Generative and probability models in image processing and computer vision

A. S. Potapov[a)]

ITMO University, St. Petersburg, Russia
S. I. Vavilov State Optical Institute, St. Petersburg, Russia

(Submitted February 19, 2015)
Opticheskiĭ Zhurnal **82**, 5–10 (August 2015)

This paper gives an analysis of the role of generative models in image processing and computer vision. Oriented and unoriented graphical models (Bayesian and Markov networks) are considered, along with the possibilities of using them in image processing, in particular, to solve problems of noise filtering, segmentation, and stereo vision. Probability programming is considered as a method of specifying arbitrary generative models that possess substantially larger expressive force than graphical models. It is shown that the main limitation of probability programming is associated with the problem of the output efficiency on arbitrary generative models, for the solution of which it is necessary to develop methods of automatic specialization of general output procedures. © *2015 Optical Society of America.*

OCIS code: (150.1135) Algorithms.
http://dx.doi.org/10.1364/JOT.82.000495

INTRODUCTION

Because computer-vision and image-processing systems are complicated, it is becoming ever more important to construct models that can cope with this complication[1]; to do this, it is efficient to resort to so-called generative models.

Generative models[2] describe the process of generating data by information sources. Such sources, in particular, can be visually observable objects and scenes, while the data can be their patterns or images (although science as a whole can be regarded as the region of application of generative models). Typical examples are generative formal grammars, which have a great influence in linguistics. It is natural in generative models to distinguish the observation conditions from the internal parameters of the objects, so that the values of the latter are invariant with respect to the former. The main problems of computer vision are associated just with restoring the inherent characteristics of objects, such as the three-dimensional shape and the reflectivity of the visible surfaces, as well as with recognizing objects regardless of the aspect from which the picture is taken and the illumination conditions.

It was noted log ago that the general problem of computer vision can be formulated as the reverse problem of computer graphics, in which a specified model is used to construct the image of a scene.[3] Not many people disagree that the problem of computer graphics is appreciably simpler. It is now possible to construct photorealistic images of virtually arbitrary scenes, whereas it still is much harder to restore high-quality descriptions or models of arbitrary scenes from images.

To solve problems of computer vision, it is hypothetically sufficient to have a system of computer graphics and to select a model of a scene so that the generated image is most similar to what is observed. The only problem of such an approach, which is fundamental, is that it is unrealistic from the computer's viewpoint to perform such a search directly. It is not surprising that preference is usually given in practice to methods that do not explicitly use generative models but that directly construct a description of an image (such methods can be called descriptive). For example, in the attribute approach to image analysis, attribute descriptions are extracted from the images without maintaining the possibility of reverse reconstruction of the original image or at least monitoring the lost information. Structural descriptions can store large amounts of information concerning the structure of the original image. However, even in that case, when alphabets of object-independent structural elements are used, developers sometimes attempt to abstract from the specifics of the shape the corresponding structural components of the original image to the maximum extent.[4] This is currently one of the leading approaches in structural image analysis.[5] Generative models, however, are important in that they divide the specifics of the object into parts, separating them from the specifically computational (although also fundamental) problem of solving the reverse problems.

In the area of machine learning, generative models became popular in the 1990s because efficient output methods were developed for a wide class of generative models—so-called graphical models (in which the combined probability distributions are represented in the form of graphs). In particular, these include Bayesian networks, in which the (statistical) dependences between all the quantities, including the hidden parameters of the system and the observable data, are represented in the form of an oriented acyclic graph, and Markov models, described by unoriented graphs.[6]

Generative models have an advantage when there is a ready-made set of tools for using them. Program libraries traditionally act as such. However, a more interesting possibility,

associated with the use of the semantics of the programming languages themselves, is implemented in the case of generative models. A program that contains random variables with specified probability distributions itself becomes the generative model, while the execution of this program, in which the *a posteriori* probability distribution is computed for specified random variables, becomes the output from the model, taking into account the limitations imposed on the program as a result of the computations that use them. Such semantics are implemented for graphical models, in particular, in the Infer.NET language,[7] which is an extension of the C# language.

Turing-complete probability languages have been developed in recent years with semantics in which a generative model can be specified in the form of an arbitrary program, including one that contains recursive calls. It is to be expected that the main limitations of such languages are associated with the fact it is impossible to efficiently derive the *a posteriori* probabilities in general. Strictly speaking, it is an algorithmically unsolvable problem to accurately derive the *a posteriori* distributions.[8] Nevertheless, the development of general output methods gradually expands the sphere in which these languages can be put into practice. Image-analysis problems now fall into this sphere.[3] The role of generative probability models is thus becoming more and more significant in the area of image analysis and computer vision.

GRAPHICAL PROBABILITY MODELS

A probability model of a certain system, as a rule, is represented in the form of a combined probability distribution of the quantities that describe this system. The probability output consists of calculating the *a posteriori* probabilities of the distribution of the hidden quantities, taking into account the values of the known observable quantities. However, the number of these quantities, particularly when working with images, can be very large, while the probability distribution function is multidimensional.

The use and thereby the evaluation of multidimensional distribution functions is problematic, in connection with which attempts have been undertaken by researchers to develop efficient representations of such functions. One of the basic ideas here is to structure the representation of the distribution functions on the basis of the conditions of the independence of random variables, and this allows the model to be made significantly more compact.

We shall consider as an example a Bayesian network given as the oriented acyclic graph $G = \langle V, E\rangle$, where V is a set of vertices, E is a set of arcs, and each vertex $v \in V$ corresponds to random variable X_v, while each arc $(u, v) \in E$ describes the direct dependence of random variable X_u on random variable X_v. In a Bayesian network, a conditional probability distribution $P(X_u|X_{v1}, \ldots, X_{vn})$ is associated with each quantity u, where $v_1, \ldots, v_n$ are all such values that $(u, v_i) \in E$. The set of such parental vertices for X_u is usually designated as $pa(X_u)$, which can be empty [that is, the distribution of the value of X_u is determined by its unconditional distribution $P(X_u)$]. As a result, the combined probability distribution of all the variables is represented via the product of conditional probabilities

$$P(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i|pa(X_i)). \quad (1)$$

Using the formula for the conditional probabilities and the marginalization, it is possible to calculate the *a posteriori* distributions of some random quantities (the hidden parameters of the model) for known values of the other (observable) random quantities. A number of methods of image analysis can be represented in the form of a Bayesian network—for example, the analysis of the main components. Because models that are given for vectors and matrices are often used in image analysis, a network is usually represented as a factor–graph. However, unoriented graphical models that describe Markov networks are more often used for image processing.

The connections between the vertices in a Markov network are undirected. The variables that correspond to any two nonadjacent vertices are conventionally statistically independent, taking into account all the other variables, and, conversely, any variable depends on its neighbors and only on them—i.e., it is conventionally statistically independent of any variable specified by a nonadjacent vertex, taking into account its neighbors.

Markov networks are associated with the formulation of problems of image processing and analysis as problems concerning the search for the maximum *a posteriori* probability when it is specified in the form of a Gibbs distribution,[1]

$$p(\mathbf{x}|\mathbf{y}) = \frac{1}{Z}\exp\left(-\sum_C E_C(\mathbf{x}_C;\mathbf{y})\right) = \frac{1}{Z}\prod_C \varphi_C(\mathbf{x}_C|\mathbf{y}), \quad (2)$$

where $\mathbf{x}$ is a vector of random variables, estimated from the observables $\mathbf{y}$; $E_C(\mathbf{x}_C;\mathbf{y})$ is a function of energy over $\mathbf{x}_C$, which is some set of elements $\mathbf{x}$ (the summation is carried out over various sets); and Z is a normalizing constant for distribution $p(\mathbf{x}|\mathbf{Y})$. The maximization of the *a posteriori* probability given by Eq. (2) obviously corresponds to minimization of the total energy function over all the sets,

$$E(\mathbf{x};\mathbf{y}) = \sum_C E_C(\mathbf{x}_C;\mathbf{y}). \quad (3)$$

The summation over various $\mathbf{x}_C$ in Eq. (2) is a consequence of the factorization of the combined $p(\mathbf{x}|\mathbf{y})$ distribution of all the variables in vector $\mathbf{x}$ on the product of the distributions $\varphi(\mathbf{x}_C|\mathbf{y})$ over vectors $\mathbf{x}_C$ of smaller dimension.

A graphical representation is convenient here because of the following: All the random variables in vector $\mathbf{x}$ can be represented as vertices of an unoriented graph, while the set of variables from which the summation is produced forms the cliques of this graph (i.e., subsets of the vertices in which each pair of vertices is united by an edge).

In a Markov model of fields, each variable in vector $\mathbf{x}$ is most often the brightness of a pixel of the output image, while each variable in vector $\mathbf{y}$ is the brightness of a pixel of the input image. Each term in Eq. (2) specifies, on one hand,

the dependence of the brightnesses of the pixels of the output image on the input image, and, on the other hand, the dependence between the pixels that correspond to the attached vertices in the output image. As a rule, vertices that correspond to the adjacent pixels are connected in a graphical model.

For example, in problems of suppressing the noise on an image, the probabilities $\varphi(\mathbf{x}_C|\mathbf{y})$ are greater, the closer the brightness $\mathbf{x}_C$ of each group of connected pixels of the output image is to the brightnesses of the corresponding pixels in the input image, as well as the closer they are to each other.[9] In the segmentation problem, the marks of the regions for each pixel appear as $\mathbf{x}$, and the correspondence between the probabilities that these marks take different or identical values depends on the degree of similarity of the pixels with respect to any attributes (for example, brightness or texture attributes).[10] When dense maps of the stereo disparateness are being constructed, the disparateness values of all the pixels appear as $\mathbf{x}$. The probability $\varphi(\mathbf{x}_C|\mathbf{y})$ is determined via the correlation energy (increasing the disparateness probabilities for which the correlation of the image fragments of the stereo pair is higher) and the bond energy (which causes sharp falloffs of depth).[11]

Graphical models are widely used and fairly popular in image processing and computer vision. There formerly was even a journal entitled *Graphical Models and Image Processing* (later renamed simply *Graphical Models*). However, it should be kept in mind that graphical models are a useful method if the probability distribution of the random quantities under consideration are factored with a sufficient degree of accuracy in the form of Eqs. (1) or (2).

However, the role of generative models in this area is not limited to graphical models. The currently popular deep-learning networks, which have broken through into image-recognition areas in recent decades, can often be represented in terms of generative models, but not always as graphical probability models. Such networks deserve to be considered separately. Here however, we will touch upon the most general tool of operating with generative models, which is still little used but possesses great potential in the area of image analysis—the tool of probability programming.

PROBABILITY PROGRAMMING

The most general method of specifying generative models is to specify them in the form of arbitrary algorithms. Actually, programs in any programming language can be used as generative models. However, if the programs include random-selection procedures (for example, by using a pseudorandom-number generator), they can be used as stochastic generative models, which implicitly specify the probability distribution in accordance with which observable objects are generated. The addition of random-selection procedures with standard distributions such as normal, multinomial, beta, gamma, etc. distributions makes it possible to compactly specify generative probability models in programming languages.

However, generative models are of interest when they can be used in probability output. Actually, such models specify the distribution of observer data, whereas the distribution of the *a posteriori* probabilities of the values of the hidden parameters of a system as the observations vary is of interest. A feature of probability programming languages is the fact that language constructions and the mechanisms that implement them are introduced into them. This make it possible to impose limitations on the values generated by the program, automatically creating a sample of the values of the requested variables in accordance with the conditions of the probability distribution.

The direct implementation of such output is to repeatedly launch a program that includes a random selection of hidden variables in accordance with specified *a priori* distributions and that returns values of the required variables, in this case sifting out the results of those launches during which the specified condition was not satisfied. Such a method, however, will be extremely inefficient in solving any nontrivial problems, since it involves blindly choosing the values of the hidden parameters.

Some probability-programming languages, such as Infer.NET,[7] are limited to directed or undirected graphical models, for which effective output models exist. However, an attempt is made in general-purpose languages such as Church to increase the selection efficiency without narrowing the class of models. The Monte Carlo Markov chain method or an analog of it is used for these purposes as a rule.[12–14] Such methods are also inadequate as a rule for working with vectors of real random variables and still more with images. Nevertheless, in some of the latter investigations, general methods of selection over programs in probability languages are sufficiently well developed to demonstrate nontrivial application in image-analysis problems.

In particular, problems of distinguishing road marking and recognizing symbols in the presence of noise are solved in Ref. 3, using a version of the Church language. The resulting solutions are described by generative models in the form of programs that consist of less than twenty lines of code (however, the generative models in this case are equipped with a library of computer graphics that contain, in particular, the type fonts used in generating the lettering). As an example, the generative model of an image of a road generates an image that consists of areas that correspond to the zones to the left and right of a road, the marking lines, and the road itself, whose size and position are determined by random quantities with certain specified *a priori* probability distributions. For each region, a compact histogram of the colors of the pixels is introduced into it, based on a training image. The condition imposed on the generative model for obtaining the *a posteriori* probability distributions of unknown random quantities (the parameters of the scene) is stochastic and takes the true value with greater probability, the higher the likelihood of the real image in the case of a specific generated scene.

The results obtained by means of generative models are comparable with the results that can be achieved by means of special methods on the basis descriptive models, the development of which requires appreciably greater efforts. Unfortunately, the authors do not show the time of operation of their programs in probability language, and this time is probably far from being practical. Moreover, the method of

comparing the generated and actual images is derived from a generative model (and is not implemented in a probability language) and cannot be applied to other image-analysis problems; this also somewhat reduces the generality of the given solution.

CONCLUSION

Stochastic generative models long ago became an important tool in image processing and analysis and computer vision. However, the most widely used still are graphical models, which have limited expressive force and are far from embracing all problems and methods in the given region.

Independent interest is felt in deeplearning networks, which have a close connection with generative models, but their effective use requires special output systems or description interpretations.

The development of probability programming languages that are capable of expressing arbitrary generative models by means of which, among other things, one can describe the solution to any problems of computer vision, is a promising step on the path to the development of the given specialization. However, a substantial obstacle here is the weakness of the output method. Can automatic output on arbitrary generative models in general be comparable in efficiency with traditional methods of image analysis?

Theoretical studies show[15] that this is possible, not when ordinary methods of statistical inference are used, but only with automatic synthesis of descriptive models as a result of specialization (the construction of an effective projection) of general output methods using specified generative models.

Generative models thus remain a promising area to be developed and one of the most important starting points for forming a generalized theory of image analysis. However, further development is required, using the theory of algorithms and metacomputations.

ACKNOWLEDGMENT

This work was carried out with the support of the Ministry of Education and Science of the Russian Federation and with the partial state support of the leading universities of the Russian Federation (Subsidy 074-U01).

[a)]Email: pas.aicv@gmail.com

[1]M. F. Tappen, "Modeling images with undirected graphical models," in *Image Processing and Analysis with Graphs: Theory and Practice*, R. Lukac, ed. (CRC Press, Boca Raton, Florida, 2012), chap. 16, pp. 475–496.

[2]N. D. Goodman and J. B. Tenenbaum, "Probabilistic models of cognition," http://probmods.org.

[3]V. Mansinghka, T. Kulkarni, Y. Perov, and J. Tenenbaum, "Approximate Bayesian image interpretation using generative probabilistic graphics programs," arXiv:1307.0060 (2013).

[4]V. Lutsiv and I. Malyshev, "Image structural analysis in the tasks of automatic navigation of unmanned vehicles and inspection of Earth surface," Proc. SPIE **8897**, 88970 (2013).

[5]V. R. Lutsiv and R. O. Malashin, "Object-independent structural image analysis: history and modern approaches," J. Opt. Technol. **81**, 642 (2014) [Opt. Zh. **81**, No. 11, 31 (2014)].

[6]D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques* (MIT, Cambridge, Massachusetts, 2009).

[7]S. S. J. Wang and M. P. Wand, "Using Infer.NET for statistical analyses," Am. Stat. **65**, 115 (2011).

[8]A. D. Gordon, Th. A. Henzinger, and A. V. Nori, "Probabilistic programming," *International Conference on Software Engineering*, May 2014, http://research.microsoft.com/pubs/208585/fose-icse2014.pdf.

[9]J. Rosa, J. Villa, Ma. A. Araiza, E. González, and E. Rosa, "A comparative study of some Markov random fields and different criteria optimization in image restoration," in *Advanced Image Acquisition, Processing Techniques and Applications*, D. I. Ventzas ed. (InTech, 2012), pp. 143–172.

[10]Z. Kato and T.-Ch. Pong, "A Markov random field image segmentation model for color textured images," Image and Vision Computing **24**, 1103 (2006).

[11]J. Sun, N.-N. Zheng, and H.-Y. Shum, "Stereo matching using belief propagation," IEEE Trans. Pattern Anal. Mach. Intell. **25**, 787 (2003).

[12]N. D. Goodman, V. K. Mansinghka, D. M. Roy, K. Bonawitz, and J. B. Tenenbaum, "Church: a language for generative models," in *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, Helsinki, Finland, July 9–12, 2008, pp. 220–229.

[13]V. Mansinghka, D. Selsam, and Y. Perov, "Venture: a higher-order probabilistic programming platform with programmable inference," arXiv 1404.0099 (2014).

[14]F. Wood, J. W. Meent, and V. Mansinghka, "A new approach to probabilistic programming inference," in *17th International Conference on Artificial Intelligence and Statistics*, 2014, pp. 1024–1032.

[15]A. Potapov and S. Rodionov, "Making universal induction efficient by specialization," *AGI 2014*, B. Goertzel, L. Orseau, and J. Snaider, eds., vol. **8598** in Lecture Notes in Artificial Intelligence, (Springer, New York, 2014), pp. 133–142.