Object Recognition Using Hierarchical Temporal Memory

Fabián Fallas-Moya[1(✉)] and Francisco Torres-Rojas[2]

[1] Universidad de Costa Rica, Cartago, Costa Rica
fabian.fallasmoya@ucr.ac.cr
[2] Instituto Tecnológico de Costa Rica, Cartago, Costa Rica
torresrojas@gmail.com

Abstract. At this time, great effort is being directed toward developing problem-solving technology that mimic human cognitive processes. Research has been done to develop object recognition using Computer Vision for daily tasks such as secure access, traffic management, and robotic behavior. For this research, four different machine learning algorithms have been developed to overcome the computer vision problem of object recognition. Hierarchical temporal memory (HTM) is an emerging technology based on biological methods of the human cortex to learn patterns. This research applied an HTM algorithm to images (video sequences) in order to compare this technique against two others: support vector machines (SVM) and artificial neural networks (ANN). It was concluded that HTM was the most effective.

Keywords: Machine learning · Computer vision · HTM

1 Introduction

One relevant issue in computer science is to try to interpret an image for a specific purpose. This research focuses on recognizing objects in an image with different occlusion percentages. Right now, computers have the computational power to achieve this objective with an acceptable running time.

Singh [1] defines *video tracking* as the process to detect one moving object throughout a video sequence, and a fundamental task of this tracking is to recognize the object in every frame. An object of interest can be: an animal, a machine, a person, etc. A system to detect the existence of objects in a frame was developed. The dataset consisted of video sequences of people, and this gave different challenges related to shape transformation (because of object movements) and occlusion (when the object is partially occluded by other objects). Figure 1 shows an example of occlusion.

The human brain has a *neocortex*. This element is in charge of visual pattern recognition and many other cognitive processes. Hierarchical temporal memory (HTM) is a technology developed to try to mimic the neocortex. It is very similar to artificial neural network (ANN), but with some architectural and conceptual

C. Brito-Loeza and A. Espinosa-Romero (Eds.): ISICS 2018, CCIS 820, pp. 1–14, 2018.
https://doi.org/10.1007/978-3-319-76261-6_1

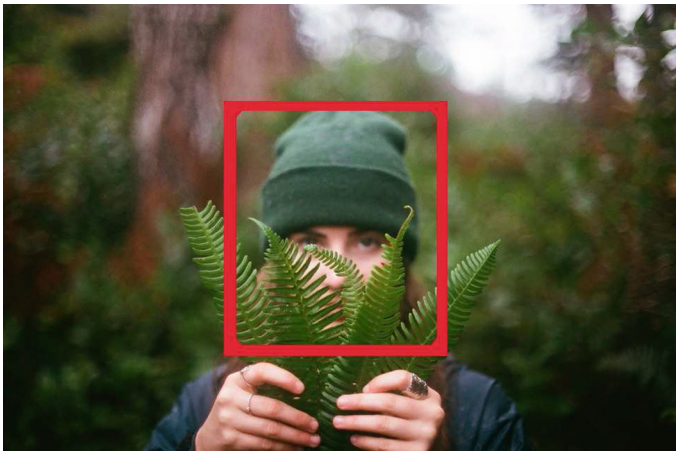

Fig. 1. An example of an occluded face.

differences. For example, the cells[1] are connected to different cells all over the region (see Fig. 2). These connections or *synapses* can change during the training period. Even more, they can change during classification. It depends on the input data. The HTM architecture have cells that are arranged into regions, and regions that can form a complex hierarchy.

This HTM technology was developed by Hawkins [2] and it is closely related to the biological structure of the neocortex that has six layers of regions. This explains why HTM uses the concepts of regions, cells, and hierarchies. The regions can be constructed using different numbers of cells. An important aspect is that the connections inside regions can change over time.

The idea behind hierarchies is to learn complex patterns. This is similar to the idea behind convolutional neural networks, as explained by Fan *et al.* [3]. There, every layer tries to detect gradient features to have an accurate classification process. In the case of HTM, every defined level[2] will be learning different degrees of an image, to get to the final layer. The idea is to learn complex patterns. See Fig. 3.

1.1 HTM and Pattern Learning

As well as in ANN, HTM network has a specific region for the input data. The outcome of this first layer is passed to the following region to be trained, and the process continues for every region until the last one.

HTM uses the concept of sparse distributed representations (SDR). It is a technique to represent different patterns, where a small number of cells are picked to be active.

[1] In the HTM terminology, cells are synonyms of neurons.
[2] The terms level and region will be used interchangeably.

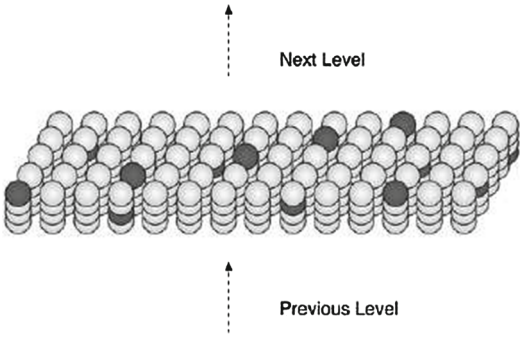

Fig. 2. An HTM region with columns of cells, the cells are connected to different cells. Also, the concept of sparse distributed representations (SDR), the patterns are just a few and distributed cells along the region.

HTM can receive an enormous input. However, every layer needs just a few cells to represent it. Higher layers apply the same concept until the final layer is reached. This process is not uniform or deterministic. This explains why different cells can be picked. In Fig. 2 shows that it is not mandatory for these cells to be close to each other. To summarize, only a few cells are need and these cells can be distributed along the region.

As Schlag [21] explains, the way HTM learns patterns is in a statistical manner, very similar to Bayesian Networks; however, they differ in the hierarchy construction and the usage of time. First, it converts raw data into proper HTM input, using some decoders (libraries developed for HTM). Second, from the data it looks for activations which occur together (spatial patterns). It then searches sequences of these patterns over time (temporal patterns). So, there is not a back propagation step, it does some kind of clustering classification. These learned patterns are used to perform inference on new inputs. In HTM the concept of *time* is important, the order in which the patterns enter the architecture impacts the connection construction in the network.

Just like a biological neuron, an HTM cell has dendrites, proximal and a distal dendrites. The dendrites contain many synapses in order to receive signals. The proximal dendrites receive a feed-forward input from regions in a lower layer. The distal receive their input from neighboring cells which belong to the same HTM region. Every cell is made up of a number of synapses, which can be potential or permanent. During the training process these synapses can change due to the classification result. Every cell can be in a different state: predictive, active and fully active. These states will defined the synapses. This cell flexibility to easily change its connections will help to make a fast training step. According to Schlag [21] (2016), "This increase and decrease in permanence is an important aspect of the Hebbian[3] learning ability of HTMs".

[3] Hebbian Theory defined by Gerstner [22].

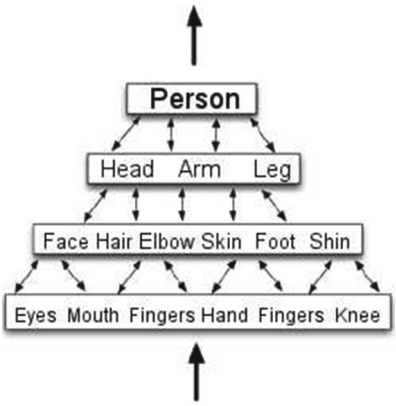

Fig. 3. An HTM hierarchy: in this example, the hierarchy has four levels of regions.

To have a full HTM implementation one can use the online prediction framework (OPF). It provides all components needed to create an architecture. A short description of these components follows.

- Raw input: for example integers.
- Encoder: it turns data into SDR.
- Spatial Pooler: it creates an SDR over a region.
- Temporal Pooler: it links the connections between cells (synapses).
- Cortical Learning Algorithm (CLA): it generates a prediction.

Another way to use HTM technology is to use isolated components. For example, using the *spatial pooler* and the *temporal pooler* to make patterns and these patterns can be connected to a classifier algorithm. The proposed approach involves using the two previous options. First the OPF tool, second, the temporal pooler combined with a high level classifier.

2 The Recognition Algorithms

Occlusion is an interesting challenge, see Fig. 1. The proposed algorithms were developed to overcome this problem. To measure this aspect accuracy[4] was used. In fact, the occlusion success rate (OSR) was used to referred to the accuracy. If the OSR has a high value, it means that with an occlusion challenge, the algorithm has a good performance. Therefore, the OSR was implemented to measure and to compare the different algorithms.

[4] "The accuracy is the proportion of true results (both true positives and true negatives) among the total number of cases examined" [5].

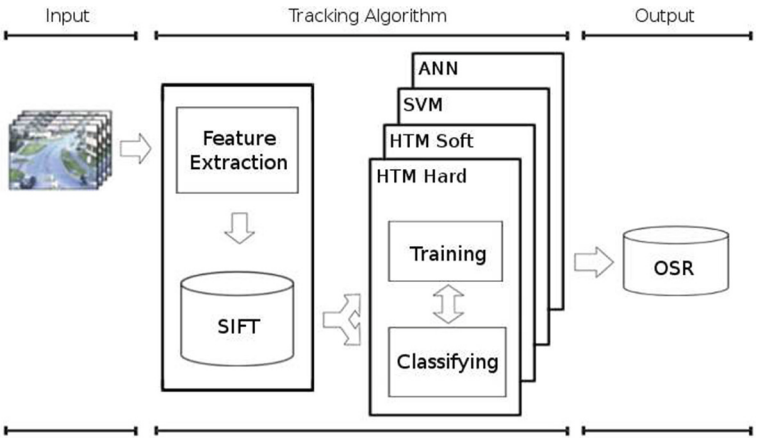

Fig. 4. The main components of the proposed algorithms.

2.1 Feature Extraction

As far as the algorithms are concerned, Fig. 4 shows the proposal. Every algorithm has the same pre-processing step. Scale invariant feature transform (SIFT) [6] was used for extracting features. Solem [7] gives a definition for SIFT: "SIFT features are invariant to rotation, intensity, and scale. In fact, they can be tracked in the presence of noise and 3D rotation". It performs two tasks: interest point detection and description of each point with data. These two tasks generate data to avoid rotation and scale differences, which are common during video frame analysis. The descriptor obtained is a vector with the following data: 16 values that describe the *interest point*, it is a matrix 4×4. And, every cell of the matrix is represented by histogram of 8 values. At the end, there are 128 floating values $(4 \times 4 \times 8 = 128)$ as shown in Fig. 5.

The SIFT process can be seen in Fig. 6. It illustrates a SIFT process over an image. The results are shown on the right of Fig. 6, with the following results: 3365 detected points and since every point has a descriptor of 128 floating values, the total is 430720 values. This quantity was not manageable by the available hardware. As a result, a constraint was implemented: the videos have a resolution of 50 pixels with only 12 interest points, as shown in Fig. 7.

Figure 7 gives an example of one frame. On the left of the image, a person with a red t-shirt is raising their hands. An important aspect is that a variant of the SIFT algorithm was used. It is called *Dense SIFT*. It is different because it allows to choose static points and to set the radiuos of each one. 12 points were chosen to cover the whole image to form a grid over it, for a total of 1536 values. Some advantages of this technique are: the SIFT information for every frame of the video will have the same size for the classifications task. Also, static points will allow to detect changes on the image. This is a well-known technique used by Hassner *et al.* [8] in a similar problem of classification. Also Han *et al.* [9] developed an algorithm using SIFT to classify people.

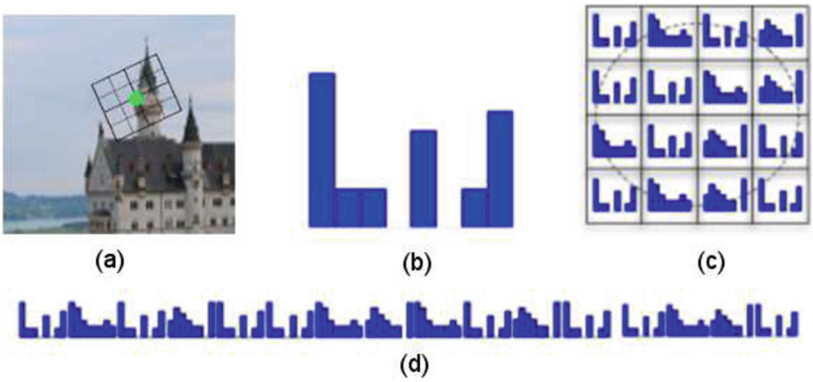

Fig. 5. A point detected by the SIFT algorithm and the descriptor generated around the point. (a) A grid generated around the interesting point. Its orientation is according to the gradient direction. (b) Every cell of the grid is represented by an 8 bin histogram. (c) Get all histograms from the grid. (d) All histograms are concatenated to have a vector. Image from [7].

Fig. 6. SIFT process over an image: it detects about 3365 points.

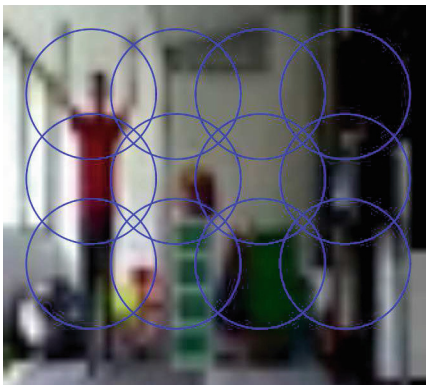

Fig. 7. Dense SIFT: there are 12 static points and they have a predetermined radius.

2.2 Proposed Algorithms

As shown in Fig. 4 there are 4 algorithms. A brief explanation of each algorithm follows.

1. HTM Hard: the word "Hard" means a full or complete implementation of an HTM algorithm. It does mean that all pieces of HTM were used. That is, Encoders, Spatial Pooler, Temporal Pooler, and Cortical Learning Algorithm. A modified version of OPF was used. This modification was implemented because of the streaming data, and it was based on Costa's proposal [10]. Numenta[5] does not recommend the use of streaming data over OPF. Anyway this modification was needed. This explains the long execution time (it lasts 60 times more). This modification was made in the input layer. Basically, it receives only one input value. There is an OPF file called *model params*, there, an for loop was added to read more than one parameter. Here, the 1536 values were read. Algorithm 1 shows the relationship between all the steps in this algorithm.
2. HTM soft: it is the recommended version for image processing. Numenta recommends this algorithm [15], where a single component (in this case the spatial pooler) is connected to a classifier. In this research a k-Nearest Neighbors [11] classifier was implemented with the spatial pooler.
3. SVM: a strong SVM library for the classification task was used, in this case the LibSVM [16]. A linear kernel function was used for this implementation.
4. ANN: the library PyBrain [12] was chosen for implementing this algorithm. A feedforward backpropagation neural network was implemented. The architecture was chosen from previous research, the number of hidden layers by Bishop [13] and the numbers of hidden neurons by Blum [14].

3 Experiments

Design of Experiments (DoE) was used as a statistical model with the R [17] language. It tries to answer if predefined factors have an influence over a study response variable, and, if there is a significant influence, the important issue is to quantify that influence [18]. The ANalysis Of VAriance (ANOVA) is an instrument of DoE. It measures the variance over the factor values against the variance of other predefined factors. OSR is the response variable and it has four factors. Table 1 shows the factors: machine learning technique, training-set size, the percentage of occlusion and the complexity of the scenario. On the one hand, a simple scenario is composed of white background and some objects. On the other hand, a complex scenario has more objects, light contrast, and different backgrounds.

As shown in Table 1, there are 4 techniques, 3 training-sets, 4 occlusion percentages, and 2 scenarios, for a total of 96 combinations. 4 replicas were picked, for a number of 384 runs ($4 \times 3 \times 4 \times 2 = 96 \times 4 = 384$). It is important to

[5] Numenta is the enterprise behind of the HTM technology.

Algorithm 1. HTM Hard

Require: *videoFrames* are Dense SIFT files

```
1: function Classification-task(videoFrames, groundTruth)
2:     resultingFrames ← videoFrames                          ▷ Encode all entries
3:     for i ← 1 to resultingFrames.size do
4:         spatialPoolerResult ← SP(resultingFrames_i)
5:         temporalPoolerResult ← TP(spatialPoolerResult)
6:         prediction ← CorticalLearningAlgo(temporalPoolerResult)
7:
8:         if prediction = groundTruth_i then
9:             δ ← δ + 1
10:        end if
11:    end for
12:    accuracy ← groundTruth.size / δ                        ▷ accuracy = OSR
13:    return accuracy
14: end function
```

highlight that every run is a sequence of frames. For instance, the training phase has $38 + 176 + 474 = 688$ images for training. With 2 scenarios, that is 1376 images. Also there are 4 replicas, for a total of 5504 images to train our algorithms. Regarding to the testing phase, there are 96 runs of 360 frames, for a total of 34560. Since there are 4 replicas, the total number of classification tasks is 138240.

Table 1. Factors and levels for this research.

	Selected factors			
	Techniques	Training sets	Occlusion(%)	Scenarios
Levels	HTM (Hard)	38	75	Simple
	HTM (Soft)	176	50	Complex
	ANN	474	25	
	SVM		0	

The data are composite of low-quality videos (in a resolution of 50×50 pixels). There was a hardware constraint, the OPF requires more RAM than the one available on the machine used (16 GB of RAM). The OPF builds a full structure of HTM which requires a lot of memory.

Another important aspect is regarding to the videos used in this research. These videos were recorded with some important rules to have good statistical results.

1. There are 3 video classes: two different persons moving around, one class per person. And the background with no motion, the final class. These persons were moving with different percentages of occlusion, this percentage does not change throughout the video. See Table 1 for the occlusion percentages.
2. One replica has 4 occlusion percentages, 2 different scenarios (completely different videos), and 3 different classes (8 videos per class). That is, 24 different videos. Since there are 4 replicas, there were 96 videos.
3. Every video has 36 s. Every second has 15 frames, that is, a total number of 540 frames per video. In Table 1 the number of frames taken for training the algorithms can be different (38, 176 and 474).

4 Results and Analysis

DoE needs to accomplish two assumptions. First, the residuals are normally distributed. Second, they are independent with a constant variance, as explained by Anderson and Whitcomb [19]. Fortunately, these assumptions were accomplished and no data transformation was needed. Table 2 shows the output of an ANOVA (Acronym explanation: Df: *Degrees of Freedom*, Sum Sq: *Sum of Squares*, Mean Sq: *Mean of Squares*, F val: *F values*, Pr(>F): *Probability values*).

Table 2. The ANOVA results.

	Df.	Sum-Sq.	Mean-Sq.	F-val	Pr(>F)
Tech	3	0.96	0.32	4.97	**0.0022**
Train	1	0.03	0.03	0.54	0.4623
Occlu	1	0.51	0.51	8.02	**0.0049**
Scenario	1	0.16	0.16	2.50	0.1148
Tech:Train	3	0.41	0.14	2.12	0.0970
Tech:Occlu	3	1.44	0.48	7.49	**7.15e-05**
Train:Occlu	1	0.01	0.01	0.11	0.7369
Tech:Scenario	3	0.31	0.10	1.62	0.1840
Train:Scenario	1	0.06	0.06	0.95	0.3303
Occlu:Scenario	1	0.08	0.08	1.20	0.2746
Tech:Train:Occlu	3	0.06	0.02	0.33	0.8012
Tech:Train:Scenario	3	0.20	0.07	1.05	0.3695
Tech:Occlu:Scenario	3	0.06	0.02	0.30	0.8247
Train:Occlu:Scenario	1	0.03	0.03	0.40	0.5255
Tech:Train:Occlu:Scenario	3	0.06	0.02	0.30	0.8256
Residuals	352	22.57	0.06		

The first important fact of Table 2, is that the probability of *Technique:Occlusion* has a value of 7.15^{-5}, which gives us a significance of 0.0. This means that there is confidence that the interaction of Technique and Occlusion is very significant. With this information, the null research hypothesis can be rejected, which says that no factor affects the response variable.

There are two other interesting facts: the factor *Technique* has a confidence of 99.78%, because its probability value is 0.0022. Similarly with the factor *Occlusion* that gives the confidence of 99.51%, its probability is about 0.0049. Consequently, the factors *Occlusion*, *Technique* and their interaction has a significant affectation over the response variable, the OSR value. With the information about the ANOVA process, the following step is to analyze the significant factors and their interaction.

4.1 Technique (Tech)

Box plots [20] were used to analyzed some aspects. On the one hand, Fig. 8 shows that ANN has the lowest quartile[6] and the mean with the lowest value, which means that ANN fails more than the others.

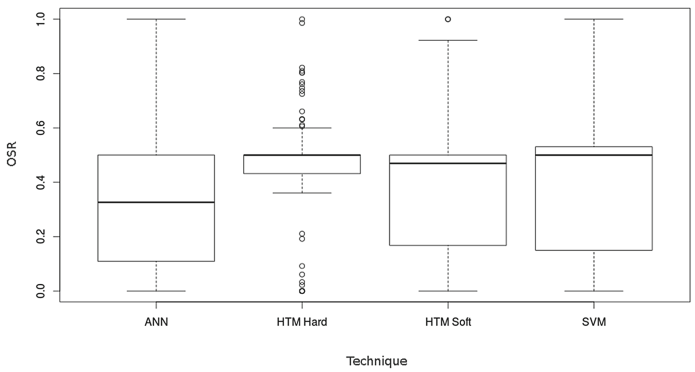

Fig. 8. Box plot for the Technique factor.

On the other hand, the HTM Hard algorithm has its 50% of data enclosed in the thinnest box, also the upper and lower quartile boundaries have the smallest range. The mean is near to 0.5 and it has a lot of outliers, this situation is normal because the upper and lower quartiles form a small range. The lower quartile shows that this algorithm had fewer failures than the others. This algorithm had a stable performance.

[6] Quartile: they are the values that divide a list of numbers into quarters.

4.2 Occlusion (Oclu)

In Fig. 9, there is the box plot of the factor *Occlusion* against OSR. It can be seen that all algorithms fail if the occlusion value is incremented, on 75% the lower boundary of the box is almost 0.0. This is a expected behavior, because the more the object is occluded, the more difficult to hit.

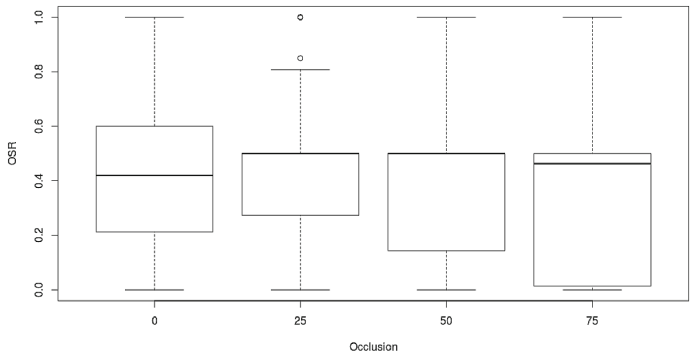

Fig. 9. Box plot for the Occlusion factor.

It can be appreciated that the mean value for 25% and 50% have almost the same value, which indicates it is easier for an algorithm to hit. This can be explained because people have more characteristics from the hips towards the head. Also, there is another interesting aspect: the values for 25%, 50%, and 75%, have their upper box boundary close to 0.5 of accuracy, but, the box of 0% has it close to 0.6, which indicates that with 0% the algorithms tend to fail less. However, its mean has the lower value. This situation tells that under a specific condition, it produces more fails with 0%. An explanation is that with 0% the objects (people) had more freedom to move, and did not have to be behind of an object to catch the occlusion.

4.3 Interaction: Occlusion (Oclu) and Technique (Tech)

The final aspect to analyze is one of the most interesting ones. The interaction between two factors: technique and occlusion. Whitcomb and Anderson give the definition of interaction [19]: "Interactions happen when the effect of one of the factors depends on another factor". Figure 10 shows many relevant situations. The first one is that all algorithms tend to decrease. It is obvious because as the occlusion raises its value, it is more difficult for all algorithms to hit.

Another aspect to note is that HTM Soft and ANN do not have an acceptable performance, unlike SVM and HTM Hard. It was mentioned that with 0% of occlusion, there is a huge challenge: the targets had more freedom to move all over the scene. It is considered that ANN and HTM Soft do not have sensibility

under this condition (target movements). Hence, the good performance of SVM and HTM Hard shows that these algorithms are better implementations for real world systems.

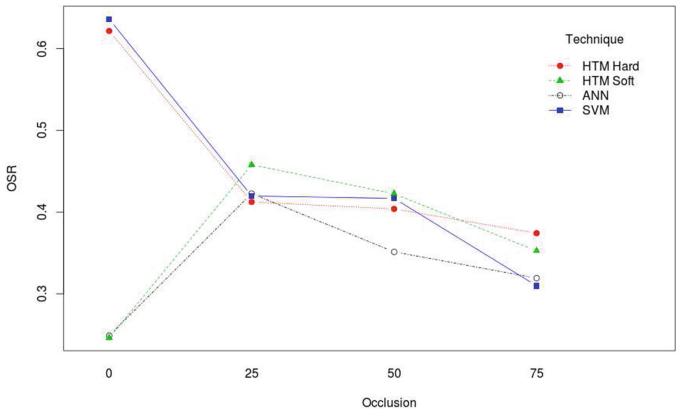

Fig. 10. Interaction of two significant elements (factors): occlusion and technique.

The third aspect analyzed is that SVM has a remarkable behavior on 0%. A stable performance on 25% and 50%, it is similar to HTM Hard. However, it fails on the 75% value. This situation indicates that it could not classify well when the object was almost totally occluded.

Lastly, there is the HTM Hard implementation. It has an excellent performance on 0%, a good sensibility to target movements. Also, good results on 25% and 50%. As seen in Fig. 10, HTM Hard had the best performance on the value of 75% of occlusion. As a result, this is the best algorithm for the occlusion challenge.

5 Conclusions

A modified version of an OPF algorithm was used (HTM Hard). Based on the statistical results, one conclusion is that this algorithm is better to recognize objects than the others. But, it was experienced an unexpected situation: it had the largest running time. It lasts 60 times more comparing with the other implementations. It is not feasible to implement with streaming data.

Another aspect is that SVM had a good performance. During tests, it was the fastest implementation. However, it had a poor performance on 75%. Also, there are the ANN and the HTM Soft performance, which had the worst results. As a result, their implementation is not recommended with the proposed architectures.

All algorithms used SIFT to classify. This technique provides enough information to feed a machine learning algorithm. For classifying purposes, Dense SIFT

was used to have a fixed amount of data per frame. Only 12 SIFT points were used. However, with a limited amount of data the results were acceptable and satisfactory.

Finally, it was discovered that the implementations of HTM Hard and SVM are more sensitive to object movements. They have an acceptable performance with a small quantity of data. HTM Hard is the recommended version for this kind of problem because in real world systems the objects are moving all over the scene.

6 Future Work

The next step is to test different combinations of algorithms. For instance, recurrent neural networks (RNN), convolutional neural networks (CNN), bayesian networks, etc. It would be interesting to use CNN, due to its similarity to HTM (many layers, feature patterns reduction, etc.).

Another interesting aspect is to test different HTM combinations. For example, to implement the usage of a temporal pooler with a classifier (in this research kNN was implemented) such as SVM, ANN, RNN, CNN, etc. Furthermore, high-quality videos can be used and get more information.

Also, to use more points (Dense SIFT values). It can be treated as an hyperparameter and adjust it to get better results.

Finally, another feature extraction technique can be used as well as different preprocessing techniques.

Acknowledgements. We would like to thank our colleagues from the Happy Few Research Group for their support during the development of this research. In addition, we thank the PARMA Group for its guidance and support publishing this research.

References

1. Jalal, A.S., Singh, V.: Visual object tracking: state of art. Int. J. Comput. Inform. **3**, 227–247 (2011). Ljubljana, Slovenia
2. Hawkins, J., Blakeslee, S.: On Intelligence. St. Martin's Griffin, USA (2005)
3. Fan, J., Xu, W., Gong, Y.: Convolutional neural networks: human traffic. IEEE Trans. About Neural Netw., 1610–1623 (2010). https://doi.org/10.1109/TNN.2010.2066286
4. Hawkins, J., Ahmad, S., Dubinsky, D.: Cortical Learning Algorithms and HTM. Numenta Inc., California (2011)
5. Metz, C.E.: Principles of ROC Analysis. University of Chicago and the Franklin McLean Memorial Research Institute, Chicago, USA, pp. 283–298 (1978). 0001-2998/78/0804-0003S02.00/0
6. Lowe, D.G.: Object recognition based on local scale invariant features. In: Computer Vision: International Conference, pp. 1150–1157. IEEE, Canada (1999). https://doi.org/10.1109/ICCV.1999.790410
7. Solem, J.E.: Python: Basics on Computer Vision. O'Reilly Medias, Sebastopol (2012)

8. Hassner, T., Mayzels, V., Zelnik-Manor, L.: About SIFT and its scale. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Rhode Island, USA (2012)
9. Han, B., Li, D., Ji, J.: DSIFT Algorithm for People Detection. Stanford University, California (2011)
10. Costa, A.: A MNIST Classifier Using OPF (2016). http://github.com/allanino/nupic-classifier-mnist
11. Altman, N.S.: Introduction to kernel and nearest-neighbor nonparametric regression. Am. Stat. **46**, 175–185 (1992)
12. Schaul, T., Bayer, J., Wierstra, D., Sun, Y., Felder, M., Sehnke, F., Rückstieß, T., Schmidhuber, J.: The PyBrain library. J. Mach. Learn. Res. **11**, 743–746 (2011)
13. Bishop, C.M.: Pattern Recognition Using Neural Networks. Oxford University Press, Oxford (1995)
14. Blum, A.: Neural Networks (C++). Wiley, New York (1992)
15. Numenta: Nupic: managing vision tasks (2017). http://github.com/numenta
16. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2017). http://www.csie.ntu.edu.tw/cjlin/libsvm/
17. Ihaka, R.: History of R: Past and Future. The University of Auckland, Auckland (1998)
18. Ferré, J., Rius, X.: Introduction to the Statistical Design of Experiments. Universitat Rovira i Virgili, Tarragona (2001)
19. Anderson, M., Whitcomb, P.: Design of Experiments: A Simplified Approach. Taylor and Francis Group, Boca Raton (2007)
20. Massart, D.L., Smeyers-Verbeke, J., Capron, X., Schlesier, K.: Means of Box Plots: Visual Presentation of Data. Vrije Universiteit Brussel, Brussel, Belgium (2005)
21. Schlag, I.: On Hierarchical Temporal Memory (2016). http://ischlag.github.io/2016/04/25/on-hierarchical-temporal-memory
22. Gerstner, W.: Hebbian learning and plasticity. From Neuron to Cognition via Computational Neuroscience, Chap. 9. MIT Press, Cambridge (2011)