Logical Vision: Meta-Interpretive Learning for Simple Geometrical Concepts

Wang-Zhou Dai[1], Stephen H. Muggleton[2] and Zhi-Hua Zhou[1]

[1] National Key Laboratory for Novel Software Technology, Nanjing University
[2] Department of Computing, Imperial College London

Abstract. Progress in statistical learning in recent years has enabled computers to recognize objects with near-human ability. However, recent studies have revealed particular drawbacks in current computer vision systems which suggest there exist considerable differences between the way these systems function compared with human visual cognition. Major differences are that: 1) current computer vision systems learn high-level notions directly from the low-level feature space and ignore the mid-level representations, which makes them difficult to incorporate background knowledge. 2) typical computer vision systems learn visual concepts discriminatively instead of encoding the knowledge necessary to produce a visual representation of the class. In this paper, we introduce a framework referred as *Logical Vision* which is demonstrated on learning visual concepts constructively and symbolically. Given a set of images, a set of first-order logic formulae of background knowledge and a set of examples of target visual concepts, *Logical Vision* extracts logical facts concerning geometrical elements from an image by sampling low-level features guided by the background knowledge and conjecturing geometrical elements as output. It first extracts logical facts of mid-level features, then generative Meta-Interpretive Learning technique is applied to learn high-level notions because it is capable of learning recursions, inventing predicates and so on. Owing to its symbolic representation paradigm, in our implementation, *Logical Vision* is fully implemented in Prolog apart from low-level image feature extraction primitives. In our implementation, Logical Vision was used to extract polygon edges as mid-level symbols, and a generalized Meta-Interpreter Learner was applied to learn high-level geometrical notions. Experiments are conducted on learning shapes (e.g. triangles, quadrilaterals, etc.), regular polygons and right-angle triangles. These demonstrates that learning visual concepts constructively and symbolically is effective.

1 Introduction

Computer vision is a sub-field of Artificial Intelligence which aims at the analysis and interpretation of visual information. It is an information-processing procedure that receives in input raw and low structured data, and gives outputs of explicit, meaningful descriptions of the structured information.

Computer vision can be categorized into high-level and low-level vision. Low-level vision was aimed at image processing tasks and identification of local features. By contrast, high-level vision was aimed at delivering overall scene analysis in terms of relations between objects in the scene [14]. Recently, almost all research has concentrated

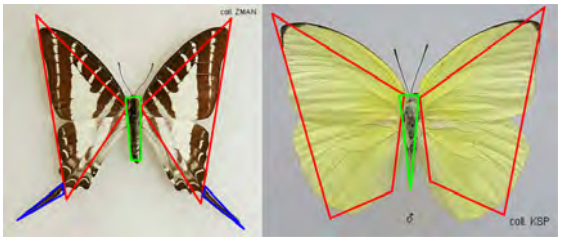


Fig. 1. Images of Papilionidae and Pieridae[†]: each butterfly can be roughly seen as a composition of different polygons; the visual concepts of butterfly species are defined by their component polygons, the (position) relations between the components and color patterns inside of each components, etc.

on low-level vision and drawn on the power and the efficiency of statistical learning to enable computer vision algorithms to identify information from low-level metrics (e.g. color and gradient information in smaller pieces of objects surrounding interest points in images). The recognition task is then based on searching for and matching the low-level features with complex statistical classifiers like Decision Trees, Neural Networks, Support Vector Machines and so on. In addition to traditional low-level features, recent popular feature descriptors like SIFT [19] and SURF [4] can even encode spatial relationships between the original low-level features. These features are developed to be invariant to changes in scale, lightness, rotation, and affine transformations. During the last decade, more and more complex classifiers and matching methods have been used to combine strong and weak features for more effective recognition.

Recently, deep neural networks (DNNs) [12, 5] have demonstrated impressive and state-of-the-art results on many pattern recognition tasks, especially image classification problems [17, 15]. DNNs are able to learn hierarchical layers of representation from sensory input, and it is possible to train neurons to be selective for high-level concepts that function as detectors for faces, human bodies, and cat faces, which enables its human-competitive ability in many tasks [17, 15]. However, recent studies revealed some major differences between them and human visual cognition [1, 31], which exist in most of statistics-based computer vision learning algorithms.

For example, it is easy to produce images that are completely unrecognizable to humans, though state-of-the-art visual learning algorithms believe them to be recognizable objects with over 99% confidence [1]. This is because it learns a discriminative model, some synthetic images that lying deep within a classification region (i.e. far from the decision boundary) in the low-level feature space can produce high confidence predictions, even though they are also far from natural images in the class [1]. In other cases small perturbations to the input images, which are imperceptible to human eyes, can arbitrarily change the classifier's prediction [31]. Analysis from the authors shows that the instability is caused by classifiers' sensitivity to small changes of low-level features in input images.

[†]Source: http://www.papua-insects.nl

Moreover, humans can typically learn from a single visual example [16], unlike statistical learning which depends on hundreds or thousands of images. Humans achieve this ability using background knowledge, which plays a critical role. By contrast, statistics-based computer vision algorithms have no general mechanisms for incorporating background knowledge. According to [21], human vision process can be postulated as a hierarchical architecture with different intermediate representations and processing levels. At each stage of recognition, the representations (symbols) obtained from previous stages play the role of background knowledge. For example in the Figure 1, to recognize a butterfly, one should first be able to detect polygons on images, then from the different shapes, position relations and color patterns between polygons, people can categorize the butterflies into different species.

In this paper we consider the approach of using modern ILP techniques to support the incorporation of background knowledge in the generation of scene analysis in terms of high-level relations. For this purpose we propose a novel visual concept learning framework, called *Logical Vision*, to realize this symbolic visual processing paradigm. *Logical Vision* first uses background knowledge on mid-level symbols to guide the sampling of low-level features, then it uses the sampled results to revise previously conjectured mid-level symbols. With the extracted mid-level feature symbols as background knowledge, a generalized Meta-Interpretive learner [23] is used to learn high-level visual concepts because it enhances the constructive paradigm of Logic Vision through its ability to learn recursive theories, inventing predicates and learning from a single example. The long-term aim of *Logical Vision* is to learn to analyze natural objects and scenes by partitioning them into colored polygons like Figure 1 shows. As the first step, in this work we applied our *Logical Vision* framework to tasks involving learning simple geometrical concepts such as triangles, quadrilaterals, regular polygons and so on. In order to model the visual process for general applications, we try to use one of the lowest feature as primitive to learn other concepts. In this work, we define a "point" as an pixel which has large gradient in its local region. Base on the primitives, we can learn more complex objects such as edges, polygons, combinations of polygons etc. stage by stage. Owing to its symbolic representation, *Logical Vision* can be fully implemented in Prolog given low-level image feature extraction primitives as the initial background knowledge. Our experimental results show its effectiveness in learning target visual concepts which are difficult for typical low-level feature based statistical computer vision algorithms.

The rest of this paper is organized as follows. Section 2 presents some related works. Section 3 proposes the *Logical Vision* framework. Section 4 describes the implementation of the $Metagol_{LogicalVision}$ approach, followed by experimental results in Section 5. Finally, Section 6 concludes and discusses about future works.

2 Related work

State-of-the-art computer vision algorithms are mostly deep neural networks (DNN) which have been trained from large-scale datasets, such as [17, 15, 29, 30]. For small-scale tasks, people usually use DNN descriptors which learned from large-scale data as feature space for learning and recognition. It has been shown that the DNN features with

standard statistical learning technique still achieved the state-of-the-art performance in these kinds of tasks [29, 30]. In in section 5, we made a comparison between the proposed approach and statistical classifier with DNN feature.

Besides of the end-to-end visual learning paradigms, there exists another classical idea which tries to parse images hierarchically like human cognition, e.g. [25]. Recently, this framework has become more tractable due to progress in machine learning and statistics. For example, [11, 9, 27] developed grammar models for hierarchical object recognition. [13] proposed the "composition machine" for constructing probabilistic hierarchical image models and encode contextual relationships. [18] proposed a hierarchical feature coding approach which uses code-word learning, coding and pooling to obtain high-level features. There are also some approaches that different levels of features together [6, 35]. Most of statistics-based computer vision systems either design objective functions as constraints in the statistical learning procedure, or manually develop specific features to enhance the learning process. However, it is difficult for them to incorporate general background knowledge in a logical formalism.

To incorporate background knowledge into statistical learning, many algorithms has been proposed in the last decade [20, 2, 22]. Some of them use background knowledge about low-level features to constraint the statistical learning process [20], some others directly learn models in a high-level feature space, in which first-order logic background knowledge can be naturally applied [2, 22]. Different to these approaches, *Logical Vision* can exploit background knowledge of different levels and can process raw image data directly.

More closely related works are those approaches which also adopt symbolic learning paradigm like *Logical Vision*. A representative work in this branch is [3]. The proposed approach first does a low-level feature descriptor extraction on the whole image. Based on these descriptors, interest points are selected to form possible interest regions. Then a statistical model is trained for categorizing the candidate interested regions, positive ones are retained and labeled as different object symbols. Finally the object symbols are used for learning high-level concepts with supplementary background knowledge that expressed by first-order logic. Our work shares the same objective to this work, however, *Logical Vision* seeks for symbolic representation in most of the vision cognition stages, which enables more flexible ways to incorporating background knowledge. Besides of directly using the statistically extracted facts as materials for relational learning, *Logical Vision* is able to guide the low-level feature extraction with first-order knowledge.

3 The proposed framework

In this section we introduce the framework of *Logical Vision*. The input for *Logical Vision* consists of a set of geometrical primitives B_P, one or a set of images $\mathcal{I}$ as background knowledge, and a set of logical facts E representing the examples as the target visual concepts. The task is to learn a hypothesis H that defines the target visual concept where $B_P, \mathcal{I}, H \models E$.

Given an input image, *Logical Vision* first alternately conjecture about mid-level visual objects and samples low-level features to support or revise those conjectures.

After obtaining mid-level features, a meta-interpreter can be executed for learning the target visual concepts.

3.1 Mid-level features extraction

The purpose of mid-level features extraction is to obtain necessary logical facts B_A representing mid-level features of $I \in \mathcal{I}$ for target visual concepts learning by ILP.

The mid-level feature extraction in *Logical Vision* is realized by repeatedly executing a "conjecturing and sampling" procedure. It uses mid-level feature conjectures to guide the sampling of low-level features, they are then used to revise previously constructed conjectures.

Here the low-level features are referred to local visual metrics such as color information, gradients, SIFT and SURF descriptors, etc. The term "mid-level feature/symbol" is a relative concept: they are logical facts that represent possible sub-parts or components of higher-level concepts. For example, a low-level feature "color gradient" can be useful for describing mid-level features such as edge discovery or contour extraction. However, if the target concept is "butterfly", the mid-level feature "edge" can be seen as sub-parts of other higher-level concepts like "shape" and "region". Together with more features like "color pattern", "region size" and background knowledge about "position relations" and so on, we can finally learn the concept of butterfly within a pure symbolic paradigm by ILP (see Figure 1).

The intuition of the "conjecturing and sampling" process is an analogy to human vision process. Suppose a man stands in front of a huge wall painting, which is very large that we can only clearly observe a small region at one time. To get a whole picture of this painting, he can try to move his eyes around to see different small regions in the painting and guess about the entire view of it. During the observation, he either can sample more details to support his conjecture, or can revise them by doing more sampling. After doing enough samples, he will believe that his final conjecture is the ground truth.

Formally, mid-level feature extraction of an image $I \in \mathcal{I}$ could be described as follows:

1. Sample low-level features F in a subarea (e.g. surrounding a focal point) of I, then add F into the sampled low-level features set $\mathcal{F}$.
2. Conjecture a mid-level feature (edge, region, texture, etc.) C according to $\mathcal{F}$.
3. Validate the conjecture C on image I by doing few more samples. If the validation failed, reject C and go to 1, otherwise go to 4.
4. When C is valid, add it to mid-level feature set B_A, then remove the low-level features $f(C)$ that encapsulated by C, the rest of low-level features $\mathcal{F}' = \mathcal{F} - f(C)$. For example, if the low-level features are pixels whose local area have a large color variance, the mid-level features to be extracted are contours on the image: once a conjectured contour C has been accepted to B_A, we should remove all other pixels on the contour, so $f(C) = \{pixel | pixel \text{ is on contour } C\}$.
5. If $\mathcal{F}' = \phi$, terminate the construction procedure and return B_A, otherwise go to 1.

Briefly speaking, *Logical Vision* uses mid-level feature conjectures to guide the sampling of low-level features, then uses sampled results to revise previously obtained

conjectures. The low-level features themselves like pixel colors, local color variances or gradient directions are usually redundant and trifling for representing higher-level visual concepts. After the background-knowledge-guided extraction, they can be compactly abduced into logical symbols such as edges, regions, textures, etc., to serve as bases for higher-level concepts learning.

This human-cognition-mimic paradigm assumes that the mid-level conjectures are constructed by pre-defined predicates, i.e. background knowledge. This assumption is reasonable because a person need certain knowledge about particular simple primitives to learn more complicated concepts. For example, we have to define "gradient" before we learn the notion "contour", and we have to understand "line segment" before we learn the concept "polygon". This procedure follows a symbolic learning paradigm, thus it can be easily implemented by logic programming tools like Prolog. This ensures that *Logical Vision* will hardly introduce typical computer vision operations such as sliding windows or image filtering. This is because we believe that in human cognition, observation is a process that happens only when necessary rather than a exhaustive enumeration, and background-knowledge/conjecture-guided abduction is a proper way to model this kind of action. By incorporating first-order formalism in feature extraction procedure, we wanted to show that symbolic reasoning is able to solve particular computer vision tasks which have been long time considered as a statistical problem.

Table 1. Prolog code for the generalized meta-interpreter. The interpreter recursively proves a series of atomic goals by matching them against the heads of meta-rules. After testing the `Order` constraint `save_subst` checks whether the meta-substitution is already in the program and otherwise adds it to form an augmented program. On completion the returned program, by construction, derives all the examples.

Generalized meta-interpreter

```
prove([], Prog, Prog).
prove([Atom|As], Prog1, Prog2):-
    metarule(Name,MetaSub, (Atom:-Body), Order),
    Order,
    save_subst(metasub(Name,MetaSub), Prog1, Prog3),
    prove(Body, Prog3, Prog4),
    prove(As, Prog4, Prog2).
```

3.2 Meta-Interpretive Learning

After obtaining mid-level logic symbols B_A, *Logical Vision* uses a generalized Meta-Interpretive Learner to learn target visual concepts. The input of generalized Meta-Interpretive Learning (MIL) [24] consists of a generalized Meta-Interpreter B_M and domain specific primitives B_P together with two sets of ground atoms as background knowledge B_A and examples E respectively. The output of MIL is a revised form of the background knowledge containing the original background knowledge B_A, domain specific primitives B_P augmented with additional ground atoms representing a hypothesis H. According to Inverse Entailment, $B, H \models E$ is equivalent to $B, \neg E \models \neg H$. In

Algorithm 1 $LogicalVision_{Poly}(B_P, I, Metagol_{LogicalVision}, E, N)$

Input: Geometrical primitives B_P, input image I, examples E, Meta-Interpretive learner $Metagol_{LogicalVision}$, sampling level N.
Output: Hypothesis of the target visual concept H;
Start:
Initialize edge points set $\mathcal{F} = \phi$ and sampled edges set $B_E = \phi$;
Randomly sample some edge points {`P1,P2,...`}, let $\mathcal{F} = \mathcal{F} \cup$ {`P1,P2,...`};
repeat
  Select a pair of edge points `P1,P2`$\in \mathcal{F}$;
  Validate whether `P1P2` forms an edge by querying `edge(P1,P2,`N`)`;
  if `edge(P1,P2,`N`)` succeeded **then**
    Extend `P1P2` on both of its directions to form a conjecture of edge C;
    $B_E = B_E \cup C$;
    Remove all edge points `P`$\in \mathcal{F}$ that lies on edge C;
  else
    Randomly sample a line which crosses the line segment `P1P2` for new edge points, if they are not encapsulated by any sampled edge in B_E then add them into $\mathcal{F}$;
  end if
until $\mathcal{F} = \phi$;
Find connected edges in B_E to construct facts of polygons B_A;
Learn a hypothesis H with $B_A, Metagol_{LogicalVision}, B_P, E$ through MIL;
Return: H.

this form we see that $B, \neg E$ is given to the meta-interpreter where $\neg E$ is a goal and the resulting abducted program $\neg H$ represents a headless Horn clause. The Prolog implementation of generalized meta-interpreter is showed in Table 1. In this work, we used dyadic meta-rules [24] to learn target theories.

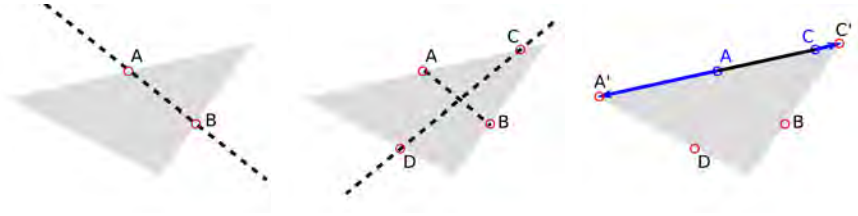
4 Implementation

Below we describe the implementation of *Logical Vision* for the task of polygon shapes learning. The target concepts of this task are definitions of different kinds of polygons (e.g. triangles, regular polygons, etc.). Our implementation is displayed as Algorithm 1, which is referred to as $LogicalVision_{Poly}$.

4.1 Polygon extraction

To learn the concepts of polygon shapes, we targeted the mid-level features B_A to be extracted as polygons. They are denoted as `polygon(Pol_i, [Edge1,...,EdgeN])`. The process of polygon extraction can be split into two stages: edge discovery and polygon construction. For simplicity, in the polygon construction stage, $LogicalVision_{Poly}$ groups of connected edges as a list as `polygon`.

Therefore, the major challenge is to discover those edges. Following the framework in section 3.1, we introduce some primitives as background knowledge to perform the conjecturing and validation. For example, the background knowledge for "edge" is defined as follow:



(a) (b) (c)

Fig. 2. (a) 2 edge points A and B are sampled; (b) Edge AB is conjectured but it is invalid because if `midpoint(A,B,P)`, then `edge_point(P)` is false. So a random line that crosses AB is sampled, 2 new edge points C and D have been discovered; (c) Edge AC is conjectured and it passes the recursive `edge` test, so AC is extended until no continuous edge points were found. Finally the edge $A'C'$ is recorded and A, C are removed from $\mathcal{F}$.

```
edge(P1,P2,0):-midpoint(P1,P2,P),edge_point(P1),edge_point(P2),
                edge_point(P).
edge(P1,P2,N):-midpoint(P1,P2,P),edge_point(P1),edge_point(P2),
                N1 is N-1,edge(P1,P,N1),edge(P,P2,N1).
```

in which `P1` and `P2` are the input conjectured end points of an edge, N is the recursion limit that controls the depth of edge validation and `midpoint/3` finds the midpoint between two pixels. By conjecturing `(P1, P2)` to be an edge, the predicate would recursively sample the points between `(P1, P2)` and test whether do they have a high local variance (color gradient) as well. The color gradient test is performed by `edge_point/1`, it is the only primitive that interacts with low-level features on images, it returns true when the color gradient magnitude of pixel `P` exceeds a pre-defined threshold. The color gradient is computed as follow:

$$G(A) = \sqrt{G_x(A)^2 + G_y(A)^2} \tag{1}$$

where

$$G_x(A) = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A \tag{2}$$

$$G_y(A) = \begin{bmatrix} -1 & 2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A. \tag{3}$$

are Sobel filters, A is the focal point that has been queried, $*$ denotes the 2-dimensional signal processing convolution operation. G_x and G_y represents the image derivatives in vertical and horizontal directions respectively. We implemented an image-processing program by OpenCV [10], and used a C++-Prolog interface [34] to enable communication between predicate `edge_point/1` and input images. A detail example of edge extraction process is illustrated in Figure 2.

4.2 $Metagol_{LogicalVision}$

Nevertheless, the polygon extraction procedure in section 4.1 sometimes results in a noisy B_A. This may cause the depth-first search in Metagol to fail or return ground hypotheses that cover only one example. Thus, we altered the original Metagol to enable it abduce imperfect hypotheses and evaluate the hypotheses using foil gain [28]. This procedure is described as follow:

1. Abduce an hypothesis P with general Meta-Interpreter.
2. If the hypothesis covers all positive examples and rejects all negative examples, return P; otherwise go to step 3.
3. Evaluate the quality of P with foil gain [28]. If P is better than current best hypothesis $\hat{P}^*$ then replace $\hat{P}^*$ with P. Go to step 1 to abduce another candidate hypothesis.

The domain specific primitives B_P of $Metagol_{LogicalVison}$ include some necessary background knowledge for learning polygon shape related target concepts. For example, in order to learn the concept of regular polygon, $Metagol_{LogicalVision}$ uses `angle_list/2` to obtain all inner angles of a polygon and uses `std_dev_bounded/2` to test whether the standard deviation of a list of double numbers is bounded by an automatically learned threshold.

Moreover, the stochastic implementation of polygon extraction occasionally discovers redundant edges (see Figure 4), therefore the B_P of *Logical Vision* included a post-processing primitive `rmv_rdndnt(L1, L2, T)`. It enables $Metagol_{LogicalVision}$ to learn a threshold `T` for automatically removing extra edges. Here `L1` and `L2` are the input and output edge lists respectively, the threshold `T` controls the degree of refinement: consider 2 connected edges AB and BC, if $(|AB| + |BC|)/|AC| \leq$ `T`, then `rmv_rdndnt/3` choose to omit them and construct a new edge AC.

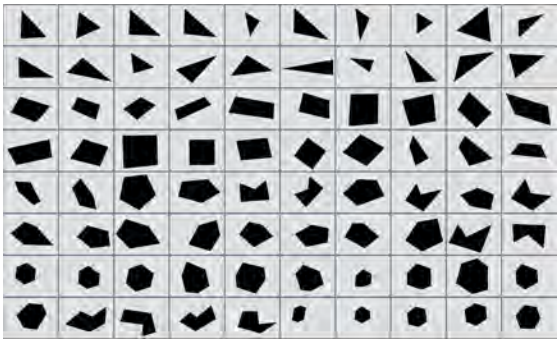
5 Experiments

In this section we describe our experiments which compare $Metagol_{LogicalVision}$ with statistics-based computer vision approaches on tasks of learning simple geometrical concepts from binary colored images.

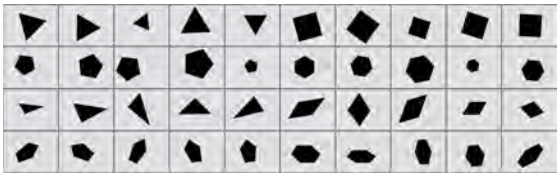
5.1 Materials

3 labeled image datasets are generated for polygon shape learning. For simplicity, the images are binary-colored, each image contains one polygon. Target concepts of the 3 tasks are: 1) `triangle/1`, `quadrilateral/1`, `pentagon/1` and `hexagon/1`; 2) `regular_poly/1` (regular polygon); 3) `right_tri/1` (right triangle). Note that in the third task, we used the best hypothesis of `triangle/1` learned in the first task as background knowledge. Part of the datasets are presented in Figure 3. All datasets were partitioned into 5-folds respectively, 4 of them were used for training and the remainder for testing. The details of all tasks are showed as follows:

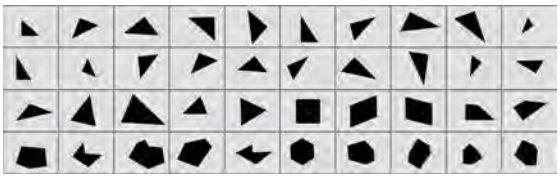
Learning polygon shapes: We randomly generated 40 images of triangles, quadrilaterals, pentagons and hexagons respectively. Each image contains one polygon. Each



(a) Learning triangles, quadrilaterals, etc.



(b) Learning regular polygons



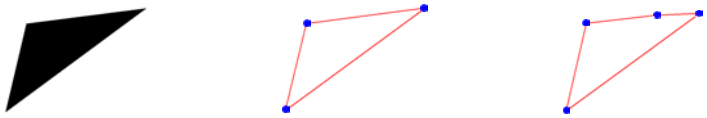
(c) Learning right-angle triangles

Fig. 3. Part of datasets for the 3 learning tasks

polygon is black-colored, displays on white canvas. Target concepts in this task are `triangle/1`, `quadrilateral/1`, `pentagon/1` and `hexagon/1`.

Learning regular polygons: We randomly generated 10 images of regular and irregular triangles, quadrilaterals, pentagons and hexagons respectively, so there are 80 images in total. In this task, the target concept is `regular/1`.

Learning right-angle polygons: We randomly generated 40 images of right-angle triangles as positive examples. The negative examples consist of 10 triangles, quadrilaterals, pentagons and hexagons respectively. In order to learn a correct definition of right-angle triangle, the generated quadrilaterals, pentagons and hexagons in negative example set may contain right-angles with probability 0.4. Please note that in this task, $Metagol_{LogicalVision}$ re-used the best hypothesis of `triangle/1` that learned from the first task as background knowledge. Target concept of this task is `right_tri/1`.



(a) (b) (c)

Fig. 4. Noise of polygon extraction: (a) is the ground truth image. (b) and (c) are two polygons extracted by our algorithm, where (c) contains a redundant vertex.

5.2 Methods

$\mathbf{LogicalVision_{Poly}}$: This is the proposed approach. In order to handle the noise caused by polygon extraction (e.g. Figure 4), for each image we ran the extraction procedure five times independently to duplicate the input instances (both for training and testing). During evaluation, the learned hypotheses were tested on all the five extracted polygons and the final prediction was based on an equal weighted vote.

Statistics-based Learning: We used a popular statistics-based computer vision toolbox VLFeat [33] to implement the statistical learning algorithms. The experiments are carried with different kinds of features. Because the sizes of datasets are small, we used support vector machine (libSVM [7]) as classifier. The parameters are selected by 5-fold cross-validation. The features we have used in the experiments are listed as follows:

- **HOG**: The Histogram of Oriented Gradients (HOG) [8] is a feature descriptor commonly used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. Specifically, in order to conserve all gradient information in the image, the HOG descriptors in our experiments are *not* summarized by k-nearest neighbor bag-of-word models. This feature is a 148800-dimensional vector.
- **Dense-SIFT**: Scale Invariant Feature Transform (SIFT) [19] is an image descriptor for image-based matching and recognition. The dense-SIFT descriptor is roughly equivalent to running SIFT on a dense gird of locations at a fixed scale and orientation. It is often used for object categorization and has been proven to be very useful in practice for image matching and object recognition under real-world conditions. In our experiments, it produces a 34048-dimensional vector for each image.
- **LBP**: Local binary pattern (LBP) [26] is a powerful feature for texture classification which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number. The LBP texture analysis can be seen as a unifying approach to the traditionally divergent statistical and structural models of texture analysis. We extracted a 69600-dimensional vector for this feature.
- **CNN**: Convolutional neural network (CNN) is a type of biological inspired feed-forward artificial neural network where the individual neurons are tiled in such a way that they respond to overlapping regions in the visual field. After integrating deep structures, they became very popular for image and video recognition [15, 29]. In the experiments, we extracted a 4096-dimensional feature for each image with a pre-trained deep CNN model called `imagenet-vgg-verydeep-16` [30] which

Table 2. Predictive accuracy of learning simple geometrical shapes on single object datasets. The feature combinations are abbreviated by the initial letters of each method.

ACC	tri	quad	pen	hex	reg	r_tri
HOG	0.83 ± 0.04	0.76 ± 0.01	0.73 ± 0.03	0.75 ± 0.07	0.63 ± 0.08	0.74 ± 0.04
dense-SIFT	0.82 ± 0.05	0.66 ± 0.06	0.64 ± 0.04	0.71 ± 0.03	0.71 ± 0.05	0.77 ± 0.07
LBP	0.87 ± 0.05	0.69 ± 0.04	0.67 ± 0.03	0.73 ± 0.03	0.65 ± 0.05	0.75 ± 0.05
CNN	0.91 ± 0.01	0.75 ± 0.00	0.75 ± 0.00	0.84 ± 0.02	0.59 ± 0.06	0.85 ± 0.04
H+d	0.82 ± 0.01	0.75 ± 0.01	0.76 ± 0.01	0.76 ± 0.01	0.64 ± 0.05	0.80 ± 0.03
C+d	0.82 ± 0.01	0.75 ± 0.00	0.76 ± 0.01	0.76 ± 0.01	0.69 ± 0.04	0.80 ± 0.03
C+L	0.87 ± 0.05	0.75 ± 0.01	0.76 ± 0.01	0.76 ± 0.01	0.61 ± 0.05	0.78 ± 0.06
C+d+L	0.82 ± 0.01	0.75 ± 0.00	0.76 ± 0.01	0.76 ± 0.01	0.64 ± 0.05	0.80 ± 0.04
LV$_{Poly}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.99 \pm 0.01}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$

Table 3. F1-measure of learning simple geometrical shapes on single object datasets. The feature combinations are abbreviated by the initial letters of each method.

F1	tri	quad	pen	hex	reg	r_tri
HOG	0.67 ± 0.11	0.80 ± 0.03	0.81 ± 0.02	0.72 ± 0.11	0.43 ± 0.05	0.31 ± 0.16
dense-SIFT	0.51 ± 0.10	0.67 ± 0.03	0.67 ± 0.04	0.67 ± 0.08	0.39 ± 0.04	0.33 ± 0.19
LBP	0.53 ± 0.15	0.72 ± 0.03	0.71 ± 0.05	0.67 ± 0.07	0.39 ± 0.03	0.36 ± 0.19
CNN	0.33 ± 0.18	0.86 ± 0.00	0.86 ± 0.00	0.71 ± 0.05	0.47 ± 0.04	0.40 ± 0.07
H+d	0.63 ± 0.12	0.85 ± 0.00	0.84 ± 0.02	0.81 ± 0.08	0.37 ± 0.06	0.22 ± 0.15
C+d	0.63 ± 0.12	0.86 ± 0.00	0.84 ± 0.02	0.81 ± 0.08	0.47 ± 0.10	0.22 ± 0.15
C+L	0.53 ± 0.15	0.84 ± 0.03	0.85 ± 0.02	0.81 ± 0.05	0.37 ± 0.10	0.31 ± 0.20
C+d+L	0.63 ± 0.12	0.86 ± 0.00	0.84 ± 0.02	0.81 ± 0.08	0.37 ± 0.06	0.24 ± 0.14
LV$_{Poly}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.97 \pm 0.02}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{0.98 \pm 0.02}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$

is implemented by MaxConvNet [32]. The descriptors were trained from ImageNet ILSVRC-2012 dataset (1.5 million photos), and these descriptors had showed state-of-the-art generalization performance in many image classification tasks [30].

– **Feature Combinations**: The experiments of statistical computer vision learning also had been carried on combinations of above feature sets.

5.3 Results

Table 2 and 3 show the results of our experiments. Performance of compared methods were evaluated by both predictive accuracy and F1-score on the hold-out test data in each fold.

Following are some examples of the hypotheses that learned by $LogicalVision_{Poly}$:

```
triangle_1(A,C,H):-rmv_rdndnt(A,B,C),list_length(B,H).
triangle_0(A,A2,B2):-polygon(A,B),triangle_1(B,A2,B2).
triangle(A):-triangle_0(A,0.04,3).
triangle_0(A,G):-polygon(A,B),list_length(B,G).
triangle(A):-triangle_0(A,3).
```

```
quadrilateral_0(A,G):-polygon(A,B),list_length(B,G).
quadrilateral(A):-quadrilateral_0(A,4).

pentagon_0(A,G):-polygon(A,B),list_length(B,G).
pentagon(A):-pentagon_0(A,5).

hexagon_1(A,C,H):-rmv_rdndnt(A,B,C),list_length(B,H).
hexagon_0(A,A2,B2):-polygon(A,B),hexagon_1(B,A2,B2).
hexagon(A):-hexagon_0(A,0.004,6).
hexagon_0(A,G):-polygon(A,B),list_length(B,G).
hexagon(A):-hexagon_0(A,6).

regular_poly_1(A,G):-angles_list(A,B),std_dev_bounded(B,G).
regular_poly_0(A,A2):-polygon(A,B),regular_poly_1(B,A2).
regular_poly(A):-regular_poly_0(A,0.02).

right_tri_2(A,G,H):-angles_list(A,B),has_angle(B,G,H).
right_tri_1(A,A2,B2):-polygon(A,B),right_tri_2(B,A2,B2).
right_tri_0(A,A2,B2):-right_tri_1(A,A2,B2),triangle(A).
right_tri(A):-right_tri_0(A,0.5,0.015).
```

where `list_length/2` returns the length of a list; `angles_list/2` returns a list of sizes of polygon's angles. `std_dev_bounded/2` examines whether the standard deviation of a set of real numbers is bounded by a threshold; `has_angle/2` checks whether a list (of angle sizes) contains a real number within an error bound; `rmv_rdndnt/3` removes redundant edges as section 4.2 introduced. For the `right_tri/1` task, we included the best hypothesis in the `triangle/1` task as background knowledge.

From the results we can see that the performance of $LogicalVision_{Poly}$ is significantly better than the compared statistics-based vision learning methods on these tasks, which suggests that symbolic learning can be of benefit to visual concepts learning.

6 Conclusion and future works

6.1 Conclusion and discussions

This paper studies a novel approach to the problem of visual concept learning, distinct from that employed by traditional computer vision learning algorithms. By using the proposed *Logical Vision* approach, we are able to exploit background knowledge flexibly and effectively in visual concepts learning tasks. Owes to its symbolic paradigm, the background-knowledge-guided mid-level features extraction and high-level visual concepts learning can be simply implemented by logic programming languages such as Prolog. The experimental results indicate that the proposed framework has potential to analyze which are traditionally hard for more statistically-oriented approaches.

The main reason for $LogicalVision_{Poly}$ outperforming statistics-based computer vision learners in our experiments is because it enables the incorporation of first-order

background knowledge which is very useful for learning the target concepts. *Logical Vision* exploits background knowledge in two ways: the first one is using high-level background knowledge to guide the observation of low-level features. This mechanism is reflected by the usage of the predicate `edge/3` in section 4.1. The second one is making use of background knowledge as primitive predicates like general Inductive Logic Programming approaches, e.g. the predicates `list_length/2`, `angles_list/2`, etc. defined in $Metagol_{LogicalVision}$. However, it is difficult for the statistical learning algorithms to incorporate these kinds of prior knowledge in its learning processes. For example, the HOG feature descriptor exploits local gradient information exactly same as the `edge_point/1` predicate, but its statistical learning framework lacks of a effective methodology to include other complex background knowledge to organize them into more informative mid-level features. On the other hand, the CNN feature descriptors can encode mid/high-level features of certain degrees, yet this ability could hardly be obtained from a small dataset.

Different to the statistics-based computer vision systems, Logical Vision treats the visual learning problem more likely to human cognition. Its learning process relies much more on the background knowledge than the quantity of data. Besides, the learned theory also can serve as background knowledge for subsequent tasks. Furthermore, it learns a constructive theory to define the target concept, rather than discriminative models that only tell differences between classes. From this angle of view, it is not hard to understand why did $LogicalVision_{Poly}$ outperforms the statistics-based methods in the experiments.

6.2 Future works

In this work, we built a primary system targeting for simple geometric concepts learning tasks such as polygon shapes learning. Although polygon identification itself cannot be directly applied to general computer vision tasks, it can be seen as a basic step for further visual recognition. As we illustrated in section 1, complex cognition tasks can be decomposed into easier sub-problems hierarchically [21]. After obtained knowledge of polygons, we can then define complex shapes with combination of separate/occluded polygons. Moreover, by taking consideration of pixels inside of polygon regions, we can define complex objects by shapes with patterns. Furthermore, we can include background knowledge about 3 dimensional geometry to help the understanding of an image by symbolic processing.

In recent future extensions of this work we hope to extend our study to more complicated tasks such as images involving a multiplicity of overlapping colored polygons as the first step.

7 Acknowledgment

This research was supported by the National Natural Science Foundation of China (61333014, 61321491), the RAEng Newton Research Collaboration Programme (NRCP/1415/133) and the Program B for Outstanding PhD candidate of Nanjing University.

References

1. Ahn, N., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition. Boston, MA (2015)
2. Andrzejewski, D., Zhu, X., Craven, M., Recht, B.: A framework for incorporating general domain knowledge into latent dirichlet allocation using first-order logic. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence. pp. 1171–1177. Barcelona, Spain (2011)
3. Antanas, L., van Otterlo, M., Oramas Mogrovejo, J., Tuytelaars, T., De Raedt, L.: There are plenty of places like home: Using relational representations in hierarchies for distance-based image understanding. Neurocomputing 123, 75–85 (2014)
4. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (SURF). Computer Vision and Image Understanding 110(3), 346 – 359 (2008)
5. Bengio, Y.: Learning deep architectures for AI. Foundations and Trends in Machine Learning 2(1), 1–127 (2009)
6. Borenstein, E., Ullman, S.: Combined top-down/bottom-up segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 30(12), 2109–2125 (2008)
7. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology 2, 27:1–27:27 (2011), software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: Proceedings of the 13rd IEEE Conference on Computer Vision and Pattern Recognition. pp. 886–893. San Diego, CA (2005)
9. Felzenszwalb, P.: Object detection grammars. In: Proceedings of the 13th IEEE International Conference on Computer Vision Workshops. pp. 691–691. Barcelona, Spain (2011)
10. Gary, B.: Opencv library. http://http://opencv.org/ (2000)
11. Hartz, J., Neumann, B.: Learning a knowledge base of ontological concepts for high-level scene interpretation. In: Proceedings of the 6th International Conference on Machine Learning and Applications. pp. 436–443. Cincinnati, OH (2007)
12. Hinton, G.E.: Learning multiple layers of representation. Trends in Cognitive Sciences 11(10), 428–434 (2007)
13. Jin, Y., Geman, S.: Context and hierarchy in a probabilistic image model. In: Proceedings of the 12nd IEEE Conference on Computer Vision and Pattern Recognition. pp. 2145–2152. New York, NY (2006)
14. Krig, S.: Computer Vision Metrics: Survey, Taxonomy, and Analysis. Apress, Berkely, CA (2014)
15. Krizhevsky, A., Sutskever, I., E. Hinton, G.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012)
16. Lake, B.M., Salakhutdinov, R., Gross, J., Tenenbaum, J.B.: One shot learning of simple visual concepts. In: Proceedings of the 33rd Annual Conference of the Cognitive Science Society. pp. 2568–2573 (2011)
17. Le, Q.V., Zou, W.Y., Yeung, S.Y., Ng, A.Y.: Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In: Proceedings of 2011 IEEE Conference on Computer Vision and Pattern Recognition. pp. 3361–3368. Colorado Springs, CO (2011)
18. Liu, J., Huang, Y., Wang, L., Wu, S.: Hierarchical feature coding for image classification. Neurocomputing 144, 509–515 (2014)

19. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
20. Maclin, R., Shavlik, J., Walker, T., Torrey, L.: Knowledge-based support-vector regression for reinforcement learning. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence Workshop on Reasoning, Representation, and Learning in Computer Games. Edinburgh, Scotland, UK (2005)
21. Marr, D.: Vision: a computational investigation into the human representation and processing of visual information. Henry Holt & Co., Inc., New York, NY (1982)
22. Mei, S., Zhu, J., Zhu, J.: Robust regbayes: Selectively incorporating first-order logic domain knowledge into bayesian models. In: Proceedings of the 31th International Conference on Machine Learning. pp. 253–261. Beijing, China (2014)
23. Muggleton, S.H., Lin, D., Pahlavi, N., Tamaddoni-Nezhad, A.: Meta-interpretive learning: application to grammatical inference. Machine Learning 94(1), 25–49 (2014)
24. Muggleton, S.H., Lin, D., Tamaddoni-Nezhad, A.: Meta-interpretive learning of higher-order dyadic datalog: Predicate invention revisited. Machine Learning (2015), published online: DOI 10.1007/s10994-014-5471-y
25. Ohta, Y.i., Kanade, T., Sakai, T.: An analysis system for scenes containing objects with substructures. In: Proceedings of the 4th International Joint Conference on Pattern Recognitions. pp. 752–754. Kyoto, Japan (1978)
26. Ojala, T., Pietikäinen, M., Mäenpää, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(7), 971–987 (2002)
27. Porway, J., Wang, Q., Zhu, S.C.: A hierarchical and contextual model for aerial image parsing. International Journal of Computer Vision 88(2), 254–283 (2010)
28. Quinlan, J.R.: Learning logical definitions from relations. Machine Learning 5, 239–266 (1990)
29. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. In: Proceedings of the 2nd International Conference on Learning Representations. Banff, Canada (2014)
30. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Proceedings of the 3rd International Conference on Learning Representations. San Diego, CA (2015)
31. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. CoRR abs/1312.6199 (2013)
32. Vedaldi, A., Lenc, K.: Matconvnet – convolutional neural networks for matlab. In: Proceedings of the 23rd Annual ACM Conference on Multimedia. Brisbane, Australia (2015)
33. Vedaldi, A., Fulkerson, B.: VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/ (2008)
34. Wielemaker, J., Schrijvers, T., Triska, M., Lager, T.: SWI-Prolog. Theory and Practice of Logic Programming 12(1-2), 67–96 (2012)
35. Zheng, S., Tu, Z., Yuille, A.: Detecting object boundaries using low-, mid-, and high-level information. In: Proceedings of 15th IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–8. Minneapolis, MN (2007)