Development of Human Poses for the Determination of On-site Construction Productivity in Real-time

BY

Yong Bai, Ph.D., P.E.
Principal Investigator and Associate Professor
Department of Civil, Environmental and Architectural Engineering
The University of Kansas
Lawrence, Kansas 66045

Jun Huan, Ph.D.
Assistant Professor
Department of Electrical Engineering and Computer Science
The University of Kansas
Lawrence, Kansas 66045

And

Abhinav Peddi
Research Assistant
Department of Electrical Engineering and Computer Science
The University of Kansas
Lawrence, Kansas 66045

A Report on Research Sponsored By

The National Science Foundation
Division of Civil, Mechanical, and Manufacturing Innovation
Award Number: 0741374

December 2008

Abstract

To enhance the capability of rapid construction, an automated on-site productivity measurement system is developed. Employing the concepts of Computer Vision and Artificial Intelligence, the developed system wirelessly acquires a sequence of images of construction activities. It first processes these images in real-time to generate human poses that are associated with construction activities at a project site. The human poses are classified into three categories as effective work, ineffective work, and contributory work. Then, a built-in neural network determines the working status of a worker by comparing in-coming images to the developed human poses. The labor productivity is determined from the comparison statistics. This system has been tested for accuracy on a bridge construction project. The results of analyses were accurate as compared to the results produced by the traditional productivity measurement method. This research project made several major contributions to the advancement of construction industry. First, it applied advanced image processing techniques for analyzing construction operations. Second, the results of this research project made it possible to automatically determine construction productivity in real-time. Thus, an instant feedback to the construction crew was possible. As a result, the capability of rapid construction was improved using the developed technology.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

Since the September 11, 2001 terrorist attacks and natural calamities like hurricane Katrina, the transportation system of the United States has been considered as vulnerable targets including highways, bridges, tunnels, seaports and airports. The White House released in February 2003 “The National Strategy for the Physical Protection of Critical Infrastructures and Key Assets” emphasizing the importance and need for protecting the U.S. transportation system. Results of previous research indicated the need to address the recovery phase in emergency management plans [1]. The focus on the recovery phase is to improve the capabilities of rapid replacements when a bridge or highway of a major transportation network is damaged by these extreme events. Rapid replacement of damaged infrastructure has been paid close attention to by government agencies, engineering and construction communities. There is an urgent need for the development of innovative technologies to enhance the capability of rapid construction.

Productivity measurement has been a wide practice to evaluate the performance of a construction activity through its entire phases. The duration to complete an activity is estimated as the ratio of quantity of work to its productivity and hence productivity plays a key role in influencing a construction project schedule. A project schedule needs to be effectively developed to be economic and also efficient, as it is governed by several constraints such as labor, finance, materials and environment. The existing methods are conducted by employing additional labor to manually collect data from the construction sites. As a result, there exist delays in analysis and also may increase the cost of the activity. This additional labor may interfere with the construction activities and lead to inaccurate results due to human errors and biases. This indicates the need to develop an advanced productivity measurement system that will overcome

these shortfalls and function in real-time. In our research, we present two human pose analyzing algorithms for automated and continuous determination of on-site construction productivity in real-time. The task is to capture a video of a construction activity and begin the detection of poses of constituent workers and analyze them to determine productivity.

1.2 Goals

Automated and real-time productivity measurement systems can strongly assist in smooth running of a construction activity. The goal of this research is two-fold.

- To develop human pose analyzing algorithms for productivity determination.
- To set-up an automated and real-time productivity measurement system by deploying these human pose analyzing algorithms.

To realize the first goal, artificial intelligent machines i.e. neural networks and support vector machines have been used for automation of the process. To realize the second goal, a **w**ireless **r**eal-t**i**me produc**t**ivity m**e**asurement (WRITE) System was used to capture the videos of the construction activity and send to the project manager at a remote location via the internet. To process this video and extract worker poses, relevant data structures and image processing algorithms have been developed over the KUIM (K.U. Image processing) framework, developed at the University of Kansas. The human pose analyzing algorithms were then employed to analyze these poses and determine the productivity. The entire system was tested on an activity of tying rebar in a bridge construction activity on Interstate I-70 and results were very encouraging and promising. With this research, we set up a baseline for such human pose based automated determination of productivity and this can be furthered to analyze all activities.

2 Background

Several approaches have been presented in the literature for the determination of construction productivity. Construction companies need to continuously track the productivity at the site to gauge its performance and also to maintain a good profit margin [2]. It also helps to improve the work-force [3]. A company that can finish a project with the minimum cost and in minimum time is a strong bidder for similar projects in the future. The duration of a construction activity is the ratio of the quantity of work to productivity and hence productivity determination plays a key role in influencing a construction project schedule.

Productivity was defined in different ways depending on the scope of research like the project-specific model, multi-factor productivity model and the activity-oriented productivity model [4]. In the activity-oriented model, the labor productivity is determined and it is the most commonly used definition in the construction industry. The productivity is considered as an output in a specific unit of activity to the input in man-hours [4]. Our research also focuses on determining the labor productivity. According to Noor [5], productivity is measured to identify the cost effective methods for construction operations and to obtain accurate labor productivity data.

Due to the sudden increase in the cost of construction labor and shortage of qualified workers since the 1970’s, several methods have evolved for improving the construction productivity based on the motion and time study [6], [7]. Some examples of such methods are stopwatch study, photographic method, taping video, time-lapse video, work-sampling and five minutes rating [5].

Stop watch study was the mainly used technique for productivity determination and it was invented by Frederick W. Taylor in 1880. This is the fundamental method for productivity determination [8]. Photographic and video filming techniques have then evolved in which the observer captures pictures/videos of the activity and comes back to the office for analysis [8], [9], [10]. Video recording techniques have grown from VHS recorders to digital capture devices [11]. The time-lapse filming was a method in which pictures of the construction activity were taken in intervals of 1-5 seconds and videos are created to look like a continuous film so that the whole construction activity can be viewed in a short time [11]. According to Sprinkle [6], this technique enables the management to record videos for training, cost verification an also as evidences for liability law suites and legal disputes. The time-lapse filming technique was used in the 1970's at the University of Michigan.

According to Teicholz [12], a decrease of 0.48%/year was observed in construction productivity from 1964 to 1968, in contrast to the increase of 3.5%/year in the manufacturing productivity. However, with the advancement of technologies, there was a substantial increase in productivity during 1980 and early 1990's. The advent of global positioning system (GPS) technology in construction operations further increased the productivity of highway earth moving operations [13]. Besides these traditional methods for measuring construction productivity, several other empirical and analytical approaches have been presented in the literature by [14], [15] and [16]. Some of these approaches are based on analysis by simulation and are application specific. Other models employing neural networks were developed by [17],[18] and [19]. Additional approach based on neural networks was presented which analyses the attitude patterns of the workers to determine the productivity [20]. However, these approaches are not automated and there is a need for human involvement. Our proposed human

pose analyzing algorithms make it possible to continuously and automatically determine the construction productivity in real-time. We use the concepts of computer vision and artificial intelligence to achieve this.

The first step is to identify and extract the workers from the video. Since workers form the majority of the moving objects in the video, they can be identified by motion segmentation. Several methods for motion segmentation exist in the literature like average background modeling [21], Gaussian background modeling [22], [23], optical flow methods [24-27], foreground modeling [28] and tensor based motion segmentation [29]. If the information of the background is known before hand, a simple background subtraction can be used to identify the workers. This method involves in a pixel to pixel subtraction of the background from the current incoming frames. However, in the real life situations which include outdoor activities, there exist variations in lighting and also a possible jitter/drift in the camera's view due to wind. So, in these situations a global background is not possible meaning that the background information is unknown. The background has to be modeled automatically for every frame and there are several methods to perform this modeling [22], [23] and [28]. A pixel is classified as background/foreground if it does/does not satisfy the background model used.

There are several other methods that do not require the background to be modeled such as the optical flow methods [15, 17] and hierarchical color segmentation by Biancardini [27]. These methods show good results but are computationally very expensive. For this research, we use a modified version of the simple background subtraction with average background modeling in which the background subtraction is done at a mask level rather than the pixel level. This is done to decrease the pixel level porosity in the resultant worker blobs.

The next step after the extraction of the workers is to track their activities along the successive frames. A match matrix method is a widely used technique in which different parameters (usually Euclidean or Manhattan distances) decide whether the current blob (to be tracked) matches with an existing track. To handle dense and complex tracking situations like a busy traffic intersection, more intelligent tracking algorithms like Kalman filters[29] and extended Kalman filters [30] and condensation algorithms[31] have been used. For our research, we also employ the match matrix in which the Euclidean distance (in x and y) of the current blob and the next predicted positions of the tracks is used as a measure of match between them. The track that has the highest match grabs the blob. These predicted next positions of the tracks are computed based on the previous position, speed and direction of the track, similar to Narayana et.al [32].

The next task is to extract poses from the constituent blobs of these tracks and estimate their productivities. The poses are extracted from these blobs as skeletons. Various image skeletonization algorithms are available in the literature such as contour based [39] and parallel processing based [33] and [40]. Comparisons of different image thinning algorithms are also presented [35], [36] and [37]. Our system uses the algorithm developed by Zhang T.Y et.al [40]. This algorithm promises a connectivity preserving single pixel thick skeleton at a great speed. These poses are then analyzed by employing neural networks and support vector machines, which will be detailed in the subsequent chapters.

3 Methodology

The human poses based productivity determination method involves two phases: (1) pose extraction and (2) productivity determination. In the pose extraction phase, image-processing algorithms are employed to identify the workers, assign each of them a unique ID and extract their corresponding poses. Then, neural network based algorithms, which the authors developed, are used to analyze these poses and hence determine the efficiency of the workers. A work flow diagram for our method is shown in Figure 1 and its constituent blocks are detailed in this section.

3.1 Data Acquisition

The authors acquire data as a series of images of the construction activities using the **w**ireless **r**eal-t**i**me produc**t**ivity m**e**asurement (WRITE) system, developed at the University of Kansas by Bai, Y et.al [38]. This system consists of a video camera mounted over a stable tripod at the construction site. It captures the images of the activities and transmits them via the internet. This system allows remotely controlling the camera's focus by adjusting its pan, tilt and zoom settings. The work flow diagram of the WRITE system is shown in Figure 2 [38] along with a typical on-site mounted video camera in Figure 3.

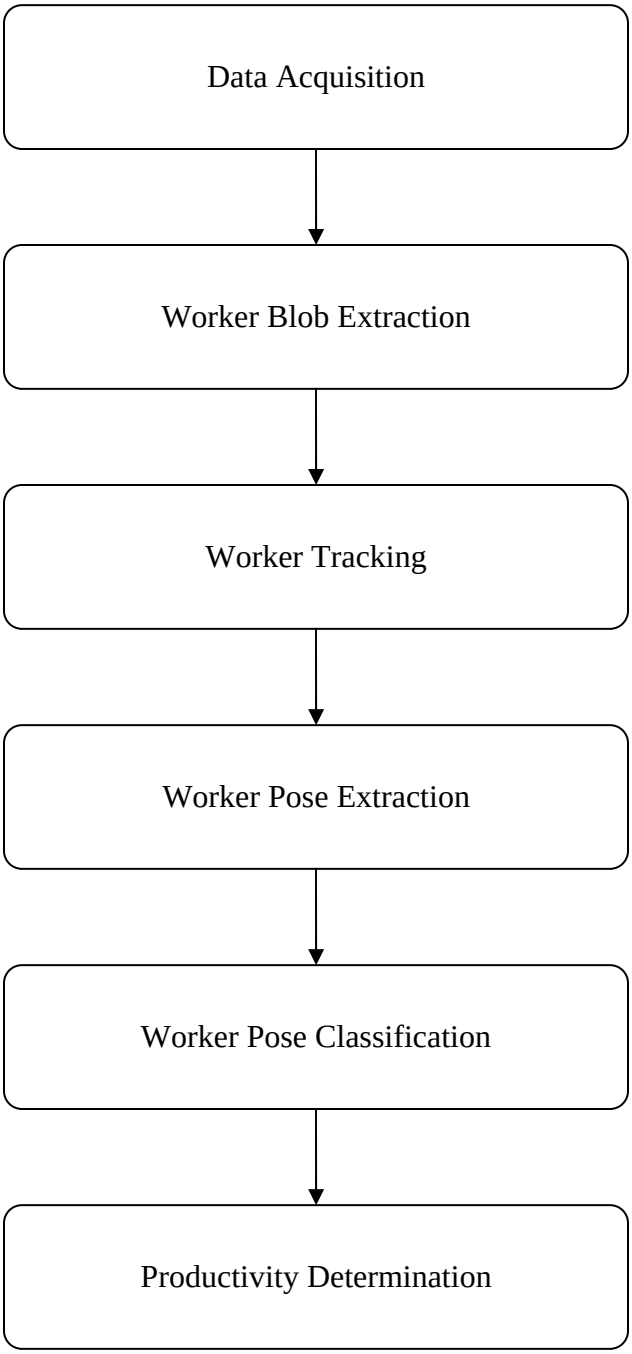


Figure 1 . Work flow diagram of the human pose based automated and real time productivity determination system.

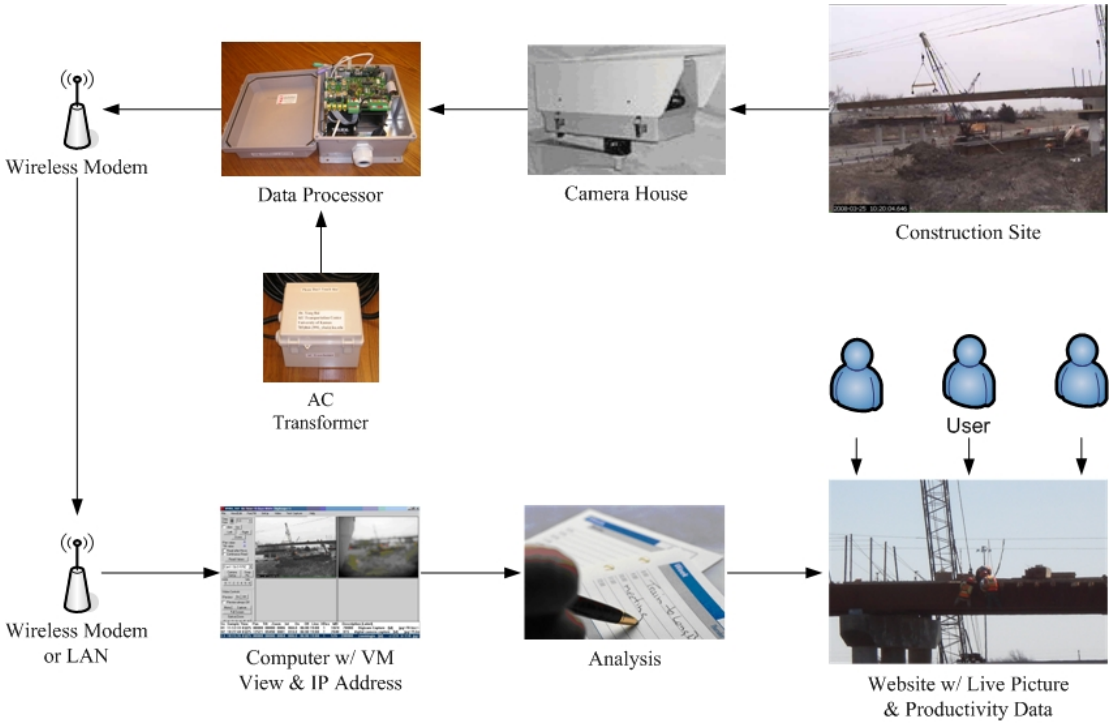


Figure 2. Work flow diagram of the WRITE System

Figure 3. An on-site mounted video camera

3.2 Data Pre-Processing

In this block, the incoming images are processed to identify and extract the workers in them. The author's assumption in worker identification is that the workers are in motion when performing an activity. Hence, a motion segmentation algorithm similar to Hong-Wen, Shao-Qing et al. [16] is developed to identify all the moving objects in these images. However, as this method captures several other uninterested moving objects such as swaying trees, birds etc, an algorithm to filter the workers from this set of moving objects is also developed.

3.2.1 Motion Segmentation

If moving objects are of interest in an image, then all the other regions which are static are collectively termed as "background". A moving average model is used to compute such background.

Definition 1. *If $I^N(x,y)$ be the intensity of the pixel at location (x,y) on N^{th} frame, then the intensity of the pixel on its background image for $BG^{N^{th}}$ at the same location (x,y), computed over K frames is defined as*

$$I^{BG}(x,y) = 1/K\left(\sum_{i=N-K/2}^{i=N+K/2} I^i(x,y)\right)$$

Once the background is computed, it is subtracted from the current frame (N^{th}) to extract the workers. Most methods focus on a simple pixel to pixel subtraction holding a fixed threshold, which is good for noiseless environments. But in this case where the construction activity is

mostly outdoors and as the camera is mounted at an elevation, a jitter exists in its focus due to wind. To overcome these inevitable sources of noise, a mask level background subtraction is developed.

Definition 2. *If* I^{N}_{avg} *and* I^{BG}_{avg} *denote the average of pixel intensities of the mask M (typically 5) in the* N^{th} *frame and its background respectively, then the intensity of a pixel* I^{SG} *of the segmented image SG, over the same mask is defined as*

$$I^{SG}(x-i, y-i) = 1; \text{ if } 1/M \left| I^{N}_{avg} - I^{BG}_{avg} \right| > T$$

$$= 0; \text{ otherwise}$$

$$\text{where } I^{N}_{avg} = \sum_{j=-M/2}^{j=M/2} I^{N}(x-j, y-j)$$

$$I^{BG}_{avg} = \sum_{j=-M/2}^{j=M/2} I^{BG}(x-j, y-j)$$

$i = [-M/2, M/2]$ and T is a predetermined threshold, typically 25.

3.2.2 Filtration

Once the moving objects are segmented, the uninteresting objects such as swaying trees and flying birds etc are filtered by employing pattern-matching. The authors assumption in filtration is that every worker in the crew wears a uniform that can be identified. The pattern of the uniform is determined and a match is run over all the objects, considering a threshold T_V (typically 40). The pattern of an object is modeled as its statistical color variance.

Definition 3. *If V denotes the variance over a mask M_V (typically 7)in the segmented image and $V_{S\tan dard}$ denotes the standard variance, then the intensity of a pixel I^F of the filtered image F, over the same mask is defined as*

I^F *(x-i, y-i) = 1; if* $(1/M_V)\left|V - V_{S\tan dard}\right| \le T_V$

= 0; otherwise

where V = $\sum_{k=-M_V/2}^{k=M_V/2}\left(I^{SG}(x-k, y-k) - 1/M \sum_{j=-M/2}^{j=M/2} I(x-j, y-j)\right)^2$

$i = \left[-M_V/2, M_V/2\right]$

3.2.3 Silhouette Creation

The next step after noise removal is to characterize the workers and assign them a specific identity. The blob of each worker is identified and a silhouette is created by clustering pixels in the blob with region growing technique [41], where similar pixels are grouped together to form a single region and are given a distinct color. Certain attributes for these silhouettes such as centroid location, area, the minimum and maximum along both the axes are computed and store dynamically in a database. The data structures of a blob are shown in Table 1. An input image and its corresponding silhouette are shown below in Figure 4(a) and 4(b).

Table 1. Data structures of a blob

Variable	Description
ID	the ID of the blob
R	The red component of the blob's color
G	The green component of the blob's color
B	The blue component of the blob's color
locY	The Y component of the blob's center (centroid)
locX	The X component of the blob's center (centroid)
Xmin	The minimum X present in the blob (boundary)
YofXmin	The corresponding Y component of Xmin
Xmax	The maximum X present in the blob
YofXmax	The corresponding Y component of Xmax
Ymin	The minimum Y present in the blob
XofYmin	The corresponding X component of Ymin
Ymax	The maximum Y present in the blob
XofYmax	The corresponding X component of Ymax
size	The area of the blob in # pixels

4 (a)

4(b)

Figure 4. (a) An input Image; (b) Silhouette in white (For worker pointed in (a))

3.3 Worker Tracking

In the tracking terminology, the silhouettes are often referred to as blobs or objects and images are often referred to as frames. Worker tracking is a process of finding the correspondences between a worker's blob in the previous frame and his/her blob in the current frame. This task is straight forward and effortless with perfect motion segmentation as there is always a one to one correspondence between the blobs in two successive frames. However, in reality, there is a lot of uncertainty involved in the segmentation due to blob occlusion and failure to detect small yet valid blobs [32]. Our goal is to develop a tracking application that is robust to these factors of uncertainty.

3.3.1 Tracking Approach

In the first frame, a new track is generated for each blob and is declared as *new*. In the subsequent frames, the matching blobs for each track in the previous frame are searched and if found, the tracks are updated. If a track in the previous frame could not be updated, then it is declared as *lost*. The tracker does not delete the track but waits for some frames in hopes to finding a matching blob and *regain* the track. If a track cannot be *regain*ed for LL (the lost limit) consecutive frames, it is then deleted. If a blob in the new frame cannot be matched to a track in the previous frame, then a new track is generated for it and is declared as *new*. So, if a track is lost due to blob occlusion, there is a scope to regain it in the subsequent frames.

3.3.2 Matching Criteria

An effortless criterion for matching a blob in the current frame to the tracks in the previous frames is by using the distance based match matrix [32]. The track with the minimum distance is updated with this blob if its distance $d \leq D$, where D is the global distance threshold. This criterion uses a global value for the distance threshold and can be applied if all workers (blobs) move at the same speed. However, in reality, workers move at different speeds and along different directions and so this global threshold cannot be applied. To handle this uncertainty, a modified tracking method is designed based on the same distance based match matrix which predicts the next position of a track based on its current speed and direction. The matrix is populated with the distances of the blobs to the new predicted positions of the tracks in the previous frame. A track with the minimum distance to a blob $d \leq D$ is updated with that blob. This method uses a global value for distance threshold, but it also considers the different motion

speeds of the workers. Thus, this method becomes a reasonable tracking approach. The data structures used to represent a track are shown in Table 2.

The tracking results by the normal method and our modified method are shown in Figures 5(a), 5(b), 5(c) and 5(d) and Figures 6(a), 6(b). 6(c) and 6(d).Both these sets of figures present a situation where a worker being tracked with ID: 1 is obstructed from the camera's view by another worker passing by. Figure 5(a) shows the worker being tracked with ID:1. Figures 5(b) and 5(c) show instances where the worker's track is lost and Figure 5(d) shows the same worker being assigned a new track with ID: 3. Whereas, Figure 6(d) shows the worker with his track regained and his ID retained.

Definition 4. *If v is instantaneous velocity of a track along the direction* θ, *then the new predicted position for the track* (x_{new}, y_{new}) *with current position at (x,y) is defined as*

$$x_{new} = x + v * LC * \sin(\theta)$$

$$y_{new} = y + v * LC * \cos(\theta)$$

where $v \Rightarrow$ *worker's instantaneous velocity* $= \sqrt[2]{(x_{new} - x)^2 + (y_{new} - y)^2}$

$\theta \Rightarrow$ *direction* $= \tan^{-1}\left((y_{new} - y)/(x_{new} - x)\right)$

$LC \Rightarrow$ *the number of consecutive frames the track for which the track was lost.*

Table 2. The data structures of a track

Variable	Description
ID	ID of the current track
R	The red color component of the worker's blob
G	The green color component of the worker's blob
B	The blue color component of the worker's blob
life	The number of frames this track lived for
lostCount	The number of frames for which the track is lost
velocity	The instantaneous velocity of the track
direction	The instantaneous direction of the track
nextX	The X component of the next predicted position
nextY	The Y component of the next predicted position
lost	The track is lost
startFrame	The frame at which the track first appeared
endFrame	The latest frame at which the track ended
dead	The track is dead

5(a)

5(b)

5(c)

5(d)

Figure 5. (a) Worker 1 being tracked; (b) and (c) Worker 1's track lost; (d) Worker 1's new track with ID 3

6(a)

6(b)

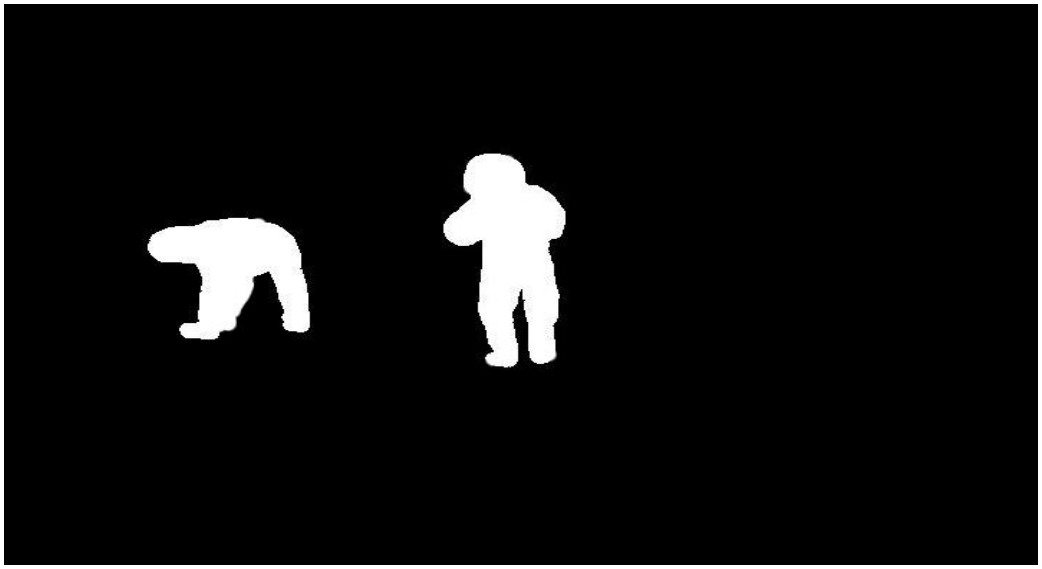
6(c)

6(d)

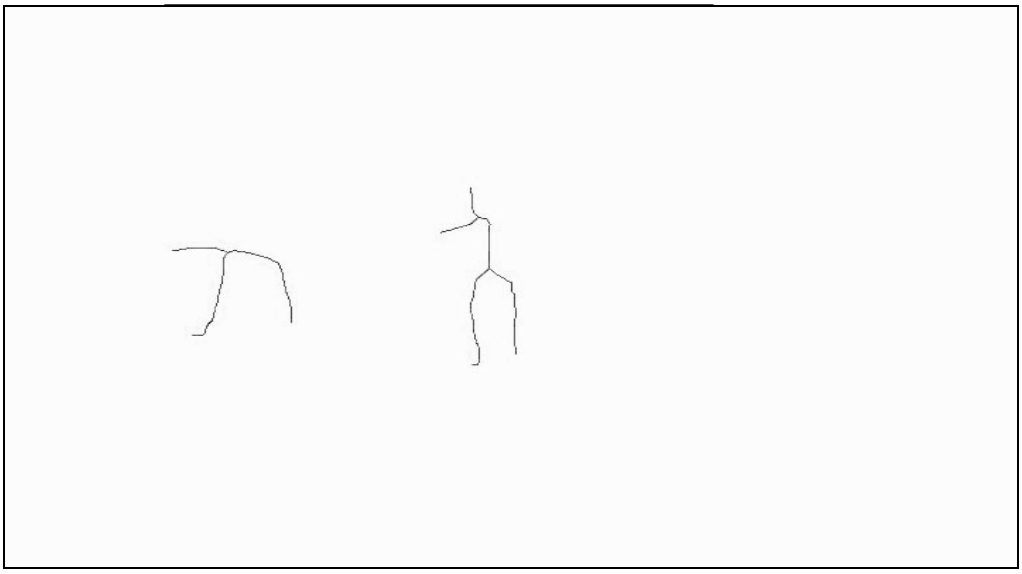
Figure 6. (a) Worker 1 being tracked; (b) and (c) Worker 1's track lost; (d) Worker 1's track regained

3.4 Worker Pose Extraction

The general problem of any pattern recognition algorithm lies in the efficiency of extracting the distinctive features from the patterns to be analyzed. The worker poses are extracted in every image by skeletonizing their corresponding blobs with the help of a fast parallel algorithm for image blob thinning developed by Zhang and Suen [40]. According to them, the algorithm involves two sub iterations; one for deleting the north-west corner points and south-east boundary points and the other for deleting the south-east corner points and north-west boundary points. Each blob is thinned to form a *skeleton* of one pixel thick. The end points and the pixel connectivity are preserved so that a continuous skeleton is obtained and also the distortion is minimal. Figure 7(a) and Figure 7(b) respectively show a silhouettes image and its corresponding skeletons obtained by this algorithm.



7(a)



7(b)

Figure 7. (a)A silhouette image; (b)Skeleton image for (a)

3.5 Worker Pose Classification

In this block, the extracted poses are categorized by performance as effective, ineffective and contributory.

Any motion that is essential for progress and that adds to the completion of a construction activity can be termed as effective work.

Examples. Lifting materials from a location or placing materials at their final locations, being very close to the work area, filling a bucket of concrete etc.

Any motion that does not add to the completion of a construction activity or no motion for a human body for a period of time can be termed as ineffective work.

Examples. waiting, walking empty handed, being very far from work area etc.

Any motion that is essential for progress but does not directly add to the completion of a construction activity can be termed as contributory work.

Examples. clean-up work, erecting structures and poles, receiving instructions from supervisor, loading/unloading a truck etc. This category of work targets on eliminating the gray area between effective and ineffective work.

However, these definitions are not enforced and are open for interpretation and can vary from one project to the other. Figures 8, Figure 9, Figure 10 show effective, ineffective and contributory sequences for the worker pointed by the arrow.

8(a)

8(b)

8(c)

8(d)

8(e)

8(f)

Figure 8. Some effective instances (for the activity of tying a rebar)

9(a)

9(b)

9(c)

9(d)

9(e)

9(f)

Figure 9. Some ineffective instances (for the activity of tying a rebar)

10(a)

10(b)

10(c) 10(d)

10(e) 10(f)

Figure 10. (b), (c), (d) and (e) Some contributory instances

To achieve this categorization of work basing on poses, two Artificial Neural Network (ANN) based algorithms are developed and are detailed in the subsequent sections followed by a brief description of a neural network.

3.5.1 Basics of neural networks

A neural network is an artificial intelligent machine that mimics the functionality of the human brain [42], [44]. It contains a large number of processing elements called “nodes” or “neurons” that grouped in “layers” and linked together by weighted connections called “synapses”. A typical neural network is as shown in Figure 11 below.

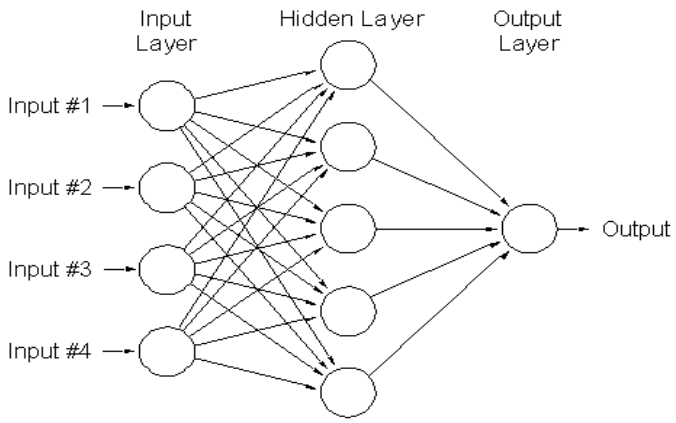


Figure 11. An artificial neural network

A feed forward neural network with a back propagation learning algorithm is used in our research [43]. An artificial neuron is a device with multiple inputs and only one output. A neuron performs two basic functions. It sums up the values at each input multiplied by the weight associated with each interconnection and then generates an output by passing this sum through an activation function *f*. There are several types of activation functions and the function we used in this study is

$$f(y_i) = 1/\left(1 + e^{-y_j}\right)$$

A neural network contains one input layer, one output layer and one or more intermediate processing layers called hidden layers. The output of a neuron (y_i), when a value (x_j) is passed to the input is

$$y_i = \sum_j W_{ji} x_j$$

where W_{ji} = the interconnection weights from neuron *j* to neuron *i*.

These interconnecting weights begin as random and are adjusted when the network undergoes training for a specific application. There are several algorithms to adjust these weights and minimize the training error and one such algorithm is the back-propagation method. In this algorithm, the training of the network begins with the inputs being fed via the input layer. The network output is then computed and compared with the desired output. The resulting error is fed back to the network via the input layer. This process is repeated iteratively until the resultant training error is acceptable or the specified number of iterations has been completed. As the feed is in forward and the error propagates backward, this network is called a feed forward back propagation network.

The performance of a neural network such as training error, speed and prediction accuracy depend upon the number of hidden layers used, the number of iterations and the learning rate, momentum values applied during the iterations.

3.5.2 Categorization Algorithms

Two algorithms are developed to analyze human poses into the above defined classes. These algorithms employ neural networks to classify these poses, one at a time. The employment of neural networks puts forth two work phases for these algorithms: learning phase and execution phase. During the learning phase, training data sets are manually annotated and are used to train these networks using the back propagation learning algorithm. During the execution phase, these trained neural networks are utilized to analyze and classify similar poses. The productivity is determined from the classification results. For reliable performance in both these work phases, it is necessary that the input data, i.e., the poses possess a generality in their

features. As the desired feature from these poses (images), is the shape of the skeleton alone, all other features/attributes like color and image size have to be fixed. The skeletonization process in the previous section provides a feature generality in color by coloring the skeleton to white (255) and the background to black (0). However, there exists poses of different sizes due to two reasons.

- No two workers are identical in shape and size.
- The orientation of the workers with respect to the camera and the depths of their positions from the camera are different. Workers facing the camera have broader poses when compared to those facing 90 degrees from the camera. Similarly, workers away from the camera have smaller poses when compared to those close to the camera.

Hence to obtain consistency in the sizes of these poses, an image scaling is done. All the poses are scaled to a fixed size of WxH irrespective of their original sizes, where W and H are the width and height of the scaled image respectively.

3.5.2.1 Algorithm for Pose Classification (Instance Algorithm)

Once the scaled poses are obtained, the next task is to analyze and classify them. This algorithm undergoes two work phases: learning phase and execution phase. In the learning phase, training data sets are manually annotated such that each pose is assigned to one of the two classes: effective and ineffective. This annotated training data is used to train the neural network using the back propagation learning algorithm.

In the execution phase, this algorithm communicates with the tracker and requests the I.D of the worker associated with each incoming pose. Using this ID, it searches for the worker's

record and creates a new one if it could not find any. It then updates this record with the results of the pose classifications and hence determines the worker's real-time productivity.

Classification Criteria

The classification is performed at an instance level, which means that the poses at every instant are analyzed individually. As poses contains information only about their effectiveness or ineffectiveness, this algorithm classifies them accordingly to those classes, relative to the current activity. During the classification process, this algorithm updates the records of the workers with the total number of effective and ineffective instance of his/her corresponding poses. The real-time productivity of the worker W is $P_i(W)$ and is computed as

$$P_i(W) = \frac{I_{effective}(W)}{\left(I_{effective}(W) + I_{ineffective}(W)\right)}$$

Where, i = suffix to denote the instance algorithm

$I_{effective}(W)$ = number of effective instances of the worker W

$I_{ineffective}(W)$ = number of ineffective instances of the worker W

Results of Instance Algorithm

Some classification results of the instance algorithm are shown below. The worker 1 (with ID 1) is chosen for demonstration as indicated in Figure 12 (a) and his productivity is being determined. A series of six sequential images containing the worker are shown in Figure 12, along with the classification by the instance algorithm. A green square near the worker's head portion denotes the classification of his current pose as effective and a red square denotes the classification of his current pose as ineffective. A cross (X) on these colored squares

indicates an incorrect classification. The bar toward the extreme top-right corner of the image shows the real-time productivity of the worker, relative to his start of the activity.

It is observed from these figures that the pose in 12(a) is incorrectly classified as ineffective and hence the real-time productivity is 0/(0+1) = 0%, as shown by the productivity bar. In figure 12(a), the pose was correctly classified as working and hence, his real-time productivity rose to 50 percent. In the next frame, it further rose to 66 percent as shown in figure 12(c). In the figures 12(d), 12(e) and 12(f), it is observed that poses were correctly classified as ineffective and the productivity varied accordingly. Also, it is observed that the productivity bar in these figures shows a constant value of 83 percent as the display has a precision to zero decimals. So the actual value of 83.12, 83.07 and 83.11 are shown as 83 only.

Hence, the instance algorithm effectively classifies the worker poses into effective and ineffective classes.

12(a)

12(b)

12(c)

12(d)

12(e)

12(f)

Figure 12. Sample Output images

Short Coming of the Instance Algorithm

The instance algorithm effectively classifies the poses as effective and ineffective. But, it suffers from a serious shortcoming of not considering the gray area between effective and ineffective classes i.e., the contributory class. Due to this reason, the instance algorithm treats the contributory work poses as ineffective poses inducing error and thus hindering the productivity. Figure 13 illustrates these situations where poses belonging to contributory work is treated as ineffective. Figures 13(a), 13(b), 13(c), 13(d) and 13(e) contain contributory poses but are detected as ineffective.

13(a)

13(b)

13(c)

13(d)

13(e)

13(f)

Figure 13. Failure to detect the contributory frames by the Instant algorithm

Since by definition, the contributory class of work does not decrease the productivity, the only class that does so is the ineffective class. Instance algorithm failed to clearly distinguish the in-effective class from the contributory class. The activity algorithm was developed to solve this problem by detecting the contributory class and thereby enhance the accuracy in productivity determination.

3.5.2.2 Activity Algorithm

3.5.2.2.1 Goal

The goal of this algorithm is to include contributory class in the classification of poses and hence enhance the accuracy in productivity determination. This goal exposes two challenges:

- Firstly, different activities interpret contributory work in different ways, and prior knowledge of the activities being progressed is required.
- Secondly, poses that represent contributory work in one instance may represent ineffective work in another. For example, the poses obtained from a worker who was standing and having a smoke are identical to the poses obtained from a worker who was walking to fetch some tools. The first set of poses represents ineffective work, whereas the second set of poses represents contributory work. Thus, a clear distinguishing of these situations is needed. We need to define metrics to identify these situations in a specific activity.

Metrics to identify contributory work class

Identification of contributory work class in an activity is a complex task as it is situation/context dependent and it is necessary to distinguish between two different contributory tasks. For example, a worker receiving instructions from the supervisor or a worker and a worker loading a truck are both contributory works but in different ways/situations. Hence, defining a global metric to identify contributory work is not possible. The different types of contributory work involved in an activity are to be manually studied and suitable metrics have to be chosen to identify them by the automated system. In this research, we analyze the phase of tying a rebar in a bridge construction. It involves an activity of tying the rebars at different locations in the work

scene. The worker needs to bend for tying the rebar and then walk to move to the next location. By the instance algorithm, the phase of bending is considered effective and the phase of walking is considered ineffective. But a closer view at this activity as a whole indicates that walking is essential for moving to the next position and hence is actually a contributive work. So, to identify this contributory work, we define a metric that it should begin immediately after an effective work. We define another metric of time, to define the end of this contributory work. In other words, if a worker walks for more than the allowed duration, then the extra time is considered ineffective. Thus, all the three classes of work have been defined for this activity.

As with the instance algorithm, the activity algorithm also undergoes two work phases: learning phase and execution phase. In the learning phase, training data sets are manually annotated such that each pose is assigned to one of the three classes: effective, ineffective and contributory. This annotated training data is used to train the neural network using the back propagation learning algorithm.

In the execution phase, this algorithm communicates with the tracker and requests the I.D of the worker associated with each incoming pose. Using this ID, it searches for the worker's record and creates a new one if it could not find any. It then updates this record with the results of the pose classifications and hence determines the worker's real-time productivity. The activity algorithm is developed as an extension of the instance algorithm by including another block called activity analyzer and it is as shown in Figure 14.

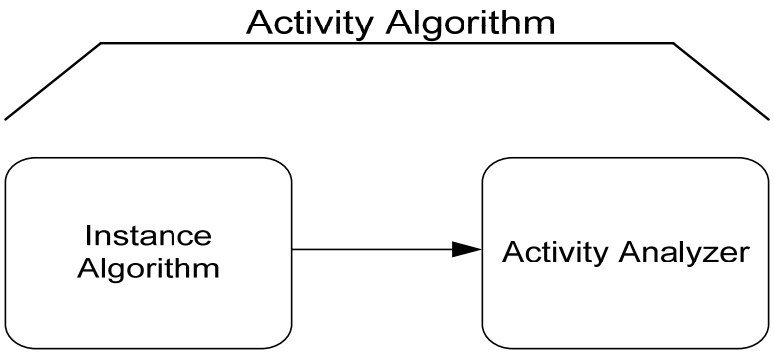


Figure 14. Activity algorithm as an extension of instance algorithm

As from the figure, the instance algorithm classifies the incoming poses as effective or ineffective and then passes the classification to the activity analyzer. The activity analyzers designed for this activity are detailed in the following section.

Productivity Determination with Activity Algorithm

In this research we assumed that contributory work also adds to the productivity and hence, its class is united with the effective work class for productivity determination. In other words, a positive is a pose which is either an effective or contributive and a negative is a pose which is ineffective. The activity algorithm contributes to the enhancement of productivity determination in two ways.

- By identifying the contributory work, which adds to the measured productivity.
- By correcting some of the incorrect classifications from the instance algorithm.

The first contribution is already explained by our goal. To explain the second contribution of correcting some ineffective classifications, it is to be recalled that the activity algorithm runs on the classifications provided by the instance algorithm and that the classification accuracy for the instance algorithm is $A_{instance} \leq 100\%$ resulting some incorrect classifications. These may include

classifying some effective instances as ineffective. If such incorrectly classified poses satisfy the metrics to detect the contributory work, then they are also classified as contributory work. Since we already assumed a union for contributory and effective classes, this classification/correction furthers the productivity level and making the process of productivity determination more accurate. The real-time productivity of a worker W is $P_a(W)$ and is computed as

$$P_a(W) = \frac{\left(I_{effective}(W) + I_{contributive}(W)\right)}{\left(I_{effective}(W) + I_{contributive}(W) + I_{ineffective}(W)\right)}$$

Where, i = suffix to denote the instance algorithm

$I_{effective}(W)$ = number of effective instances of the worker W

$I_{contributive}(W)$ = number of contributive instances of the worker W

$I_{ineffective}(W)$ = number of ineffective instances of the worker W

Activity Analyzer

All the functionality for the activity algorithm is implemented in this analyzer. Two different activity analyzers were developed to analyze the activity described above.

- Activity analyzer by ratio comparison.
- Activity analyzer by using a SVM (support vector machine).

Activity analyzer by ratio comparison

The instance algorithm produces a series of poses classified as effective and ineffective. For easy understanding, let us use the word *positive* to denote a pose classified as effective,

negative to denote a pose classified as ineffective and *contributive* to denote a pose classified as contributory. To detect if a given pose is *contributive,* we use the classifications of the worker's poses over a window *W* of length L, with *(L-1)* poses after the current pose. A window is called *positive window* if the number of positives in that window is greater than *N where (N ≤ L-1)* and the first pose in that window is a positive. A window is called *negative window* if the number of negatives is greater than *N* i.e., the number of positives is less than *(L-N).* Similarly, a window is called a *contributory window* if it is a negative window and immediately occurs after a positive window.

For every incoming pose p_i, a window of L poses is loaded with poses p_i to p_{i+L-1}. We begin our algorithm by scanning these windows in search of a positive window. Once a positive window is found, the first pose of a this window is considered as the start of a new activity and end of a previous activity, if any. We then begin our search for a negative window. If a negative window is found immediately after a positive one, then that window is a *contributive window.* This transition from a positive window to a negative window denotes the transition from effective to contributive work.

The first pose in the contributory window is marked as contributive and with this pose as start, a total of *t-1* poses are individually scanned and are marked as contributive if they were negative. After the *t* incoming poses are scanned, all the occurring negatives are discarded until a positive window is found. Once a positive window is found, the first pose of this window is treated as the end of the current activity and the start of the new activity. The algorithmic representation for this activity analyzer is detailed below.

List of Symbols

$W^{'}$ //A sequence of length L

C_L //The contributory limit

$C_C \leftarrow 0$ //The contributory count

N //The ratio threshold

$F_{PW} \leftarrow false$ // A flag denoting the occurrence of a positive window

$F_{NW} \leftarrow false$ // A flag denoting the occurrence of a negative window

Function *ActivityAnalyzerbyRatioComparison*(W)

Input: The sequence of classifications from the instance algorithm, *W*.

Output: The modified sequence *W*.

for $\mathrm{i} = \frac{L}{2} + 1, \ |W| - \frac{L}{2}$

$W^{'} \leftarrow W\left[i - \frac{L}{2}, i + \frac{L}{2} \right]$ //populate the sequence W^1 from W

if $\sum_{j=1}^{|W^{'}|} W_j^{'} \geq N$ and $W_1^{'}$ = positive $F_{PW} \leftarrow true$ endif

if $F_{PW} = true$ and $\sum_{j=1}^{|W^{'}|} W_j^{'} \leq \left(|W^{'}| - N \right)$ $F_{NW} \leftarrow true$ endif

if $F_{PW} = true$ and $F_{NW} = true$

if $C_C < C_L$ $C_C \leftarrow C_C + 1$ $W_1^{'} \leftarrow contributive$

else $F_{PW} \leftarrow false$ and $F_{NW} \leftarrow false$ endif

endif

end *ActivityAnalyzerbyRatioComparison*

// Algorithm for the activity analyzer using ratio comparison

Activity analyzer by using a SVM

An activity analyzer using a support vector machine (SVM) is developed. A sequence W^1 of L poses is treated as a feature for the SVM. For every pose, its corresponding feature is extracted as a sequence W^1 of length L containing the its classification followed by (L-1) poses of the same worker. The SVM is trained over a set of these features and is then applied to classify the incoming real-time poses. A brief overview of the concepts of support vector machines (SVM's) is given in the following section.

3.5.2.2.1.1 Basics of support vector machines

Support vector machines are a new class of supervised learning models used for pattern recognition, classification and regression [45]. They belong to the family of linear classifiers. The data sets used to train the support vector machines are termed as *support vectors* and they compute solutions in terms of these support vectors. A training set can be defined as

$$\Theta = \{(x_1, c_1), (x_2, c_2), (x_3, c_3), ..., (x_N, c_N)\}$$

Where $i \in \{1, N\}$

$c_i \in \{1, -1\}$

$x_i \in R^N$ is an N-dimensional real-vector.

A hyper plane g(x) can be written as a set of points in the training data such that

$g(x) = w.x - b$, where w is a vector normal to the hyper plane.

The offset of the hyper plane g(x) from the origin along the normal w is $\frac{b}{|w|}$, where $|w|$ is the magnitude of w. The optimization problem is to choose values of b and $|w|$ such that the distance between two parallel hyper planes is the maximum.

If the two hyper planes are $w.x - b = 1$ and $w.x - b = -1$, then the distance between these two hyper planes is $\frac{2b}{|w|}$ can be geometrically computed if the training data is linearly separable. The value of $|w|$ should be minimized to maximize this distance. To prevent the data points from falling into the margins between the two hyper planes, a constraint k_i is added to the equation, making the optimization problem as solving

$\min_{b,w} \frac{1}{2}|w|^2$, such that $k_i(w.x_i - b) \geq 1$ for $1 \leq i \leq N$

3.5.2.2.1.2 Analysis Criterion

The previous analyzer used the ration metric to detect positive and negative windows. The value of the ratio threshold (N) needs to be manually decided by the user and fed during its execution. A deviation in this value would produce an erroneous classification making it a vital factor in governing the performance of the classifier. To avoid the need for such human intervention, we developed this SVM based analyzer which automatically analyzes the window based on its previous knowledge. The employment of a support vector machine puts forth two work phases for this analyzer: learning phase and execution phase. During the learning phase, training data sets are manually annotated and are used to train this machine. During the execution phase, this trained machine is utilized to analyze and classify similar poses.

For every incoming pose p_i, a window of L poses is loaded with poses p_i to p_{i+L-1}. We begin our algorithm by scanning these windows in search of a positive window. Once a positive window is found, the first pose of a this window is considered as the start of a new activity and end of a previous activity, if any. We then begin our search for a negative window. If a negative window is found immediately after a positive one, then that window is a *contributive window.* This transition from a positive window to a negative window denotes the transition from effective to contributive work.

The first pose in the contributory window is marked as contributive and with this pose as start, a total of *t-1* poses are individually scanned and are marked as contributive if they were negative. After the *t* incoming poses are scanned, all the occurring negatives are discarded until a positive window is found. Once a positive window is found, the first pose of this window is treated as the end of the current activity and the start of the new activity. The algorithmic representation for this activity analyzer is detailed below.

4 Experimental Study

An experimental study of the performance characteristics of the human pose analyzing algorithms was conducted. The ground truth data for both the algorithms was collected by manually annotation and the classification performance of the algorithms was compared to this manual annotation.

4.1 Data Sets

A steel girder bridge reconstruction project was utilized for the study. Construction activities were collected using the WRITE System as a series of color images at a rate of 1 frame per second, a technique similar to the time-lapse filming [6]. These images were of size 720x480 pixels with a resolution of 96 dots per inch. A set of 1,000 such images was selected for analysis. Two workers performing the activity of tying rebar were chosen and were labeled W1 and W2 respectively. The respective poses of these workers were manually classified to generate the ground truth data. The ground truth data for the workers for both algorithms are shown in Table 3.

Table 3. Characteristics of the Data Sets

Worker	Samples	For Algorithm	Total frames as effective	Total frames as ineffective	Total frames as contributive*
W1	1000	Instance	637	363	--
	1000	Activity	637	183	180
W2	1000	Instance	717	283	--
	1000	Activity	717	235	48

* The contributory class does not apply to the Instance Algorithm

4.2 Experimental Protocol

Every incoming image from the video is passed through the video processing blocks to extract the constituent workers. For the motion segmentation, the background model is computed over a window of K=10 and is eliminated using a mask of size M=3 with threshold T=25. The objects are then filtered for pattern-match using a mask size of $M_v = 7$ and threshold $T_v = 40$. These objects are characterized as workers and their centroid values are recorded as their positions in the image. Workers are each assigned a unique ID (i = 1,2,3,…) and they tracked for there new positions in the subsequent images with a search limit N = 30.

All the images related to a single worker are isolated from the image and scaled to a size of 140x280, irrespective of their original sizes. Their corresponding poses are extracted by using the pixel-based image thinning software by Zhang and Suen [40], downloaded at http://www.cs.utexas.edu/users/qr/software/evg-thin.html. To analyze these poses, artificial neural networks were used as a part of JOONE (Java Object Oriented Neural Engine), a free java neural net framework downloaded at http://www.jooneworld.com/. Feed forward neural networks with a back propagation learning algorithm were used in this project and their configuration is listed below in Table 4.

Table 4. Neural Network Configuration.

Input Layers	1
Hidden Layers	1
Output Layers	1
Input Nodes	39200
Hidden Nodes	70
Output Nodes	1
Learning Rate	0.2
Momentum	0.7
Training Cycles	300

The poses obtained for worker W1 were used to train the neural networks. These poses were first manually divided into two groups as effective and ineffective. Training sets of sizes 2,4,6,8….26,28,30 were created by randomly selecting equal number of poses from each group. The neural networks were trained over these training sets. The trained neural networks were used to analyze and classify the poses. The productivity computed from these classification was noted. Testing of these neural networks was performed on the poses of both thw workers: W1 and W2. Since these neural networks were trained on W1 poses, testing it on W1 poses was *straight-testing* and testing it on poses from W2 was *cross-testing*. It is to be noted that worker W2 performed the same activity as worker W1. This whole procedure of training and testing formed a single experimentation cycle.

Straight testing: training on W1 and testing on W1.

Cross-testing: training on W1 and testing on W2.

The above experimentation cycle was repeated for 200 times and the average, 95 percent and 5 percent values of precision, recall and accuracy values were determined for the instance algorithm and the activity algorithm. All the experiments were performed on a desktop computer with a 3 GHz dual-core processor and 2GB of RAM. An overview of the experimental work flow is shown in Figure 18. Some blocks of the original system are also included for reader's ease of understanding.

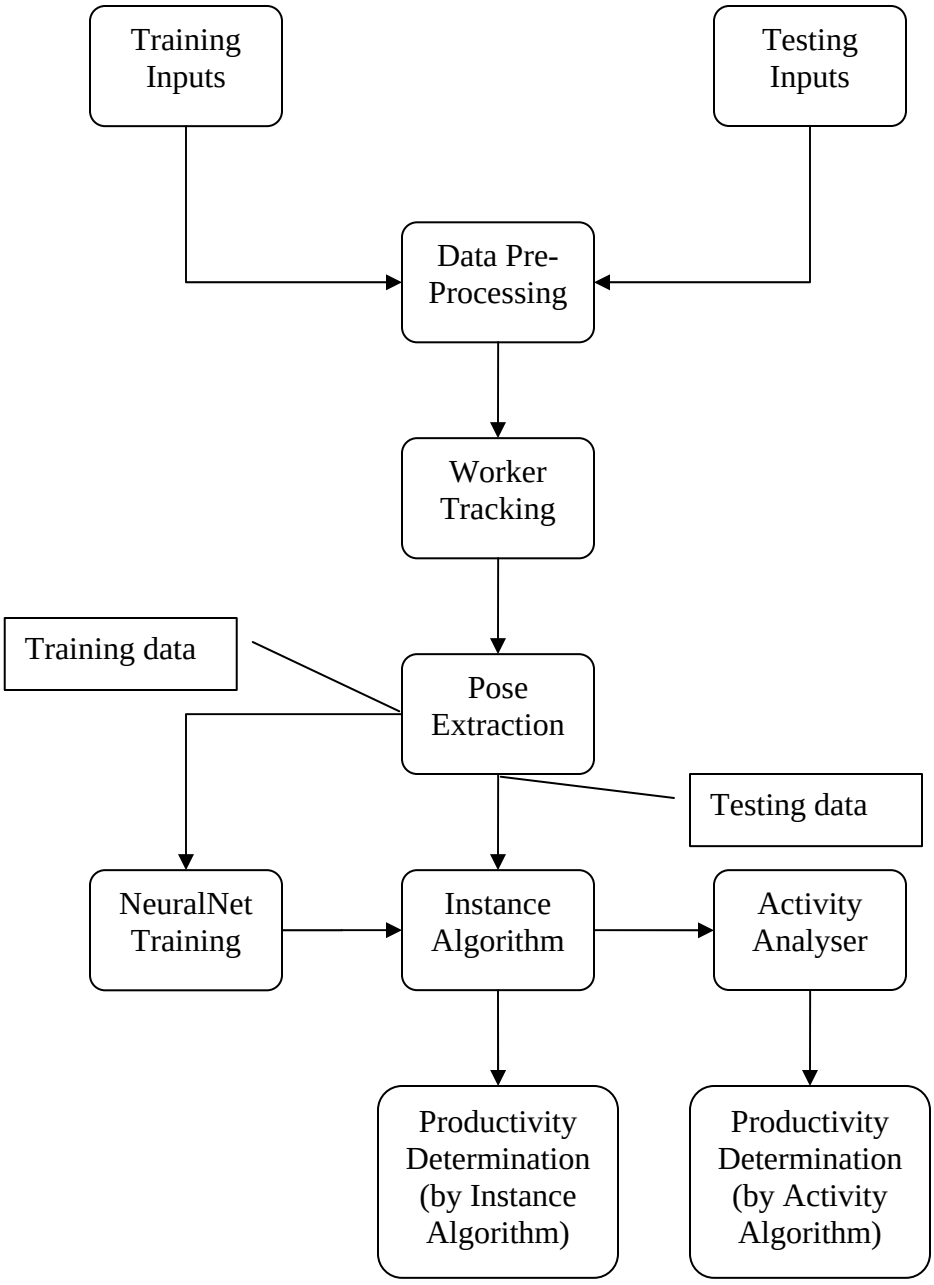


Figure 15. Experimental work flow diagram

4.3 Experimental Results

The performance of the algorithms is gauged by the precision, recall and accuracy that they provide for productivity determination.

Precision is defined as the ratio of true positives to the sum of true positives and false positives [46].

Precision $P = TruePositives/(TruePositives + FalsePositives)$

Recall is defined as the ratio of true positives to the sum of true positives and false negatives [46].

Recall $R = TruePositives/(TruePositives + FalseNegatives)$

Accuracy is defined as the ratio of sum of true positives and true negatives to the total predictions.

Accuracy A = $(TruePositives + TrueNegatives)/Test\_Data\_Size$

A true positive occurs when the algorithms classify a pose as representing effective work, and the pose is indeed representing effective work. A false positive occurs when the algorithms classify the pose as representing effective work, but the pose is actually in the ineffective work status. For the instance algorithm, effective work poses are considered as positives and ineffective work poses are considered as negatives. For the activity algorithm, effective and contributory work poses are considered as positives and ineffective work poses are considered as negatives.

4.3.1 Results from Instance Algorithm

Tests for precision, recall and accuracy were conducted on the instance algorithm and their results were plotted. The graphs for straight testing are shown in Figure 16 (precision),

Figure 17 (recall) and Figure 18 (accuracy) and the corresponding tables for these graphs containing their data points are shown in Table 5, Table 6 and Table 7.

The graphs for cross testing are shown in Figure 19 (precision), Figure 20 (recall) and Figure 21 (accuracy) and the corresponding tables for these graphs are shown in Table 8, Table 9 and Table 10. In each figure, the average, 95 percent and 5 percent curves are shown that are obtained over 200 experiments. The average curve is bolded for ease of viewing to the reader.

4.3.2 Results from Activity Algorithm

Tests for precision, recall and accuracy were conducted on the activity algorithm by employing the two constituent activity analyzers: activity analyzer by ratio comparison, activity analyzer by SVM. The graphs obtained by straight testing of the algorithm employing analyzer by ratio comparison are shown in Figure 22 (precision), Figure 23 (recall) and Figure 24 (accuracy). The corresponding tables for these graphs are shown in Table 11, Table 12 and Table 13. The graphs obtained by cross-testing of the same algorithm are shown in Figure 25 (precision), Figure 26 (recall) and Figure 27 (accuracy) with their corresponding tables shown in Table 14, Table 15 and Table 16.

The graphs for straight-testing on the activity algorithm employing an analyzer by SVM are shown in Figure 28 (precision), Figure 29 (recall) and Figure 30 (accuracy) with their corresponding tables in Table 17, Table 18 and Table 19. The graphs for cross-testing on the same algorithm are shown in Figure 31, Figure 32 and Figure 33 with their corresponding tables in Table 20, Table 21 and Table 22.

In each figure, the average, 95 percent and 5 percent curves are shown that are obtained over 200 experiments. The average curve is bolded for ease of viewing to the reader.

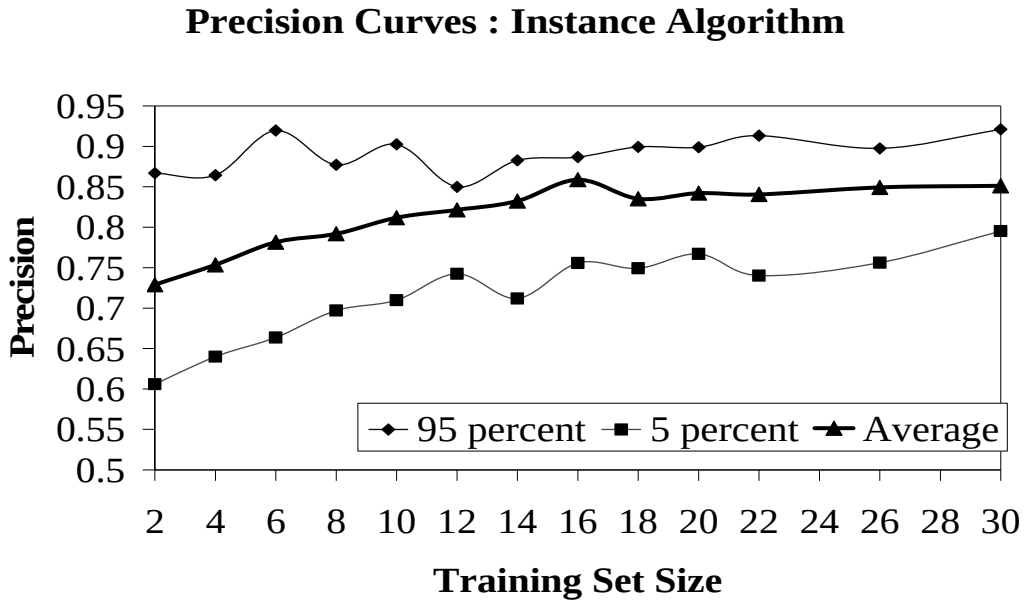


Figure 16. Precision Curves for the Instance Algorithm (Straight-testing)

Table 5. Average Precision, 95 percent of max and 5 percent of max precision for the Instance Algorithm (Straight-testing)

Training Set Size	Average	95 percent	5 percent
2	0.7292	0.8669	0.6061
4	0.7535	0.8644	0.6408
6	0.7814	0.9196	0.6636
8	0.7919	0.8772	0.6971
10	0.8119	0.9025	0.7098
12	0.8215	0.8497	0.7423
14	0.8324	0.8826	0.7118
16	0.8589	0.8867	0.7558
18	0.8353	0.8992	0.7493
20	0.8422	0.8988	0.7674
22	0.8406	0.9132	0.7402
26	0.8491	0.8975	0.7562
30	0.8512	0.9211	0.7952

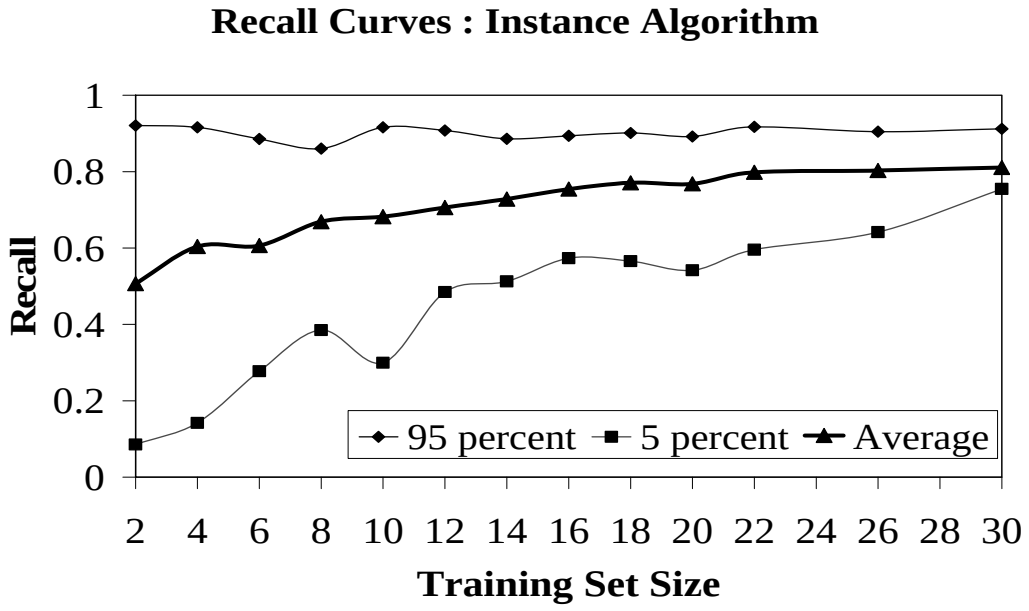


Figure 17. Recall Curves for the Instance Algorithm (Straight-testing)

Table 6. Average recall, 95 percent of max and 5 percent of max recall for the Instance Algorithm (Straight-testing)

Training Set Size	Average	95 percent	5 percent
2	0.5069	0.9207	0.0855
4	0.6038	0.9163	0.1422
6	0.6066	0.8858	0.2773
8	0.6691	0.8605	0.3851
10	0.6823	0.9162	0.2995
12	0.7063	0.9081	0.4849
14	0.7281	0.8859	0.5128
16	0.7544	0.8938	0.5737
18	0.7707	0.9012	0.5657
20	0.7684	0.8922	0.5419
22	0.7984	0.9175	0.5955
26	0.8032	0.9049	0.6418
30	0.8108	0.9122	0.7548

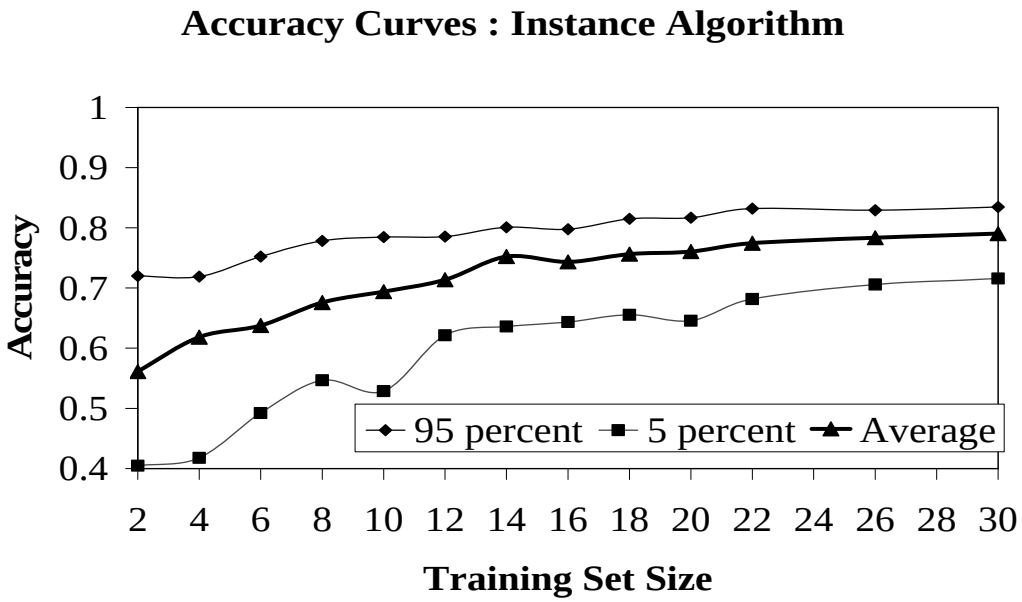


Figure 18. Accuracy Curves for the Instance Algorithm (Straight-testing)

Table 7. Average accuracy, 95 percent of max and 5 percent of max accuracy for the Instance Algorithm (Straight-testing)

Training Set Size	Average	95 percent	5 percent
2	0.5614	0.7200	0.4048
4	0.6183	0.7193	0.4179
6	0.6377	0.7523	0.4922
8	0.6759	0.7784	0.5468
10	0.6942	0.7845	0.5287
12	0.7136	0.7854	0.6213
14	0.7521	0.8008	0.6358
16	0.7433	0.7975	0.6435
18	0.7563	0.8147	0.6556
20	0.7603	0.8167	0.6459
22	0.7744	0.8318	0.6817
26	0.7836	0.8291	0.7058
30	0.7904	0.8345	0.7156

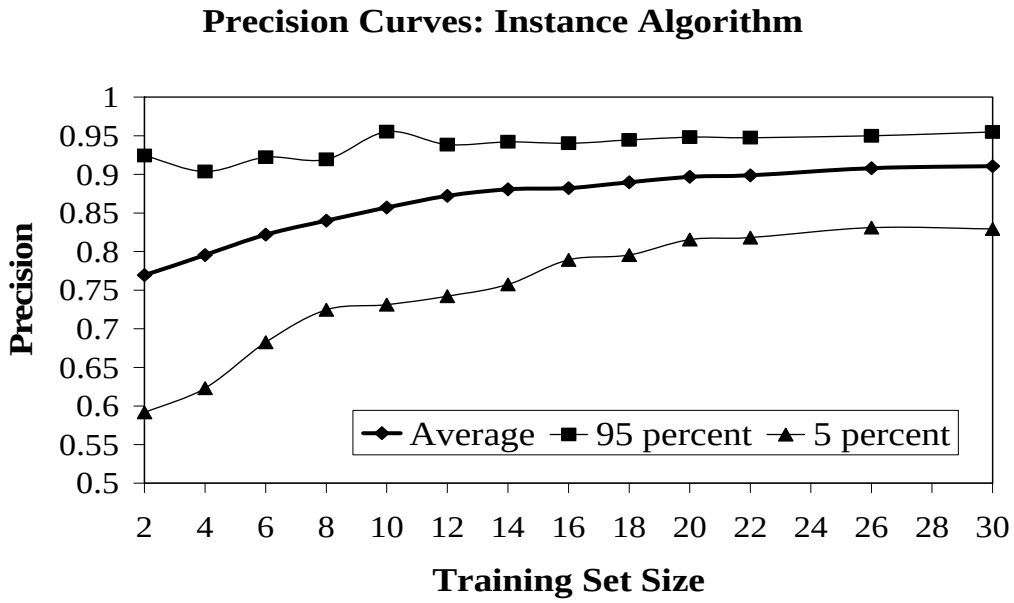


Figure 19. Precision Curves for the Instance Algorithm (Cross-testing)

Table 8. Average Precision, 95 percent value and 5 percent of max precision for the Instance Algorithm (Cross-testing)

Training Set Size	Average	95 percent	5 percent
2	0.7695	0.9243	0.5917
4	0.7954	0.9037	0.6230
6	0.8219	0.9221	0.6826
8	0.8400	0.9192	0.7245
10	0.8570	0.9551	0.7312
12	0.8720	0.9384	0.7421
14	0.8806	0.9420	0.7573
16	0.8820	0.9402	0.7893
18	0.8897	0.9445	0.7955
20	0.8969	0.9480	0.8155
22	0.8987	0.9474	0.8180
26	0.9079	0.9498	0.8309
30	0.9105	0.9548	0.8291

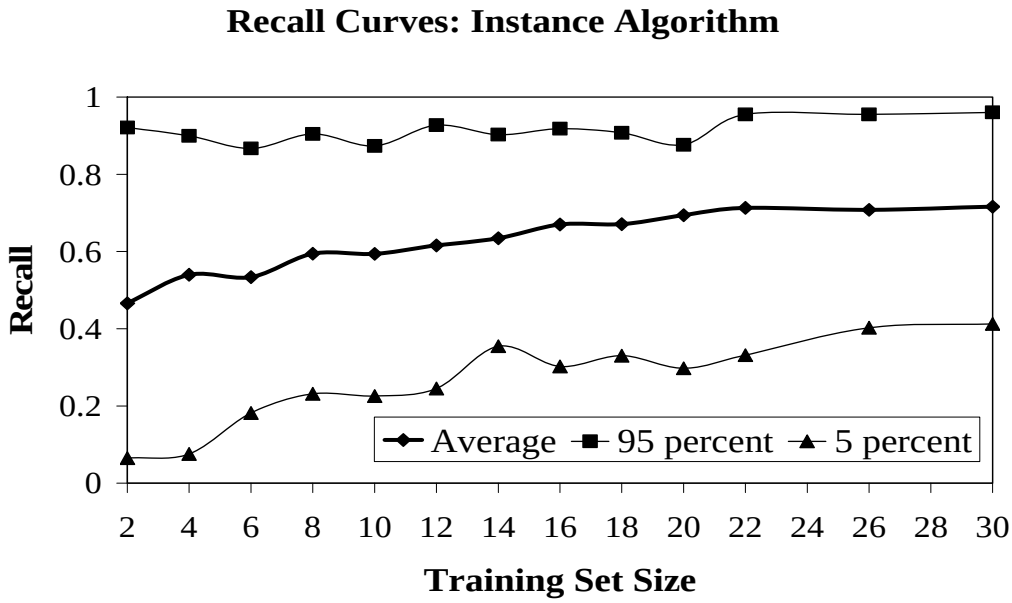


Figure 20. Recall Curves for the Instance Algorithm (Cross-testing)

Table 9. Average Recall, 95 percent value and 5 percent of max recall for the Instance Algorithm (Cross-testing)

Training Set Size	Average	95 percent	5 percent
2	0.4657	0.9212	0.0648
4	0.5398	0.8996	0.0756
6	0.5337	0.8672	0.1820
8	0.5942	0.9043	0.2314
10	0.5940	0.8734	0.2259
12	0.6157	0.9274	0.2453
14	0.6346	0.9027	0.3549
16	0.6699	0.9182	0.3025
18	0.6709	0.9074	0.3302
20	0.6940	0.8765	0.2978
22	0.7132	0.9552	0.3318
26	0.7076	0.9552	0.4028
30	0.7159	0.9603	0.4125

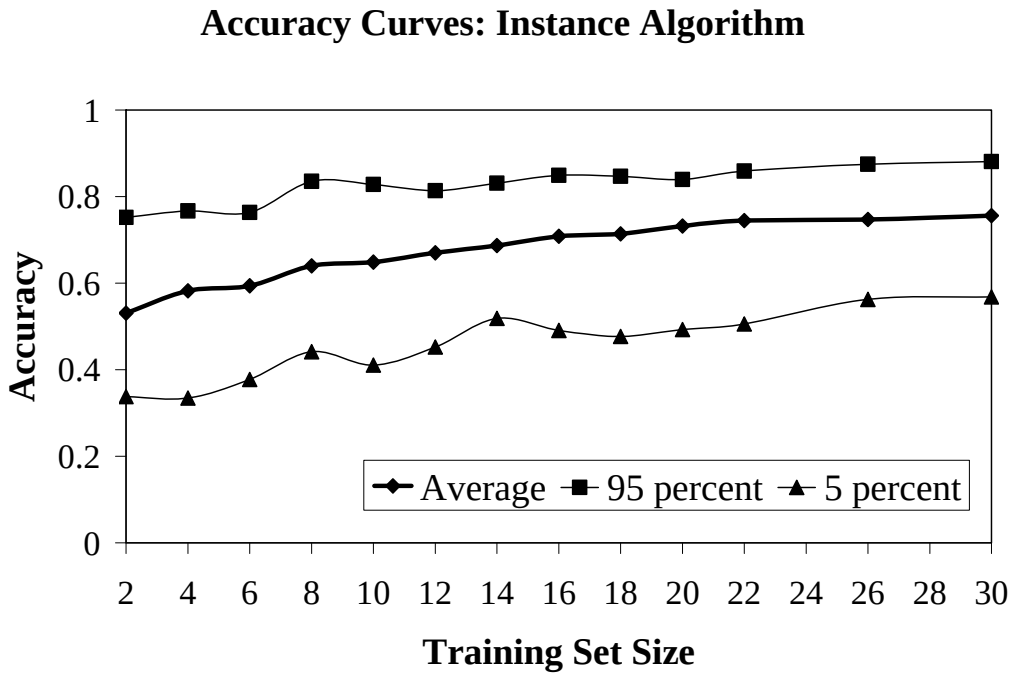


Figure 21. Accuracy Curves for the Instance Algorithm (Cross-testing)

Table 10. Average Accuracy, 95 percent value and 5 percent of max recall for the Instance Algorithm (Cross-testing)

Training Set Size	Average	95 percent	5 percent
2	0.5309	0.7518	0.3379
4	0.5825	0.7668	0.3347
6	0.5941	0.7636	0.3775
8	0.6399	0.8352	0.4417
10	0.6486	0.8278	0.4105
12	0.6701	0.8139	0.4524
14	0.6867	0.8310	0.5187
16	0.7082	0.8491	0.4909
18	0.7137	0.8470	0.4770
20	0.7319	0.8395	0.4930
22	0.7446	0.8588	0.5058
26	0.7470	0.8748	0.5625
30	0.7561	0.8807	0.5679

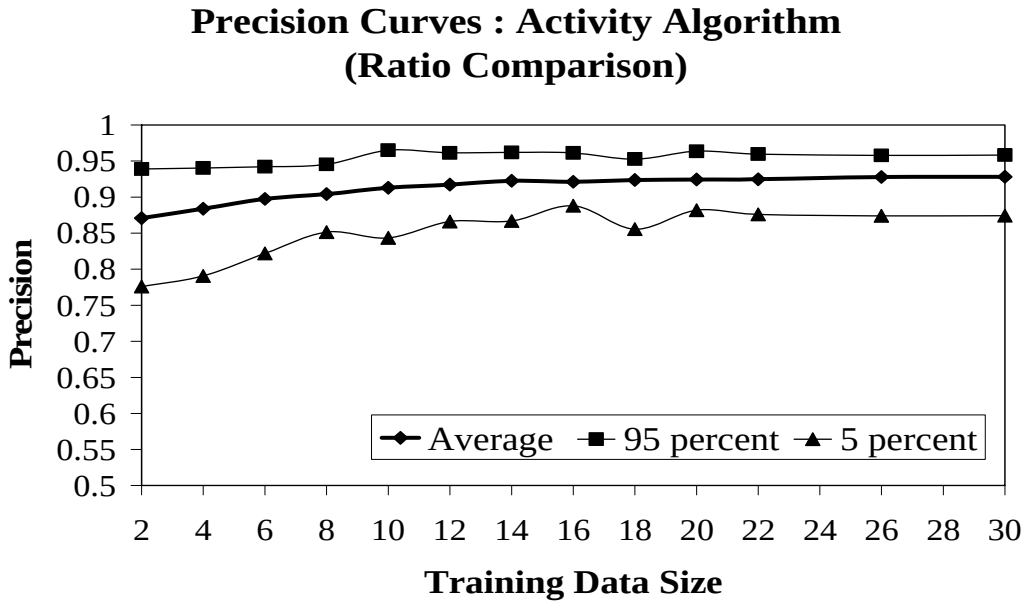


Figure 22. Precision Curves for the Activity Algorithm using analyzer by ratio comparison (Straight-testing)

Table 11. Average Precision, 95 percent value and 5 percent of max precision for the Activity Algorithm using analyzer by ratio comparison (Straight-testing)

Training Set Size	Average	95 percent	5 percent
2	0.8707	0.9389	0.776
4	0.8839	0.9403	0.7907
6	0.8976	0.9421	0.8225
8	0.9041	0.9454	0.8515
10	0.9129	0.9650	0.8434
12	0.9173	0.9614	0.8663
14	0.9226	0.9621	0.8669
16	0.9211	0.9613	0.8881
18	0.9238	0.9529	0.8557
20	0.9244	0.9636	0.8819
22	0.9245	0.9597	0.8761
26	0.9278	0.9579	0.8741
30	0.9280	0.9583	0.8743

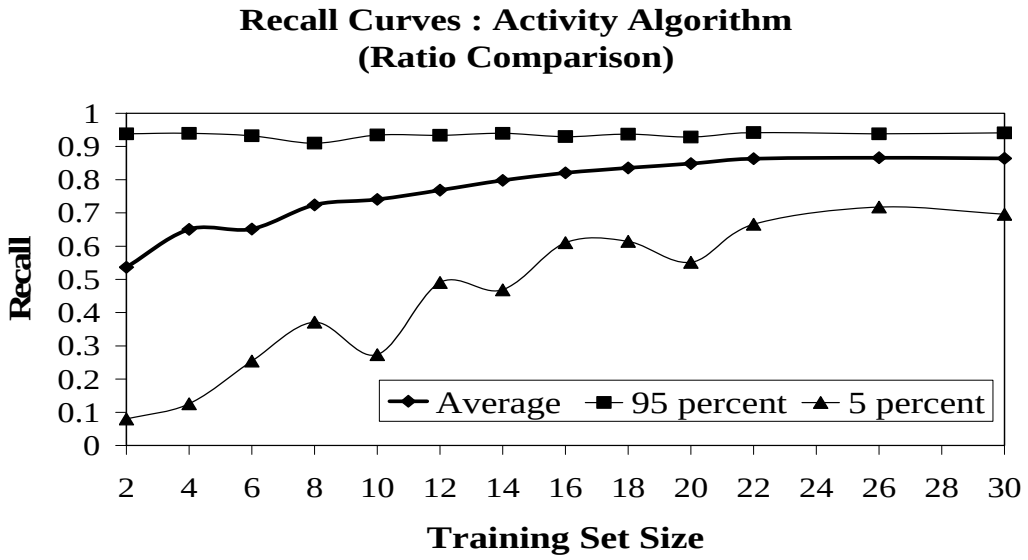


Figure 23. Recall Curves for the Activity Algorithm using analyzer by ratio comparison (Straight-testing)

Table 12. Average Recall, 95 percent value and 5 percent of max recall for the Activity Algorithm using analyzer by ratio comparison (Straight-testing)

Training Set Size	Average	95 percent	5 percent
2	0.5366	0.9383	0.0801
4	0.6504	0.9396	0.1258
6	0.6516	0.9322	0.2541
8	0.7241	0.9103	0.3711
10	0.7407	0.9342	0.2737
12	0.7686	0.9334	0.4908
14	0.7981	0.9396	0.4686
16	0.8204	0.9297	0.6104
18	0.8354	0.9371	0.6141
20	0.8486	0.9285	0.5512
22	0.8633	0.9421	0.6658
26	0.8661	0.9383	0.7176
30	0.8642	0.9407	0.6958

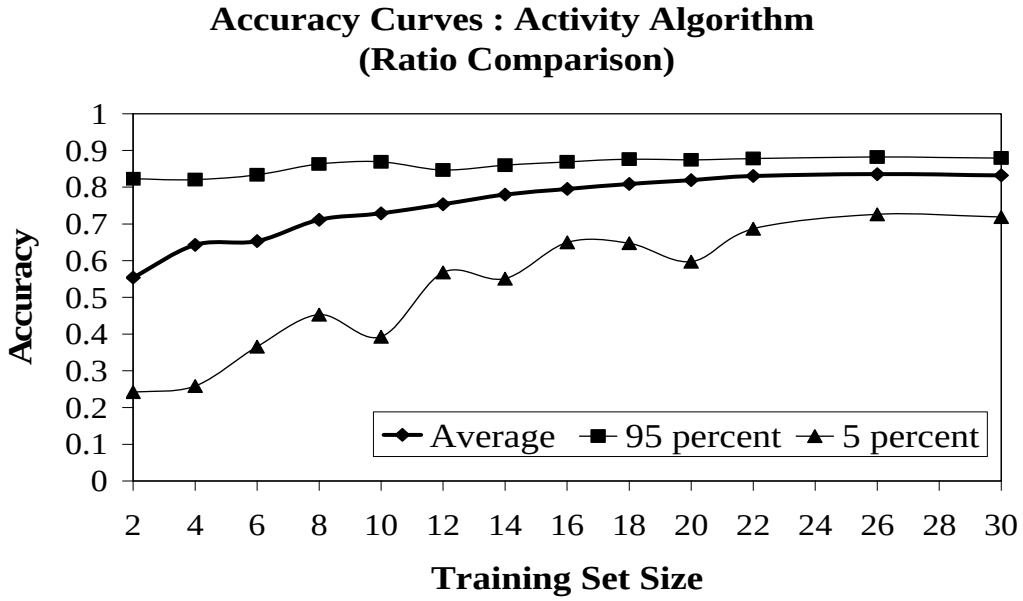


Figure 24. Accuracy Curves for the Activity Algorithm using analyzer by ratio comparison (Straight-testing)

Table 13. Average Accuracy, 95 percent value and 5 percent of max accuracy for the Activity Algorithm using analyzer by ratio comparison (Straight-testing)

Training Set Size	Average	95 percent	5 percent
2	0.5538	0.8228	0.2417
4	0.6429	0.8207	0.2588
6	0.6531	0.8338	0.3656
8	0.7112	0.8631	0.4532
10	0.7288	0.8691	0.3927
12	0.7535	0.8469	0.5681
14	0.7799	0.8608	0.5509
16	0.7953	0.8691	0.6495
18	0.8087	0.8761	0.6475
20	0.8193	0.8741	0.5972
22	0.8304	0.8781	0.6868
26	0.8353	0.8822	0.7261
30	0.8323	0.8793	0.7188

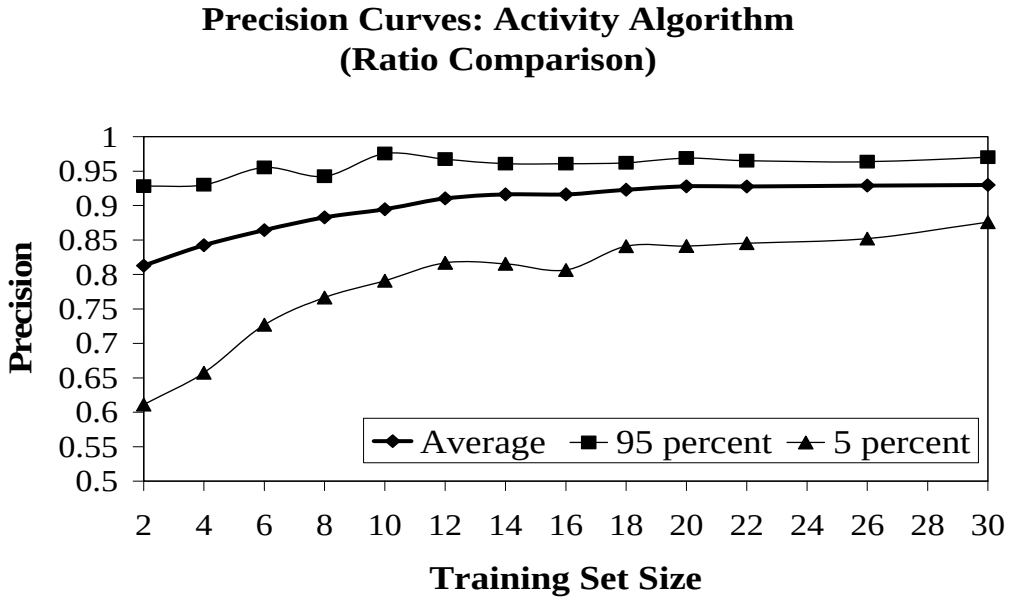


Figure 25. Precision Curves for the Activity Algorithm using analyzer by ratio comparison (Cross-testing)

Table 14. Average Precision, 95 percent value and 5 percent of max precision for the Activity Algorithm using analyzer by ratio comparison (Cross-testing)

Training Set Size	Average	95 percent	5 percent
2	0.8128	0.9282	0.6111
4	0.8423	0.9302	0.6573
6	0.8643	0.9551	0.7269
8	0.8827	0.9424	0.7666
10	0.8948	0.9755	0.7908
12	0.9102	0.9673	0.8168
14	0.9161	0.9607	0.8155
16	0.9160	0.9608	0.8066
18	0.9228	0.9621	0.8411
20	0.9278	0.9688	0.8411
22	0.9276	0.9651	0.8452
26	0.9290	0.9636	0.8521
30	0.9299	0.9701	0.8758

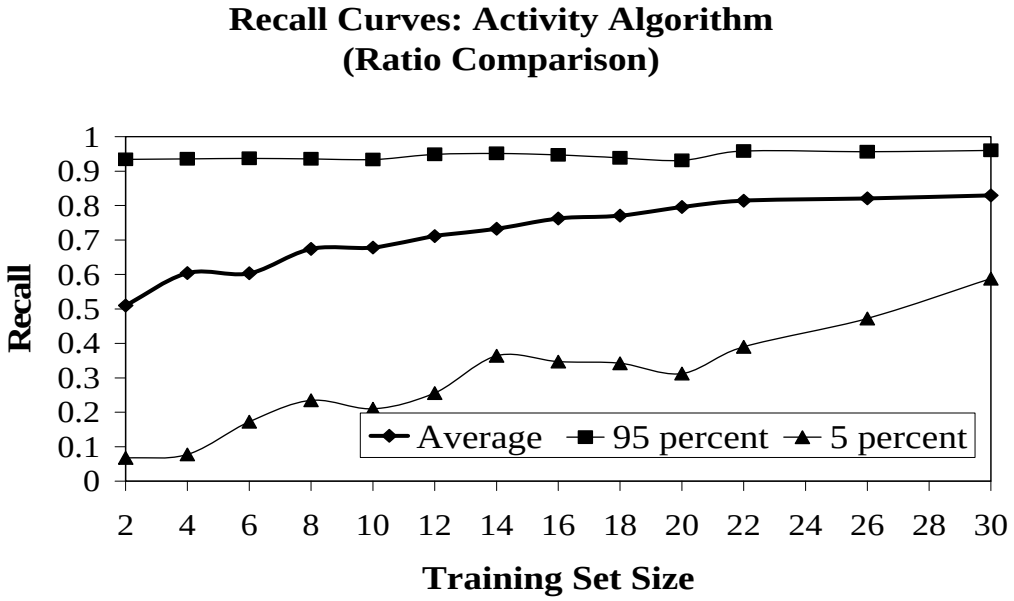


Figure 26. Recall Curves for the Activity Algorithm using analyzer by ratio comparison (Cross-testing)

Table 15. Average Recall, 95 percent value and 5 percent of max recall for the Activity Algorithm using analyzer by ratio comparison (Cross-testing)

Training Set Size	Average	95 percent	5 percent
2	0.5098	0.9338	0.0676
4	0.6037	0.9353	0.0777
6	0.6031	0.9367	0.1727
8	0.6741	0.9353	0.2345
10	0.6779	0.9333	0.2101
12	0.7113	0.9482	0.2561
14	0.7326	0.9511	0.3641
16	0.7624	0.9468	0.3468
18	0.7703	0.9381	0.3424
20	0.7955	0.9309	0.3122
22	0.8140	0.9583	0.3899
26	0.8209	0.9561	0.4721
30	0.8295	0.9601	0.5879

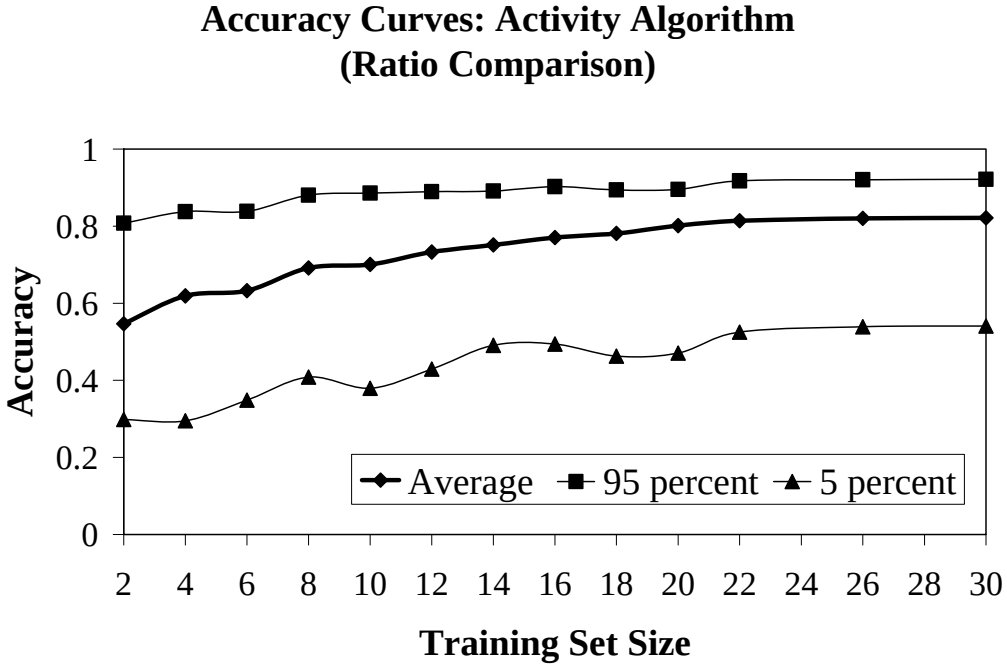


Figure 27. Accuracy Curves for the Activity Algorithm using analyzer by ratio comparison (Cross-testing)

Table 16. Average Accuracy, 95 percent value and 5 percent of max accuracy for the Activity Algorithm using analyzer by ratio comparison (Cross-testing)

Training Set Size	Average	95 percent	5 percent
2	0.5469	0.8075	0.2984
4	0.6193	0.8374	0.2952
6	0.6328	0.8385	0.3487
8	0.6913	0.8802	0.4086
10	0.7004	0.8856	0.3797
12	0.7328	0.8898	0.4289
14	0.7510	0.8909	0.4909
16	0.7706	0.9027	0.4941
18	0.7809	0.8941	0.4631
20	0.8013	0.8952	0.4706
22	0.8137	0.9176	0.5251
26	0.8201	0.9206	0.5391
30	0.8214	0.9215	0.54097

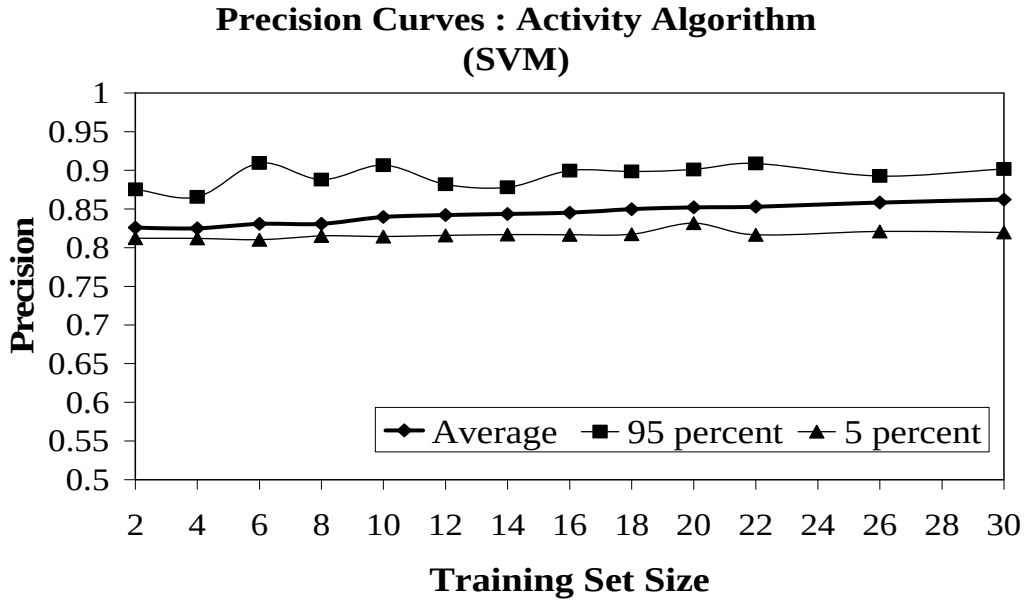


Figure 28. Precision Curves for the Activity Algorithm using analyzer by SVM (Straight-testing)

Table 17. Average Precision, 95 percent value and 5 percent of max precision for the Activity Algorithm using analyzer by SVM (Straight-testing)

Training Set Size	Average	95 percent	5 percent
2	0.8258	0.8753	0.8121
4	0.8249	0.8657	0.8119
6	0.8307	0.9094	0.8103
8	0.8308	0.8881	0.8152
10	0.8396	0.9066	0.8143
12	0.8421	0.8819	0.8159
14	0.8434	0.8781	0.8168
16	0.8451	0.8996	0.8166
18	0.8497	0.8984	0.8173
20	0.8520	0.9011	0.8316
22	0.8528	0.9088	0.8166
26	0.8582	0.8927	0.8209
30	0.8622	0.9017	0.8196

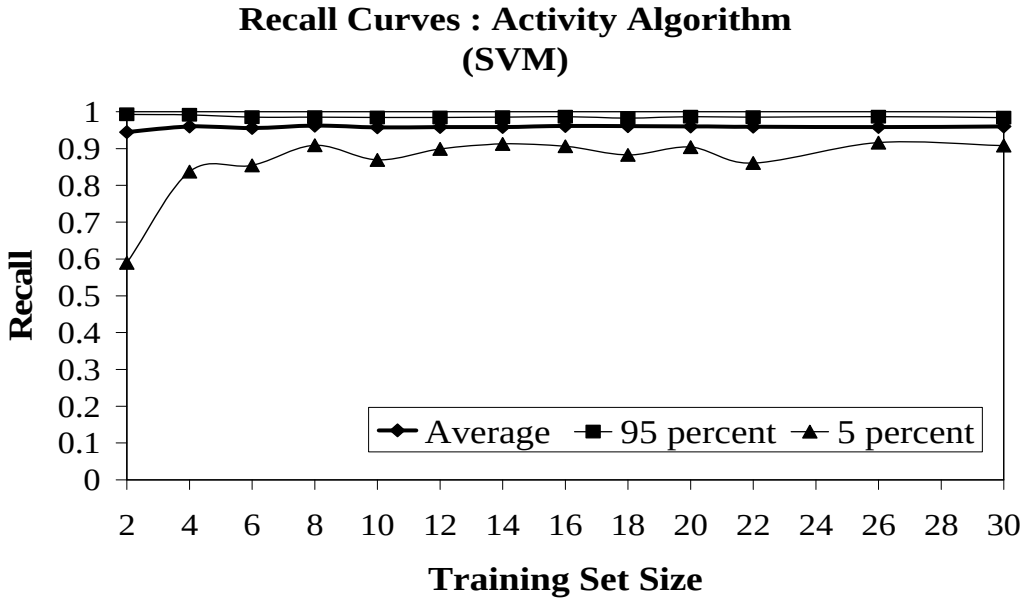


Figure 29. Recall Curves for the Activity Algorithm using analyzer by SVM (Straight-testing)

Table 18. Average Recall, 95 percent value and 5 percent of max recall for the Activity Algorithm using analyzer by SVM (Straight-testing)

Training Set Size	Average	95 percent	5 percent
2	0.9447	0.9926	0.5894
4	0.9596	0.9914	0.8372
6	0.9558	0.9852	0.8545
8	0.9622	0.9852	0.9088
10	0.9575	0.9843	0.8693
12	0.9585	0.9843	0.8989
14	0.9586	0.9852	0.9125
16	0.9613	0.9864	0.9063
18	0.9607	0.9827	0.8829
20	0.9598	0.9864	0.9038
22	0.9595	0.9852	0.8607
26	0.9584	0.9864	0.9162
30	0.9597	0.9838	0.9082

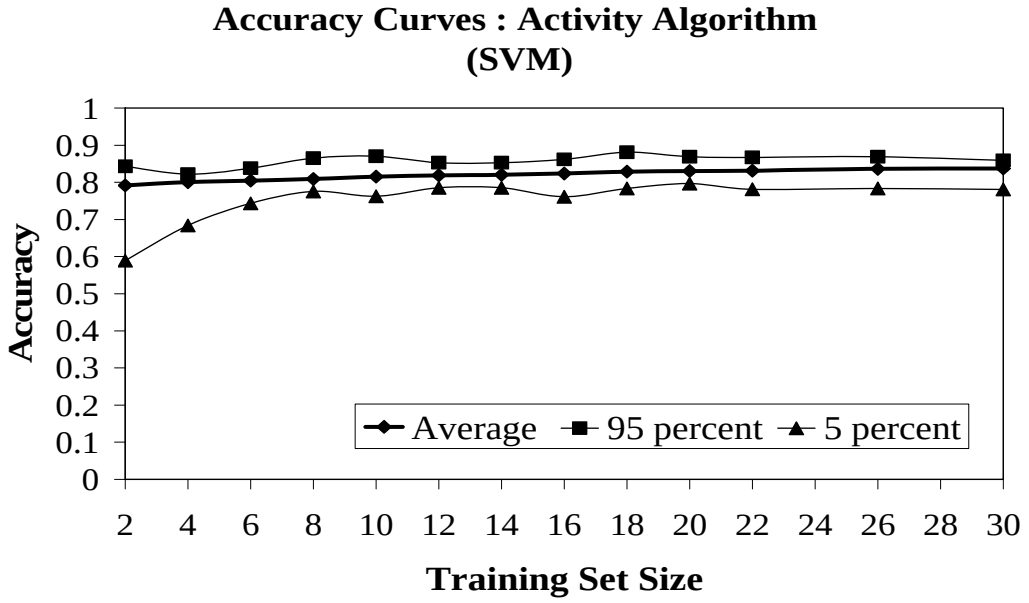


Figure 30. Accuracy Curves for the Activity Algorithm using analyzer by SVM (Straight-testing)

Table 19. Average Accuracy, 95 percent value and 5 percent of max accuracy for the Activity Algorithm using analyzer by SVM (Straight-testing)

Training Set Size	Average	95 percent	5 percent
2	0.7914	0.8429	0.5891
4	0.8003	0.8218	0.6838
6	0.8043	0.8379	0.7432
8	0.8088	0.8651	0.7754
10	0.8150	0.8701	0.7623
12	0.8188	0.8532	0.7855
14	0.8204	0.8532	0.7855
16	0.8240	0.8621	0.7613
18	0.8286	0.8812	0.7835
20	0.8305	0.8691	0.7966
22	0.8311	0.8671	0.7815
26	0.8362	0.8691	0.7835
30	0.8369	0.8592	0.7808

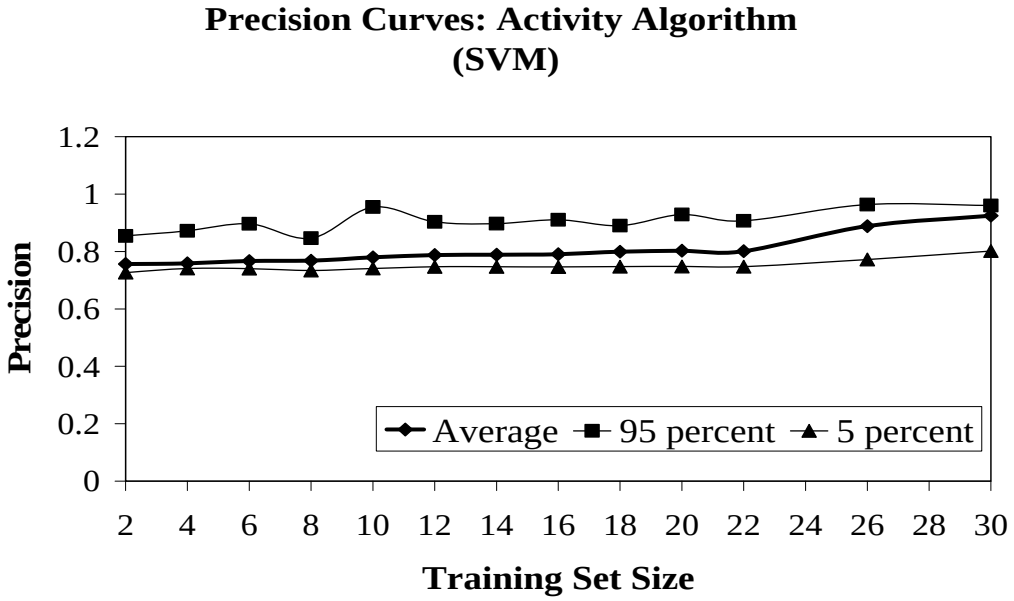


Figure 31. Precision Curves for the Activity Algorithm using analyzer by SVM (Cross-testing)

Table 20. Average precision, 95 percent value and 5 percent of max precision for the Activity Algorithm using analyzer by SVM (Cross-testing)

Training Set Size	Average	95 percent	5 percent
2	0.7566	0.8543	0.7265
4	0.7587	0.8719	0.7407
6	0.7670	0.8962	0.7403
8	0.7677	0.8463	0.7336
10	0.7795	0.9547	0.7414
12	0.7875	0.9032	0.7464
14	0.7885	0.8967	0.7467
16	0.7903	0.9101	0.7462
18	0.7990	0.8902	0.7469
20	0.8024	0.9286	0.7478
22	0.8008	0.9071	0.7474
26	0.8881	0.9636	0.7721
30	0.9245	0.9598	0.8015

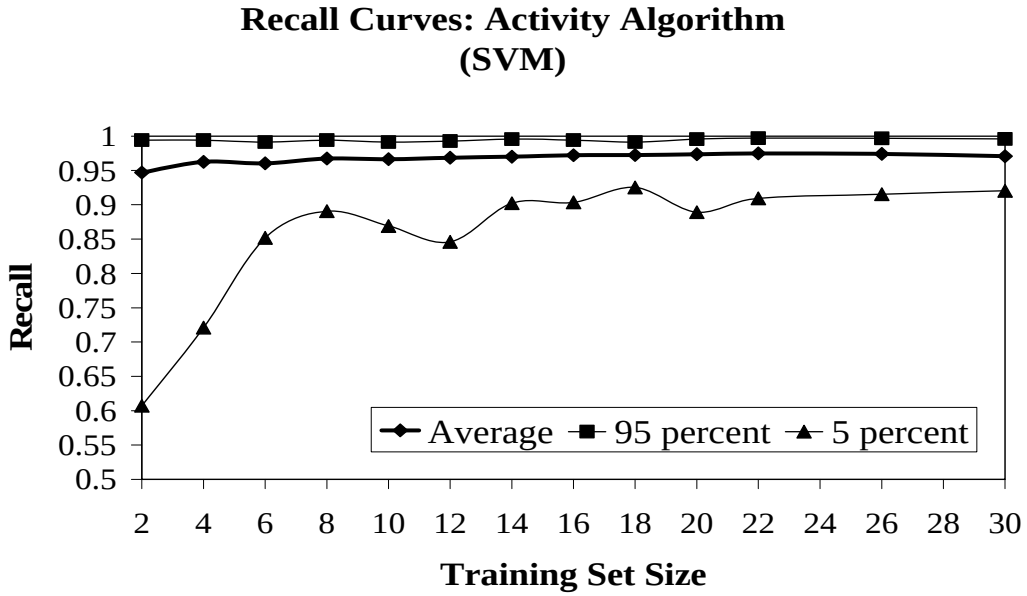


Figure 32. Recall Curves for the Activity Algorithm using analyzer by SVM (Cross-testing)

Table 21. Average recall, 95 percent value and 5 percent of max recall for the Activity Algorithm using analyzer by SVM (Cross-testing)

Training Set Size	Average	95 percent	5 percent
2	0.9471	0.9942	0.6072
4	0.9625	0.9942	0.7211
6	0.9602	0.9914	0.8518
8	0.9672	0.9942	0.8906
10	0.9663	0.9914	0.8691
12	0.9685	0.9928	0.8461
14	0.9699	0.9957	0.9022
16	0.9721	0.9942	0.9036
18	0.9723	0.9914	0.9252
20	0.9735	0.9957	0.8892
22	0.9747	0.9971	0.9094
26	0.9740	0.9969	0.9154
30	0.9709	0.9958	0.9205

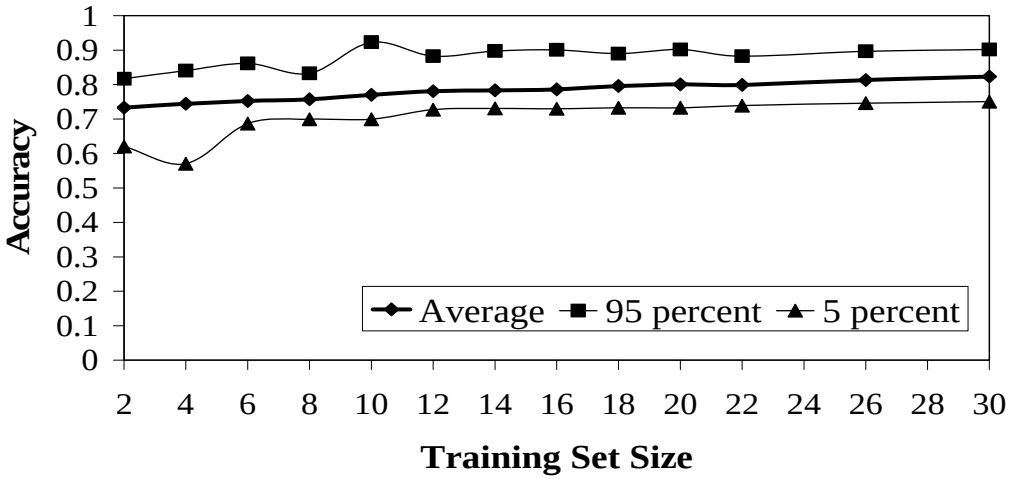


Figure 33

Accuracy Curves for the Activity Algorithm using analyzer by SVM (Cross-testing)

Table 22. Average accuracy, 95 percent value and 5 percent of max accuracy for the Activity Algorithm using analyzer by SVM (Cross-testing)

Training Set Size	Average	95 percent	5 percent
2	0.7328	0.8171	0.6203
4	0.7439	0.8406	0.5701
6	0.7523	0.8613	0.6866
8	0.7571	0.8321	0.6995
10	0.7699	0.9231	0.6995
12	0.7806	0.8824	0.7273
14	0.7830	0.8973	0.7305
16	0.7860	0.9005	0.7294
18	0.7958	0.8898	0.7326
20	0.8003	0.9016	0.7326
22	0.7992	0.8824	0.7391
26	0.8129	0.8966	0.7459
30	0.8233	0.9015	0.7502

4.3.3 Productivity Measurements

The productivities for the system were computed with the classification results from the algorithms. From the experiments conducted, it is observed that the system employing these two classification algorithms exhibits a stable performance when the constituent neural networks were trained with training sets of sizes of [22, 30] samples. At any point in this range, the system exhibits a reliable, stable and maximum performance. Hence a training set size of 26 samples was chosen within this range and the productivities were computed as the average over the 200 experiments. The results are tabulated in Table 23.

Table 23. Classification Results from the developed algorithms

Worker	Samples	Algorithm	Total frames as effective	Total frames as ineffective	Total frames as contributive*
W1	1000	Manual (activity based)	637	183	180
	1000	Instance	583	417	--
	1000	Activity (ratio comparison)	583	235	182
	1000	Activity (SVM)	583	105	312
W2	1000	Manual (activity based)	717	235	48
	1000	Instance	531	469	--
	1000	Activity (ratio comparison)	531	327	142
	1000	Activity (SVM)	531	322	147

* The contributory class does not apply to the Instance Algorithm

Worker W1

Productivity by manual annotation (activity based): 81.7 %:

Productivity by the Instance Algorithm: 58.3 %

Productivity by the Activity Algorithm:

With activity analyzer using ratio comparison: 76.5 %

With activity analyzer using SVM: 89.35 %

Worker W2

Productivity by manual annotation (activity based) : 72.18 %:

Productivity by the Instance Algorithm: 53.1 %

Productivity by the Activity Algorithm:

With activity analyzer using ratio comparison: 67.3 %

With activity analyzer using SVM: 67.8 %

For both the workers, the productivities measured by the instance algorithm were very low when compared to the productivities computed by manual annotation. This is because the instance algorithm did not consider the contributory class in its classification. The productivities measured for the two workers using the activity algorithms were much closer to the manual ones. Hence, the introduction of activity algorithms greatly increased the accuracy in productivity determination.

The activity algorithms employed two different activity analyzers. On comparing the performance of these two activity analyzers we observe that the productivities measured with the *analyzer by ratio comparison* were lower for worker W2 than the corresponding manually

computed value. This is due to the occurrence of many falseNegatives in the classification as evident from the low recall value (0.866 when compared to precision: 92.7) .Similarly, the productivities measured with *analyzer using the SVM* were greater than the normal due to many falsePositives (a low precision value: 0.85 when compared to recall: 0.96).

4.3.4 System characteristics

By these experiments, it is noticed that the precision, recall and accuracy curves steadily ascend as the training set size increases. This proves the positive response of the classifiers to training size. Also, the precision, recall and accuracy values obtained by the activity algorithm are much higher and stable when compared to those of the instance algorithm. This proves the tolerance to error by the activity algorithm that is provided by: making $N \leq L-1$ in analyzer by ratio comparison and applying a learned classifier in the analyzer by using a SVM.

4.3.4.1 System Stability

It can be observed from the graphs that the system exhibits a stable performance for training set size ≥ 22 for both the instance algorithm and the activity algorithms.

4.3.4.2 Speed of Processing

The processing speed of this system when tested on one worker was found to be 0.98 frames per sec during classification and 0.673 frames per sec during the performance testing. Since, the classification speed of 0.98 frames per sec is almost equal to the rate at which the input image sequence is sampled (1 frame per sec), the system effectively works in real-time and thus enables a continuous and automated real time determination of the construction

productivity. To analyze multiple workers, the rate of the input samples is to be reduced proportionally.

5 Conclusions and Contributions

Productivity measurement has been a wide practice to gauge the performance of a construction activity. Existing productivity measurement methods require a continuous human involvement to collect and analyze the data. Such methods are prone to errors due to human biases and limitations. To address this issue, two intelligent human pose analyzing algorithms were developed that automatically and continuously determine the productivity in real time, thus completely eliminating the need for human involvement. A full scale system that deploys these algorithms was also developed to determine productivity. This system extracts the poses of the workers in the video and analyzes them using these developed algorithms.

Simple background subtraction was used to identify these workers. This resulted in blobs (of workers) that had high degree of porosity due to pixel level subtraction and thus led to very noisy skeletons. To avoid this problem, a mask-level background subtraction was implemented which significantly reduced the porosity in the blobs. The unwanted moving objects like flying birds and swaying trees were eliminated by using filtration by pattern matching. These additions to the basic segmentation resulted in a robust algorithm.

The object tracking by match matrix method showed good results. However, this basic tracking algorithm could not handle situations when a track is temporarily lost for a few frames. To add this functionality, the tracker was we modified similar to [32] to wait for a few frames when a track is lost, in hopes of regaining it. This greatly helped in avoiding cases of same workers having multiple track records.

The poses extracted from these blobs were of different sizes due to the variation of depth in their positions from the camera and as a result these poses could not be used for analysis. The poses were then scaled to a fixed size of WxH, irrespective of their original sizes leading to a

consistency in the size of inputs to the analyzing algorithms. In other words, we handled the effect of depth factor in pose analysis by this scaling.

The basic algorithm what is called the *Instance Algorithm* provided good results by classifying poses as effective and ineffective poses. However, it did not consider the contributory class and the productivity results were low compared to the ones obtained by manual annotation. To account for the contributory class, an algorithm was developed what we call *the Activity Algorithm* as an extension of the instance algorithm. This effectively identified and distinguished the contributory poses from the ineffective ones. A significant improvement was noticed in the productivity results comparing to ones obtained by the instance algorithm.

An experimental study was conducted to determine the accuracy of the developed algorithms. Results of analyses indicated that the developed algorithms produced productivity measurements with accuracy around 85-89 percent compared to the manual method. This research project made several major contributions to the advancement of construction industry. First, it applied advanced image processing techniques for analyzing construction operations. Second, the results of this research project made it possible to automatically determine construction productivity in real-time. Thus, an instant feedback to the construction crew was possible. As a result, the capability of rapid construction was improved using the developed technology.

6 Future Work

Our research has a scope for future expansion along two directions: improvements in the current system and extensions to our algorithms to work on multiple activities.

6.1 Improvements in the current system

6.1.1 Worker Extraction

In this system, a simple motion segmentation was used to identify the workers and extract them from the input video. This method requires that the camera is static and even a slight motion in the camera of a jitter in its focus due to wind will cause to detect a large number of false objects. There is a scope to develop advanced motion segmentation algorithms that are unaffected by the camera motion.

6.1.2 Object Tracking

There is a lot of scope for extension of the object tracking method that we used in our thesis. By the current method, only position information is used to track the workers in successive frames. Due to this, incorrect tracking may occur when one track crosses the other. Since tracks have no special identification features, the tracker is confused when one track crosses the other. Hence, more information is necessary to track these workers. Introducing RFID tags to the workers uniforms will lead to better tracking as a worker can uniquely be identified by his tag. This helps in tracking even though the visual contact is lost.

We use only one camera to capture the video in our system. This restricts the tracking capabilities of the system as a worker cannot be tracked when he is obstructed by an untracked object, ex: a truck. The introduction of RFID technology can prevent losing the track and also provide information of the worker's movements, but the workers poses cannot be extracted. A promising future work could be setting up a data acquisition system with multiple cameras placed at different locations of the site. Three cameras place 120 degrees apart can provide a 3 dimensional view of a worker in the site. This significantly solves the above problem as at any moment, a worker is captured by at least one camera, thus making it possible to extract his poses.

6.2 Extensions to our algorithms

The algorithms developed in this system were trained to work on a specific activity of tying a rebar in bridge construction. A future work could be analyzing more activities, identifying the effective, ineffective and constituent poses of the workers and developing appropriate activity analyzers for use with these algorithms extending their use along multiple activities.

References:

[1] Bai, Y., and Burkett, W. R. “Rapid bridge replacement: Processes, techniques, and needs for improvements”. *Journal of Construction Engineering and Management*, 132(11), *American Society of Civil Engineers*, Reston, VA, United States, pp. 1139-1147, 2006.

[2] Ghanem, A. G., and Abdelrazig, Y. A. “A framework for real-time construction project progress tracking”. *American Society of Civil Engineers*, Reston, VA, United States, Earth and Space 2006 - *Proceedings of the 10th Biennial International Conference on Engineering, Construction, and Operations in Challenging Environments*, League City/Houston, TX, United States, pp. 112, 2006.

[3] Chang, L.-M. “Measuring construction productivity”. *Cost Engineering (Morgantown, West Virginia)*, 33(10), pp. 19-25, 1991.

[4] Liu, C., and Song, Y. “Multifactor productivity measures of construction sectors using OECD input-output database”. *Journal of Construction Research*, 6(2), World Scientific Publishing Co. Pte Ltd, Singapore, 596224, Singapore, pp. 209-222, 2005.

[5] Noor, I. “Measuring construction labor productivity by daily visits”. AACE Inc., Morgantown, WV, USA, AACE International. Transactions of the Annual Meeting, Cincinnati, OH, USA, 1998.

[6] Sprinkle, H. B. “ANALYSIS OF TIME-LAPSE CONSTRUCTION FILMS”. *American Society of Civil Engineers, Journal of the Construction Division*, 98(C2), pp. 183-199, 1972.

[7] Thomas, H. R., and Daily, J. “CREW PERFORMANCE MEASUREMENT VIA ACTIVITY SAMPLING”. *Journal of Construction Engineering and Management*, 109(3), pp. 309-320, 1983.

[8] Oglesby, C. H., Parker, H. W., and Howell, G. A. “Data Collection Method”. *Productivity Improvement in Construction*. Chapter 7: Data Gathering for On-Site Productivity-Improvement Studies, McGraw-Hill, Inc., New York, N.Y., pp. 146-210, 1989.

[9] Abudayyeh, O. Y. “Multimedia construction delay management system”. *Microcomputers in Civil Engineering*, 12(3), Blackwell Publishers, Malden, MA, USA, pp. 183-192, 1977.

[10] Fondahl, J. W. “Photographic analysis for construction operations”. *ASCE -- Proceedings -- Journal of the Construction Division*, 86(C2, Part 1), American Society of Civil Engineers (ASCE), pp. 9-25, 1960.

[11] Everett, J. G., and Halkali, H. “Time-Lapse Video Applications for Construction Project Management Schedule”. *J. Constr. Eng. Manage*, 124(3), pp. 2004-2009, 1998.

[12] Teicholz, P. “U.S. construction labor productivity trends, 1970-1998”. *Journal of Construction Engineering and Management*, 127(5), American Society of Civil Engineers, pp. 427, 2001.

[13] Han, S.-W., Lee, S.-Y., and Halpin, D. W. “Productivity evaluation of the conventional and GPS-based earthmoving systems using construction simulation”. American Society of Civil Engineers, Reston, VA, United States, *Construction Research Congress 2005:*

Broadening Perspectives - Proceedings of the Congress, San Diego, CA, United States, pp. 351-359, 2005.

[14] Eldon, C., and Janaka Y. R. "Situation Based Modeling for Construction Productivity," ASCE Construction Research Congress, San Diego, California, Vol. 183, No. 40754, 2005.

[15] Dissanayake, M., Aminah R. F., Russell, A.D., and Pedrycz, W. "A Hybrid Neural Network for Predicting Construction Labour Productivity," ASCE International Conference on Computing in Civil Engineering, Cancun, Mexico, Vol. 179, No. 40794, 2005.

[16] Lin, Hong-Wen, Yang, Shao-Qing, Xia, Zhi-Jun, and Kang, Chun-Yu. "A Moving Objects Detection Approach for Smart Sensor". International Conference on Machine Learning and Cybernetics, Dalian, China, pp 3751-3754, 2006.

[17] Ezeldin, A. S. and Lokman, M. S. "Neural Networks for Estimating the Productivity of Concreting Activities." Journal of Construction Engineering and Management, 132 (6), pp 650-656, 2006.

[18] Li-Chung, C., and Miroslaw, J. S. "Estimating Construction Productivity: Neural-Network-Based Approach." Journal of Computing in Civil Engineering, 8 (2): pp 234-251, 1994.

[19] Ming, L., S. M. Abou Rizk. "Estimating Labor Productivity Using Probability Inference Neural Network," Journal of Computing in Civil Engineering 14 (4), pp 241-248, 2000.

[20] Mengov, G. D., Zinovieva, I. L. and Sotirov, G. R. "Patterns of Work Attitudes: A Neural Network Approach, The 3rd International Conference AIP, Liege, Belgium, Vol. 517, pp 221-229, 2000.

[21] G. Li, Zhang, J. Lin, Tu, H., D. and Zhang, M. "A moving object detection approach using Integrated Background template for smart video sensor". In *Proceedings of the 21st IEEE Instrumentation and Measurement Technology Conference*, volume 1, pages 462-466, May 2004.

[22] Zhao, T. and Nevatia, R. "Tracking multiple humans in crowded environment". *CVPR*, 02:406-413, 2004.

[23] Staurer, C., Eric,W. and Grimson, L., "Learning patterns of activity using real-time tracking". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747-757, 2000.

[24] Barron, J.L., Fleet, D.J., Beauchemin, S.S. and Burkitt, T.A. "Performance of optical flow techniques". *CVPR*, 92:236-242.

[25] Berthold, K., Horn, P. and Schunck., B. G. "Determining optical flow" *Artificial Intelligence*, 17(1-3):185-203, 1981.

[26] Wang, J. and Adelson, E. "Spatio-temporal segmentation of video data". *In SPIE Proc. Image and Video Processing II*, 1994.

[27] Biancardini, L., Dokadalova, E. S. Beucher, and L. Letellier. "From moving edges to moving regions". In *Proceedings Second Iberian Conference IbPRIA 2005*, volume 1, pages 119-127, 2005.

[28] Kuhne, G., Richter, S. and Beier, M. “Motion-based segmentation and Contour-based classification of video objects”. In *MULTIMEDIA '01: Proceedings of the Ninth ACM International Conference on Multimedia* pages 41-50, New York, NY, USA, 2001. ACM Press.

[29] Kalman, R. E. “A new approach to Linear Filtering and Prediction Problems”. *Transactions of the ASME-Journal of Basic Engineering*, 82(Series D):35-45, 1960.

[30] Lou, J. Tan, T. and Hu, W. “Visual vehicle tracking algorithm”, *IEEE Electronics Letters*, 38(18):1024-1025, August 2002.

[31] Isard, M. and Blake, A. “Condensation-Conditional density propagation for visual tracking”. *International Journal of Computer Vision*, 29(1):5-28, 1998.

[32] Narayana, M. and Haverkamp, D., “A Bayesian algorithm for tracking multiple moving objects in outdoor surveillance video”. In *Fourth Joint IEEE International Workshop on Object Tracking and Classification in and Beyond the Visual Spectrum, In conjunction with IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2007*, June 2007.

[33] Haralick, R.M. ”Performance Characterization in image analysis: thinning, a case point, *Pattern Recognition Letters”*, Vol.13, pp. 5-12, 1992.

[34] Guo, Z., and Hall, R.W. “Fast fully parallel thinning algorithms”, *CVGIP: Image Understanding*, Vol. 55, No. 3, pp. 317-328, 1992.

[35] Plamondon, R., and Suen, C.Y., "On the definition of reference skeletons for comparing thinning algorithms", in *Proc. Vision Interface 1988,* Edmonton, Canada, pp. 70-75, 1988.

[36] Jaisimha, M.Y., Haralick, R.M. "A methodology for the characterization of the performance of thinning algorithms", in *ICDAR,* pp. 282-296, 1993.

[37] Lam, L., Lee, S.W. and Suen, C.Y "Thinning Methodologies- a comprehensive survey", *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 14, No.9, pp. 869-885, 1992.

[38] Bai, Y., Kim, S., and Burkett,W.R., "Enhancing the capability of rapid bridge replacement after extreme events." *Engineering, Construction and Architectural Management*, Vol.14, No. 4, pp 375-386. 2007

[39] Arcelli, C., "Pattern thinning by contour tracing", *Computer Graphics Image Processing*, Vol. 17, pp. 130-144, 1981.

[40] Zhang, T.Y. and Suen, C.Y. "A fast parallel Algorithm for Thinning Digital Patterns", *Communications of the ACM,* Vol. 27, Issue. 3, 1984.

[41] Adams, R.; Bischof, L. "Seeding region growing", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, Issue. 6, pp. 641 – 647, 1994.

[42] Tsukamoto, Y. Namatame, A. "Evolving neural network models", *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1996., pp. 689-693, 1996.

[43] Amin, Minesh B. and Shekhar, Shashi. ``Customizing Parallel Formulations of Back propagation Learning Algorithm to Neural Network Architecture: A summary of Results". *IEEE Transactions on Parallel and Distributed Systems*, 1994.

[44] Beale, R. and Jackson, T. ``*Neural Computing: An Introduction"*. Hilger, Philadelphia, PA, 1991.

[45] Katagiri, S. and Shigeo, A. “Incremental training of support vector machines using hyperspheres” *Pattern Recognition Letters*, Vol. 27, Issue. 13, pp. 1495-1507, 2006.

[46] David, J. and Goadrich, M. “The relationship between precision, recall and ROC curves” *Proceedings of the 23rd International Conference on Machine Learning*, 2006.