource location (Maalek & Sadeghpour, 2013; Teizer, 2015), and safety management (Park, Kim, & Cho, 2016; Seo, Han, Lee, & Kim, 2015).

At the current stage, radio frequency identification (RFID), Global Positioning System (GPS), and 3D range imaging camera (LADAR) are some of the most commonly used automatic or semi-automatic jobsite monitoring techniques.

RFID has been adopted in the construction industry to identify and track a large variety of project-related objects. For example, it has been used to track and locate highly customized precast concrete components to avoid late deliveries, double-handling, and incorrect installation (Ergen, Akinci, & Sacks, 2007). Costin, Teizer, and

Schoner (2015) integrated RFID and Building Information Modeling to track construction crews in real time for safety, security, and verification of maintaining local hiring mandates.

GPS has been applied on construction sites for real-time materials management (Caldas, Torrent, & Haas, 2004; Ergen et al., 2007) and construction equipment tracking (Lu, Chen, Shen, Lam, & Liu, 2007; Pradhananga & Teizer, 2013). The availability of this technique depends on the triangulation of groups of satellites, ground control stations, and signal receivers. To monitor the concrete production and delivery processes on and off construction sites, Lu et al. (2007) proposed a continuous real-time tracking system for construction vehicles by integrating GPS with vehicle navigation technology. Their system was tested in the urban environment of Hong Kong; the system proved accurate and reliable for recording the key event times of a mixer truck under practical site conditions.

The LADAR system has also been used on construction sites to enhance on-site safety. Teizer, Caldas, and Haas (2007) proposed a methodology to model, detect, and track the position of static and moving objects such as construction crews and heavy equipment in real time, based on data obtained from video range cameras. Experimental results demonstrated that position, dimension, direction, and speed measurements acquired by the LADAR system have an accuracy level compatible with the requirements of active safety monitoring for construction.

The above studies have shown the possibility and necessity of applying different techniques to improve the construction management process from a large variety of perspectives. Replacing traditional manual processes conducted by project management personnel with automatic or semi-automatic systems could improve the efficiency of documentation, data analysis, and decision-making processes. However, on large-scale sites, deploying, maintaining, and removing such systems can be costly and time-consuming. In addition, privacy issues with attached personnel tracking devices often limit the usability of these technologies on construction sites (Brilakis, Park, & Jog, 2011).

With sophisticated algorithms and powerful computing hardware made available in recent years, vision-based techniques have arrived as an alternative to existing systems. Vision-based systems are built on top of computer vision and machine learning technologies. By combining such technologies, vision-based systems have the following advantages:

- They are highly applicable for dynamic, busy construction sites involving large numbers of equipment, personnel, and materials (Memarzadeh, Golparvar-Fard, & Niebles, 2013).

- They are more desirable for personnel who wish to avoid being "tagged" with sensors (Brilakis et al., 2011).

- The continuous advent of vision-based tracking algorithms could effectively reduce the effects of illumination conditions and occlusions (Brilakis et al., 2011).

- It is profitable to apply vision-based systems in construction operations due to simplicity, commercial availability, and low costs associated with video equipment (Teizer, Caldas, & Haas, 2007).

With further development of such technologies, a vision-based system can contribute to a fully automated construction monitoring system. The monitoring system can integrate a large variety of capabilities, including but not limited to sounding an alarm when unsafe behavior occurs; automatically generating an optimized site layout for the working area, rest area, and material lay-down area; recommending an activity sequence; recognizing discrepancies between the current and as-planned construction schedule; and providing suggestions to update the management plan.

At the current stage of research, the purposes of applying vision-based systems can be summarized as follows:

- For safety purposes: to automatically detect unsafe behavior and unsafe site conditions in real time (Du, Shehata, & Badawy, 2011; Seo et al., 2015);

- For construction progress control: to continuously compare as-built and as-planned construction progress (Golparvar-Fard & Pena-Mora, 2007) and to provide construction professionals with immediate and accurate information regarding specific

project issues for their project control decision-making process (Yang, Park, Vela, & Golparvar-Fard, 2015);

- For construction personnel and equipment productivity control: to collect information, such as count, location, and moving trajectory of construction personnel (Brilakis et al., 2011; Park & Brilakis, 2012; Gong, Caldas, & Gordon, 2011; Memarzadeh et al., 2013) and certain equipment (Azar & McCabe, 2011 & 2012; Chi & Caldas, 2011). The information collected could be utilized to calculate and measure productivity and performance and to improve travel path conflicts.

Construction personnel recognition and tracking is one of the major research areas in applying vision-based systems for construction monitoring. However, most research efforts recognize construction personnel as objects, which creates hurdles for revealing the full benefits of vision-based systems. It is essential for the systems to understand construction workers' activities because more accurate data collection results in further analysis, such as workers' productivity measurement and safety evaluation. However, it is still a challenge to accurately recognize construction workers' activities from still images or videos (Gong et al., 2011).

2. MOTIVATION

As previous studies have demonstrated, vision-based technologies have promising potential for monitoring construction personnel. However, it is still a challenge to recognize construction workers' activities from the images or videos collected from a jobsite. Representation of activity is important because data can be provided for management personnel and computer systems to perform further analysis more accurately, such as workers' productivity measurement and safety evaluation. Even though Gong et al. (2011) proposed a method of recognizing workers' on-site activities from videos, their approach has a relatively high false recognition rate at 21%, and the recognition accuracy could be improved.

Until now, all previous studies regarding construction worker activity recognition have only applied the overall features of a construction worker's body area, only capturing the worker's body pose feature. This recognition approach is referred as a single-layered approach (Aggarwal & Ryoo, 2011). More details on construction worker action recognition can be found in Chapter 6.1.2, Construction Worker Recognition. Single-layered approaches (Sheikh, Sheikh, & Shah, 2005; Yilmaz & Shah, 2005) are based on sequenced images as input data. As the nature of a single-layered approach, it is suitable for recognizing gestures and actions, where gestures refer to simple movements of a person's body part, such as jumping or waving a hand, and actions refer to compositions of multiple gestures.

However, actions appearing on construction sites are more dynamic and complex than simple gestures and actions such as running and jumping, and construction-related actions usually include human interaction or operation of certain objects or tools.

3. SUGGESTIONS

In the community of action recognition, the concept of semantic understanding has been applied in the recognition process to further improve the performance of action recognition. Semantic understanding enables users to apply prior knowledge of certain activities to the recognition processes. Semantics interprets an action as a relation among its features. The features include all or at least two elements of pose/poselet (Ikizler & Duygulu, 2007; Ukita, 2013), related objects (Gupta & Davis, 2007; Yao & Fei-Fei, 2010), scene (Marszalek & Laptev, 2009; Zhang, Qu, & Wang, 2014), and object attributes (Bourdev, Maji, & Malik, 2011; Farhadi, Endres, Hoiem, & Forsyth, 2009). The meaning of each action generally can be decomposed into the meanings of its features (Ziaeefard & Bergevin, 2015).

If a human is looking at an image, the first data piece to arrive is the pose of the person in general, as shown in Figure 1(a). However, we cannot conclude what that person is doing by just looking at the pose. If we apply the concept of semantic understanding to recognize construction crew activity, we could look around trying to find clues other than body pose. In Figure 1(b), we can see that this person is holding a hammer in his hand. We then determine that the action being performed by this person is nailing.

(a) (b)

Figure 1 Understanding a construction activity. (a) Only looking at the pose, (b) looking at both the pose and the tool.

Can we apply our prior knowledge of the relationship between a construction worker's body pose and certain tools to the process of worker action recognition? Compared with existing construction worker action recognition systems based only on features of the workers, does the new action recognition system integrating the human-tool relationship perform better than existing systems?

So far, all previous applications were developed to look only at the human features, never taking a closer look at any other related objects. This study proposes to add recognition processes to look not only at human body pose features, but also to recognize the co-related tools for each specific action. The proposed approach has two recognition tasks, pose recognition and tool recognition. By applying reversed recognition sequences of pose and tool, the new approach was further defined and developed into two frameworks, Pose-Tool Action Recognition System (PTARS) and Tool-Pose Action Recognition System (TPARS). Both frameworks have four major

steps; the only difference between PTARS and TPARS is the detection sequence of pose and tool—PTARS detects pose first, while TPARS detects tool first. Figure 2 illustrates the workflow of PTARS. Taking PTARS as an example, the four major steps are as follows:

1) Applies a sliding window to go through the entire image and extract histogram of oriented gradients (HOG) for each image patch that the sliding window went through. Figure 3 gives an example of the image patches generated by applying a 300- × 300-pixel sliding window that moves every 150 pixels horizontally and vertically over a 1920- ×-1080 pixel image;

2) Classifies the HOG features of each image patch with a pretrained pose classifier to predict if it is the pose of an activity or if it is not a body pose;

3) Defines a potential tool area based on the location and size of the image patch and the predefined interactive pose-tool relationship;

4) Applies another sliding window within the potential tool area to detect the associated tool. If and only if both the pose and tool are detected, the framework predicts the area in the image as the activity and highlights that area.

Implementation details of the frameworks are covered in Chapter 7, Methodology.

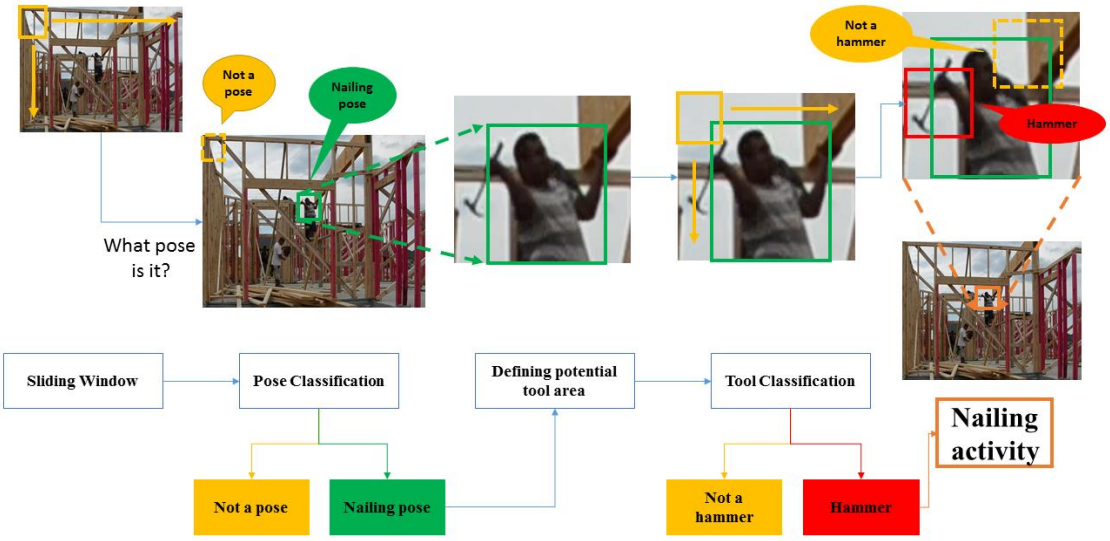


Figure 2 Workflow of PTARS.

(a) (b)

Figure 3 Illustration of sliding window. (a) Original image, (b) image patches.

4. RESEARCH QUESTION

Do the proposed frameworks, PTARS and/or TPARS, perform better[a] than the construction personnel action recognition system based only on human pose?

[a] To define a "better result," the evaluation metric system is introduced in Chapter 6.3, Evaluation of Algorithm Performance.

5. RESEARCH OBJECTIVE

The researcher could not find any previous work using human-tool spatial relationship as part of an action recognition system in the construction community. The objective of this study is to find out whether combining human pose recognition and tool recognition provides a better result for construction worker action recognition than the action recognition system based only on human pose.

6. LITERATURE REVIEW

The literature review is organized in the following sequence:

Chapter 6.1 reviews previous studies of construction personnel recognition and construction personnel activity recognition.

Chapter 6.2 reviews, explains, and illustrates some of the major feature extraction and machine learning algorithms that have been applied in previous studies. Part of this chapter follows Dr. Andrew Ng's Machine Learning class on Coursera.

Chapter 6.3, Evaluation of Algorithm Performance, introduces object recognition performance evaluation methods.

6.1 Application of Object Recognition in Construction

6.1.1 Why Apply Object Recognition in the Construction Industry?

Research on object recognition techniques has become more popular in recent years because of the demand for automatic real-time site monitoring for safety management, work progress control, and productivity control.

Formerly, researchers and professionals in the construction industry relied on other techniques such as RFID (Jaselskis & El-Misalami, 2003; Teizer, Lao, & Sofer, 2007), GPS (Caldas, Torrent, & Hass 2004; Ergen et al., 2007; Lu et al., 2007), and 3D range imaging camera, also known as flash LADAR system (Teizer et al., 2007).

However, the existing approaches depend heavily on manual operations to achieve high accuracy and precise interpretation and usually require extra equipment or tools, which are costly and time-consuming for most construction projects (Teizer & Vela, 2009; Teizer et al., 2007).

Automated vision-based detection and tracking came as an alternative technique to previous techniques, and it requires very low human involvement and intervention and has minimum requirements for data acquisition and data analysis equipment (Chi & Caldas, 2011; Park & Brilakis, 2012; Teizer & Vela, 2009).

Construction worker recognition is one of the major focus areas of previous research.

6.1.2 Construction Worker Recognition

Chi and Caldas (2011) combined moving object detection, object correspondence, and object classification techniques to detect construction workers.

Moving object detection is a technique where the program continuously compares a series of video frames, subtracts the static "background," and leaves the "foreground," which is the moving object for further usage, as shown in Figure 4.

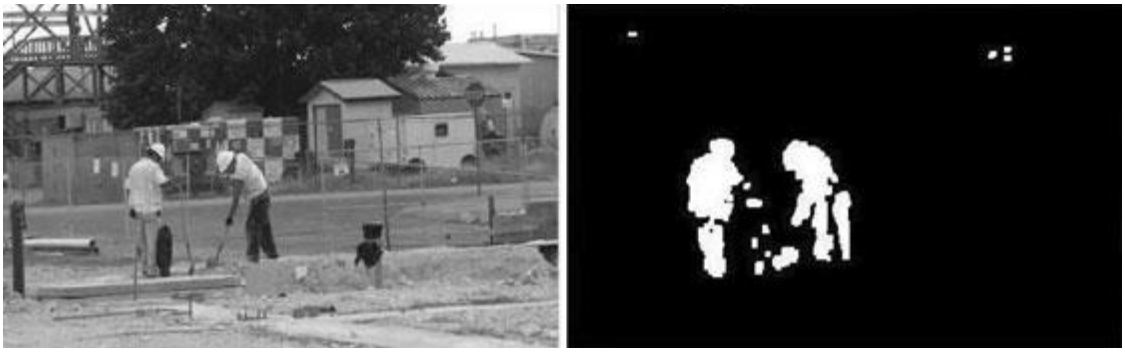


Figure 4 Illustration of a background subtraction result. (Left) Original image, (right) result of background subtraction/moving object detection (reprinted from Chi & Caldas, 2011).

Object correspondence was adopted to match the detected foreground objects between frames. Finally, they adopted and compared two different classifiers, normal Bayes classifier and neural network classifier, to classify and highlight the object in the image, as shown in Figure 5.

Figure 5 Illustration of a worker detection result. A worker was detected and circled in an image captured by a standard video camera on a construction jobsite (reprinted from Chi & Caldas, 2011).

Per Park and Brilakis (2012), vision tracking is an efficient tool to monitor a large outdoor site because it does not require targets to be tagged as with techniques such

as RFID or GPS. To initiate the vision tracking process, object recognition is a prerequisite to locate and mark the target of tracking. However, the lack of object recognition methods is a major obstacle to applying the tracking process. Park and Brilakis (2012) employed background subtraction, HOG, and hue-saturation-value (HSV) color histogram to minimize detection region step by step, from moving objects to people and eventually to construction workers, as shown in Figure 6. The performance of their approach was evaluated with precision and recall. They achieved precision at 99% and recall at 81%. Definitions of precision and recall values are introduced in Chapter 2.3.

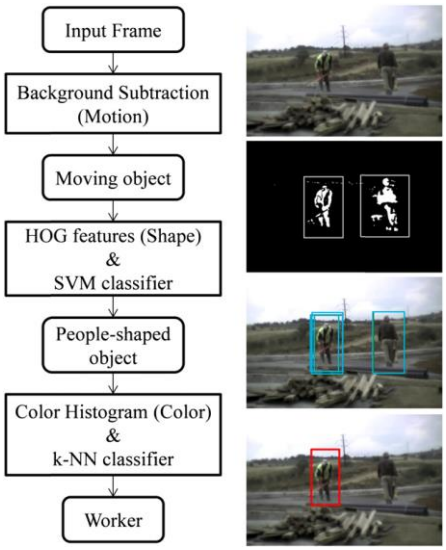
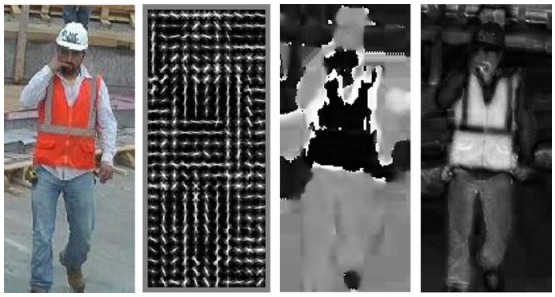


Figure 6 Workflow of a construction worker detection method (reprinted from Park & Brilakis, 2012).

To deal with large variations in image illumination, weather condition, resolution, and scale of target objects, Memarzadeh et al. (2013) introduced multiscale sliding detection windows to solve the problem of changing scales of objects in images.

They also combined HOG and histogram of color (HOC) to handle continuously changing image quality. Figure 7 shows an example of HOG and HOC features for a human. They were also able to recognize multiple different objects from one image by running several classifiers together, as shown in Figure 8. They received an average accuracy of 98.83%. Average accuracy is defined in Chapter 6.3, Evaluation of Algorithm Performance.



(a) (b) (c) (d)

Figure 7 Illustration of applying HOG and HSV features to an image of a construction worker. (a) Test image; (b) oriented gradients; (c) hue map; and (d) saturation map (reprinted from Memarzadeh et al., 2013).

Figure 8 Detection of multiple excavators and construction workers (reprinted from Memarzadeh et al., 2013).

Even though progress has been made in the last few years for construction worker recognition and tracking, it has always been a challenge to classify actions of construction workers and equipment. To improve the action recognition process, Gong et al. (2011) integrated the Bag-of-Video-Feature-Words model with Bayesian learning methods for construction worker action analysis. Their approach is composed of four steps: 1) applies Harris 3D detector (Sipiran & Bustos, 2011) to locate interest corner points that have significantly changed between consecutive video frames; 2) applies and compares HOG and histogram of optical flow (HOF) (Horn & Schunck, 1981) to describe the surrounding regions of each interest point; 3) applies the Bag-of-Words technique (Niebles, Wang, & Fei-Fei, 2008) to form the features of each image category; and 4) applies Bayesian network models (Fergus, Perona, & Zisserman, 2003) to train a construction crew activities classifier. The method is illustrated in Figure 9; it achieved an average accuracy of 73.6%.

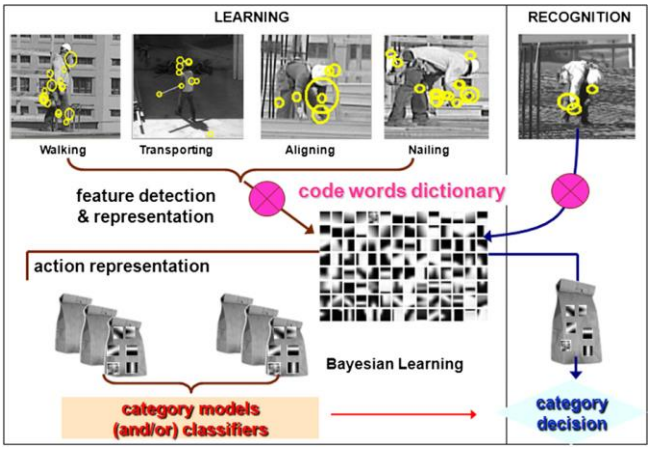


Figure 9 Illustration of Bag-of-Video-Feature-Words model (reprinted from Gong et al., 2011).

6.2 Algorithms and Terms

6.2.1 Feature Descriptor

Histogram of Oriented Gradients

Dalal and Triggs (2005) introduced HOG as a feature descriptor for human detection, and their experiment results showed that HOG outperforms other feature descriptors for human shape figures. It takes the following major steps to generate HOG features of an image, as shown in Figure 10.

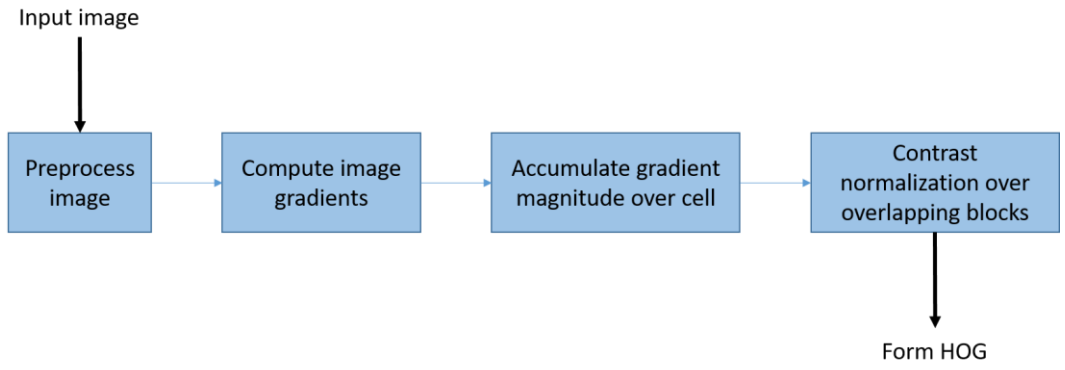


Figure 10 Workflow for generating HOG features.

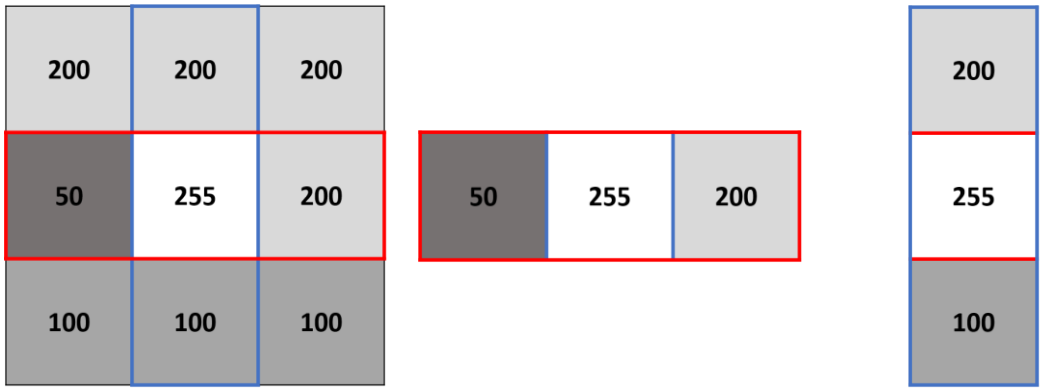
The first step of generating an HOG feature is preprocessing the image. HOG can be formed with different pixel representations, such as grayscale and red-green-blue (RGB), with modest effect on the performance (Dalal & Triggs, 2005). Because grayscale images use one value to represent each pixel and RGB images use three values to represent each pixel, applying HOG with grayscale images is more computationally reasonable for later steps.

It is common to use 64- × 128-pixel images for human shape figure recognition with the HOG feature. Previous studies have not suggested whether other image sizes would perform better or worse. Typically, training images and detection windows could be set at any size, but the width:height ratio has always been set at 1:2, and the images are resized to 64 × 128 during the training and testing phase.

Once the image is preprocessed, the next step is to compute image gradient. An image gradient can be described with a direction, gradient orientation (θ), and its strength can be described by the change, gradient magnitude (g). Assuming Figure 11(a) is a 3- × 3-pixel area within an image and the pixel in the center is at the cth column, rth row of the image, the intensity of this pixel could be represented as I (c, r), the intensity of the right pixel as I (c + 1, r), the intensity of the left pixel as I (c – 1, r), the intensity of the top pixel as I (c, r – 1), and the intensity of the bottom pixel as I (c, r + 1). To compute the gradient orientation and magnitude of this center pixel, horizontal gradients and vertical gradients, as shown in Figure 11(b) and (c), can be computed as follows:

Horizontal gradient $d_x = \text{I (c + 1, r)} - \text{I (c – 1, r)} = 200 - 50 = 150$ Eq. 1

Vertical gradient $d_y = \text{I (c, r – 1)} - \text{I (c, r + 1)} = 200 - 100 = 100$ Eq. 2



(a) (b) (c)

Figure 11 Example of an image gradient. (b) Horizontal gradient, (c) vertical gradient.

Then, the gradient of the center pixel can be represented as shown in Figure 12.

Thus, the gradient direction, θ, and gradient magnitude, g, can be computed as:

Gradient direction $\theta = \arctan(d_y/d_x) = \arctan(100/150) = 33.69$ Eq. 3

Gradient magnitude $g = \sqrt{d_y^2 + d_x^2} = \sqrt{100^2 + 150^2} = 180.28$ Eq. 4

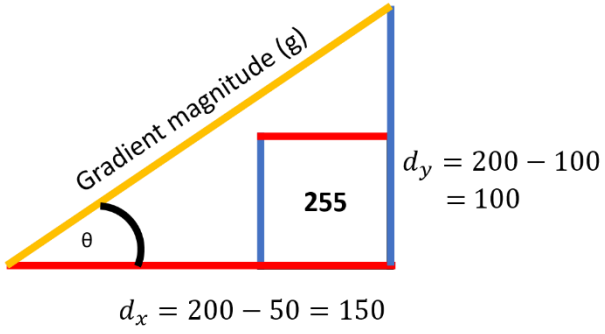
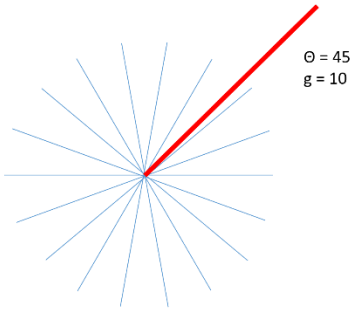


Figure 12 Gradient direction and gradient magnitude.

Instead of using the gradient of each pixel to represent the original image, “cell” was introduced to HOG as the unit for summarizing features to describe a patch of an image. Using the idea of a cell not only gives a more compact feature representation, but also makes the representation more robust in the face of noise.

However, as shown in the example, a gradient can point in any direction. Practically, researchers need more control over the data, so gradient orientations are usually predefined. Unsigned gradients are usually applied here, meaning that the angles are between 0 and 180 instead of between 0 and 360, and the same numbers are used to represent a gradient orientation and its opposite direction. Increasing the number of orientation bins to nine significantly improves performance, but makes less difference after that (Dalal & Triggs, 2005). For example, setting the number of orientations at nine gives one orientation bin per 20°, and the nine orientation bins are 0°, 20°, 40°, 60°, 80°, 100°, 120°, 140°, and 160°. Then, all the original gradients within each cell contribute to the nine orientations. For example, if an original gradient has a gradient orientation equal to 45° and a magnitude equaling 10, the original gradient could be proportionally contributed to its nearby directions, the 40° bin, and the 60° bin, as shown in Figure 13. Thus, for each cell, a 1-row × 9-column vector could be formed to summarize the gradients within the cell.



0	20	40	60	80	100	120	140	160
		7.5	2.5					

Figure 13 Contributing a gradient to selected directions.

Then, all the histograms for each cell are combined to form the descriptor of the image. In order to achieve a better result, the histogram for each cell is contrast-normalized (Papageorgiou et al., 1998). Contrast normalization can be done by applying an intensity measure across the entire block, which is a region larger than a cell, and then using the results to normalize every cell in that block. Figure 14 gives an example of the HOG feature, the red star-like shape, overlapping an image of a human figure. Implementation details of HOG are given in Chapter 7.5.2, Implementing Features of Histogram of Oriented Gradients.

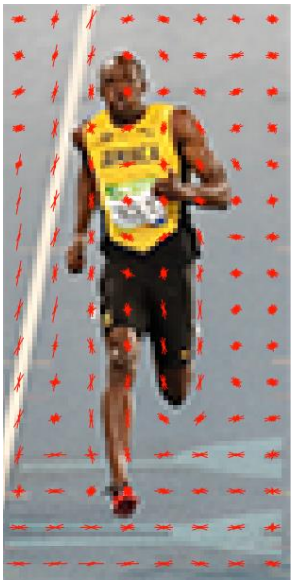


Figure 14 HOG example of a human (reprinted from Mallick, 2016).

Bag-of-Words Model

The Bag-of-Words model, also referred as Bag-of-Features, is another type of commonly used feature descriptor for recognizing/classifying object categories. A typical Bag-of-Features classifier usually contains the following steps: interest point detection, interest point description, k-means clustering, and classification model building.

The first step to applying the Bag-of-Features model is to detect interest points and descriptors around each interest point. There is no requirement or verification regarding which algorithms perform better than others. The selection of algorithms for interest point detection and descriptor varies from one case to another. There are several algorithms that have been commonly used for interest point detection. They are scale-

invariant feature transform (SIFT), speeded-up robust feature (SURF), Harris, and dense (taking every nth pixel as interest point) methods. Once interest points have been selected, SIFT, HOG, or SURF is introduced as the feature descriptor to summarize the feature around each interest point.

The second step is clustering patches of features together and calculating the occurrence frequencies of each cluster. k-means is the most common approach to clustering features. “k” refers to the total number of clusters, and “means” refers to the mean distance value of the cluster centroid to each other point in that cluster. Each cluster centroid is referred as a visual word.

The k-means clustering algorithm is an unsupervised learning process, where each descriptor is considered a training example $x^{(i)}$ without any label, and X_descriptors is a matrix summarizing all the training examples. Within the clustering algorithm, k cluster centroids, shown as the black crosses in Figure 15, are randomly selected from the training set, X_descriptors, and named $\mu_1, \mu_2, \ldots, \mu_k$.

The algorithm iteratively groups the descriptors, $x^{(1)}, x^{(2)}, \ldots, x^{(n)}$, into k mutually exclusive clusters. The clustering optimization process has two steps. First, it indexes through each descriptor and assigns an extra index value $c^{(i)}$ to each descriptor, where $c^{(i)}$ equals the index of the cluster centroid that is closest to $x^{(i)}$. Second, it updates the location of all the centroids 1 to k, from the original location to the mean location of points assigned to each cluster. The algorithm iteratively repeats the two steps until it finds a partition where the objects within each cluster are as close to each

other as possible and as far from objects in other clusters as possible. The optimization problem can be summarized with Eq. 5. The clustering process can be illustrated as shown in Figure 15.

$$J\left(c^{(1)}, c^{(2)}, \ldots, c^{(m)}, \mu_1, \mu_2, \ldots, \mu_k\right) = \frac{1}{m}\sum_{i=1}^{m} ||x^{(i)} - \mu_{c(i)}||^2 \quad \text{Eq. 5}$$

where

$c^{(i)}$ = index of cluster (1, 2, …, k) to which example $x^{(i)}$ is currently assigned;

μ_k = cluster centroid k;

$\mu_{c(i)}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned;

$||x^{(i)} - \mu_{c(i)}||$ = distance between descriptor $x^{(i)}$ and the centroid $\mu_{c(i)}$ that it has been currently assigned to.

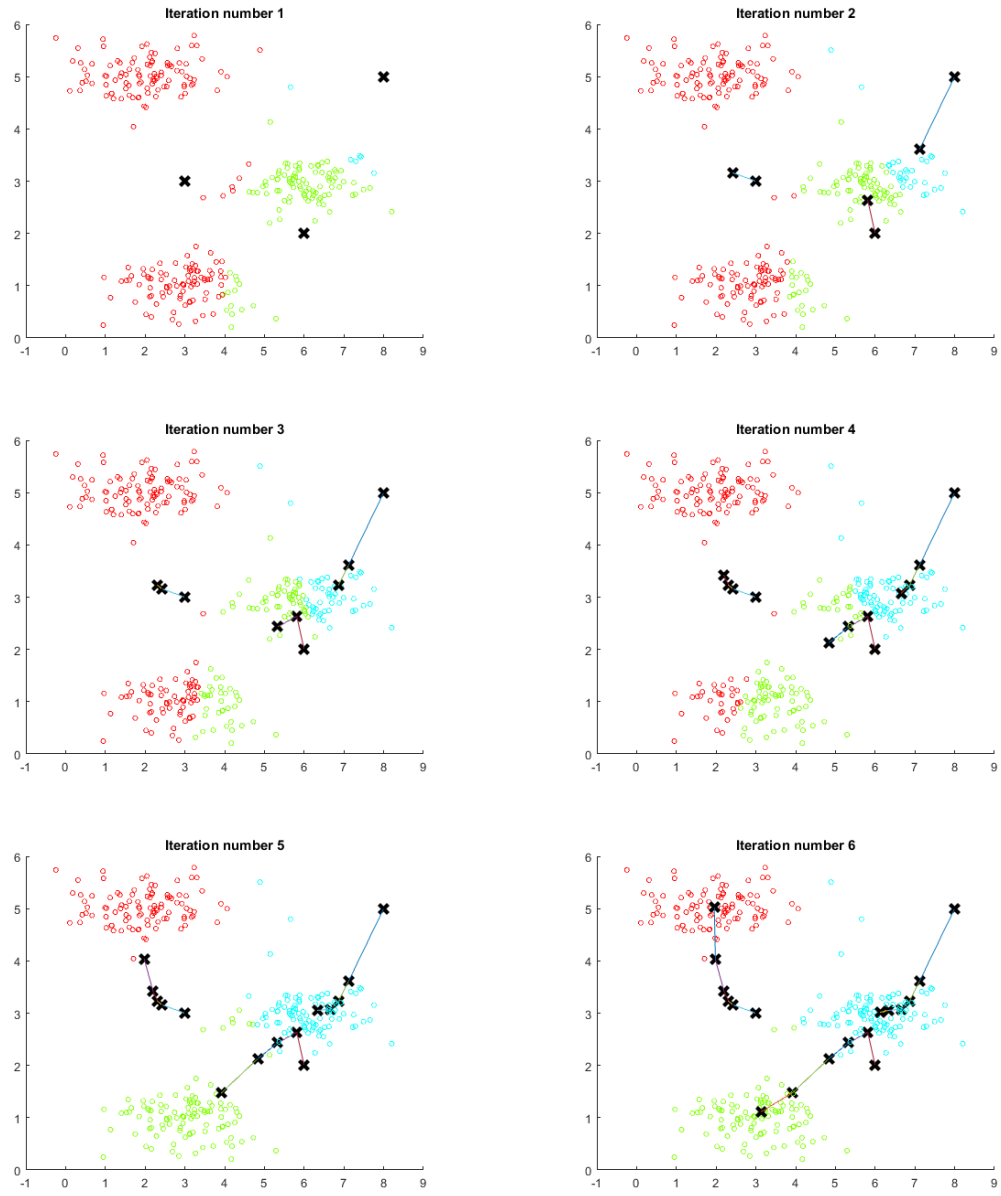


Figure 15 Illustration of clustering process.

The process of generating visual words and forming a feature histogram is illustrated in Figure 16. The Y-axis in the Histogram column represents the occurrence frequency of each visual word in each image.

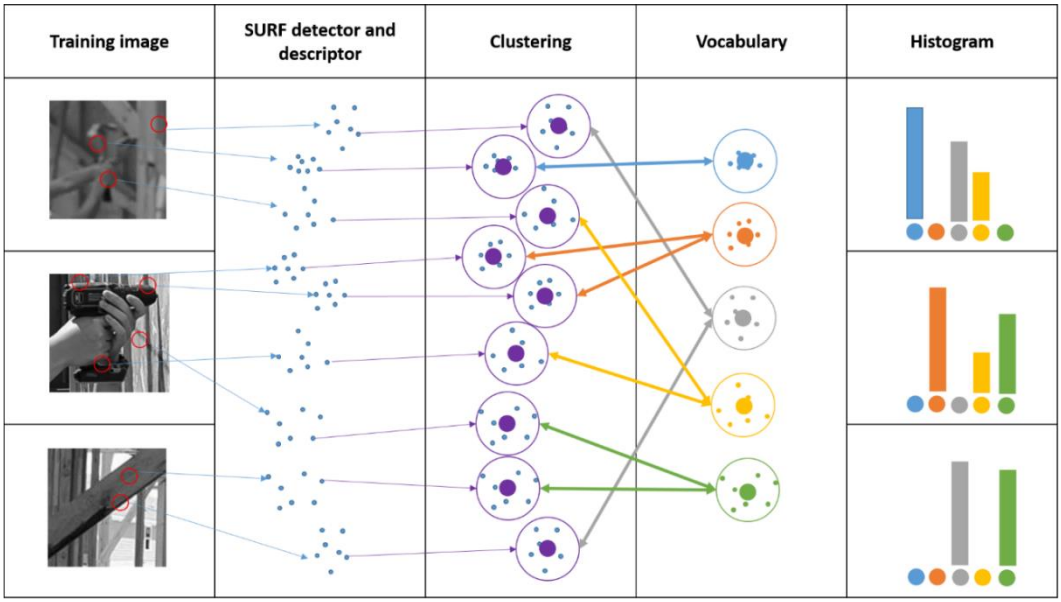


Figure 16 Illustration of the process of generating visual words and forming a feature histogram.

6.2.2 Classifier

The terms artificial intelligence (AI), machine learning, and deep learning were prevalent in the media when Google's AlphaGo defeated the South Korean board game master Lee Se-dol. Thinking of the relationship among the three techniques, AI is the broadest concept, machine learning comes second, and deep learning brings AI to the next level (Copeland, 2016). The relationship is illustrated in Figure 17.

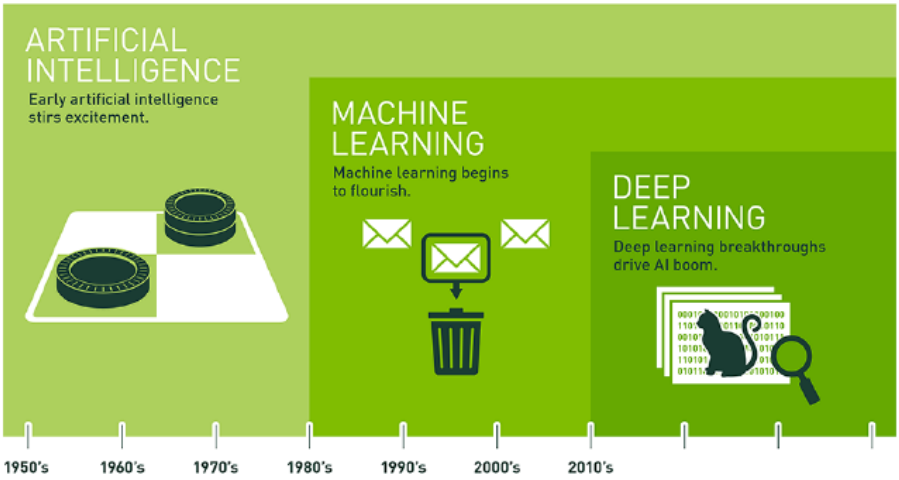


Figure 17 Relationship between AI, machine learning, and deep learning (reprinted from Copeland, 2016).

The result of the techniques is a classification model or classifier. A classifier is made with a training image dataset, where each training image may or may not be labeled, and with sets of machine learning algorithms. The learning process can be categorized as supervised learning or unsupervised learning (Joshi, Cherian, & Shivalingam, 2016).

Supervised learning is being given a training image dataset of two or more categories, with each image in the training dataset labeled with the name of its category. By the end of the training process, a classifier is built that can classify the category of a new query image. In the real world, supervised learning has a wide range of applications, such as handwriting recognition, speech recognition, and computer vision.

Unsupervised learning is being given a set of training images with no categorical label assigned to any image. By the end of the process, the classifier can differentiate

images without knowing what is in the image. Unsupervised learning has also been widely applied in life, such as for finding different market segmentations and for finding a malfunctioning computer from large computing clusters.

For example, we give different systems the same set of images containing either human faces or rabbits. In supervised learning, each human face and each rabbit in the training dataset needs to be labeled with its associated category. By the end of the process, the classifier can distinguish between a human face and a rabbit in the query image. With unsupervised learning, the classifier only learns that there are two different categories within the training dataset, but it is not able to tell what each is.

For this study, it is more appropriate to apply supervised learning because according to the objective of this study, we would like to know what the action is in the query images. The following sections introduce some of the most widely used supervised learning methods for training a classifier.

Support Vector Machine

In the support vector machine (SVM) method, a hyperplane is introduced to separate one class from another. The best hyperplane is the one that gives the largest margin between the two classes. Margin means the maximal width of the slab parallel to the hyperplane that has no interior data points. The data points closest to the hyperplane and sitting on the slabs are referred as support vectors (Figure 18) (MathWorks, 2016).

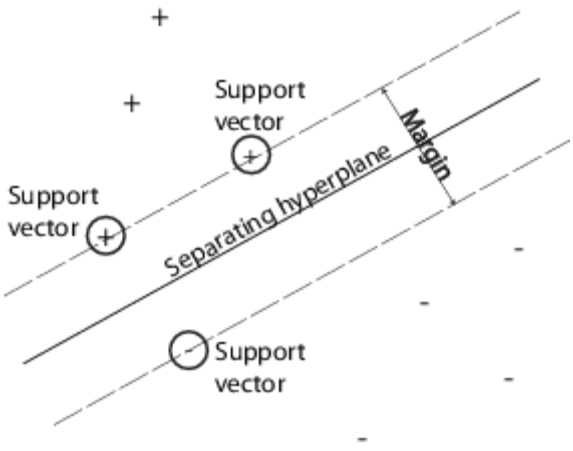


Figure 18 Illustration of SVM (reprinted from MathWorks, 2016).

Mathematically, if we were giving a binary classification problem with a training set of m examples and n features for each example, which is an m × n feature matrix as shown in Figure 19(a), and a corresponding categorical vector of y, which is an m × 1 vector of 1s and 0s as shown in Figure 19(b), the objective of SVM is to train a hypothesis, $h_\theta(x)$, that minimizes the cost function, $J(\theta)$. The hypothesis predicts the category of the query data to be either 1 or 0, and it can be described by Eq. 6.

$$h_\theta(x) = \begin{cases} 1 \ \ if \ \theta^T x \ \geq 0 \\ 0 \ \ otherwise \end{cases} \qquad \text{Eq. 6}$$

where *x* is a vector of the features of an example and

Θ is also a vector containing the coefficient of each element in *x*.

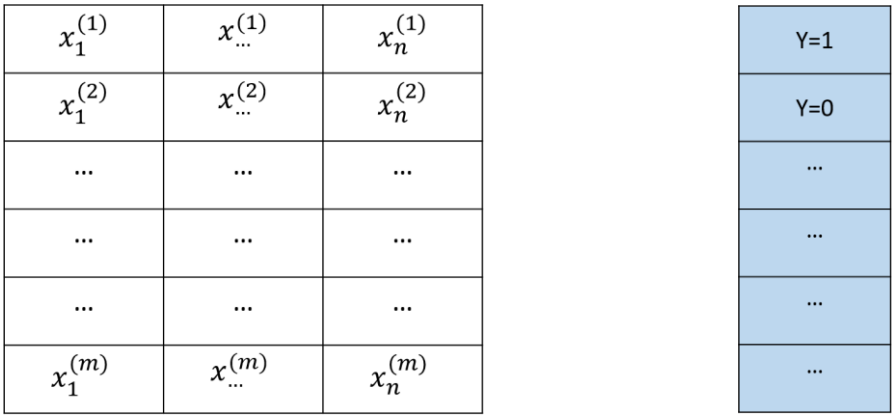


Figure 19 Illustration of training dataset.

The optimized values of the elements in Θ are obtained by minimizing the cost function of SVM, $J(\theta)$, illustrated in Figure 20, and iteratively updating Θ (Eq. 7 and Eq. 8).

$$J(\theta) = C\sum_{i=1}^{m}(y^{(i)}cost_1(z) + (1-y^{(i)})cost_0(z) + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2) \quad \text{Eq. 7}$$

$$\text{Repeat}\left\{\begin{array}{c}\theta_0 \coloneqq \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^i \\ \theta_j \coloneqq \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^i + \frac{\lambda}{m}\theta_j\right]\end{array}\right\} \quad \text{Eq. 8}$$

where

z = $\theta^T x^{(i)}$;

$cost_0(z) = \max(0, k(1+\theta^T x^{(i)}))$;

$cost_1(z) = \max(0, k(1-\theta^T x^{(i)}))$;

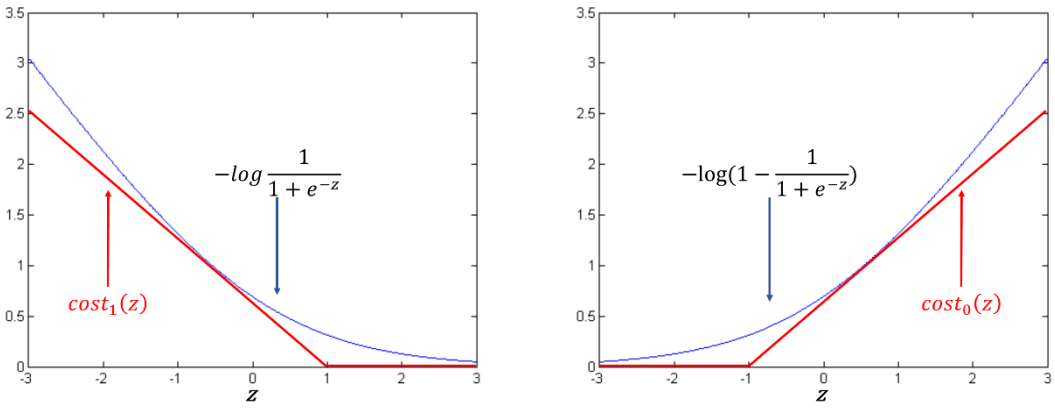
k is an arbitrary constant defining the magnitude of the slope of the line, as shown in Figure 20(a)(b);

$\frac{1}{2}\sum_{j=1}^{n}\theta_j^2$ is the regularization term for cost function to prevent overfitting problem;

α is the learning rate of gradient descent algorithm;

λ is the regularization parameter;

$\frac{\lambda}{m}\theta_j$ is the regularization term for gradient descent to prevent overfitting problem.



(a) (b)

Figure 20 Illustration of SVM cost functions. (a) If $z \geq 1$, the penalty of predicting the category of query data to 1 is 0; (b) if $z \leq -1$, the penalty of predicting the category of query data to 0 is 0.

More complex, non-linear classifiers can be built with the Gaussian kernel, which is the most commonly used kernel function. It measures the similarity between each example, x, and some landmarks, $l^{(i)}$, shown in Eq. 9.

$$f_i = similarity(x, l^i) = \exp(-\frac{\sum_{j=1}^{n}(x_j - l_j^{(i)})^2}{2\sigma^2}) \quad \text{Eq. 9}$$

If $x \approx l^{(i)}$, $f_i \approx 1$.

If $x\ is\ far\ away\ from\ l^{(i)}$, $f_i \approx 0$.

In practice, the landmarks are selected as the exact locations as all the m training examples. Given an example $x^{(i)}$ and applying the similarity function to measure the similarity between $x^{(i)}$ and each of the landmarks, we can build a new feature vector, shown as Eq. 10, to substitute the original $x^{(i)}$.

$$x^{(i)} \rightarrow \begin{bmatrix} f_1^{(i)} = similarity\left(x^{(i)}, l^{(1)}\right) \\ f_2^{(i)} = similarity\left(x^{(i)}, l^{(2)}\right) \\ \vdots \\ f_m^{(i)} = similarity\left(x^{(i)}, l^{(m)}\right) \end{bmatrix} \quad \text{Eq. 10}$$

Thus, the original optimization problem can be rewritten as Eq. 11. By resolving this optimization problem, a kernel function can be built to solve a more complex, non-linear classification problem. More illustrations and implementation details regarding SVM can be found in Chapter 7.5.3, Implementing Support Vector Machine.

$$J(\theta) = C\sum_{i=1}^{m}\left(y^{(i)} cost_1\left(\theta^T f^{(i)}\right) + (1 - y^{(i)}) cost_0\left(\theta^T f^{(i)}\right) + \frac{1}{2}\sum_{j=1}^{n}\theta_j^2\right) \quad \text{Eq. 11}$$

k-Nearest Neighbor

k-nearest neighbor (k-NN) has been widely applied as a benchmark for machine learning rules. With its simplicity, k-NN is easy to use and makes it easy to compare the results against other classification methods. The idea of k-NN is to measure the distance between the query data and the data in the training dataset. The distance can be measured with Euclidean distance:

$$d = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} = (q_i - p_i) \text{ x } (q_i - p_i)' \quad \text{Eq. 12}$$

where

d is the Euclidean distance;

q_i is a 1 × n feature vector of a training example;

p_i is a 1 × n feature vector of a query example;

$(q_i - p_i) \times (q_i - p_i)'$ is the vectorization implementation to calculate the distance.

Then, k closest points can be selected from the training dataset. The algorithm predicts the category of the query image as the category receiving the most votes out of the k closest training data. More illustrations and implementation details regarding k-NN can be found in Chapter 7.5.4, Implementing k-Nearest Neighbor**.**

6.3 Evaluation of Algorithm Performance

The most commonly used method to analyze the results of a classification system is called confusion matrix, as shown in Figure 21. The Y-axis represents the actual category of a testing image; the X-axis represents the predicted category of the image.

Assuming we have developed a classification system that supposedly can distinguish hammer images from hammer-free images and assuming we are given two images, as shown in Figure 22, to classify the category for each of them, we use “1” to represent the positive category, “hammer,” and “0” to represent the negative category, “not a hammer.”

The actual category of each image is given by human observation results. Thus, the actual category of Figure 22(a) is “hammer” or “1,” and the actual category of Figure 22(b) is “not a hammer” or “0.” The predicted category represents the system’s classification result. It is a true result if the predicted category of an image matches its actual category, and it is a false result if the predicted category of an image does not match its actual category.

With the concept of the positive or negative category and the true/false result, four values have been widely applied in previous studies to analyze the results of a classification system:

- True positive, as shown in the upper left corner in Figure 21: Figure 22(a) has been predicted as the positive category, 1. The predicted result matches its actual category, which is also 1;

- False positive, as shown in the lower left corner in Figure 21: Figure 22(b) has been predicted as the positive category, 1. However, the predicted result does not match its actual category, which is 0;

- False negative, as shown in the upper right corner in Figure 21: Figure 22(a) has been predicted as the negative category, 0. However, the predicted result does not match its actual category, which is 1;

- True negative, as shown in the lower right corner in Figure 21: Figure 22(b) has been predicted as the negative category, 0. The predicted result matches its actual category, which is also 0.

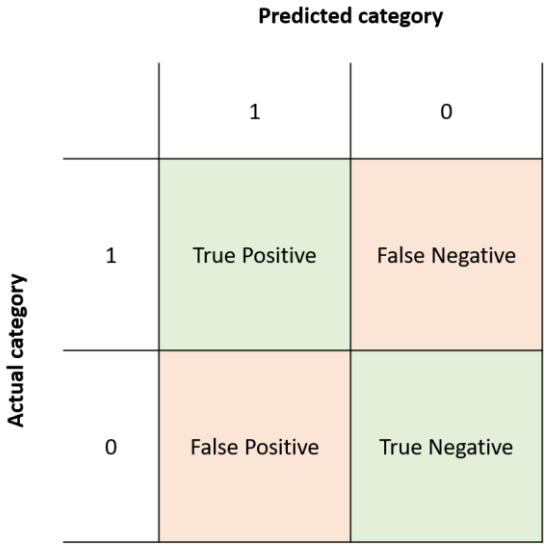
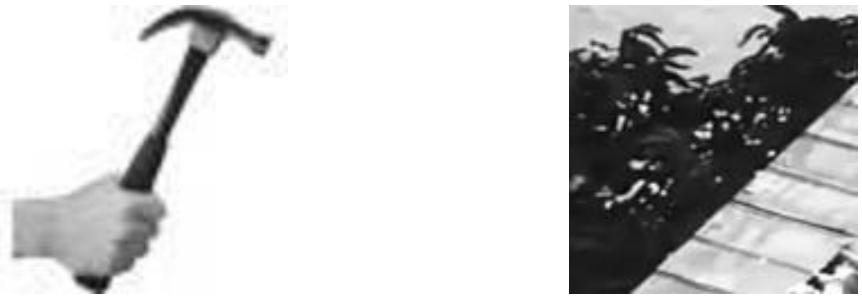


Figure 21 Confusion matrix.



(a) (b)

Figure 22 Example image for confusion matrix.

Once the four values, true positive, false positive, false negative, and true negative, have been collected by running the classification system with the testing image set, two other values can be computed to evaluate the performance of the system. The two values are precision and recall.

Precision measures the accuracy of the prediction result:

$$\text{Precision} = \text{True Positive} / (\text{True Positive} + \text{False Positive}) \quad \text{Eq. 13}$$

Recall measures the sensitivity of the recognition system to its target objects:

$$\text{Recall} = \text{True Positive} / (\text{True Positive} + \text{False Negative}) \quad \text{Eq. 14}$$

The values of both precision and recall range from 0 to 1, and the larger the values are, the better the system performs. However, there is a controllable trade-off between precision and recall. Usually, if we want to have more confidence on the prediction result, we can select the system with a higher precision value. But such a system usually has a lower recall value. Practically, the trade-off relationship between precision and recall can be illustrated as shown in Figure 23.

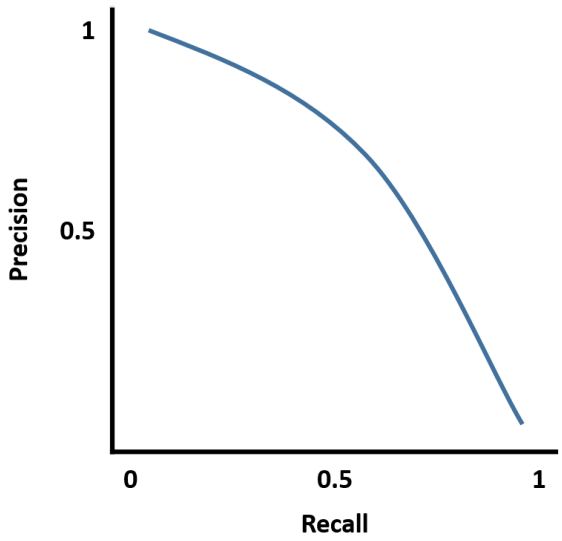


Figure 23 Illustration of precision and recall trade-off.

As there is a trade-off between precision and recall, sometimes when we are evaluating the performance of several systems, it is not a rare situation that a system has a higher precision and a lower recall. This leaves the problem of which system is the best. Thus, in the community of machine learning, the F_1 score, which is calculated from precision and recall, has been commonly used as a single real-number evaluation metric system. The F_1 score measures the overall performance of the recognition system:

$$F1\ Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \qquad \text{Eq. 15}$$

The F1 score is a fair way of evaluating a system because it gives the same weight to both precision and recall, and it is also ranges from 0 to 1. In order to have a large F1 score, both precision and recall need to be large. In contrast, for either precision or recall to be small, F1 reduces dramatically.

Other evaluation metric systems have been used in previous studies. For instance, Gong et al. (2011) applied accuracy, as shown in Eq. 16, and error, as shown in Eq. 17.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad \text{Eq. 16}$$

$$Error = 1 - Accuracy \quad \text{Eq. 17}$$

However, it is not appropriate to apply accuracy and error in this study, which is a typical case of skewed classes. Skewed classes means that a significant of the testing dataset is described in the negative category, and only a very small portion of the dataset is in the positive category. In this study, we applied a sliding window to recognize construction worker activities through every testing image or every frame of a video, producing a large test dataset with a very small positive category; the concept is shown in Figure 3. For example, assuming that the classification system has been given 500 testing images sized 1920 × 1080 pixels, among the 500 images, 50 have a nailing activity, 450 do not have nailing activity, the size of the sliding window is 200 × 200 pixels, and the sliding window moves every 10 pixels. Thus, for every image, a testing image set is generated of over 15,000 images, and over the 500 images, it generates over 7.5 million images, with only 50 of them being in the positive category and the rest being in the negative category.

In this case, even if the recognition system fails to recognize any nailing action, meaning the true positive equals 0, the accuracy is still extremely close to 100%, and the

error is still extremely close to 0, as calculated with Eq. 16 and Eq. 17. Thus, the accuracy and error evolution metric system cannot truly present the performance of the system, and it is not suitable for the setup of the proposed frameworks or for this study. Instead, this study has chosen to use precision, recall, and F1 score to measure the performance of the proposed frameworks.

7. METHODOLOGY

In response to the research objective mentioned in Chapter 5, to test whether combining pose recognition and tool recognition provides a better result of construction worker action recognition than an action recognition system based only on human pose features, this study compares the two proposed frameworks, PTARS and TPARS, against a benchmark framework, Pose-Based Action Recognition System (PARS). The benchmark framework, PARS, only applies body pose features for construction personnel activity recognition, while each of the two proposed frameworks is combined with pose recognition, tool recognition, and the interactive relationship between the body pose and the tool.

7.1 Hypotheses

The three frameworks were compared with the three values of precision, recall, and F_1 score because the three values have proved to be the most appropriate evaluation metric system for skewed-classes classification where the positive category only accounts for a very small portion of the total testing dataset. By summarizing the potential comparison results, the research hypotheses are as follows:

Hypothesis 1: PTARS has a higher precision value than PARS, meaning that the recognition result of PTARS is more accurate.

Hypothesis 2: TPARS has a higher precision value than PARS, meaning that the recognition result of TPARS is more accurate.

Hypothesis 3: PTARS has a higher recall value than PARS, meaning that PTARS is more sensitive than PARS.

Hypothesis 4: TPARS has a higher recall value than PARS, meaning that TPARS is more sensitive than PARS.

Hypothesis 5: PTARS has a higher F_1 score than PARS, meaning that PTARS performs better overall than PARS.

Hypothesis 6: TPARS has a higher F_1 score than PARS, meaning that TPARS performs better overall than PARS.

Null hypothesis: Neither of the proposed frameworks, PTARS or TPARS, has a better performance than PARS from any perspective.

7.2 Scope of the Test

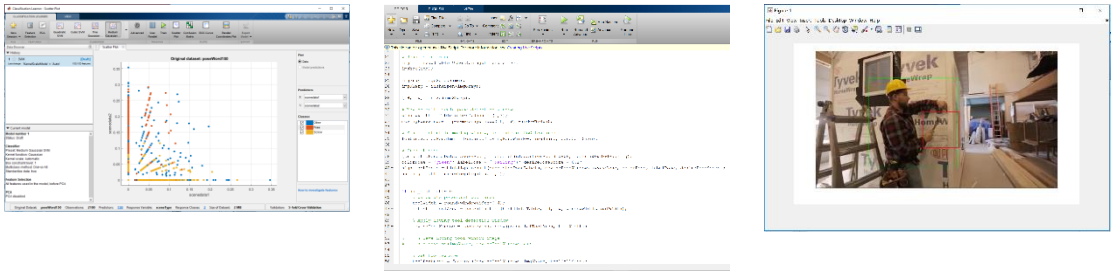
This study used construction personnel nailing activity to test the hypotheses because 1) nailing is one of the most common activities in construction and 2) nailing was applied as the testing target activity in previous research.

In this study, nailing activity is defined with three parts: 1) a construction worker with his/her upper body straight up; 2) a bent arm; and 3) a hammer within the potential

tool area. Detailed information regarding potential tool area is provided in the later part of this chapter.

7.3 Implementation Environment

The three frameworks, PARS, PTARS, and TPARS, were developed into three applications to test the hypotheses as described in Chapter 7.1, Hypotheses. The frameworks and applications were implemented and tested in Matlab R2016b with some of its toolboxes, including but not limited to Computer Vision System, Image Processing, and Statistics and Machine Learning. Figure 24 gives some screenshots of the application.



(a) (b) (c)

Figure 24 Screenshots of the application. (a) Using Classification Learner app trains a body pose classifier; (b) part of the code of the PTARS framework; (c) recognition result of PTARS framework.

The programming, training, and testing were accomplished on a MacBook Pro computer running the Microsoft Windows 10 operating system. More specifications of the computer are listed in Table 1.

The programming work of the framework was completed by the author.

Table 1 Computer specification

Processor	Intel Core i7-2635QM CPU @ 2.00GHz
Memory (RAM)	4.00GB

7.4 Image Datasets for Training and Testing

Twenty video clips were collected and used for training the two classifiers, body pose classifier and tool classifier. Screenshots of some of the video clips are shown in Figure 25. The 20 video clips have a combined total duration of 142 minutes. They were collected from three different residential jobsites at 15 different scenes while construction crews were performing framing work. Video backgrounds and illumination conditions vary from each scene. These video clips were collected at 1080P resolution (1920 × 1080) at 24 frames/second.

Testing images are free-to-use images collected from Google, Bing, and YouTube. The testing image set has 500 images, in which 50 are nailing activities. The rest of the 500 includes over nine other activities, including placing rebar, pouring concrete, shoveling, surveying, welding, observing, walking, climbing a ladder, and others. Each test image has at least one construction activity, and some of the test images have multiple construction workers performing various activities.

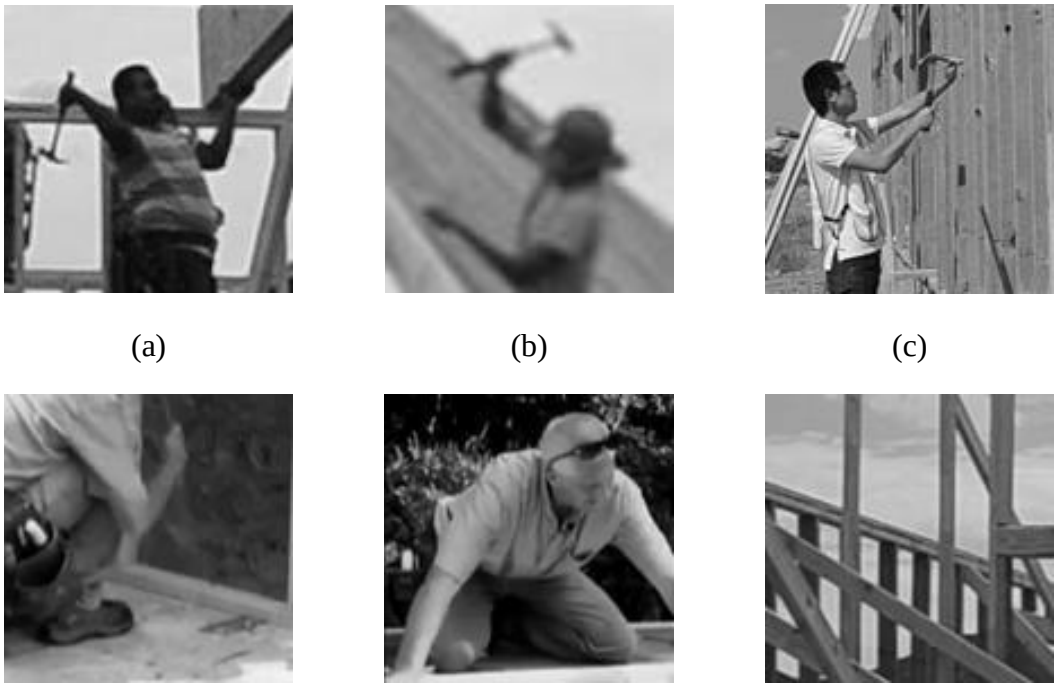
Figure 25 Examples of video footage.

7.4.1 Preparing Images for Pose Classifier Training

In this study, a nailing pose is defined as a construction worker with a straight-up upper body and a bent arm. One-thousand images of nailing poses, the positive images, were manually cropped from the original video frames for training purposes, as shown in Figure 26(a)(b)(c). The 1,000 nailing poses images were left-right reflected, as shown in Figure 27, to create a positive nailing pose training set of 2,000 images. In the positive pose training images, a nailing pose is located approximately in the center of each training image.

Negative images are the training images without nailing activity. There are three types of negative pose training images: images that have partial nailing activity, images that have other activities, and images of construction background, as shown in Figure

26(d)(e)(f). A fixed set of 20,000 negative poses was also collected. The negative dataset includes 1,000 images that have partial nailing activity, as shown in Figure 26(d); 1,000 images that have other activities, as shown in Figure 26(e); and 18,000 images of construction background that are randomly sampled from 1,000 nailing-activity-free images, as shown in Figure 26(f).



(a) (b) (c)

(d) (e) (f)

Figure 26 Example of training images for pose classifier. First row shows examples of positive image; second row shows examples of negative image.

(a) (b)

Figure 27 Illustration of left-right reflection of an image. (a) Original image that has been manually cropped from the original video frame; (b) left-right reflection of the original cropped image.

The cropped positive images and negative images have various sizes ranging from 100 × 100 to 1000 × 1000 pixels. To train a classifier with the positive and negative images, the HOG features of each image must have the same vector size. To ensure that each image has the same HOG feature vector size, the size of each image must be the same. Thus, all the training images for the pose classifier were scaled to the same size, 128 × 128 pixels, which has been widely used as the training image size for human shape recognition and has proved to perform well.

Previous studies (Banko & Brill, 2001) have shown that increasing training dataset size improves classifier performance. This study tried to maximize training dataset size; the size is comparable to previous studies, such as Dalal and Triggs (2005) with 2,478 positive images and 12,180 negative images and Park and Brilakis (2012) with 500 positive images and 2,200 negative images. All the training images were converted to grayscale images to save computational cost.

7.4.2 Preparing Images for Tool Classifier Training

Images of a hammer are treated as positive tool classifier training images. There are two types of positive tool training images: 1) 150 images of a hammer being held by a construction worker, as shown in Figure 28(a), and 2) 100 images of only a hammer, as shown in Figure 28(b). Images of the first type were selected and cropped from the original video frames, and images of the second type were collected from Google and Bing. The 250 (150 + 100 = 250) positive tool images were left-right mirrored and up-down mirrored, as shown in Figure 29, creating a total of 1,000 images considered positive tool training images.

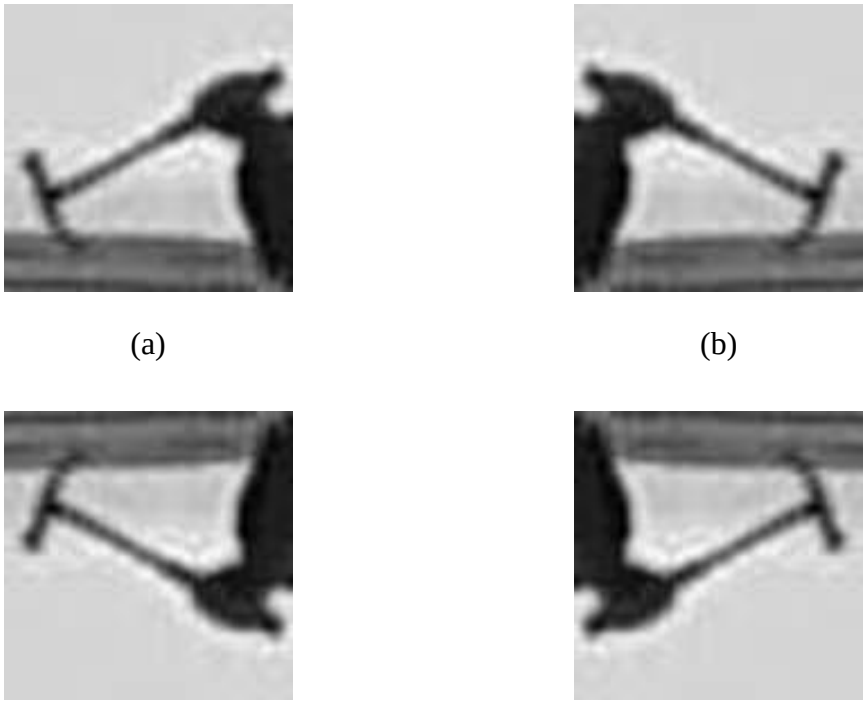
Negative tool training images are those without a hammer; 20,000 negative tool images were collected. There are three types of negative tool training images: 1) 2,000 images that have a partial hammer, as shown in Figure 28(c); 2) 5,000 images of a partial upper body such as head, torso, and arm, as shown in Figure 28(d); and 3) 13,000 images of construction background, as shown in Figure 28(e).

Both positive and negative images were scaled to the same size, 128 × 128 pixels, to ensure that each image has the same HOG feature vector size. All the training images were converted to grayscale images to save computational cost.

(a) (b)

(c) (d) (e)

Figure 28 Example of training images for tool classifier. First row shows examples of positive image; second row shows examples of negative image.



(a) (b)

(c) (d)

Figure 29 Example of left-right and up-down reflection of an image. (a) Original image; (b) left-right reflection of the original image; (c) and (d) up-down reflection of (a) and (b).

7.4.3 Testing Image Dataset

Testing images were collected to test the performance of the frameworks. The process of collecting testing images was completely separate from the collection of training images. Images of the testing dataset were free-to-use images collected from Google and Bing, as well as video frames downloaded from YouTube.

The testing image dataset has 500 images, in which 50 are nailing activities, as shown in the first row of Figure 30. The rest of the 500 includes over nine other

activities, including placing rebar, pouring concrete, shoveling, surveying, welding, observing, walking, climbing a ladder, and others, as shown in the second row of Figure 30. Each test image has at least one construction activity, and some of the test images have multiple construction workers performing various activities. The testing images vary in image size, illumination condition, background environment, size of construction worker appearing in the image, appearance of construction worker, and appearance of a hammer.

Figure 30 Examples of testing images. First row shows examples of nailing activity images; second row shows examples of images of other activities.

7.5 Implementation of the Algorithms

This section gives technical details of the techniques and algorithms applied in the frameworks, including sliding window, feature descriptors, classification algorithms, and pose-tool spatial relationship. The proposed frameworks take advantage of three elements of a construction activity: the shape features of a construction worker's body pose, the shape features of an associated tool, and the spatial relationship between the body pose and the tool. Regardless of the differences in detection sequence of the three frameworks, each of two proposed frameworks contains four major parts or techniques, while the benchmark framework, PARS, has two steps: sliding window and pose recognition. The four major parts are sliding window, pose recognition, tool recognition, and pose-tool spatial relationship.

7.5.1 Sliding Window

Sliding window was applied to scan through the testing image, the potential tool area, and the potential pose area. The sliding window generates overlapping patch images, as shown in Figure 3(b), for the image or the area. Every patch image was scaled to a 128- × 128-pixel grayscale image, which is the same as the training images to ensure the same HOG feature vector size.

In this study, the sliding window was set to be a square area that can cover the entire body pose area or the hammer area, regardless of the facing direction of the construction worker and the rotation of the hammer. The step size, also called stride, was

set to be one-fifth of the sliding window's width for nailing pose detection and 5 pixels for hammer detection. Starting from the original point, which is the upper left corner of a testing image, the sliding window moves horizontally and vertically every one-fifth of the sliding window's width, or every 5 pixels for tool detection, as shown in Figure 31.

Figure 31 Illustration of sliding window. The sliding window starts from the origin point [1, 1] located at the upper left corner of the image. The red square represents the first window. The window moves every stride size, and the green square represents the second window. The sliding window keeps moving vertically and horizontally until its lower right corner reaches the lower right corner of the testing image.

The width of the sliding window can be set according to the size of the patch image of a nailing action. In this study, the size of the nailing action varies dramatically in the test image set, so the sliding window was set as a multiscale sliding window. For nailing body pose recognition in PARS and PTARS, the width starts from 100 pixels, with 50-pixel incrementation per iteration, until the window size is equal to the width or height of the image. For hammer recognition in TPARS, the width of the sliding window

starts from 50 pixels, with 25-pixel incrementation per iteration, until the window size is equal to half of the image width or height.

By measuring 1,000 training images cropped from the video footage, the sizes of the original nailing action image patch range from 123 × 123 pixels to 434 × 434 pixels, the sizes of the original hammer image patch range from 55 × 55 pixels to 192 × 192 pixels. The coefficient between the width of nailing pose image patch and its associated hammer image patch was calculated as shown in Eq. 18.

$$Coefficient = \frac{\sum_{i=0}^{1,000} \frac{width\ of\ nailing\ pose\ image\ patch\ i}{width\ of\ the\ associated\ hammer\ image\ patch\ i}}{1,000} = 0.45 \quad \text{Eq. 18}$$

For nailing body pose recognition in TPARS and hammer recognition in PTARS, the relationship between the width of the sliding windows for pose and hammer recognition are defined as Eq. 19 and Eq. 20.

$$width\ tool\ sliding\ window = width\ pose\ sliding\ window \times Coefficient \quad \text{Eq. 19}$$

$$width\ pose\ sliding\ window = \frac{width\ tool\ sliding\ window}{Coefficient} \quad \text{Eq. 20}$$

7.5.2 Implementing Features of Histogram of Oriented Gradients

HOG was applied in this study to extract features from the patch images generated by the sliding window, as well as for summarizing the features of the training image dataset.

HOG has proved to be one of the most robust feature descriptors to summarize the shape features for an object, especially for human bodies. The HOG feature vector is formed by accumulating each pixel's gradient magnitude to its corresponding bin that the gradient orientation falls into.

This study applied the "extractHOGFeatures" function in Matlab. The coding syntax is shown as follows:

features = extractHOGFeatures (I, Name, Value)

For each training image, the function extracts HOG features from an input image "I" and returns a 1 × N vector where "N" is the HOG feature length. Repeating the feature extraction process for each of the images in the training set with "m" number of images returns the training data, which is an m × N table; the table was used in later steps to train a classifier.

The function extracts HOG features from each patch image and returns a 1 × N vector. The vector is processed by the classifiers and classified into nailing pose or non-nailing pose, or hammer or non-hammer.

"Name, Value" are extra arguments that the function handles to customize the feature extraction process. "Name" is the name of the argument, and "Value" is the corresponding value. In this study, the following "Name, Value" arguments have been applied as shown in Table 2.

Table 2 Parameters for extracting HOG features

Name	Value	Function
CellSize	[2-element vector]	Set the size of each cell. Increasing the cell size could capture larger area's spatial information.
BlockSize	[2-element vector]	Set the number of cells in each block. Reducing block size could help capture the significance of local pixels and help suppress illumination changes.
BlockOverLap	[2-element vector]	Set the number of overlapping cells between adjacent blocks. The overlapping area must be at least half of the block size to ensure adequate contrast normalization. Increasing its value could help capture more information, but also increases the vector size.
NumBins	Positive scalar	Set the number of orientation histogram bins.

Based on the experimental results of previous studies on human detection (Dalal & Triggs, 2005) and construction personnel detection (Park & Brilakis, 2012), CellSize was set as [8 8], BlockSize was set as [2 2], BlockOverLap was set as BlockSize/2, and NumBins was set as 9.

The parameters for applying HOG features on tool images were also set as [8 8] for CellSize, [2 2] for BlockSize, BlockSize/2 for BlockOverLap, and 9 for NumBins.

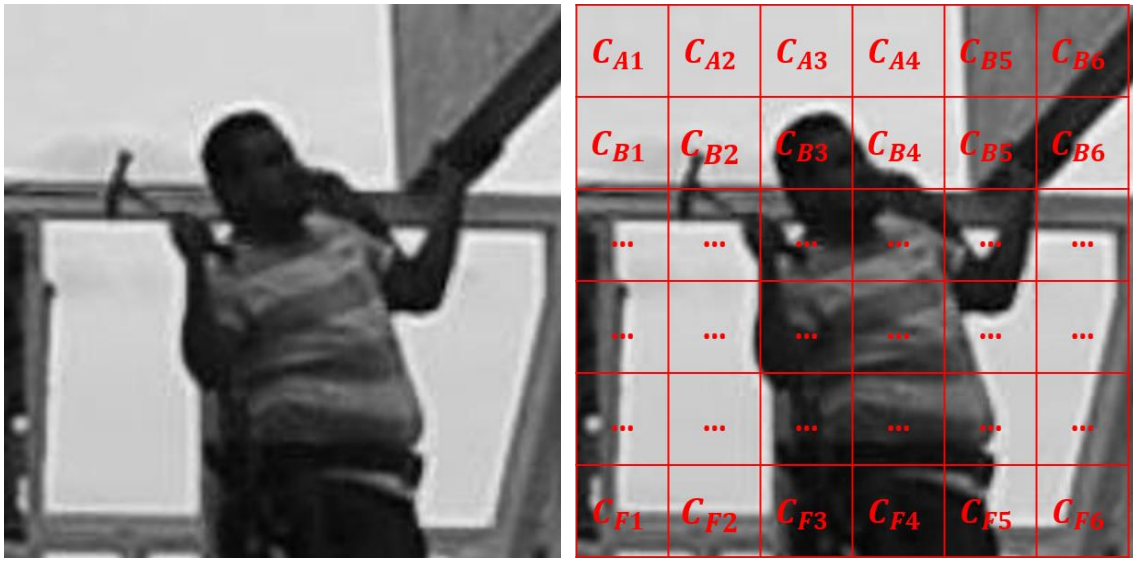
The following gives an example of the function's input argument and output argument and explains how the function works. For presentation purposes, this example uses a 384- × 384-pixel image and [64 64] CellSize instead of a 128- ×128-pixel image

and [8 8] CellSize, which are the setups in the real experiments. To extract the HOG features, the command can be written as follows:

TestImgFeature = extractHOGFeatures(TestImg.jpg, CellSize, [64 64], BlockSize, [2 2], BlockOverLap, ceiling(BlockSize/2), NumBins, 9)

The above command extracts HOG features from an image named as TestImg.jpg at cell size [64 64], block size [2 2], overlapping area at half of the block size, and nine orientation histogram bins. The original image is shown in Figure 32(a).

The gradient orientation and gradient magnitude of each pixel were first calculated as shown in Eq. 3 and Eq. 4. Then, the original 384- × 384-pixel image was divided into 64- × 64-pixel small cells, with 36 cells, as shown in Figure 32(b); each cell can be named with letters and numbers.

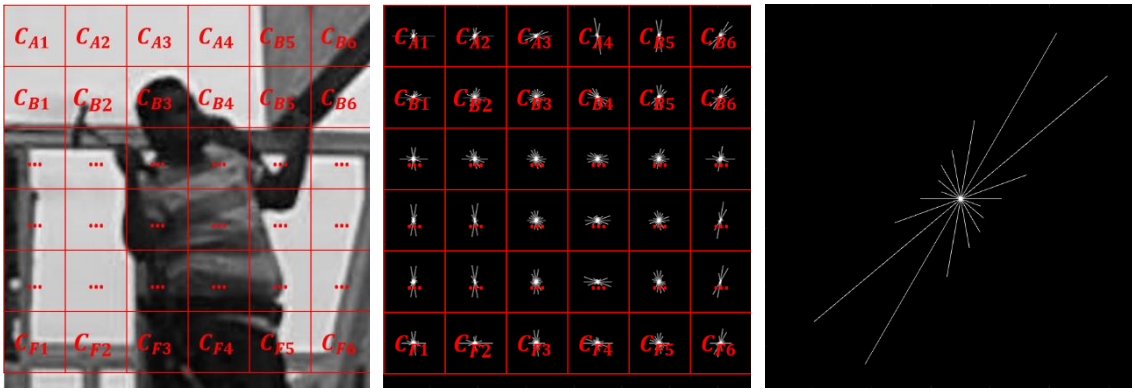


(a) (b)

Figure 32 Original example image and illustration of cells over the original image.

HOG features are extracted and summarized with each cell as a unit. Image gradient represents the intensity changes across pixels in an image. The intensity changes from lighter tones to darker shades, or conversely, can be measured along any direction.

In this example, the number of orientation histogram bins was set to nine, meaning that the intensity changes were measured along nine unsigned orientations. As shown in Figure 33(b), the intensity changes along each of the nine directions are summarized within each cell. Figure 33(c) enlarges the feature summarized from Cell B6, and by comparing the content of Cell B6 in (a) and (c), it can be observed that more intensity changes occur along 45°; thus, the magnitude of 40° and 60° orientations are much stronger than the magnitudes of other directions.



(a) (b) (c)

Figure 33 Illustration of HOG features summarized from each cell in the example image.

For each cell, the histogram, $H(C_{yx})$, can be summarized into a 1 × NumBins vector; in this example, NumBins equals nine.

The HOG feature vectors of the entire image are arranged by HOG blocks. As the BlockSize was set as [2 2], every HOG block is composed of 2 × 2 cells, as shown in the highlighted area in Figure 34(a). Also, as the BlockOverLap was set as half of the block size, meaning for each block, half of the block area is overlapped by each adjacent block, as shown in Figure 34(b). Thus, the arrangement of HOG feature vector can be illustrated as shown in Figure 34(c).

For each image, the function eventually outputs a 1 × N vector, where N represents HOG feature length. The value of N can be calculated based on the image size and the function parameter values discussed above.

$$N = \text{BlocksPerImage} \times \text{CellsPerBlock} \times \text{NumBins} \quad \text{Eq. 21}$$

In this example, there are 25 blocks within the 384- × 384-pixel image, four cells per block, and nine orientations per cell. Thus, employing the extractHOGFeatures function on the test image returns a 1 × 900 vector. If we have a training image set of m images, the function returns an m × 900 matrix, which describes the entire image set.

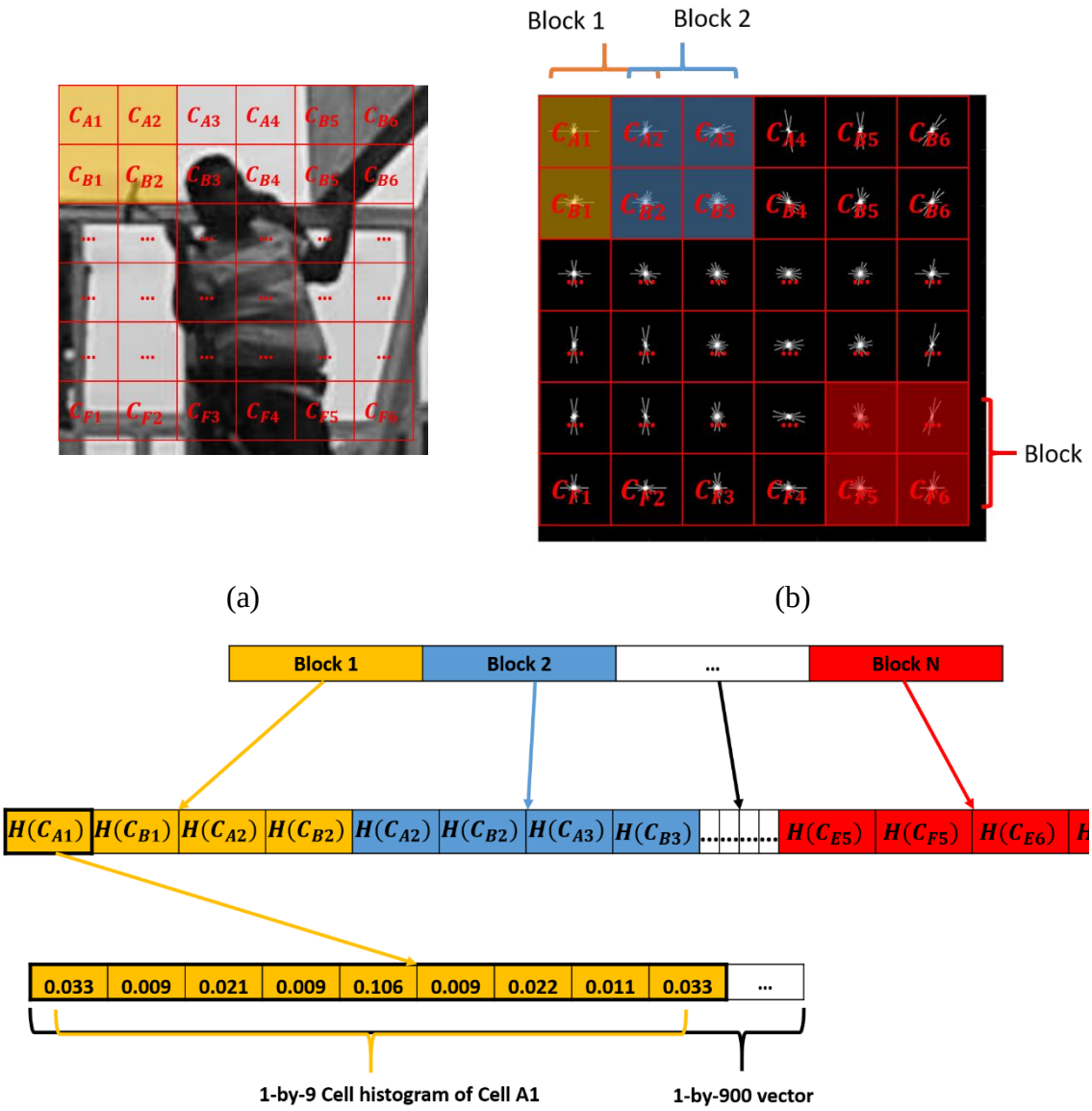


Figure 34 Illustration of Block, BlockOverLap, and arrangement of feature vector.

7.5.3 Implementing Support Vector Machine

The Classification Learner app in Matlab was applied in this study to train the classifiers, including both SVM and k-NN. The app takes the previously introduced

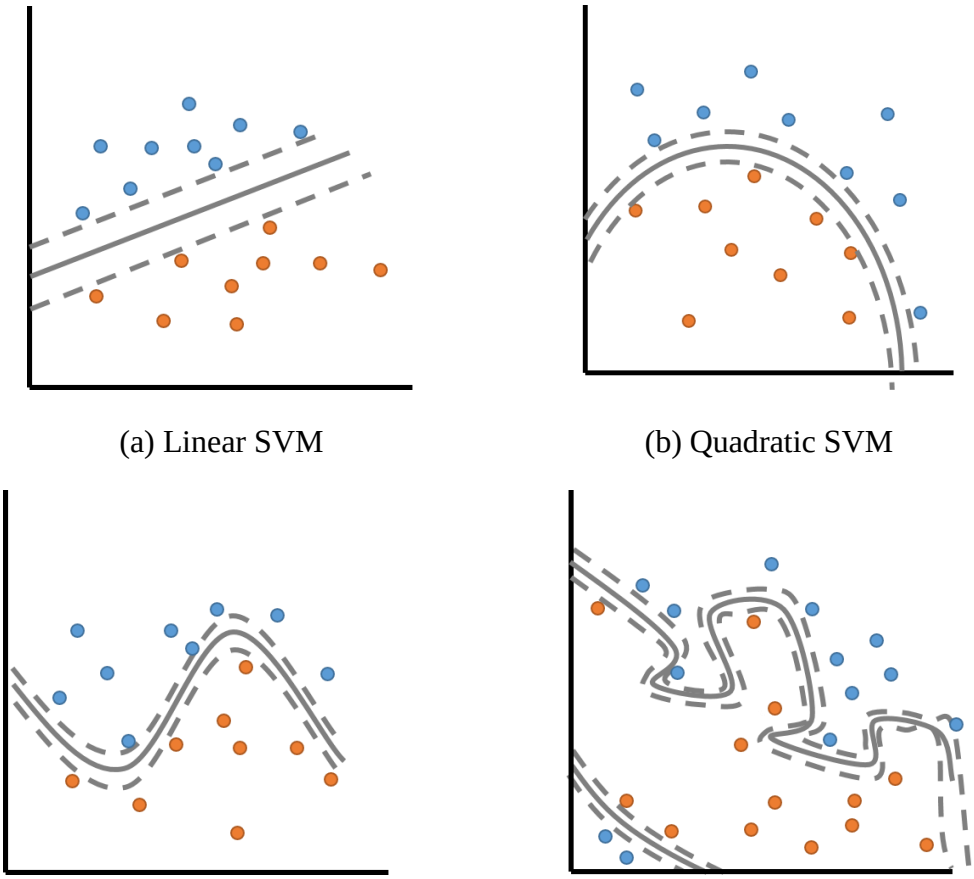
HOG feature matrix as the input and returns a classification model that can predict the category of the new query data or image. Using the Classification Learning app has several advantages, and it can reduce a lot of redundant programming work. First, the application can automatically reduce a multiclass classification problem into a set of binary classification subproblems, with one classifier for each subproblem. Second, it has the most commonly used options for each type of classification technique. The user can easily try all the options and evaluate the performance of each setup. Third, the app automatically separates the input training set into a training set and a validation set. Introducing the idea of validation can prevent the problem of data overfitting during the classification model selection process. In the machine learning area, the term "overfitting" means that a trained model fits well with the training data, but doesn't perform as well with new query data. More specifically, in this study, 20% of the original training set was randomly selected and held out as the validation set. The app trains a model using all the data outside the validation set. The app tests the performance of the model using the data inside the validation set.

As described in Chapter 6.2.2, Classifier, SVM classifies data from different categories by finding the best separating hyperplane that leaves the largest margin between the two categories. The data piece here is the matrix returned by the feature extraction functions. With the given data, the Classification Learner app can test certain or all different SVM options, including linear, quadratic, cubic, and Gaussian kernel SVM, as illustrated in Figure 35, to compare which option produces the best model that separates data from different categories; eventually, we could select a classifier with the

highest accuracy in predicting the data categories in the validation set. The SVM options are different shapes of the separating hyperplane, as shown in Figure 35. In the figure, the solid gray line represents the separating hyperplane, and the X-axis and Y-axis represent values Feature 1 and Feature 2 from the matrix, respectively. As shown in Table 3, blue dots represent Category I, orange dots represent Category II, and the dots on the gray dashed lines are support vectors.

Table 3 Example of training feature table

Row #	Feature 1	Feature 2	…	Feature n	Category
1	0.08	0.4	…	…	I
…	…	…	…	…	…
…	0.1	0.25	…	…	II
m	…	…	…	…	…



(a) Linear SVM (b) Quadratic SVM

(c) Cubic SVM (d) Gaussian SVM

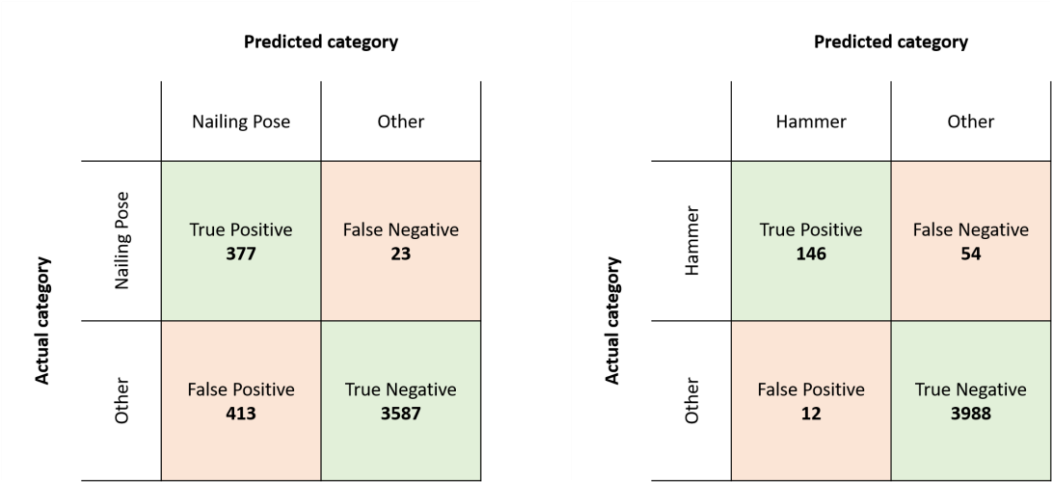
Figure 35 Illustration of SVM options.

By testing the four SVM options, this study applied Gaussian SVM for training the pose classifier and tool classifier because of its good performance, and it also fitted the suggestion provided by previous studies for similar purposes. A pose classifier and a tool classifier were trained with the Gaussian SVM algorithm.

During the training process, one-fifth of the training images were randomly selected and held out as the validation set, meaning 400 positive images and 4,000 negative images were held out from the original pose training image set, and 200

positive images and 4,000 negative images were held out from the original tool training image set.

The pose and tool classification models' test results with the validation set are shown in Figure 36(a) and (b), respectively.



(a) (b)

Figure 36 SVM classifier validation set test results.

7.5.4 Implementing k-Nearest Neighbor

k-NN algorithms are widely used as the benchmark training technique in machine learning tasks. This study applied k-NN with the Classification Learner app. The k-NN classification algorithms categorizes the query points/features based on their distance to the corresponding points/features in the training dataset.

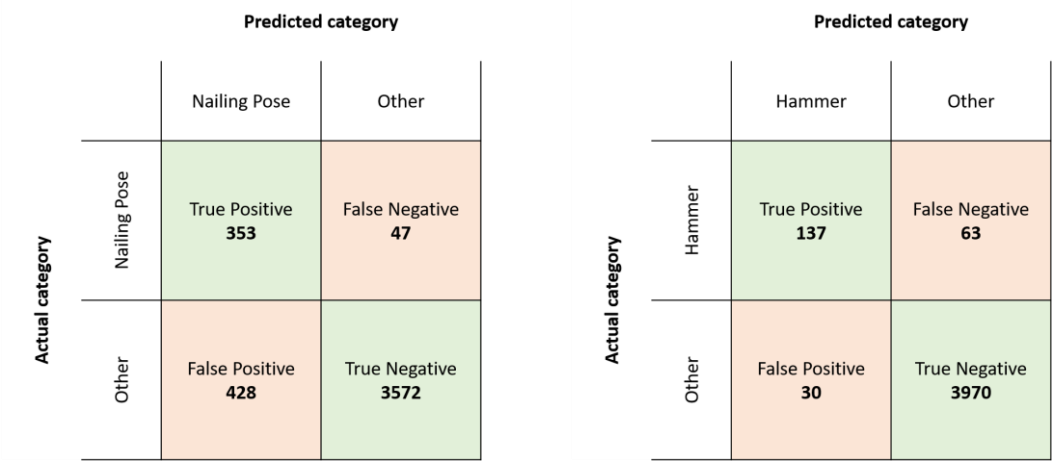
With the Classification Learner app, there are four parameters that can be customized, as shown in Table 4.

Table 4 k-NN parameters table

Number of neighbors	Set the number of nearest neighbors to find for classifying each query point.
Distance metric	Set the metric to measure the distance between the query point and the nearest neighbors. This study selected Euclidean distance as the metric for distance measurement.
Distance weight	Set the weight function from the options of: Equal: no weight Inverse: weight = 1/distance Squared inverse: weight = $1/distance^2$
Standardize data	Set either scale each coordinate distance or not. If the predictors have widely different scales, for example, if the value of feature 1 ranges from 0 to 1 while feature 2 ranges from 2 to 1000, standardizing can improve the result.

In this study, the number of neighbors, k, was set to five; distance metric was set to Euclidean distance, as shown in Eq. 12; distance weight was set to equal; and standardize data was set to true. These parameters were set up based on suggestions from previous studies, as well as preliminary test results of this study. A pose classifier and a tool classifier were trained with the k-NN algorithm.

The same validation set was held out during the k-NN training process. The validation set test results are shown in Figure 37.



(a) (b)

Figure 37 k-NN classifier validation set test results.

Figure 38 compares the validation set test results of SVM and k-NN classifiers; it compares the precision and recall values of SVM and k-NN classifiers. As mentioned in Chapter 6.3, Evaluation of Algorithm Performance, precision measures the accuracy of the prediction/classification model, while recall measures the sensibility of the model. They represent the most appropriate evaluation metric to measure performance in this study.

From the comparison results, it is not hard to conclude that SVM performs better than k-NN for body pose HOG features and tool HOG features. Thus, the SVM pose classifier and SVM tool classifier were chosen to build the frameworks.

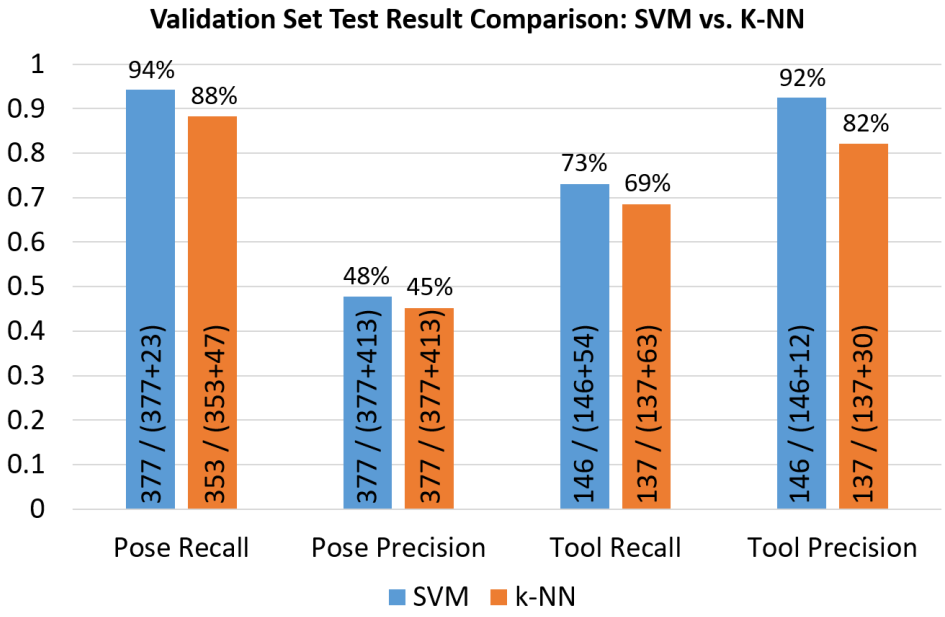


Figure 38 Validation set test results comparison of SVM and k-NN classifiers.

7.5.5 Pose-Tool Spatial Relationship

The pose-tool spatial relationship was applied within PTARS, one of the proposed frameworks. PTARS detects a construction worker's body pose first and then searches for the associated tool within the potential tool area. The potential tool area, the red rectangle in Figure 39, is defined with the location and size of an image patch, which has been recognized as body pose, shown as the green rectangle in the figure.

Figure 39 Pose-tool spatial relationship.

The upper left corner of the image is $P_0(0, 0)$; the relationship between the width of the pose sliding window, BW, and the width of the tool sliding window, TW, is shown in Eq. 19 and Eq. 20. TW = 0.45 × BW. The potential tool area can be defined and calculated as follows:

Upper left corner of potential tool area	$Pt(x_t, y_t) = P((x - TW), (y - TW))$	Eq. 22
Height of potential tool area	$H_t = BW + TW$	Eq. 23
Width of potential tool area	$W_t = BW + TW \times 2$	Eq. 24

Once the potential tool area is defined, another sliding window, shown as the blue dashed rectangle in the figure, is applied within the area for tool detection. If and only if both the nailing body pose and the hammer are detected, PTARS recognizes the area as nailing action.

7.5.6 Tool-Pose Spatial Relationship

The tool-pose spatial relationship was applied within TPARS, which is a reversed version of pose-tool spatial relationship. Once an image patch has been recognized as the tool, shown as the green rectangle in Figure 40, it defines the potential pose area, which is shown as the red rectangle.

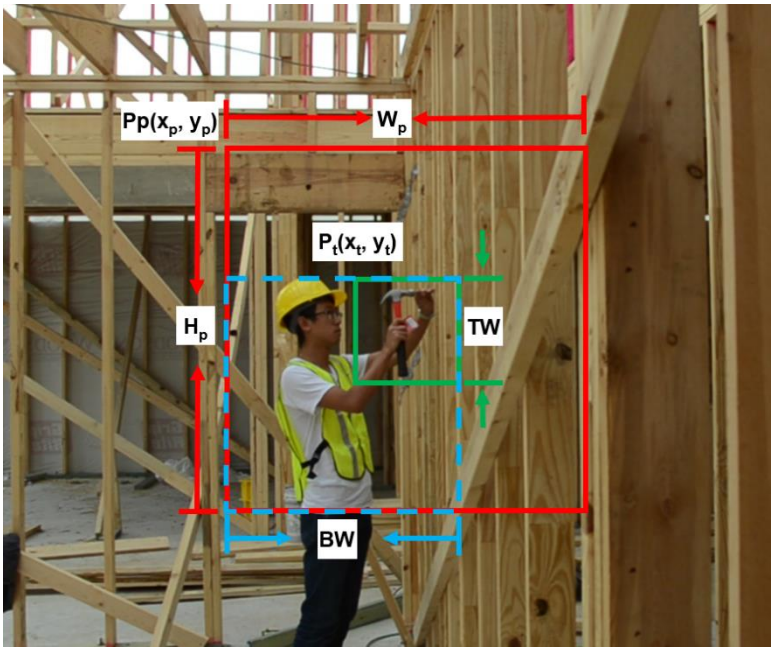


Figure 40 Tool-body relationship.

The potential pose area can be defined and calculated as follows:

Upper left corner of potential pose area	$P_p(x_p, y_p) = P_t((x_t + TW – BW), (y_t + TW – BW))$	Eq. 25
Height of potential pose area	$H_P = BW \times 2 – TW$	Eq. 26
Width of potential pose area	$W_P = H_P$	Eq. 27

Once the potential pose area is defined, another sliding window, shown as the blue dashed rectangle in the figure, is applied within the area for pose detection. If and only if both the hammer and the nailing body pose are detected, PTARS recognizes the area as nailing action.

7.6 Results

As discussed in Chapter 6.3, Evaluation of Algorithm Performance, precision, recall, and F1 score are used to measure the performance of the three frameworks, PARS, PTARS, and TPARS.

To calculate precision, recall, and F1 score, three values of each framework were collected as raw data. The three values are true positive, false positive, and false negative. In this study, the two categories are nailing and non-nailing. The test results of

each framework were collected as a table, as shown in Figure 41. Figure 42 gives an example of a testing image, the red rectangle and the green rectangle represent two image patches predicted/recognized as nailing activity (in the real experiment, all highlighted rectangles are the same color). By human observation, the actual category of the red rectangle is non-nailing, and the actual category of the green rectangle is nailing. However, the framework predicts both image patches as nailing. Thus, the red rectangle is a false positive, the green rectangle is a true positive, and it is a false negative if the framework fails to predict the image patch in the green rectangle as in the nailing category.

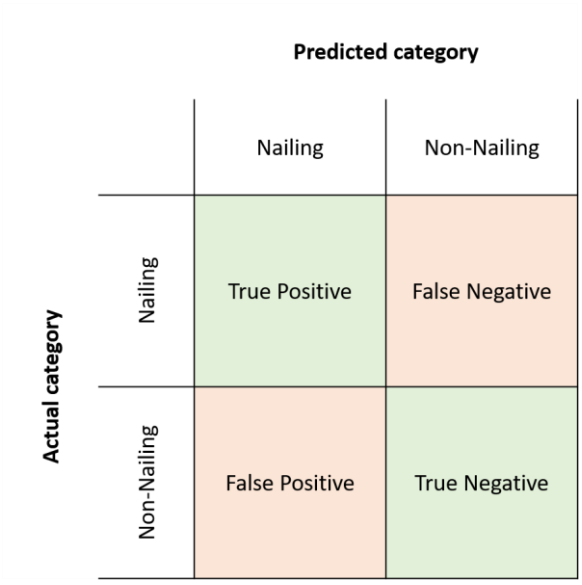


Figure 41 Raw data collection table.

Figure 42 Example test image.

Based on the implementation details of each algorithm and technique, as mentioned in Chapter 7.5, Implementation of the Algorithms, the testing image dataset, as described in Chapter 7.4.3, Testing Image Dataset, was tested with the three frameworks. Figure 43 shows two examples of the detection result made by PARS, PTARS, and TPARS. In Figure 43(a), PARS successfully recognized the nailing activity; PTARS also successfully recognized the nailing activity; and TPARS not only recognized the nailing action, but also mistakenly recognized another image patch on the left of the image as nailing action. In Figure 43(b), PARS mistakenly recognized the construction worker with a bended arm on the left of the image as nailing activity; PTARS did not recognize any image patch as nailing activity because it did not recognize any hammer within the potential tool area and eliminated false detection made by PARS; and although TPARS mistakenly recognized two image patches as hammers,

it eventually did not make any false detection because it did not recognize any nailing body pose in the potential pose areas associated with the two detected hammers.

The experimental results were collected as shown in Figure 44.

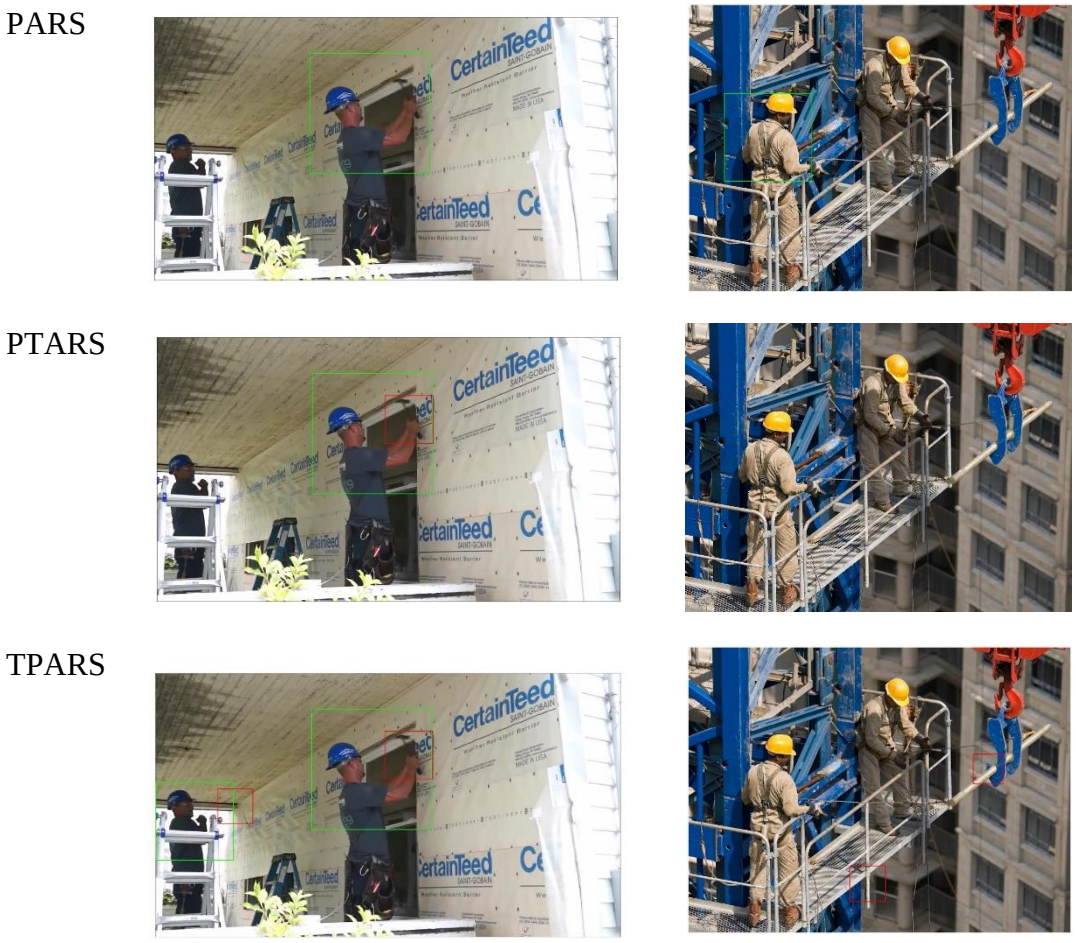


Figure 43 Examples of detection results.

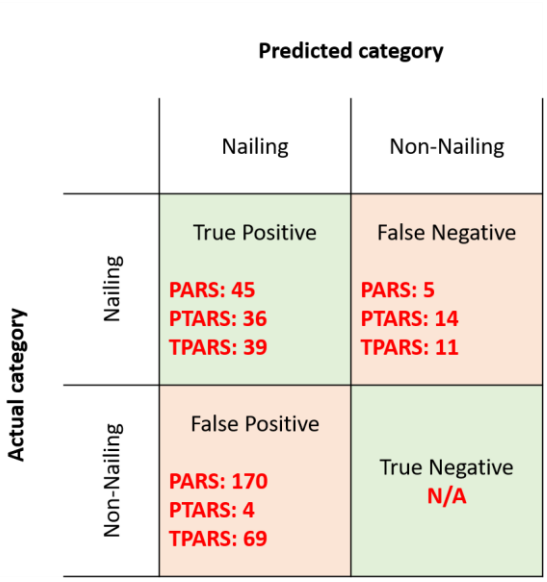


Figure 44 Raw data collection.

Precision measures the accuracy of a framework's prediction result; recall measures the sensitivity of a framework to the target activity; and F1 score measures the overall performance of a framework. According to Eq. 13 to Eq. 15:

Precision = True Positive / (True Positive + False Positive)

Recall = True Positive / (True Positive + False Negative)

F1 Score = 2 × Precision x Recall / (Precision + Recall)

The results and the computation time for detection of each image of each framework are recorded in Table 5.

Table 5 Table of experiment results.

	Precision	Recall	F1 Score	Time (s)
PARS	21%	90%	0.3405	<0.5
PTARS	90%	72%	0.8	0.5~1.5
TPARS	36%	78%	0.4926	>180

8. CONCLUSIONS AND DISCUSSION

8.1 Discussion

From the experimental results shown in Table 5, PTARS has the highest precision and F1 scores, PARS has the highest recall but the smallest precision scores, and TPARS has moderate precision, recall, and F1 scores.

PTARS, one of the proposed frameworks, has a much higher precision than PARS because it eliminated plenty of patch images of poses similar to nailing poses but without a hammer. As shown in Figure 44(b), a construction worker was mistakenly recognized as nailing activity by PARS, while PTARS successfully eliminated the patch image as it did not recognize any hammer within the potential tool area. However, because PTARS also failed to detect some of the hammers in the potential tool area, PTARS has a lower recall than PARS, as shown in Figure 45.

(a) (b)

Figure 45 Example of a false negative. (a) True positive made by PARS; (b) false negative made by PTARS because it failed to detect the hammer within the potential tool area.

TPARS has a much longer average computational time for the detection of each testing image because the width of its sliding window is much smaller than PARS and PTARS and it has a smaller stride size, making it detect many more patch images than the other two frameworks.

As mentioned in Chapter 6.3, Evaluation of Algorithm Performance, human activity recognition from image or video is a typical case of skewed class where a significant portion of the testing dataset describes the negative category and only a very small portion of the dataset is in the positive category. In the case of this experiment, the positive category has 50 images of nailing activity, and the size of the negative category is at the multimillion level. Because of the small sliding window size and small stride size of TPARS, the size of its negative category is over 10 times larger than PARS and PTARS. Based on the multimillion-level negative category size, even the tool classifier has a very high accuracy, but it still generated many more false positive results than PTARS. Remembering Eq. 13, the equation of precision, with the very close amount of true positive (36 for PTARS, 39 for TPARS), TPARS has many more false positive results (4 for PTARS, 69 for TPARS); thus, TPARS has a lower precision value than PTARS.

PTARS achieved the highest F1 score, meaning that it has the best overall performance compared with PARS and PTARS. However, it is necessary to take precision and recall into consideration while choosing a framework for practical purposes. As mentioned in Chapter 1, with further development, a vision-based system

such as a construction personnel activity recognition system has two major potential applications: 1) it could be used as a warning system when unsafe behavior, accident, or injury occurs, 2) it could be used as part of a tracking system that can collect information, such as count, location, and moving trajectory of construction personnel and equipment, for productivity and performance analysis.

A system with a higher recall (sensitivity), such as PARS and TPARS, is more desirable for a warning system because we don't want to miss any unsafe behavior, accident, or injury, especially those that may lead to fatality. Although an activity detection system with a higher recall may have a lower precision (accuracy), it is still worth looking into the warning and making a decision on whether to take further action or determining a false alarm.

In contrast, a system with a high precision (accuracy) and a moderate recall and computational speed, such as PTARS, is more appropriate for a tracking system because it provides more accurate detection and tracking results.

8.2 Conclusion

Replacing traditional manual jobsite monitoring processes conducted by project management personnel with an automatic or semi-automatic system could improve the efficiency of documentation, data analyzing, and decision-making processes. The vision-based system has proved to be one of the most cost- and time-effective approaches.

Until now, all previous studies regarding construction worker activity recognition only applied features of construction worker poses. However, actions appearing on construction sites are more dynamic and complex, and these construction-related actions are usually related to human interactions (operations) with certain types of objects (tools). This research developed frameworks that mimic a human's way of understanding an action. It compared the recognition results of three frameworks with four values, including precision, recall, F1 score, and time consumption. The three frameworks are 1) PARS, detection based only on construction worker pose; 2) PTARS, which first detects construction worker pose and then detects the tool within the potential tool area defined by spatial relationship; and 3) TPARS, which first detects a tool and then detects the pose within the potential pose area defined by spatial relationship. The frameworks were tested with images recorded in various environments. Based on the experimental results and the discussion, the following conclusions can be made:

1) PTARS has the best overall performance, and it is the most appropriate approach to be applied as part of a tracking system that can collect information, such as count, location, and trajectory of construction personnel, for productivity and performance analysis;

2) At the current stage, or with the pose classifier and the tool classifier trained in this study, PARS is a better option than PTARS and TPARS for a warning system;

3) At the current stage, TPARS is not an appropriate framework to be applied practically because of its moderate precision, recall, and F1 score and its long computational time.

8.3 Limitations

This study recognizes and acknowledges that a multitude of factors contributes to the results and outcomes. Because of this, it is virtually impossible to account for every factor or contributing nuance. The following are some of the limitations particular to this study.

8.3.1 Selection of Algorithms

There are many algorithms and methods we can select and test for generating a feature descriptor or training a classifier. However, the goal of this research was to test whether combining pose recognition with tool recognition would provide a better action recognition result than only using pose recognition. Therefore, the research did not test and compare every possible combination of feature descriptors and classifiers, but selected HOG feature and SVM and k-NN classifiers as the techniques for this study because of their extraordinary performance in previous related research. The study of other combinations of object recognition algorithms for action recognition offers one direction for future research.

8.3.2 Selection of the Action

On a regular construction site, there are numerous construction crew actions associated with a specific body pose and a certain type of tool. This study did not test the hypotheses through every single action, but selected nailing action for the test. Nailing action was selected because 1) it is a very typical action on a daily construction site; 2) it has been tested in previous construction crew action recognition frameworks based only on crew pose; and 3) it requires a very specific body pose and a certain type of tool. More actions of construction workers and equipment could be tested in future studies.

8.3.3 Training Images and Test Images

The training and testing data influenced the results of the framework. However, it is not practical to visit all construction sites to collect data. Instead, this research collected training images at various residential construction sites and collected testing images of various construction activities with various backgrounds and image illuminations. This may have had a certain influence on the result. However, as the experiment was designed (three frameworks were tested with exactly the same images), the influence should have been minimized.

8.4 Future Study

This research studied the most fundamental algorithms and frameworks for construction worker action recognition. However, in the area of computer vision-based

construction operation monitoring, the study could be improved in the future in many different ways.

First, more advanced classification techniques and feature descriptors could be applied, studied, and tested for construction operation monitoring. Many other existing classification models could be applied in the construction industry.

Also, as mentioned in previous chapters, in the community of computer vision, there are several benchmark image datasets that have been repetitively used by different studies. It is necessary for the construction industry to build up its own benchmark image dataset with multiple construction-related categories such as construction workers, actions, tools, and equipment.

As the most famous human action recognition databases, KTH (Schuldt, Laptev, & Caputo, 2004), UCF (Rodriguez, Ahmed, & Shah, 2008), and Hollywood2 (Marszałek, Laptev, & Schmid, 2009) all have a much larger number of images for training at higher resolution. It is necessary to build a similar dataset for construction action recognition to provide enough data and to form a benchmark dataset for future studies.

REFERENCES

Aggarwal, J. K., & Ryoo, M. S. (2011). Human activity analysis: A review. ACM Computing Surveys (CSUR), 43(3), 16.

Azar, E. R., & McCabe, B. (2011). Automated visual recognition of dump trucks in construction videos. Journal of Computing in Civil Engineering, 26(6), 769-781.

Azar, E. R., & McCabe, B. (2012). Part based model and spatial–temporal reasoning to recognize hydraulic excavators in construction images and videos. Automation in construction, 24, 194-202.

Christianini, N., & Shawe-Taylor, J. (2000). An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, Cambridge, 2000, xiii+ 189 pp., ISBN 0-521-78019-5.

Banko, M., & Brill, E. (2001, July). Scaling to very very large corpora for natural language disambiguation. In Proceedings of the 39th annual meeting on association for computational linguistics (pp. 26-33). Association for Computational Linguistics.

Borensteln, G., Histogram of Oriented Gradients with ofxCv. Retrieved on August 2016, from https://www.flickr.com/photos/unavoidablegrain/8123343395/in/photostream/

Bourdev, L., Maji, S., & Malik, J. (2011, November). Describing people: A poselet-based approach to attribute classification. In Computer Vision (ICCV), 2011 IEEE International Conference on (pp. 1543-1550). IEEE.

Brilakis, I., Park, M. W., & Jog, G. (2011). Automated vision tracking of project related entities. *Advanced Engineering Informatics*, *25*(4), 713-724.

Brilakis, I. K., Soibelman, L., & Shinagawa, Y. (2006). Construction site image retrieval based on material cluster recognition. *Advanced Engineering Informatics*, *20*(4), 443-452.

Caldas, C. H., Torrent, D. G., & Haas, C. T. (2004, September). Integration of automated data collection technologies for real-time field materials management. In Proceedings of the 21st International Symposium on Automation and Robotics in Construction, Jeju, Korea.

Chi, S., & Caldas, C. H. (2011). Automated object identification using optical video cameras on construction sites. *Computer-Aided Civil and Infrastructure Engineering*, *26*(5), 368-380.

Copeland, M., NVIDIA. (2016). What's the Difference Between Artificial Intelligence, Machine Learning, and Deep Learning? Retrieved on September 7, 2016, from https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

Costin, A. M., Teizer, J., & Schoner, B. (2015). RFID and BIM-enabled worker location tracking to support real-time building protocol and data visualization. Journal of Information Technology in Construction (ITcon), 20(29), 495-517.

Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (Vol. 1, pp. 886-893). IEEE.

Du, S., Shehata, M., & Badawy, W. (2011, March). Hard hat detection in video sequences based on face features, motion and color information. In *Computer Research and Development (ICCRD), 2011 3rd International Conference on* (Vol. 4, pp. 25-29). IEEE.

Eichner, M., Marin-Jimenez, M., Zisserman, A., & Ferrari, V. (2010). Articulated human pose estimation and search in (almost) unconstrained still images. *ETH Zurich, D-ITET, BIWI, Technical Report No, 272*.

Ergen, E., Akinci, B., & Sacks, R. (2007). Tracking and locating components in a precast storage yard utilizing radio frequency identification technology and GPS. Automation in construction, 16(3), 354-367.

Farhadi, A., Endres, I., Hoiem, D., & Forsyth, D. (2009, June). Describing objects by their attributes. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (pp. 1778-1785). IEEE.

Fergus, R., Perona, P., & Zisserman, A. (2003, June). Object class recognition by unsupervised scale-invariant learning. In Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on (Vol. 2, pp. II-II). IEEE.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). The elements of statistical learning (Vol. 1). Springer, Berlin: Springer series in statistics.

Golparvar-Fard, M., Peña-Mora, F., Arboleda, C. A., & Lee, S. (2009). Visualization of construction progress monitoring with 4D simulation model overlaid on time-lapsed photographs. Journal of Computing in Civil Engineering, 23(6), 391-404.

Gong, J., Caldas, C. H., & Gordon, C. (2011). Learning and classifying actions of construction workers and equipment using Bag-of-Video-Feature-Words and Bayesian network models. *Advanced Engineering Informatics*, *25*(4), 771-782.

Gupta, A., & Davis, L. S. (2007, June). Objects in action: An approach for combining action understanding and object perception. In Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on (pp. 1-8). IEEE.

Horn, B. K., & Schunck, B. G. (1981). Determining optical flow. Artificial intelligence, 17(1-3), 185-203.

Ikizler, N., & Duygulu, P. (2007). Human action recognition using distribution of oriented rectangular patches. Human Motion–Understanding, Modeling, Capture and Animation, 271-284.

Jaselskis, E. J., & El-Misalami, T. (2003). Implementing radio frequency identification in the construction process. *Journal of Construction Engineering and Management*, *129*(6), 680-688.

Joshi, A., Cherian, A., Shivalingam, R. Machine Learning in Computer vision. Retrieve on August 14, 2016, from http://www-users.cs.umn.edu/~cherian/ppt/MachineLearningTut.pdf

Liu, K., & Golparvar-Fard, M. (2015). Crowdsourcing construction activity analysis from jobsite video streams. Journal of Construction Engineering and Management, 141(11), 04015035.

Lu, M., Chen, W., Shen, X., Lam, H. C., & Liu, J. (2007). Positioning and tracking construction vehicles in highly dense urban areas and building construction sites. Automation in construction, 16(5), 647-656.

Maalek, R., Lichti, D., & Ruwanpura, J. (2015). Development of an automated 3D/4D as-built model generation system for construction progress monitoring and quality control.

Maalek, R., & Sadeghpour, F. (2013). Accuracy assessment of Ultra-Wide Band technology in tracking static resources in indoor construction scenarios. Automation in Construction, 30, 170-183.

Mallic, S. (2016). Histogram of oriented gradients. Retrieved on March 25, 2017, from http://www.learnopencv.com/histogram-of-oriented-gradients/

The MathWorks, Inc. Train A Stop Sign Detector. Retrieved on July 24, 2016, from http://www.mathworks.com/help/vision/ug/train-a-stop-sign detector.html?requestedDomain=www.mathworks.com

MathWorks, Support vector machines for binary classification, retrieved from http://www.mathworks.com/help/stats/support-vector-machines-for-binary-classification.html, on October 12, 2016.

MathWorks. ExtractHOGFeature. Retrieved on August 2016, from http://www.mathworks.com/help/vision/ref/extracthogfeatures.html?requestedDomain=www.mathworks.com

MathWorks, Classification using nearest neighbors, retrieved from http://www.mathworks.com/help/stats/classification-using-nearest-neighbors.html#btap66m, on October 12, 2016.

MathWorks, Naïve Bayes Classification, retrieved from http://www.mathworks.com/help/stats/naive-bayes-classification.html, on October 12, 2016.

MathWorks. Train a Cascade Object Detector. Retrieved on September 6, 2016, from http://www.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html

Marszalek, M., Laptev, I., & Schmid, C. (2009, June). Actions in context. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (pp. 2929-2936). IEEE.

Memarzadeh, M., Golparvar-Fard, M., & Niebles, J. C. (2013). Automated 2D detection of construction equipment and workers from site video streams using histograms of oriented gradients and colors. *Automation in Construction*, *32*, 24-37.

Niebles, J. C., Wang, H., & Fei-Fei, L. (2008). Unsupervised learning of human action categories using spatial-temporal words. International journal of computer vision, 79(3), 299-318.

Park, M. W., & Brilakis, I. (2012). Construction worker detection in video frames for initializing vision trackers. *Automation in Construction*, *28*, 15-25.

Park, J., Kim, K., & Cho, Y. K. (2016). Framework of Automated Construction-Safety Monitoring Using Cloud-Enabled BIM and BLE Mobile Tracking Sensors. Journal of Construction Engineering and Management, 05016019.

Pradhananga, N., & Teizer, J. (2013). Automatic spatio-temporal analysis of construction site equipment operations using GPS data. Automation in Construction, 29, 107-122.

Seo, J., Han, S., Lee, S., & Kim, H. (2015). Computer vision techniques for construction safety and health monitoring. Advanced Engineering Informatics, 29(2), 239-251.

Sheikh, Y., Sheikh, M., & Shah, M. (2005, October). Exploring the space of a human action. In Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1 (Vol. 1, pp. 144-149). IEEE.

Sipiran, I., & Bustos, B. (2011). Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes. The Visual Computer, 27(11), 963-976.

Teizer, J. (2015). Status quo and open challenges in vision-based sensing and tracking of temporary resources on infrastructure construction sites. Advanced Engineering Informatics, 29(2), 225-238.

Teizer, J., Caldas, C. H., & Haas, C. T. (2007). Real-time three-dimensional occupancy grid modeling for the detection and tracking of construction resources. *Journal of Construction Engineering and Management*, *133*(11), 880-888.

Teizer, J., Cheng, T., & Fang, Y. (2013). Location tracking and data visualization technology to advance construction ironworkers' education and training in safety and productivity. Automation in Construction, 35, 53-68.

Teizer, J., Lao, D., & Sofer, M. (2007, September). Rapid automated monitoring of construction site activities using ultra-wideband. In *Proceedings of the 24th International Symposium on Automation and Robotics in Construction, Kochi, Kerala, India* (pp. 19-21).

Theaveragebody.com, Average Hand, retrieved from http://www.theaveragebody.com/average_hand_size.php on September 18, 2016

Ukita, N. (2013). Iterative Action and Pose Recognition using Global-and-Pose Features and Action-specific Models. In Proceedings of the IEEE International Conference on Computer Vision Workshops (pp. 476-483).

Yang, J., Park, M. W., Vela, P. A., & Golparvar-Fard, M. (2015). Construction performance monitoring via still images, time-lapse photos, and video streams: Now, tomorrow, and the future. Advanced Engineering Informatics, 29(2), 211-224.

Yao, B., & Fei-Fei, L. (2010, June). Grouplet: A structured image representation for recognizing human and object interactions. In Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on (pp. 9-16). IEEE.

Yi, W., & Chan, A. P. (2013). Critical review of labor productivity research in construction journals. Journal of Management in Engineering, 30(2), 214-225.

Yilmaz, A., & Shah, M. (2005, June). Actions sketch: A novel action representation. In 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) (Vol. 1, pp. 984-989). IEEE.

Zhang, Y., Qu, W., & Wang, D. (2014). Action-scene model for human action recognition from videos. AASRI Procedia, 6, 111-117.

Ziaeefard, M., & Bergevin, R. (2015). Semantic human activity recognition: a literature review. Pattern Recognition, 48(8), 2329-23