EFFICIENT SIMULATION FOR PROLOG IMPLEMENTATION OF IMAGE RECOGNITION PROBLEM

Tatiana Kosovskaya, Maria Vlasova

Abstract: *Pattern recognition problem using properties of parts of a recognizable objects and relations between them is NP-hard. That is why quick recognition of an object with many elements is a difficult problem. To overcome such a difficulty the use of upper bound of steps for an algorithm solving such a problem based on logic-objective approach is suggested. The possibility of the use of Prolog language family for image recognition is investigated. An example of such a problem for which the use of these upper bounds of complexity permits to decrease the time of its solution is presented.*

Keywords: *pattern recognition, predicate calculus, complexity of algorithms, Prolog language.*

ACM Classification Keywords: *I.2.4 ARTIFICIAL INTELLIGENCE Knowledge Representation Formalisms and Methods – Predicate logic, I.5.1 PATTERN RECOGNITION Models – Deterministic, F.2.2 Nonnumerical Algorithms and Problems – Complexity of proof procedures.*

Introduction

Recognition of an image on the display screen is an important problem of Artificial Intelligence. One of the approaches to its decision is that based on the use of predicate calculus [Kosovskaya, 2011].

Let every recognisable object ω be a set of its elements $\omega=\{\omega_1,\ldots,\omega_l\}$. Let Ω be a collection of objects. Any subset τ of ω will be called its part. A partition $\Omega = \cup_{j=1}^{K}\Omega_j$ of the set Ω on K (possibly intersected) classes is done. A set of predicates $p_1,\ldots, p_n$ characterizes the properties of and relations between the elements of ω.

Logical description $S(\omega)$ of an object ω is a set of all true constant formulas of the type $p_i(\bar{\tau})$ or $\neg p_i(\bar{\tau})$ calculated for all possible parts τ of the object ω.

Logical description $A_j(\bar{x})$ of the class Ω_j is a disjunction of elementary conjunction containing atomic formulas only with predicates $p_1,\ldots, p_n$. If $\bar{\tau} \in \Omega_j$ then $A_j(\bar{\tau})$ is valid.

Here and below the notation $\bar{x}$ is used for a list of elements of a finite set x, corresponding to some permutation of its elements. The fact that elements of the list $\bar{x}$ are the elements of y will be written in the form $x \subseteq y$. In order to write that there exists a list of different values for variables $\bar{x}$ that satisfies the formula $A(\bar{x})$ the notation $\exists_{\bar{x}\neq} A(\bar{x})$ will be used instead of the formula

$$\exists x_1 \ldots \exists x_n \left(\&_{i=1}^{n-1} \&_{j=i+1}^{n} (x_i \neq x_j) \,\&\, A(x_1,\ldots,x_n) \right).$$

It is shown in [Kosovskaya, 2011] that the most Artificial Intelligence problems may be reduced to the proof of the formula of the type

$$S(\omega) \Rightarrow \exists \bar{x}_{\neq} A_j(\bar{x}) \quad (1)$$

This problem is an NP-hard one [Kosovskaya, 2007].

In particular, upper bounds of number of steps for an algorithm solving this problem are the following: $O(t^m \, ||A|| \, ||S||)$ for exhaustive search method and $O(s^a)$ for logical methods, where t and m are the number of elements in ω and the number of variables in $A_j(\bar{x})$; $||S||$ and $||A||$ are the lengths of $S(\omega)$ and $A_j(\bar{x})$; s and a are the maximal and the summary numbers of occurrences of the same predicate in $S(\omega)$ and $A_j(\bar{x})$ respectively.

It seems that the programming language Prolog is the most acceptable language for the implementation of an algorithm solving an Artificial Intelligence problem. A language from the Prolog languages family uses resolution method. The upper bound

$$O(s^a) \quad (2)$$

was received, in particular, for resolution method.

Additionally it is required that the class descriptions would be stored in a text file, because while solving an image recognition problem it is often necessary to keep the class descriptions in the computer memory.

Input data for a Prolog program and predicates for description of an image

Input data for a recognition system may be an image represented by .jpeg (.jpg), .bmp or .png format. Methods of data compression are frequently used while saving an image in .jpeg or .bmp formats. As a result some artefacts may appear which does not allow to recognize an object. That is why a .png format is used.

To receive input data for a Prolog program recognizing an image a C# program is written which creates logical description of an image.

An image is described pixel-by-pixel. Every pixel is characterized by its coordinates and its brightness calculated according the formula *0.3 R +0.59 G + 0.11 B*, where *R, G, B* are red, green and blue components of pixel color respectively.

If we use the only one predicate *p(i, j, b) "pixel with coordinates (i,j) has the brightness b"* then the value s_1 of the parameter *s* in the estimate (2) equals to the size (the number of pixels) of the image. This value may be huge.

If we use a series of predicates *pb(i, j) "pixel with coordinates (i,j) has the brightness b"* for every value of brightness *b*, then the value s_2 of parameter *s* in the estimate (2) equals to the maximal number of pixels with the same brightness. It is less then s_1 if the image is not a monochrome one.

We may take in the account brightness of surrounding pixels (see Fig. 1). In such a case a predicate $p(i, j, b_1,...,b_9)$ or a series of predicates $pb_1...b_9(i, j)$ may be used. The number of predicates in the series is r^9 where *r* is the number of the used brightness gradations. The value s_3 of parameter *s* in the estimate (2) is the maximal number of equal fragments containing 9 pixels with the same brightnesses. The value s_3 usually is much less than s_1 and s_2.

In the terms of such a series of predicates the description of the pixel with coordinates *(i,j)* represented on the fig. 1 is *p342301321(i,j)* if we use 5 brightness gradations.

We are able not to use brightness gradation and use only normal evaluated brightness. But in such a case the length of the predicates will be larger and the time for the processing with the text files will be also larger.

The C# program develops the image description in the terms of predicates $pb_1...b_9(i, j)$.

All further examples and computations are made using SWI Prolog language [SWI-Prolog]. It's the most useful and free kind of the Prolog language. It's possible to use Visual Prolog [Visual Prolog, Visual Prolog 5], Strawberry Prolog [Strawberry Prolog] or other kinds of it, but it will be necessary to write interpreter or something else.

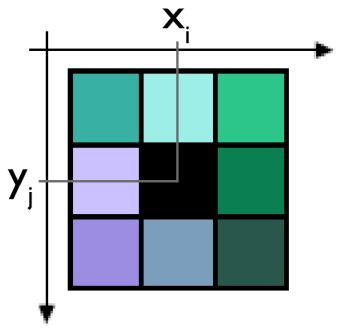

Figure 1. Characteristic of a pixel by its surrounding pixels

Search of a fragment within a complex image

A problem of the search of an etalon image within a large complex one is considered.

Let a complex image has $M \times N$ pixels and an etalon one has $m \times n$ pixels. The description of the etalon may be regarded as a description of a class if we change every constant in the etalon description by a variable (the same one for equal constants and different ones for different constants). In such a case the value of parameter a in the estimate (2) is $m \times n$. Usually it is large enough to be an exponent in the number of steps estimate.

For example, an attempt of the search of a fragment within Yandex map presented on the fig. 2 was not fulfilled. It becomes clear that the value of parameter a in the estimate (2) must be decreased.

An additional natural parameter k is introduced as a degree of compression of an image.

Let

$$M = k\,M' + rest_M\,, \quad N = k\,N' + rest_N\,,$$
$$m = k\,m' + rest_n\,, \quad n = k\,n' + rest_n\,,$$

where $0 \le rest_M\,, rest_N\,, rest_n\,, rest_n < k$.

Decompose both the complex image and the etalon one on squares $k \times k$ pixels. Every square is represented by one *mega-pixel* the brightness of which is an arithmetic mean of all pixels in the square. The remained (at the right and at the bottom) pixels are rejected.

After such a decomposition the value of parameter a in the estimate (2) decreases in k^2 times and becomes $a' = m'/k\ n'/k = [m/k]\ [n/k]$ instead of $a = m\ n$. So $a' = a/k^2$.

If an etalon image is a part of the complex image then one pixel of the etalon corresponds to one pixel of the complex image. That is why one program run is enough to find the etalon or to be sure that this etalon is not a part of the complex image.

In the case of mega-pixels introducing one program run may be not enough because such mega-pixels may be not identical. But k^2 calls of a program is sufficient to solve the problem. An example of "coincidence" and "non coincidence" of mega-pixels is presented on fig. 4.

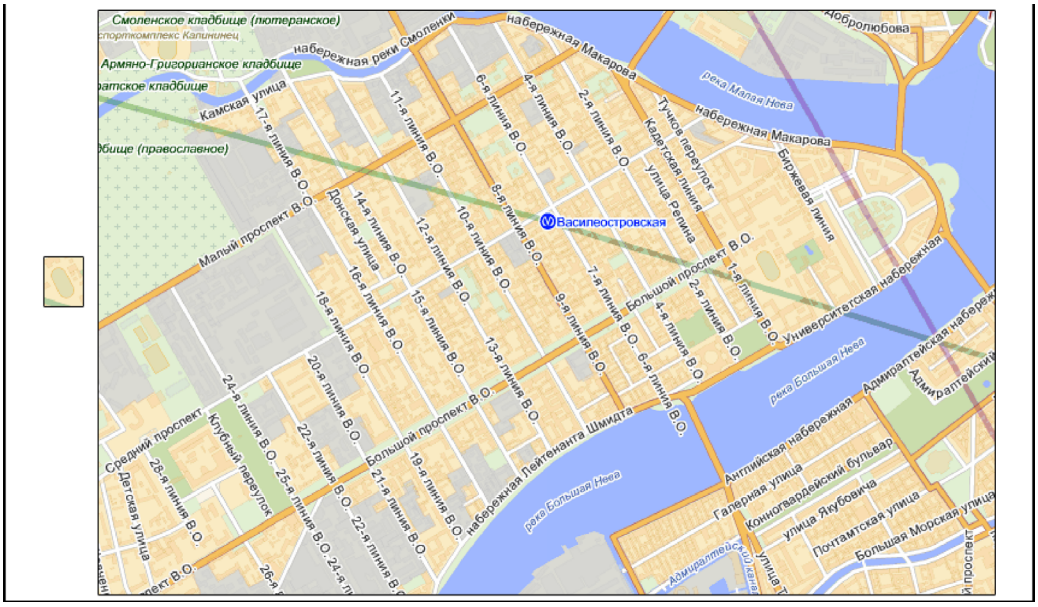

Figure 2. Representation of a fragment (at the left) and a complex image (at the right).

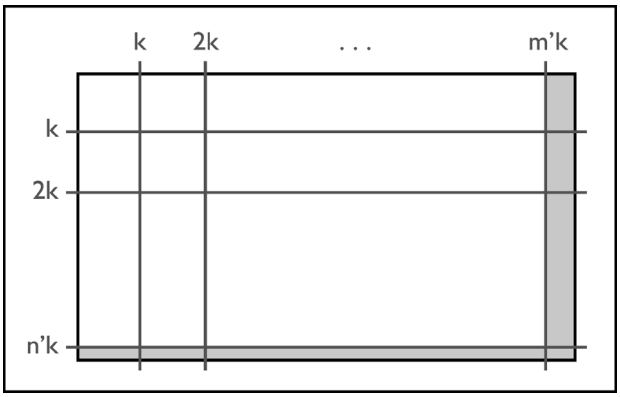

Figure 3. Decomposing of an image with restriction of some pixels

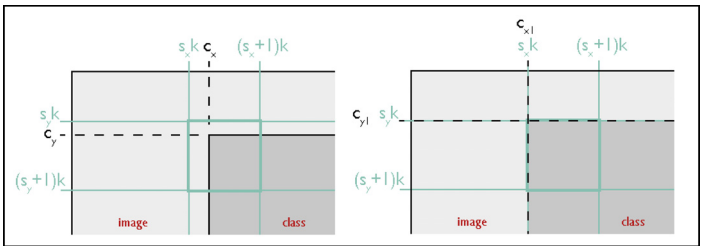

Figure 4. Example of "coincidence" and "non coincidence" of mega-pixels

So, the new estimate of the Prolog program run is

$$(k^2 \cdot time) \cdot (s')^{\overline{k^2}},$$

where s' is the maximal number of the same predicate in the description of a complex image represented by mega-pixels, *time* is the mean time of the Prolog program call.

Examples of the Prolog program runs

Consider some examples of input data and results of Prolog program runs.

The first example of input data is presented on fig. 2. The etalon image has the size 32*px* × *40px*. The complex one has the size *744px* × *480px*. The etalon image was found within the complex one with the coefficient of compression k = 3.

The result of the Prolog program run is presented on fig. 5.

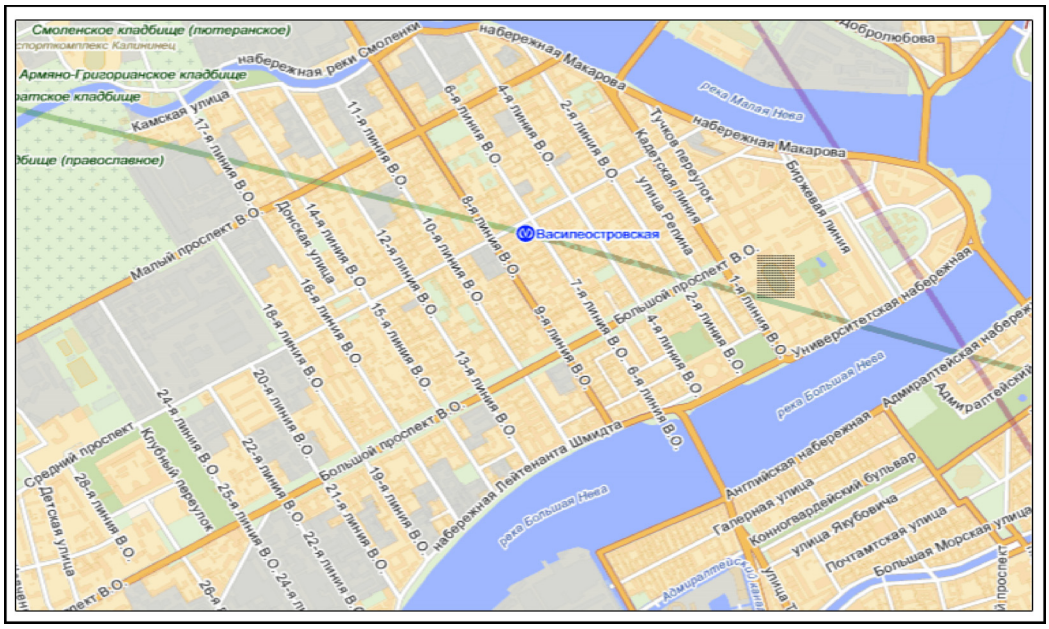

Figure 5. The result of the Prolog program run

The second example concerns the problem appearing while taking an image in different formats which use a compression algorithm with quality loosing.

The etalon image and the complex one presented on fig.6. have sizes *40px* × *34px* and *300px* × *277px* respectively. The images were saved in .png and .jpg formats. While saving in .jpg format the artefacts appeared.

The result was received only in the case of .png saving.

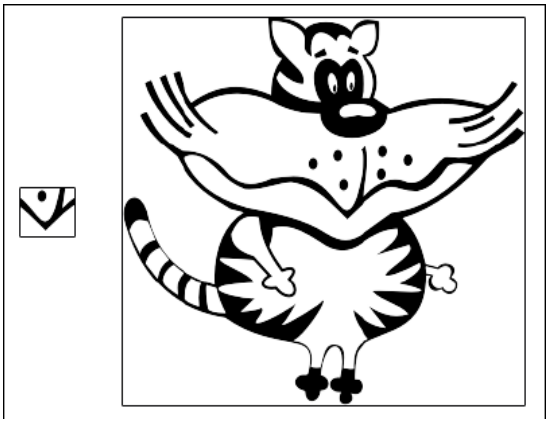

Figure 6. Etalon and complex images

Conclusion

The influence of number of steps estimation for an algorithm solving a problem to its mathematical modeling is demonstrated in the paper. It is the mostly important for an NP-hard problem.

Such estimations for different algorithms solving an Artificial Intelligence problem influence to many its parameters, for example, the choice of initial characteristics of an object and the choice of a heuristic method providing the decreasing of a problem dimensionality.

It is shown in the paper that the use of the seemed to be natural predicate describing the brightness of a pixel is adequate for description of a picture but implies huge number of program run steps. Nevertheless the use of 5^9 different predicates allows to recognize an etalon image within a complex one represented on fig. 6.

The main influence upon the decreasing of a program run steps has the value of parameter which is in the exponent of an estimate. For different algorithms these parameters may be also different. An investigator has two ways to decrease their value. The first one is to choice an algorithm according to the already chosen predicates [Kosovskaya, 2011]. The second one is represented in this paper and is a heuristic method of decreasing the problem dimensionality. As a result of such a heuristic method is recognition of an etalon image within a complex one represented on fig. 2 and fig. 4.

Bibliography

[Kosovskaya, 2007] Kosovskaya T.M. Proofs of the number of steps bounds for solving of some pattern recognition problems with logical description // Vestnik of St.Petersburg University, Ser. 1, 2007. Ed. (2), pp. 82-90. (In Russian)

[Kosovskaya, 2011] Kosovskaya T. Discrete Artificial Intelligence Problems and Number of Steps of their Solution // International Journal "Information Theories and Applications", Vol. 18, Number 1, 2011. P. 93 – 99.

[Strawberry Prolog] Strawberry Prolog: Professional tool for programming in Prolog. Free Light edition with many example programs in it. URL: http://www.dobrev.com/

[SWI-Prolog] SWI-Prolog's home. URL: http://www.swi-prolog.org/

[Visual Prolog 5] Visual Prolog Version 5.x. // Language Tutorial, Prolog Development Center A/S H.J. Holst Vej 3-5C, DK – 2605 Broendby, Copenhagen, Denmark.

[Visual Prolog] Visual Prolog Programming Language, Compiler, IDE, Download Free Personal Edition: Visual Prolog is a full-featured programming environment for making commercial applications. Prolog compiler, IDE, tutorials, examples. New: Visual Prolog 7.3. URL: http://www.visual-prolog.com/

Acknowledgement

The paper is published with financial support of the project ITHEA XXI of the Institute of Information Theories and Applications FOI ITHEA (www.ithea.org) and the Association of Developers and Users of Intelligent Systems ADUIS Ukraine (www.aduis.com.ua).

Authors' Information

Tatiana Kosovskaya *– Dr., Senior researcher, St.Petersburg Institute of Informatics and Automation of Russian Academy of Science, 14 line, 39, St.Petersburg, 199178, Russia; Professor of St.Petersburg State Marine Technical University, Lotsmanskaya ul., 3, St.Petersburg, 190008, Russia; Professor of St.Petersburg State University, University av., 28, Stary Petergof, St.Petersburg, 198504, Russia, e-mail: kosov@NK1022.spb.edu*

Major Fields of Scientific Research: Logical approach to artificial intelligence problems, theory of complexity of algorithms.

Maria Vlasova *– PhD student, St.Petersburg State University, University av., 28, Stary Petergof, St.Petersburg, 198504, Russia, e-mail: mvlasova.spb@gmail.com*

Major Fields of Scientific Research: Logical approach to Artificial Intelligence problems.