

# Classifying via Hierarchical Temporal Memory

Fabián Fallas-Moya

Sede del Atlántico

Universidad de Costa Rica

Cartago, Costa Rica

Email: fabian.fallasmoya@gmail.com

Francisco Torres-Rojas

Escuela de Computación

Instituto Tecnológico de Costa Rica

San José, Costa Rica

Email: torresrojas@gmail.com

**Abstract**—With recently advances in technology (hardware and software) there is an interest of humanity in having machines that behave like humans do. One aspect that researchers have to overcome is how to imitate the cognitive processes of the brain; cognitive processes like visual pattern recognition, speech recognition, space comprehension and so on. This task needs an algorithm that receives raw information from the environment, thus a signal processing method is needed to convert the raw input into useful information. Computer Vision is an interesting field of research, because the process of capturing images is simple and the hardware to process these images is available with current technology. This research focuses on the field of classifying images using hierarchical temporal memory (HTM), a machine learning technique that imitates the neocortex and emulates cognitive processes.

## I. INTRODUCTION

The computing power of a computer is greater than anyone could imagine in the past. The main advantage using computers is their capability to process information faster than a human can do it; in this context the advance in computer vision is remarkable. Computer vision is a field that includes methods for acquiring, processing, analyzing, and understanding images (that come from electronic devices) to produce useful information [13].

Video classification could be applied to a full body tracking (people). This type of classification is a common research issue, due to its relevance in robotics and surveillance. This research proposes a technique to classifying objects (people) using a new algorithm based on hierarchical temporal memory (HTM).

### A. Video classification

Video tracking is the process of finding the location of one or more moving objects in each frame (image) of a video sequence [11], as shown in Figure 1. A *target* is the *object of interest* tracked by a system. One system can track animals, vehicles, micro-organisms or another specific object. However, people has an enormous amount of attention because it can produce important information in areas like public transport, traffic congestion, tourism, security, and business intelligence.

There is an important aspect in a tracking system: to detect if the target is present on a frame. That is why an algorithm to classify can be implemented to detect if a specific target is present on an image or not. It is the first step to track an object and it is the main concern of this research.



Fig. 1: Object tracking: Graphical description. Taken from [1]

### B. Challenges

A video is composed by a sequence of frames. A particularity of video is that its content can be affected by environmental aspects, for example noise or shape transformations. There are some challenges for a video classifying system to overcome [8]. The first one is *clutter*, which happens when the appearance of any object or the background are similar to the appearance of the target, possibly interfering with its observation. Changes in pose also make it harder the video classification process. When a target rotates, it could have a lot of new features never seen before by the system. Another challenge is occlusion, that is to say, when a target is not observed while partially or totally occluded by other objects. For instance, when a person moves behind a car or any other object, some frames from the video sequence lose the target. This last aspect (occlusion) is the main focus of this research.

Video tracking can be defined as a process involving three tasks: feature extraction, target representation, and localization. This research proposes a new algorithm structure, where a classification process is implemented to verify the presence of the target in the frames. Our algorithm uses an emerging technology for classification: HTM networks.

### C. HTM

Hierarchical Temporal Memory (HTM) is a model of algorithms based on the neocortex. HTM can be understood as a specialization of Neural Networks, because it uses the concept of neuron<sup>1</sup> (in HTM is more common use the term “cell” instead of “neuron”) but with a different implementation and topology.

1) *Region*: As Hawkins [6] explains, the concept of regions comes from biology where regions are sets of cells connected to other regions in the neocortex. Figure 2 shows a graphical architecture of a region. An HTM region consists of a large number of cells, arranged into columns connected to other cells. A cell has  $n$  connections.

<sup>1</sup>The terms *cell* and *neuron* will be used interchangeably in this research.

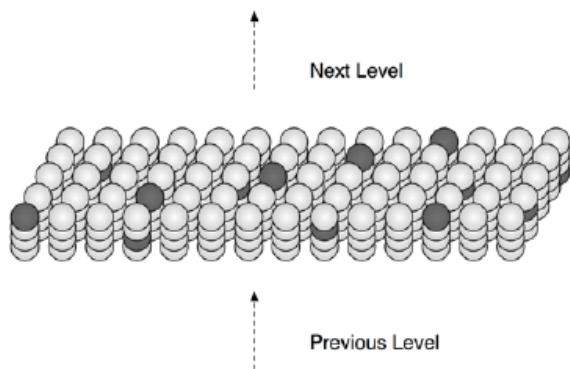


Fig. 2: Region of a HTM: just a few cells are active at a time. Taken from [5, p. 11]

2) *Hierarchy*: A hierarchy is a set of regions distributed into levels. These concepts are very similar to convolutional neural networks [3]. In HTM, each region represents one level in the hierarchy. Lower levels represents *raw* features, and as one ascend through the hierarchy, complex features are represented. Figure 3 illustrates this idea. The first levels have *small* features of an object, which are combined in higher levels to form more complex objects.

3) *Prediction*: HTM can perform inference on inputs [5]. It can match these inputs to previous spatial and temporal patterns. By matching stored sequences with the current input, a region forms a prediction.

4) *Creating structures*: A full (or complete) HTM implementation use the following components:

- 1) Raw data: it is the data in its simplest form: integers, letters, floating numbers, etc.
- 2) Encoders: Transform raw data into sparse distributed representations (SDR<sup>2</sup>).
- 3) Spatial Pooler: receives the SDR and chooses a subset of columns (from its region).
- 4) Temporal Pooler: receives the selected columns from the spatial pooler and connects cells (synapses).
- 5) CLA: turns the prediction of the temporal pooler into the predicted value.

One option to build a full (complete) implementation of HTM is to use the online prediction framework (OPF). It builds a full structure with all components and their connections.

#### D. Training

The training begins in the lower regions up to the higher regions. Every region learns different patterns through cells. These cells are connected to other cells arranged in columns. The cells activation function works with different cells, as seen in Figure 2 it is not necessary that cells must be close to each other.

## II. IMPLEMENTED ALGORITHM

Figure 4 illustrates the implementation of our algorithms. The case of artificial neural networks (ANN) and support

<sup>2</sup>All cells are interconnected, but only a small number of them are active at one time. *Sparse*: only a small quantity of cells are active at a time. *Distributed* means that the activation of different cells are needed to represent a pattern.

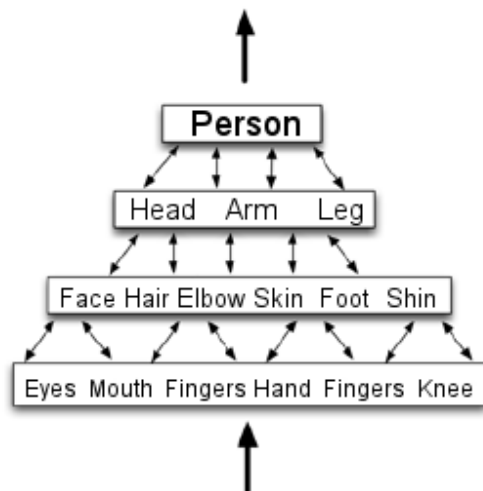


Fig. 3: HTM hierarchy: a hierarchy comprised of four levels (regions).

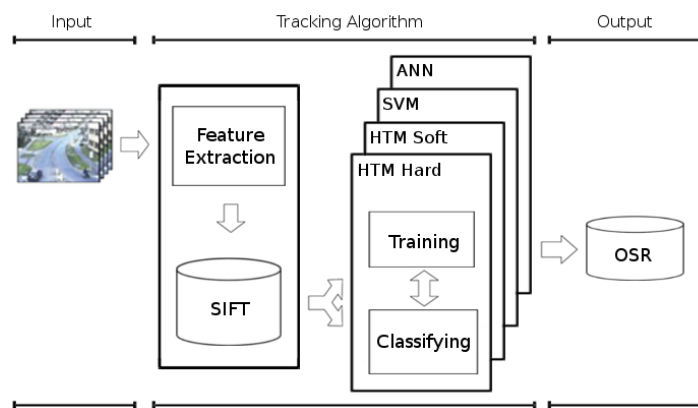


Fig. 4: The Object Tracking algorithm structures, developed for this research.

vector machines (SVM) are similar. They receive the data directly to a classifier. ANN uses a feedforward backpropagation structure, using the library PyBrain. The SVM uses the LIBSVM library. Regarding to HTM Soft, it combines a temporal pooler with a k-nearest neighbor (kNN) classifier [10].

We use the name “Hard” to describe a full or complete implementation of HTM, using all the HTM components. It receives raw data and process it with all the component shown in Figure 5. Talking of raw data, it is obtained by the feature extraction step, for this we used scale-invariant feature transform (SIFT).

#### A. SIFT

The first task to do is the feature extraction (see Figure 4). SIFT was used to achieved this goal. It is one of the most successful feature extraction techniques [7]. It does two main actions: detects interest points from an image (called *interest point detector*) and gets a local descriptor from each point (called the *descriptor*). Its main advantage is to avoid differences in rotation and scale.



Fig. 5: A complete structure of a HTM network. It is a complete hierarchy.

Figure 6 explains the whole SIFT process. First, all interest points from the image are detected using *difference-of-Gaussian* functions [7]. Then, as Solem [12] describes, it starts by taking the dominant gradient of the point based on the direction and magnitude of the image gradient around each point. Next, it takes a grid of subregions around the point, and for each computes an image gradient orientation histogram. Finally, the histograms are concatenated to form a descriptor vector.

Over a high-quality image, we can have approximately 430720 SIFT points. This quantity is too big for the hardware used on this research. As a result, some restrictions were imposed: the usage of images of  $50 \times 50$  pixels and 12 interest points.

### B. Dense SIFT

Figure 7 shows an example of an image used for doing our classification process. It has a person on the left side wearing a red shirt and raising his hands. It is important to indicate that this implementation is different from the normal SIFT technique. Because the points are chosen statically with a fixed radius. The benefits of this approach is to have SIFT files of the same size and to recognize any changes in the image.

In summary, *dense* SIFT can be described as [4]: (a) the location of each keypoint is not from the gradient feature of the pixel, but from a predesigned location; (b) the scale of each keypoint is all the same which is also predesigned; (c) the orientation of each keypoint is always zero. With this assumptions, *dense* SIFT can acquire more feature in less time than SIFT does. As Han *et al.* [4] did, this research focuses on people.

### C. A new algorithm

In order to create a complete structure of an HTM network (as seen in Figure 5) the OPF tool was used. This is a framework which creates an HTM structure with all components of this technology.

That is why a deep modification had to be done (based on Costa's proposal [2]), because a classifier is needed. The first step was to create a model in OPF, this model is created to handle streaming data but in a low scale, for example receiving one to ten values at a time. We use 12 interest points (from a *Dense SIFT* process), each point has 128 values (from the *descriptor*), for a total of 1536 values. We modified an OPF model to accept 1536 floating values. The *Spatial Pooler* and the *Temporal Pooler* were modified to accept our input size (185856 values).

As can be seen on Figure 5 there is a sequence process. First, the raw data (in our case SIFT files) is converted into SDR data by the *encoders*. Then, these SDR are processed by the *Spatial Pooler* and its output is processed by the *Temporal Pooler*, and finally by the CLA. The latter one gives a predicted

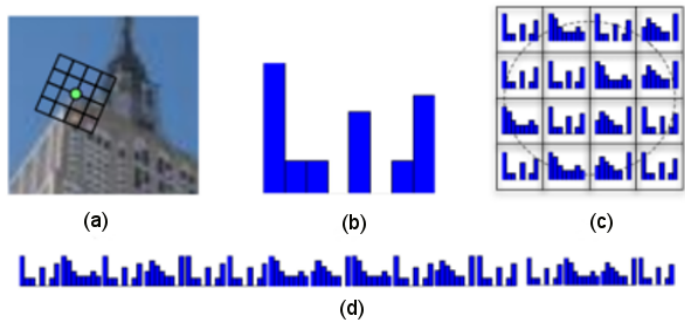


Fig. 6: One SIFT point. (a) a frame around an interest point - according to the dominant gradient. (b) an 8 bin histogram over the direction of the gradient. (c) histograms are extracted in each grid location. (d) the histograms concatenated. Taken from [12]

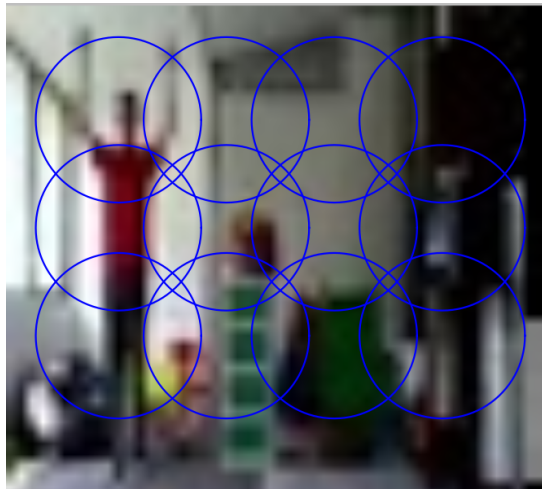


Fig. 7: Implementation of SIFT. It uses a technique called *dense* SIFT, establishing 12 points and increasing the radius of each one. This image is taken from one of the videos used for this research.

value, that is, our classification result. The last step of the algorithm is to calculate the Occlusion Success Rate (**OSR**), it is the accuracy of the algorithm.

The training algorithm is similar as the shown on Figure 5, the difference is that the learning process is activated on training and there is no calculation of the OSR.

## III. RESULTS AND ANALYSIS

Our algorithm was compared with another three implementations: Artificial Neural Network, Support Vector Machine and HTM Soft (using the *Temporal Pooler* connected with a K-Nearest Neighbor classifier). Figure 8 shows the box plot (as define by Massart *et al.* [9]) as a result of these four implementations. This box plot helps to identify the middle 50% of the data, the median (the thick horizontal line inside the boxes) and the extreme points. One aspect is that the technique of ANN has the lowest quartile<sup>3</sup> and the lowest median, showing that this technique tends to failed more than the others.

<sup>3</sup>The quartiles of a ranked set of data values are the three points that divide the data set into four equal groups, each group comprising a quarter of the data.

HTM Hard and SVM have similar boxes and median, but SVM has the upper quartile and the median slightly higher, given better results. Also the maximum point of the SVM and ANN reaches the accuracy value of 1.0 unlike the others, showing that these technique were accurate in some runs (under specific conditions).

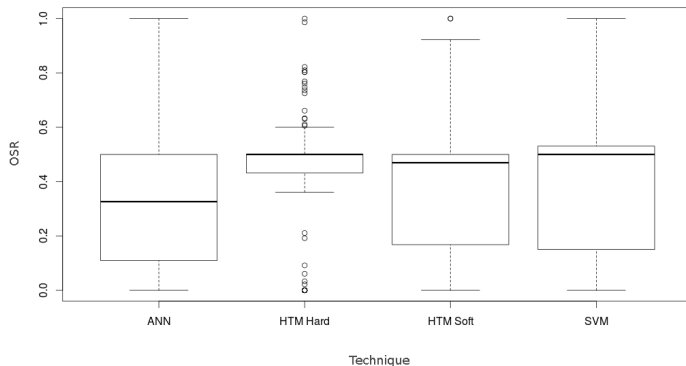


Fig. 8: Box plot for the significant factor Technique.

Finally, the most interesting aspect is the HTM Hard box encloses its 50% of values in the thinnest box and the smallest range from the upper quartile to the maximum point and from the lower quartile to the minimum point. This means that most of the values from this technique are around the accuracy value of 0.5. This value is a similar to the others, but it shows that this particular one tends to have less failures during tests. The Figure 8 also shows many outliers, which is normal because of the small range of quartiles. In comparison with the other techniques, these outliers represents that failures and hits are atypical. Anyway, it is the most stable one (because its small range).

#### IV. CONCLUSIONS

This research has shown the development of a new algorithm using the OPF tool to implement an HTM structure to classify patterns. The evidence shows that the proposed algorithm has a higher accuracy than using other classification techniques. However, the execution time is considerably higher than for other implementations. ANN, SVM and HTM Soft ran fast on a limited hardware (an Intel i7 processor with 8GB of RAM). Approximately, they had a run time of 5 seconds (per run). However, HTM Hard lasted 20 minutes per run, which is considerably higher than the other techniques.

In addition, more evidence was provided to support the idea of using SIFT features for pattern recognition. The well known technique of *dense* SIFT was used, and due to the given results, it was found that it is an excellent tool for feature extraction. Furthermore, it was demonstrated that with few SIFT points the results are satisfactory. It is not necessary to process big amount of information to get acceptable results.

#### V. FUTURE WORK

It would be interesting to use another combination of technologies for the HTM Soft algorithm. For this research we combined the temporal pooler with the kNN classifier.

However, the spatial pooler and the temporal pooler can be combined with different technologies such as: recurrent neural networks, Bayesian networks, reinforcement learning, sparse dictionary learning, genetic algorithms, etc.

Also, it can be used high resolution video. Here the resolution of  $50 \times 50$  pixels resolutions was used, due to some constraints. And used more SIFT points (more than 12), too.

Furthermore, to combine classifying processes with *state-of-the-art* tracking algorithms. For example the *particle filtering* technique. This combination can help to overcome a lot more of challenges, such as: clutter, rotation, illumination, etc.

#### REFERENCES

- [1] Reading University Computational. *Performance Evaluation of Tracking and Surveillance. A dataset for Computer Vision and Pattern Recognition*. 2009. URL: [www.cvg.rdg.ac.uk/PETS2009/a.html](http://www.cvg.rdg.ac.uk/PETS2009/a.html) (visited on 03/15/2009).
- [2] Allan Costa. *Nupic Classifier MNIST*. 2014. URL: <http://github.com/allanino/nupic-classifier-mnist> (visited on 08/08/2014).
- [3] Jialue Fan, Wei Xu, and Yihong Gong. "Human Tracking Using Convolutional Neural Networks". In: *IEEE Transactions on Neural Networks* 21.10 (2010), pp. 1610–1623. DOI: 10.1109/TNN.2010.2066286.
- [4] Bing Han, Dingyi Li, and Jia Ji. "People Detection with DSIFT Algorithm". In: (2011).
- [5] Jeff Hawkins, Subutai Ahmad, and Donna Dubinsky. "Hierarchical Temporal Memory and Cortical Learning Algorithms". In: (2011).
- [6] Jeff Hawkins and Sandra Blakeslee. *On Intelligence*. USA: St. Martin's Griffin, 2005.
- [7] David G. Lowe. "Object recognition from local scale-invariant features." In: *International Conference on Computer Vision* (1999), pp. 1150–1157. DOI: 10.1109/ICCV.1999.790410.
- [8] Emilio Maggio and Andrea Cavallaro. *Video Tracking: Theory and Practice*. UK: Wiley: a John Wiley and Sons, Ltd, 2011.
- [9] D.L. Massart et al. "Visual Presentation of Data by Means of Box Plots". In: (2005).
- [10] Numenta. *Numenta: nupic vision*. 2015. URL: <http://github.com/numenta/nupic.vision> (visited on 11/05/2015).
- [11] Anand Singh Jal al and Vrijendra Singh. "The State-of-the-Art in Visual Object Tracking". In: *Informatica: An International Journal of Computing and Informatics* 36.3 (2011), pp. 227–247.
- [12] Jan Erik Solem. *Programming Computer Vision with Python*. USA: O'Reilly Media, 2012.
- [13] Wikipedia. *Computer Vision – Wikipedia, The Free Encyclopedia*. 2014. URL: [http://en.wikipedia.org/wiki/Computer\\_vision](http://en.wikipedia.org/wiki/Computer_vision) (visited on 05/19/2014).