Z Language Based an Algorithm for Event Detection, Analysis and Classification in Machine Vision

Sukhpal Singh
CSED
Thapar University, Patiala
ssgill@hotmail.co.in

Dr. Inderveer Chana
CSED
Thapar University, Patiala
inderveer@thapar.edu

Dr. Maninder Singh
CSED
Thapar University, Patiala
msingh@thapar.edu

Abstract—**A common task in various machine learning (ML) application areas involves observing regularly gathered data for 'interesting' events. This mission is predominant in reconnaissance, but also in responsibilities fluctuating from the investigation of scientific data to the observing of unsurprisingly happening events, and from controlling engineering procedures to noticing human behavior. We will refer to this observing procedure with the determination of classifying remarkable manifestations, as event detection, analysis and classification. With the appearance of personal computers (PCs), a lot of efforts have been made to substitute manual investigation by a computerized manner. Data, nevertheless, have become gradually difficult, and the sizes of gathered data have become enormously bulky in latest years. Text documents, JPEG images, MP3, videos and even relational data are now regularly gathered. In this paper, we propose an algorithm for event detection, analysis and classification in machine vision. Till proposed algorithm is deliberated a significant facility, required degree of event detection cannot be achieved. Finally, we use K-means algorithm for classification of incoming events and proposed algorithm has been validated by Z Formal specification language in general. The proposed algorithm has been implemented in Matlab and results have been gathered through a data mining tool. Using the proposed algorithm, the events are easily detected, analyzed and classified in machine vision.**

Keywords— Expert System; Artificial intelligence; Machine Learning; Computer Vision; Machine Vision; Software Architecture; Matlab; Data Mining; Formal Language

I. INTRODUCTION

Expert System (ES) is a set of programs that handle encrypted knowledge to solve problems in a particular area, which usually needs human proficiency [1]. Several renowned expert systems are DENDRAL, MYCIN, PROSPECTOR and XCON (R1). ES are systems which provide professional quality guidance, analyzes and recommendations given real world problems [2]. Nevertheless, ES is intended to resolve real problems which generally would need a dedicated human expert (such as a doctor or a mineralogist). Expert systems have been used to explain an extensive variety of problems in areas such as treatment, calculation, engineering, geology, computer science and learning [3]. Artificial Intelligence (AI) is a branch of computer science that studies and develops intelligent machines and software [4]. An intelligent manager is a system that observes its surroundings and takes actions that increase its probabilities of realization [5].

Machine Learning, a branch of AI, is about the creation and study of systems that can acquire from data [6]. For example, a ML system could be trained on e-mails to learn to differentiate between non-junk and junk mails. Subsequently learning, it can then be used to categorize new emails into non-junk and junk folders [7]. Computer Vision (CV) is a domain that comprises approaches for obtaining, handling, scrutinizing and recognizing JPEG images and, in broad, high-dimensional data from the real world in demand to create figurative information, e.g., in the forms of judgments [8]. A theme in the progress of this domain has been to replica the capabilities of human vision by automatically recognizing JPEG image [9]. This image understanding can be seen as the unscrambling of figurative information from image data using prototypes created with the help of geometry, physics, indicators, and learning philosophy. Computer vision has been designated as the creativity of computerizing and adding a extensive variety of processes and demonstrations for vision observation [10]. Machine Vision (MV) is the technology and methods used to provide imaging-based automatic inspection and analysis for such applications as automatic inspection, event control, and computer guidance in industry [11]. The field of machine vision, or computer vision, has been developing at a fast speed [9]. As in most fast-emerging areas, not all characteristics of MV that are of interest to energetic researchers are useful to the software designers and customers a vision system for a definite application [12].

The word Software Architecture (SA) instinctively represents the high level configurations of a software system [13]. SA can be defined as the set of configurations desired to purpose about the software system, which include the software components, the relationships among them, and the properties of both components and relationships [14]. The use of patterns and styles of software design is inescapable in many engineering disciplines like machine learning, computer vision etc. [15]. MATLAB (matrix laboratory) is a mathematical computing environment and 4G programming language (PL) developed by MathWorks. MATLAB permits matrix operations, plotting of functions and data, execution of algorithms, development of customer interfaces, and

interfacing with programs coded in other programming languages, comprising C, C++ and Java [16]. The Z language is a formal specification language that creates it easier to generate mathematical depiction of complex dynamic software's. The explanations are frequently lesser and simpler than other PL can offer. Other PLs include a combination of formal and informal parts [17].

The main goal of our work is to establish an event detector system. The motivation behind this paper is to propose an algorithm for event detection, analysis and classification in machine vision. The paper is structured as follows: Related work has been presented in Section 2. In Section 3, a description of Event Detection Model (EDM) has been presented. The Validation of Event Detection Model (EDM) has been presented in Section 4. In Section 5, the Results and Discussions has been described. The conclusion and the future work have been presented in Section 6.

II. RELATED WORK

As a result, to address event detection issues, innovative methodologies have to be generated in order to address their specific problems and to fill the corresponding gaps in ML research. With respect to the ML algorithms, we were predominantly engrossed in how event detection algorithms solves complex forms of data, include valuable domain knowledge, and on how these event detection algorithms are verified [18]. We also observed for methods that researchers have working for event detection task breakdown and at recognizing how the suggested methods can be comprehensive for different kinds of tasks [19]. One of the paramount challenges for ML when implemented to event detection tasks is the fact that data is not sufficient if the detection of anomalous events is of interest [20].

Most of the methods so far include fragments of domain expertise in an ad-hoc way while implementing the learning algorithms or while training the models on the presented data [21]. Normally such resolutions need important re-engineering if deployed or if employed on (even slightly) unlike tasks [22]. Neill and Cooper extend univariate Bayesian detection framework into a principled multivariate Bayesian method that incorporates earlier domain knowledge for an extremely powerful detector of developing patterns [23]. Their multivariate Bayesian scan statistic (MBSS) approach stands out because of its flexibility and applicability to an extensive variety of multivariate detection issues. Nikovski and Jain present novel, scalable algorithms for detecting sudden variation in streaming data [24].

Unlike distinctive change point detection algorithms which suppose the data are drawn from a scattering of a recognized parametric form, the algorithms in Nikovski and Jain make no distributional suppositions of the data. The authors define dynamic programming approaches to creating their non-parametric approaches scalable and also provide a hypothetical study of their algorithms. Afterward, Tandon and Chan look definitely at addressing one of the most puzzling real-world event detection tasks, namely computer network intrusions [25]. Their methods create guidelines that can be amended and judged by experts. The paper cautiously investigates the tradeoff between trimming and different instruction coverage intensification techniques. The proposed Hybrid coverage intensification leads to the maximum detection rate while sustaining a false positive rate of less than 1%—both network and host-based detection.

Finally, Singliar and Hauskrecht defined detection desires, developed, and analyzed over different parameter ranges, a series of detectors for traffic events [26]. They exactly address two of the most significant features of controlling of streaming data that is gathered from sensors: data affected by noise and data that is not associated. Their learning of the Tree-Augmented Naïve Bayes (TAN) approach addresses the association problem in a very sophisticated way, and undoubtedly increases detection. Based on literature review, various event detection research problems stay open [18] [19] [20] [21] [22] [23] [24] [25] [26], such as:

- How can we determine spatio-temporal events proficiently?
- How can events in relational data be exposed?
- How should learners interface with domain experts for reeducating or refocusing on different event detection tasks, via a high level language? How can highly interactive perfect event detection be attained?
- How should complex data be controlled proficiently?
- What are the greatest methods of controlling confidence assessments of a detector?
- How can extra knowledge that is more complex and structured than probabilistic priors are incorporated into event detectors?
- What are the best arithmetical investigations for assessing anomaly and event detectors?

The important role of this study is to construct a basis for detection, analysis and classification of events and then verifying their occurrence using Z formal language.

III. EVENT DETECTION MODEL (EDM)

We have proposed an event detection model where events are analyzed and classified based on their characteristics. The implementation of EDM enables the system to analyze incoming events.

A. Blackboard Architecture Based Event Detector (BABED)

BABED basically implies gathering events. In a BABED, there are two distinct kinds of components: a central data structure represents the current state and a collection of independent events operate on the Blackboard Shared Repository (BSR). Interactions between the BSR and events can vary. Figure 1 illustrates the simple view of BABED. Where common repository i.e. blackboard is iteratively updated by a diverse group. The components of BABED are shown in Figure 2. The BABED is usually presented with three major parts:

- Event Source: There are event specialist modules for the incoming events.

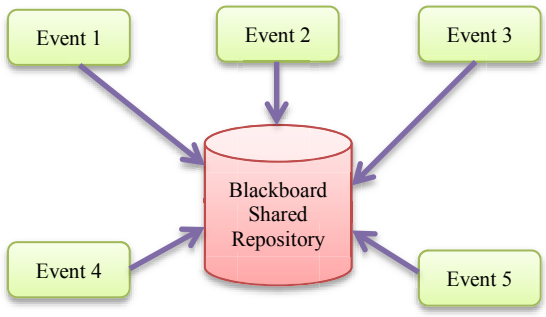


Figure 1. Blackboard Architecture Based Event Detector (BABED)

- Event Repositories: It is like a dynamic library, which is continuously updated by multiple event sources.
- Event Controller: To control the flow of event detection activity. It decides about the event sources that can make change in Blackboard Shared Repository (BSR).

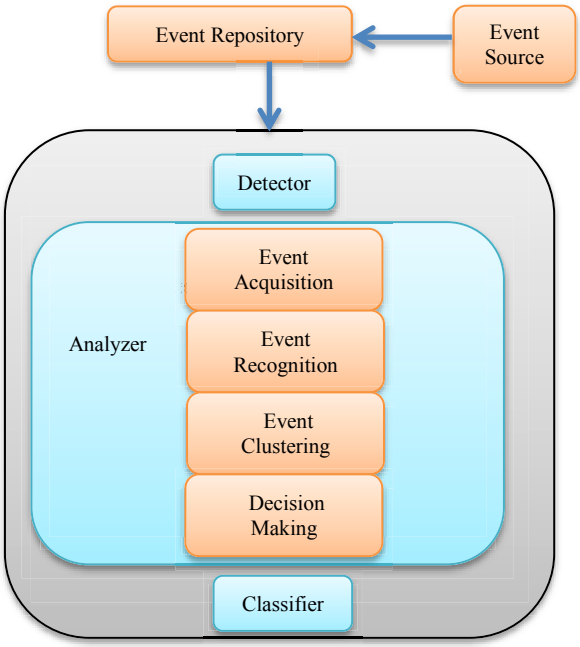


Figure 2. Components of BABED

The main components of BABED are discussed below:

- Event Acquisition: An event is created by one or some sensors, depending on the category of sensor, the resulting event data is a conventional 2D/3D image, or an image categorization.
- Event Recognition: The traditional issue in CV, image processing and MV is that of finding whether or not the event data comprises some particular item or activity. This mission can usually be elucidated vigorously and lacking work by a human, but is still not adequately resolved in CV.
- Event Clustering: Cluster analysis is the assignment of a set of observations into subsets (called clusters) so that observations within the same cluster are similar according to some predestinated benchmarks, while interpretations drawn from dissimilar clusters are different. Different clustering methods make different suppositions on the structure of the event data, often defined by some similarity metric EDR. In our work, K-Means clustering algorithm has been used for clustering of events [27].
- Decision Making: Decision tree knowledge uses a decision tree as an analytical model which plots interpretations about an event to deductions about the event's target value. Creating the last decision mandatory for the event, how to handle it.

B. Process of Event Detection (PED)

Following is the explanation of stages shown below in Figure 3 of Process of Event Detection (PED):

- Recognition: Recognize class of issues that the EDM will be predictable to resolve, comprising the data that the EDM will work with, and the norms that results must meet.
- Perception: Find the important ideas and interactions among them. This should comprise a characterization of different types of event, the stream of information and the essential structure of domain in terms of incoming events.
- Ratification: Effort to understand the nature of fundamental examines event and the character of event that will have to be directed. Main problems contain the certainty and comprehensiveness of information and other restrictions upon the incoming events.
- Execution: In revolving a ratification of event knowledge into a runnable program, one is mainly concerned with the requirement of control and details of event information flow. Decisions must be made about the BSR and the degree of freedom among different events.
- Corroboration and Investigation: The appraisal of EDM is far from being a particular case, but it is clear that the task of event detection can be made easier if one is able to execute the event detection program on a large and illustrative section of test cases to detect a particular event.

C. K-Means Based Event Detection (KMBED) Algorithm

The K-Means Algorithm is using in data mining for the clustering of similar events based on the minimum distance calculated [27]. In our work, we have modified K-Means Algorithm for event detection. The K-Means Based Event Detection (KMBED) Algorithm is shown in Figure 4.

D. Event Detection Rate (EDR)

The event detection rate (ERD) is calculated as the ratio of number of events detected to the total number of events. The fitness formula derived for event detection is given below:

$$EDR = \frac{Number\ of\ Events\ Deteected}{Total\ Number\ of\ Events}$$

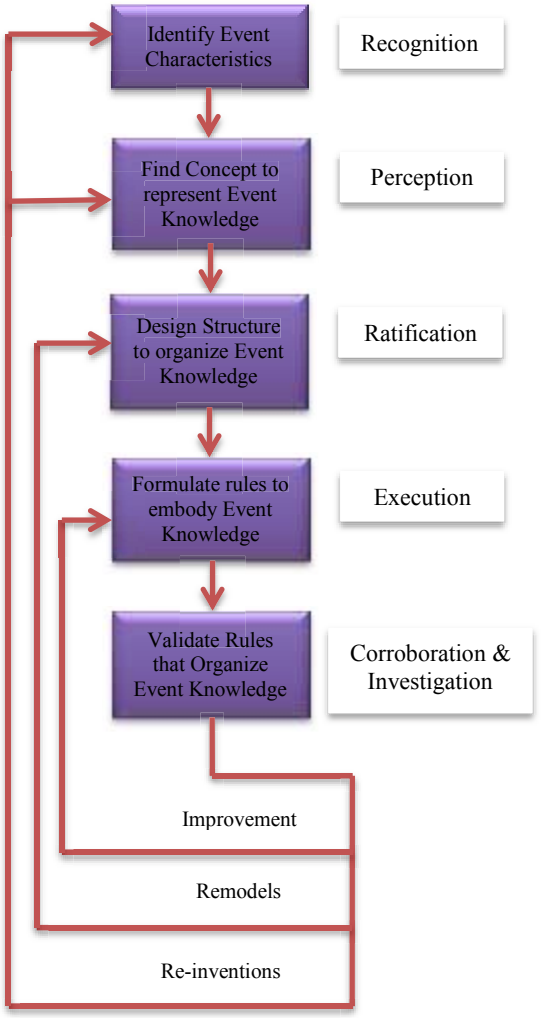


Figure 3. Process of Event Detection (PED)

IV. VALIDATION OF EVENT DETECTION MODEL

Formal specification can serve as a single, reliable reference point for who investigate events; detect the current events by tracing existing events and those who verify the results [17]. In Z specification [17], schemas are used to describe both the static and dynamic aspects of a system. Z decomposes specifications into manageably sized module's called schemas: Schemas are divided into three parts: 1. A state, 2. A collection of state variables and their values and 3. Operations that can change its state [17]. This section explains how the Event Detection Model deals with the current and existing events. The set of all current events and existed events are the basic types of the specifications [17].

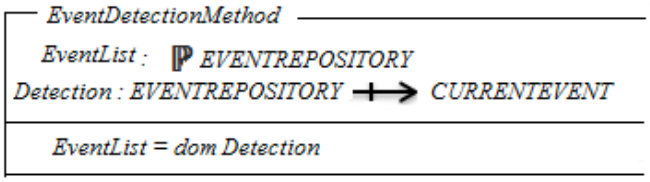
[EVENTREPOSITORY, CURRENTEVENT]

In our work, the space of event detection has been described and the two variables represent important observations which can make of the state [17].

Algorithm: K-Means Based Event Detection (KMBED) Algorithm

Input: E = Number of Events
K = Number of Existed Events (Clusters)
Output: Set of K Centroids c ϵ C representing a good partition of E into K Existed Clusters and Value of Event Detection Rate (ERD)
1. Select the initial event centroids c
2. Repeat
3. Changed = 0
// Discover the nearby centroid to each data point e….and ERD
4. For all data point e_i ϵ E do
5. assigned.Event = e_i.Event
6. For all event c_j ϵ C do
7. Compute the Squared Euclidian Distance dist = dist (e_i, c_j)
8. If dist < e_i.eventDistance then
9. e_i.eventDistance = dist
10. e_i.center = j
11. endif
12. end for
13. If e_i.event < > assigned.Event then
14. Changed ++
15. Recompute c_j new for next event detection
16. endIf
17. endFor
18. Until Changed == 0

Figure 4. K-Means Based Event Detection (KMBED) Algorithm

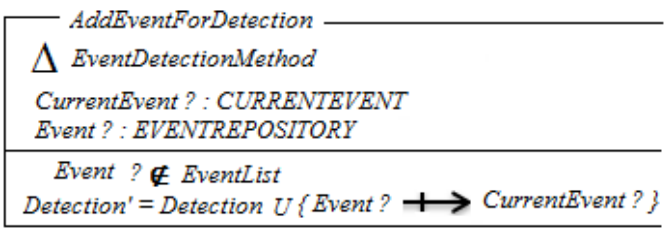


- *EventList* is the set of existing events.
- *Detection* is a function that when applied to certain existing events, it detects the events associated with them.
- set *EventState* is the same as the domain of the function detect the events to which it can be validly applied.

EventList = {Event1, Event2, Event3}

Detection = {Event1 → CurrentEvent3
Event2 → CurrentEvent2
Event3 → CurrentEvent1}

The invariant is satisfied because detection details a CURRENTEVENT for tree EVENTREPOSITORY in detection. There are some operations that can apply on the event detector. The Δ EventDetectionMethod alerts us to the fact that the schema is describing a state change: it introduces four variable EventList, Detection, EventList' and Detection'. The first of all there is to add a new event, and we describe it with schema:



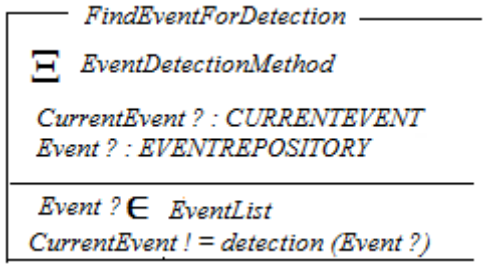
The first two are observations of the state earlier the modification, and the most recent two are interpretations of the state after the change [17]. We expect that the set of event known to EventDetectionMethod will augmented with new event.

Detection' = Detection U {Event?}

We can prove this from the specification of AddEventForDetection using the invariants on the state before and after.

EventList ' = dom Detection '
= dom (Detection U { Event ? ⟼ CurrentEvent ? })
= dom Detection U dom { Event ? ⟼ CurrentEvent ? })
= dom Detection U dom { Event ? }
= EventList U { Event ?}

In FindEventForDetection for, traceability schema finds the events for coming applications.



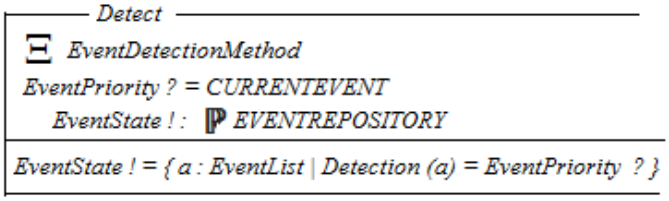
The declaration Ξ EventDetectionMethod indicates that this is an operation in which the state does not change, the value of EventList' and Detection' of the observations after the operation are equal to these values EventList' and Detection'. Including Ξ EventDetectionMethod above the line has the same effect as including Δ EventDetectionMethod above the line and two equations below it.

EventList ' = EventList

Detection ' = Detection

The other notation (!) for an output the FindEventForDetection operations take an EventList as input and yield corresponding detection as output.

The most useful operation on event detector is one to find which Current Event detect with Existing Events. The operation has an input EventPriority? And one output, EventState! Which is set of events for detection? There may be zero, one or more event detect with particular existing events, to whom existing event EventState should be sent.



This time there is no pre-condition. The Eventstate! is specified to be equal to the set of all values a drawn from the set item such that the value of the detection function at a is EventPriority? [17]. In general, i is a member of the set {h: Z |……..h……} exactly if i is a member of Z and the condition ……i……, obtained by replacing h with i, is satisfied:

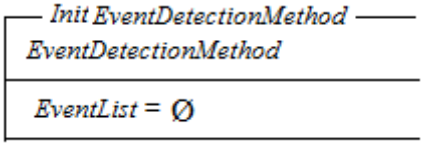
i ∈ {h: Z|…h…..} ↔ i ∈ Z ∧ (….i….)

w ∈ {a: EventState ∧ Detection (w)= EventPriority? }

↔ w∈ EventState ∧ Detection (w) = EventPriority?

A name w is in output set EventState! exactly if it is known to the event detector and the detection recorded for it is EventPriority? [17].

The given below schema identify the initial state of the event detector:

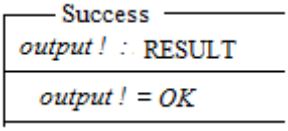


This schema describes an EventDetectionMethod in which the set known is empty: in consequence, the function detection empty too.

We shall add an extra output! to each action in event detector. After successful execution of given process the outcome will be OK [5], but it may take the other value already_detected and not_detected when error is discovered. RESULT defines the set contains three values.

RESULT::=OK/ already_detected / not_detected

The result should be OK after proper execution of success schema without saying how the state changes.



EventDetectionMethod ∧ Success

The conjunction operator ∧ of the schema calculates allows us to combine this description with our previous description of EventDetectionMethod. The process for accurate input has

defines both acts as described by EventDetectionMethod and produces the result OK. Schema specified that the result already_detected should be produced when input event? Is already a member of detected.

$$
\begin{array}{|l}
\textit{AlreadyDetected} \\
\hline
\Xi\ \textit{EventDetectionMethod} \\
\textit{Event ?} : \textit{EVENTREPOSITORY} \\
\textit{output !} : \text{RESULT} \\
\hline
\textit{Event ?} \in \textit{Detected} \\
\textit{output !} : \textit{already\_detected} \\
\hline
\end{array}
$$

Ξ EventDetectionMethod specifies that if the error occurs, the state of the event detector should not change.

$$\textit{REventDetectionMethod} \mathrel{\hat{=}} (\textit{EventDetectionMethod}) \vee \textit{AlreadyDetected}$$

$$
\begin{array}{|l}
\textit{REventDetectionMethod} \\
\hline
\Delta\ \textit{EventDetectionMethod} \\
\textit{CurrentEvent ?} : \textit{CURRENTEVENT} \\
\textit{Event ?} : \textit{EVENTREPOSITORY} \\
\textit{output !} : \text{RESULT} \\
\hline
\textit{Event ?} \in \textit{Detected} \wedge \\
\textit{Detection}' = \textit{Detection} \cup \{\textit{Event ?} \rightarrowtail \textit{CurrentEvent}\} \wedge \\
\textit{output !} = \textit{OK}) \vee \\
\textit{Event ?} \in \textit{Detected} \wedge \textit{Detection}' = \textit{Detection} \wedge \\
\textit{output !} : \textit{already\_detected} \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\textit{NotDetected} \\
\hline
\Xi\ \textit{EventDetectionMethod} \\
\textit{Event ?} : \textit{EVENTREPOSITORY} \\
\textit{output !} : \text{RESULT} \\
\hline
\textit{Event ?} \notin \textit{Detected} \\
\textit{output !} : \textit{not\_detected} \\
\hline
\end{array}
$$

$$\textit{REventDetectionMethod} \mathrel{\hat{=}} \textit{FindEventForDetection} \wedge \text{Success}\ ) \vee \textit{NotDetected}$$

Detected operation can be called at any time, it never results an error robust version need only add reporting of success.

$$\textit{RDetected} \mathrel{\hat{=}} \textit{Detected} \wedge \text{Success}$$

V. RESULTS AND DISCUSSION

The event detection model has been validated by Z Formal specification language [17] and implementation of proposed algorithm has been done through the Matlab [16]. The results were taken by a data mining tool i.e. WEKA [28]. In this, there are two types of events: red color represents the existing events while current events are represented by blue color as shown in Figure 5.

There are five different scenarios as shown in Figure; the events are coming at different rate. All the incoming events are recognized in different five cases. Existed events are used to recognize the incoming events. After that all the current detected events are analyzed and clustered based on particular event characteristics. The classification of current events is represented in the form of plot matrix as shown in Figure 6.

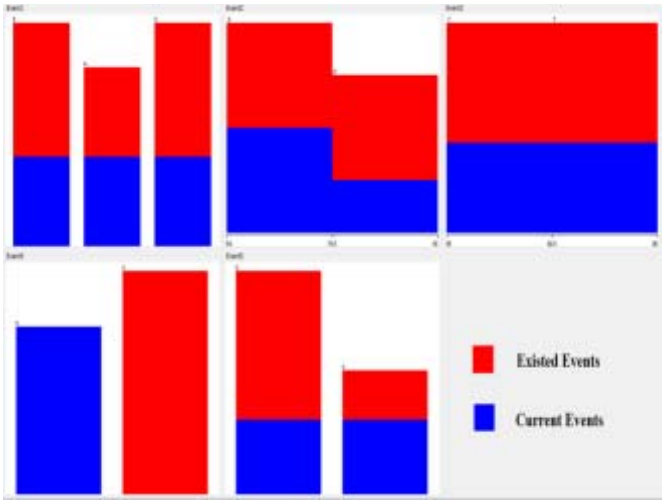


Figure 5. Clustering of existed and current events

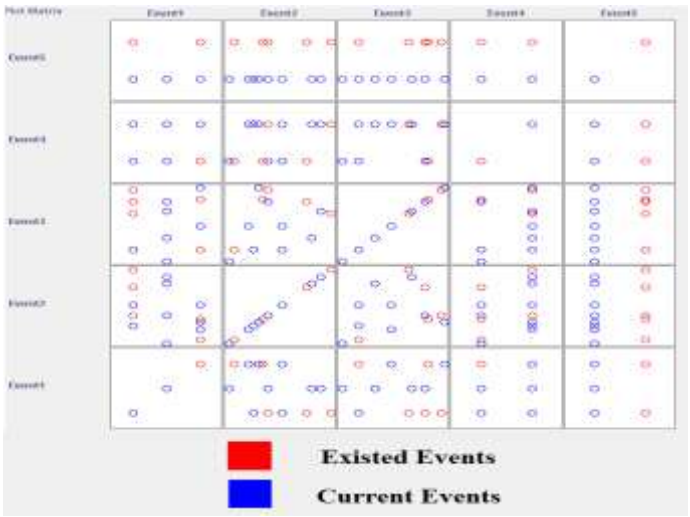


Figure 6. Plot matrix of existed and current events

VI. CONCLUSION AND FUTURE SCOPE

In this paper, we have presented an algorithm for event detection, analysis and classification in machine vision. The objective is to minimize the complexity of event detection. Through this work, incoming events are easily detected, analyzed and classified. We have used K-means algorithm for classification of incoming events. Verification of proposed algorithm has been done with the help of Z specification formal language. Formal specification will ensure that the all the events are detected properly and will make the detection process more clear. The implementation of proposed algorithm

has been done in Matlab and results have been gathered through a data mining tool.

The future scope of this work is to analyze and to incorporate risk factors in EDM and find the critical success factors of the EDM and also identify the various risk factors using risk analysis. Event detection along with analysis and classification can also be automated in machine vision using an automated tool. The automated model solves the problems of event detection in machine vision to a large extent. The current results have been gathered through data mining tool but in future same results would be verified on actual network.

REFERENCES

[1] Waterman, Donald. "A guide to expert systems." (1986).

[2] Chandrasekaran, Balakrishnan. "Generic tasks in knowledge-based reasoning: High-level building blocks for expert system design." IEEE expert 1, no. 3 (1986): 23-30.

[3] Zhang, Wenzhu, and Brian T. Chait. "ProFound: an expert system for protein identification using mass spectrometric peptide mapping information." Analytical chemistry 72, no. 11 (2000): 2482-2489.

[4] Russell, Stuart Jonathan, Peter Norvig, John F. Canny, Jitendra M. Malik, and Douglas D. Edwards. Artificial intelligence: a modern approach. Vol. 74. Englewood Cliffs: Prentice hall, 1995.

[5] M. Ginsberg, W. Harvey, Iterative broadening, Artificial Intelligence 55 (2) (1992) 367–383.

[6] Goldberg, David E., and John H. Holland. "Genetic algorithms and machine learning." Machine learning 3, no. 2 (1988): 95-99.

[7] Goldberg, David E. "Genetic algorithms in search, optimization, and machine learning." (1989).

[8] Hartley, Richard, and Andrew Zisserman. Multiple view geometry in computer vision. Vol. 2. Cambridge, 2000.

[9] Haralock, Robert M., and Linda G. Shapiro. Computer and robot vision. Addison-Wesley Longman Publishing Co., Inc., 1991.

[10] Bradski, Gary R. "Computer vision face tracking for use in a perceptual user interface." (1998).

[11] Jain, Ramesh, Rangachar Kasturi, and Brian G. Schunck. Machine vision. Vol. 5. New York: McGraw-Hill, 1995.

[12] Tsai, Roger. "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses." Robotics and Automation, IEEE Journal of 3, no. 4 (1987): 323-344.

[13] Shaw, Mary, and David Garlan. "Software architecture: perspectives on an emerging discipline." (1996).

[14] Bosch, Jan. "Software architecture: The next step." In Software architecture, pp. 194-199. Springer Berlin Heidelberg, 2004.

[15] Bass, Len. Software Architecture in Practice, 2/E. Pearson Education India, 1998.

[16] Grant, Michael, Stephen Boyd, and Yinyu Ye. "CVX: Matlab software for disciplined convex programming." (2008).

[17] Spivey, J.M.: The Z notation: a reference manual, 2nd edn. Programming Research Group, University of Oxford.

[18] Williamson, Richard, and Brian J. Andrews. "Gait event detection for FES using accelerometers and supervised machine learning." Rehabilitation Engineering, IEEE Transactions on 8, no. 3 (2000): 312-319.

[19] Sadlier, David A., and Noel E. O'Connor. "Event detection in field sports video using audio-visual features and a support vector machine." Circuits and Systems for Video Technology, IEEE Transactions on 15, no. 10 (2005): 1225-1233.

[20] Sinclair, Chris, Lyn Pierce, and Sara Matzner. "An application of machine learning to network intrusion detection." In Computer Security Applications Conference, 1999.(ACSAC'99) Proceedings. 15th Annual, pp. 371-377. IEEE, 1999.

[21] Davatzikos, Christos, Kosha Ruparel, Yong Fan, Dinggang Shen, M. Acharyya, James Loughead, Ruben Gur, and Daniel D. Langleben. "Classifying spatial patterns of brain activity with machine learning methods: application to lie detection." Neuroethics Publications (2005): 10.

[22] Hansen, Morten, Morten Kristian Haugland, and Thomas Sinkjær. "Evaluating robustness of gait event detection based on machine learning and natural sensors." Neural Systems and Rehabilitation Engineering, IEEE Transactions on 12, no. 1 (2004): 81-88.

[23] Neill, Daniel B., and Gregory F. Cooper. "A multivariate Bayesian scan statistic for early event detection and characterization." Machine Learning 79, no. 3 (2010): 261-282.

[24] Nikovski, Daniel, and Ankur Jain. "Fast adaptive algorithms for abrupt change detection." Machine learning 79, no. 3 (2010): 283-306.

[25] Tandon, Gaurav, and Philip K. Chan. "Increasing coverage to improve detection of network and host anomalies." Machine learning 79, no. 3 (2010): 307-334.

[26] Šingliar, Tomáš, and Miloš Hauskrecht. "Learning to detect incidents from noisily labeled data." Machine learning 79, no. 3 (2010): 335-354.

[27] Hartigan, John A., and Manchek A. Wong. "Algorithm AS 136: A k-means clustering algorithm." Journal of the Royal Statistical Society. Series C (Applied Statistics) 28, no. 1 (1979): 100-108.

[28] Hall, Mark, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. "The WEKA data mining software: an update." ACM SIGKDD Explorations Newsletter 11, no. 1 (2009): 10-18.