Action categorization from video sequences

Jean-Christophe Baillie[1] and **Jean-Gabriel Ganascia**[2]

Abstract.

This article presents a framework for extracting relevant qualitative chunks from a video sequence. The notion of qualitative descriptors, used to perform the qualitative extraction, will be first described. A grouping algorithm operates on the qualitative descriptions to generate a real-time qualitative segmentation of the image flow. Then, simple pattern recognition methods are used to extract abstract description of basic actions such as "push", "take" or "pull". The method proposed here provides an unsupervised learning technique to generate abstract description of actions from a video sequence.

1 INTRODUCTION

1.1 The perception problem

Perception is probably one of the most challenging problems in Artificial Intelligence. Even if we could design a smart system to compute difficult and subtle reasoning inferences, we still need a strong perception capability for the system to be able to learn and to be of some use in the real world. Pattern recognition [3, 1], artificial vision or active vision [2] are some well known research areas that address this problem.

In this work, we focus our attention on symbolic representation, which means that we will not discuss the approaches based on connectionist, non-symbolic views. We assume that the aim of the system is to generate a complete or partial symbolic representation of the world it perceives. In 1998, Stuart Russel proposed a challenge for Artificial Intelligence: building a system capable of driving a car from Paolo Alto to San Francisco. His point of view was to use only numerical, non symbolic information with no internal representation of the problem. This contradicts the usual trends of classical AI, which are mainly concerned with symbolic data. The approach we propose lies somewhere between these two views, as we use symbolic data flows computed from numerical data flows. In fact, we believe the whole perception problem starts with this well-known transition from numerical data to symbolic and qualitative data. This will be the main scope of the article.

1.2 Qualitative modules and psychocognitive analysis

The model we propose is to use what we call *qualitative descriptors*, that we shall describe in more detail below. However, it would be interesting to start with the psychocognitive background that guided us in this research.

[1] LIP6 - Université Pierre et Marie Curie - 4, place Jussieu 75005 PARIS - Jean-Christophe.Baillie@lip6.fr tel : 01 44 27 88 03 fax : 01 44 27 70 00

[2] LIP6 - Université Pierre et Marie Curie - 4, place Jussieu 75005 PARIS - Jean-Gabriel.Ganascia@lip6.fr fax : 01 44 27 70 00

We start with the classical idea that a complex problem should be divided into small parts to facilitate its solution. Therefore, we will consider the problem of perception as the extraction of different qualitative features from a raw data flow. Each qualitative feature is to be taken in charge by a qualitative descriptor. But our perception capabilities seem to go a little further than just extracting a collection of qualitative features from a data flow. Let us take an example: on a table, we see two red balls, one rolling to the right edge of the table. Our claim is that we do not only perceive two balls in a flow of qualitative positions, with a flow of qualitative colors. We also perceive the fact that they are of the same color, that one ball is moving with constant speed to the right and that the distance between the two balls is increasing. Qualitative descriptors allow us to do exactly the same, by generating symbolic, qualitative description flows from the data flows and by extracting qualitative regularities from these qualitative flows.

2 DESCRIPTION OF THE PROBLEM

In computational terms, our problem is to design a transition system that allows us to transform quantitative data into qualitative data. But we want a little more than just a translation from numeric to symbolic. We also want, as we said in the previous section, a system capable of extracting qualitative properties of the data and relations between them. Our system should be designed as follows:

Input: raw data flow, most of the time numerical (for example, images, numerical flows), but not only.

Output: a qualitative flow and a set of qualitative properties describing certain qualitative regularities in the flow.

Constraint: real-time processing

Properties: a modular system, each qualitative property being described by a single qualitative descriptor. Because they are reusable, the set of qualitative descriptors is a kind of toolbox.

Qualitative descriptors are the tools we use to build such a system.

3 QUALITATIVE DESCRIPTORS

A qualitative descriptor can be seen as a qualitative processor, described by a quadruplet (D, Q, T, G). It receives a flow of data as an input and produces a flow of qualitative, symbolic data as an output. The type of input data will be called the domain D of the descriptor. In practice, this domain can be anything, not only a numerical domain. For example, the domain of a qualitative descriptor can be the output of another qualitative descriptor. This leads to networks of interconnected descriptors, as we shall see below.

A qualitative transform function T is used to change the data of D into qualitative data of Q, the qualitative description domain. The

elements of Q will be called *qualitative types*. The important idea is that there is a generalization function G from Q^2 to Q int order to generalize two qualitative types of Q. This particularity is used to detect regularities and common points inside the qualitative flow.

Usually, Q will be a set of first order predicates, for it defines a natural generalization function G.

3.1 Order of descriptors

The notion of descriptor order is very important. Some descriptors transform one element of D into one of Q. This is a simple function that abstracts information, according to a specific qualitative perspective. It will be called a qualitative descriptor with order 1. For example, a very simple first order descriptor could be working on the sign of a real number, which is a qualitative feature of this number. If $x < 0$ then $T(x) = [neg]$ else $T(x) = [pos]$. Note that we adopt the convention of writing elements of Q between square brackets.

Some descriptors need two elements of D to produce a qualitative response in Q. The typical example is a descriptor stating whether two numbers are in decreasing or increasing order. If $x > y$ then $T(x, y) = [>]$ else $T(x, y) = [<]$. They are called second order descriptors.

In a way, the order of a descriptor can be seen as the arity of the function T. First order descriptors just transform one piece of information into another. Second order descriptors give information about the evolution of some elements of D. One could speak of static and dynamic descriptors and the notion of order is then closely related to the notion of derivative order in mathematics.

3.2 Examples

Here are some typical examples of qualitative descriptors. The first and second ones are not directly used in the action perception problem but can be useful in other situations (see [4]). The third is used in the action perception problem and is quite typical too since it uses predicates as an output.

3.2.1 Alignment descriptor

For this qualitative descriptor, the domain is the set of points in the plan. It is a second order descriptor, and therefore it needs two points as an input. The qualitative result given by the function T is a description of the line going through the two points. The qualitative representation of a line (i.e. the elements of Q) are couples made up of a director vector and one point of the line, such as $[D(\vec{v}, P)]$. Two lines can be generalized into a line, a vectorial line or nothing, as follows:

$$\begin{aligned} G(D(\vec{v_1}, P_1), D(\vec{v_1}, P_1)) &= D(\vec{v_1}, P_1) \\ G(D(\vec{v_1}, P_1), D(\vec{v_1}, P_2)) &= D(\vec{v_1}, *) \\ G(D(\vec{v_1}, P_1), D(\vec{v_2}, P_2)) &= D(*, *) \end{aligned}$$

Note that we have not expressed how points are described for this descriptor. In any practical implementation, a cartesian representation, using a given reference, will be appropriate. In this case, the domain becomes $\mathbb{R}^2$, but it is not necessary.

3.2.2 Monotonicity descriptor

This descriptor operates on number flows. Its domain is $\mathbb{R}$. It is a second order descriptor which describes the monotonicity of a series of numbers. The function T is:

$$\begin{aligned} \text{if} \quad x > y, \quad T(x, y) &= [dec] \\ \text{if} \quad x < y, \quad T(x, y) &= [inc] \\ \text{if} \quad x = y, \quad T(x, y) &= [equal(x)] \end{aligned}$$

As a generalization, we could propose $G([equal(x)], [equal(y)]) = [equal(*)]$. In other cases, $G(X, Y) = [*]$ if $X \neq Y$ and $G(X, X) = X$. This could allow the system to recognize a staircase function for example.

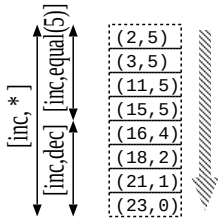
3.2.3 Move descriptor

This qualitative descriptor is given a speed as an input. The domain is then a vectorial space. If the speed module is not zero, with a certain tolerance, the qualitative result given by T is a predicate *move*. The generalization is the natural generalization used on predicates. In practice, the vector in input is associated with a label to name the object whose speed is given. Here, we call it a for example.

$$\begin{aligned} \text{if} \quad |\vec{v}| > \epsilon, \quad T(\vec{v}) &= [move(a)] \\ \text{if} \quad |\vec{v}| \leq \epsilon, \quad T(\vec{v}) &= [] \end{aligned}$$

3.3 Grouping algorithm

A grouping algorithm has been designed (see [4]) to use the generalization capabilities of qualitative descriptors. This algorithm works in real time to provide a qualitative flow of elements of Q from a flow of element of D, given a specific qualitative descriptor. To illustrate the results of this algorithm, we will use a simple example. Let us consider a flow of couple (a, b) where a and b are numbers. We will use the algorithm together with a monotonicity descriptor operating on both elements of the couple. The following figure describe the expected result when running the algorithm.



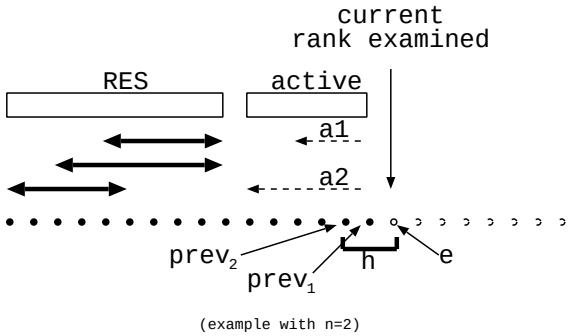
The labeled arrows on the left are qualitative groupings generated by the algorithm. They represent maximal groupings in the sense that it is impossible to extend them without changing their qualitative type. The algorithm generates every maximal grouping, according to the generalization specification of the qualitative descriptor.

3.3.1 Short description

Here is a short description of the algorithm. We consider a flow of elements of D, analyzed by a given descriptor, with order n. The elements of the flow are given in real time to the algorithm, one by one. In the following, we shall call *grouping* a series of elements of the flow, consecutive, and bounded by the rank of the first and the last element. A grouping is also associated a qualitative type, i.e. an element of Q, written $QT(g)$ for a grouping g.

Two sets of groupings are used in this algorithm: RES and $active$. Initially, they are empty sets. As the algorithm is run, RES will be filled with the groupings already built (i.e. whose final rank is strictly smaller than the rank currently being processed). The groupings in RES are *closed* groupings. In $active$, we find the groupings currently being built.

To see how it works, let us suppose the algorithm has already started and is in progress:



The RES list has three groupings. The $active$ list contains a_1 and a_2. e is the current flow element that has to be integrated. $prev_k$ is the element located k places to the left of e ($prev_0$ is e itself, $prev_1$ is the previous element, added just before e). First, the algorithm builds a small grouping, h, whose extension ranges from $prev_n$ to e and with qualitative type $T(prev_n, prev_{n-1}, ..., prev_1, e)$. Then, for each grouping a in $active$, a is extended to include e if $QT(a) = QT(h)$. If not, then a new grouping is created, with the extension of a plus e and the qualitative type of a generalized with the qualitative type of h, according to G. This new grouping is called g. If $QT(a) = QT(g)$, a is replaced by g in the $active$ list. Else, a moves from $active$ to RES since it cannot be extended without changing its qualitative type. If there is no grouping in $active$ with the same qualitative type as g, then g is to be added to $active$. We proceed through the entire $active$ list in this way. If there is no grouping a such that $QT(a) = QT(h)$, h is added to $active$.

At the end, the algorithm just copies the groupings of $active$ into RES and then stops.

4 APPLICATION

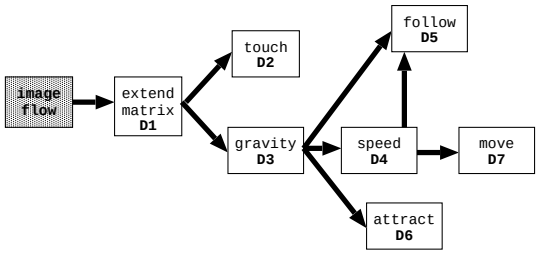
4.1 Problem

Our main application is related to action perception. We give the system a series of video sequences showing simple basic actions like "take", "push" or "pull". This is unsupervised learning since the sequences are not labeled and, for this reason, we have here a different point of view from [5, 9, 8]. To be able to learn from video sequences, there are two main stages: first to create a qualitative flow from the image flow and then to extract regular patterns from this qualitative flow. In this article we will focus on the first part mainly, although we shall briefly present an outline of the second part at the end.

Note that our preoccupation is not really image segmentation or artificial vision. Therefore, the objects in the video sequences are very easy to recognize, clzarly colored on a black background. This simplification allows the use of simple, yet robust algorithms to do the image segmentation.

4.2 Qualitative descriptors used

As already mentioned, qualitative descriptors can be connected one to the other. The figure below presents the structure we use for the qualitative analyzer:



The qualitative descriptors used are described below and the Input, Output and Transfer functions given in detail:

D1: extend matrix

Input: the raw image flow

Ouput: an occupation grid where every pixel of the image is associated with a number corresponding to the label of an object.

Transfer function: image segmentation. The descriptor extracts the number of objects and their spatial extension. Note that the image is made up of clearly colored objects that are easy to distinguish. A simple region-growing algorithm is used.

D2: touch

Input: an occupation grid from $D1$.

Ouput: a predicate indicating what objects are touching. The objects are named with the labels from the occupation grid. The predicate is $touch$.

Transfer function: uses the occupation grid to detect two touching points (with tolerance) with different labels.

D3: gravity

Input: an occupation grid from $D1$.

Output: a set of positions corresponding to the gravity centers of the objects

Transfer function: calculates the isobarycenter of the set of all points of a given label.

D4: speed

Input: a set of positions from $D3$.

Output: a set of vectors corresponding to the instant speed of the objects. This descriptor is a the second order one.

Transfer function: simple discrete derivative.

D5: follow

Input: both the gravity centers and the speeds of the objects. This is an example of a more complex domain.

Output: a predicate indicating which objects are following or avoiding each other. Predicates: $follow, avoid$.

Transfer function: an object A is following an object B if its speed or the variation of its speed is directed in a cone pointing to the center of gravity of B.

D6: attract

Input: a set of positions from $D3$.

Output: a predicate indicating which objects are attracting, repulsing or equidistant. Predicates: $attract, repulse, equid$.

Transfer function: an object A is attracting an object B if their relative distance is decreasing.

D7: move

Input: a set of speeds from $D4$.

Output: a predicate indicating which objects are moving or immobile. Predicates: $move, immobile$.

Transfer function: compares the module of the speed with zero, with a tolerance ϵ.

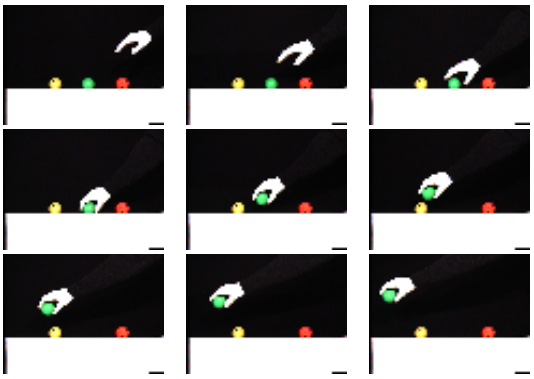
4.3 Tracks

With this structure of qualitative descriptors, we only generate qualitative flows. To benefit from the generalization capabilities of the qualitative descriptors, we should use the grouping algorithm with the qualitative flow. To connect the algorithm to this structure of qualitative descriptors, we can plug some *tracks* on to the output of any descriptor. A track records the qualitative flow produced by the descriptor, then, the grouping algorithm works on it to produce useful qualitative groupings according to the generalization function of the descriptor.

5 RESULTS

The grouping algorithm and the qualitative descriptor structure have been implemented on a pentium 400MHz, with a Sony EVI-D31 camera for the video sequences.

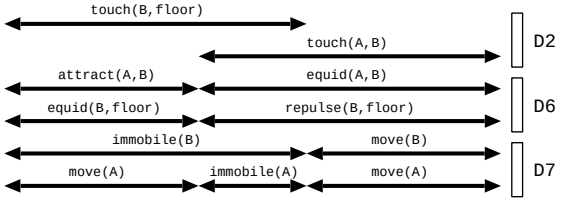
Here is a typical sequence given to the system:



We have only one image out of five here, the complete sequence is made up of 80 images.

We plugged a track on $D2$, $D6$ and $D7$ to get the following qualitative groupings:

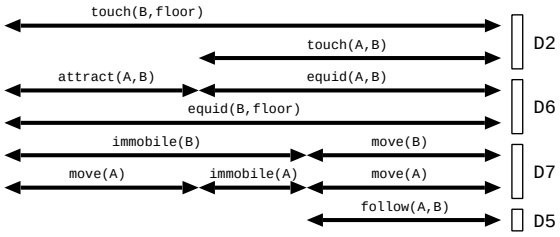
SEQUENCE 1



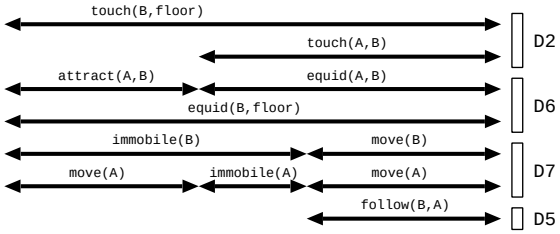
The names of the predicates are explicit enough, and we can recognize a typical "take" sequence. The objects have been given explicit labels, so we have here an object A taking an object B left on the $floor$.

Another interesting sequence are "push" and "pull". To be able to distinguish between these two sequences, we have to plug a track on $D5$:

SEQUENCE 2 (push)



SEQUENCE 3 (pull)



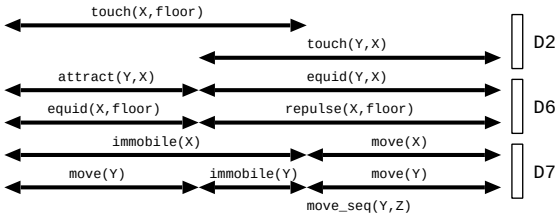
As we can see, the only difference between these actions is the fact that A follows B or the contrary.

6 EXTRACTING ABSTRACT QUALITATVE PATTERNS

6.1 Principle

Our future workThe next part of our work will be to extract some regular patterns from the qualitative flow. In the previous example, the case of "take" could be generalized by:

TAKE (generalization)



The difficulty lies in the fact that this qualitative sequence is lost in a wider qualitative flow. Therefore, we have to extract this subsequence from the global flow. This is indeed the well-known problem of pattern extraction (see [7, 6]).

To do so, we can use a simple algorithm with complexity $O(p^2)$, p being the number of elementary sub-sequences to be compared one to the other. There are other algorithms, working in $O(p)$ but with a

proportionality factor α exponential with the size of the predicates. They are not necessarily better for our particular problem of action perception, since the factor α may be greater than p.

7 CONCLUSION

The notion of qualitative descriptor allows the formalization of the transition from numerical data flow to qualitative flow. This tool is general enough to be used in any situation and yet can be processed by the same algorithm in all cases. It creates qualitative flow and qualitative groupings, using the generalization capabilities of the descriptors. It has been used successfully in the example of action perception and produces a qualitative flow in which it seems possible to recognize typical "take" or "push" sequences, with simple pattern extraction algorithms.

Modularity is also an advantage for the clarity and the ease of use of the qualitative notions involved.

Many points are still to be studied, both in practical and theoretical terms. One very interesting question is the ability to learn new descriptors. Such learning could be based on the building of macro descriptors: in the action perception problem, for example, once the system has identified a new action, it could create a descriptor for it. We would then have a "take" descriptor or a "push" descriptor. Another interesting question is to know how to choose the descriptors for a specific problem. The fact is that, given the type constraint on a description input, there are not many possibilities and therefore the right question would be to know if there is a "natural" set of elementary descriptors that could be used in any practical situation or if we have to define a new set for each application. They is agood chance that a "natural" set of elementary descriptors can be defined. This would lead to the building of a kind of qualitative descriptor toolbox, with possible cross references to cognitive psychology. Finally, we hope to build a system capable of creating its own abstraction from observation and if possible from interaction with its environment, reproducing thus part of the capabilities of a young baby.

REFERENCES

[1] J. Aloimonos, A. Basu, and C. M. Brown, 'Contour, orientation and motion', in *Proceedings: Image Understanding Workshop (Miami Beach, FL, December 9–10, 1985)*, ed., L. S. Baumann, pp. 129–136, San Mateo, CA, (1985). Defense Advanced Research Projects Agency, Morgan Kaufmann.

[2] J. Aloimonos, I. Weiss, and A. Bandyopadhyay, 'Active vision', *International Journal of Computer Vision*, **1**, 333–356, (1988).

[3] A. Bijaoui, *Image et information : introduction au traitement numrique des images*, Masson, 1981.

[4] Baillie J.-C. and Ganascia J.-G., 'Segmentation qualitative sur des series de donnees', *Actes de CAP'99, Conference d'apprentissage*, 1–8, (1999).

[5] R. Mann, A. Jepson, and J.M. Siskind, 'The computational perception of scene dynamics', *CVIU*, **65**(2), 113–128, (February 1997).

[6] Rolland P.-Y. and Ganascia J.-G., 'Flexpat : a novel algorithm for musical pattern discovery', *Proceedings of the 12th Colloquium on Musical Informatics*, 125–128, (1998).

[7] Rolland P.-Y. and Ganascia J.-G., 'Flexible mining of patterns in sequential data (paraitre)', *ICAI99*, (1999).

[8] J.M. Siskind and Q. Morris, 'A maximum-likelihood approach to visual event classification', in *ECCV96*, pp. II:347–360, (1996).

[9] Hirochika Inoue Yasuo Kuniyoshi, 'Qualitative recognition of ongoing human action sequences', *Robotics and vision*, 1600–1609, (1993).