An Inference Language for Imaging

Stefano Pedemonte[1 2], Ciprian Catana[1], Koen Van Leemput[1 2 3]

[1] Athinoula A. Martinos Center for Biomedical Imaging, MGH/Harvard, MA, USA
[2] Department of Information and Computer Science, Aalto University, FI
[3] Department of Applied Mathematics and Computer Science, Technical University of Denmark, DE

Abstract. We introduce iLang, a language and software framework for probabilistic inference. The iLang framework enables the definition of directed and undirected probabilistic graphical models and the automated synthesis of high performance inference algorithms for imaging applications. The iLang framework is composed of a set of language primitives and of an inference engine based on a message-passing system that integrates cutting-edge computational tools, including proximal algorithms and high performance Hamiltonian Markov Chain Monte Carlo techniques. A set of domain-specific highly optimized GPU-accelerated primitives specializes iLang to the spatial data-structures that arise in imaging applications. We illustrate the framework through a challenging application: spatio-temporal tomographic reconstruction with compressive sensing.

1 Introduction

Probabilistic reasoning combines deductive logic with the capacity of probability theory to handle uncertainty, providing an expressive formalism with a broad range of applications in many areas of artificial intelligence and machine learning. Stochastic programming languages address the model-building process by giving a formal language which provides simple, uniform, and re-usable descriptions of a wide class of models, and supports generic inference techniques [9, 11, 12, 10]. Probabilistic graphical models express explicitly the structure of probabilistic models by means of a graph, constituting a natural data structure for the design of stochastic programming languages.

The iLang framework is aimed at enabling the construction of models for imaging applications, focusing in particular on volumetric biomedical imaging. In this domain, probabilistic graphical models have been employed recently in a number of applications including image segmentation, tomographic reconstruction and multi-modal image processing. The *integrated modeling paradigm* has emerged in the work of K. Van Leemput [1], J. Ashburner [2], B. Fischl [3] and others in the context of medical image classification and alignment, adopting a model-based approach to devise algorithms for the joint estimation of multiple model parameters. Other instances of the integrated probabilistic modeling paradigm include the fusion of functional and structural information for the purpose of inferring anatomical-functional networks of the brain [4], the fusion of

information from MRI and PET [5] and the use of population-derived information for intensity-based classification of image structures [6]. In the aforementioned publications, probabilistic graphical models are employed for the purpose of describing the models and aiding the derivation of the symbolic expressions that the models imply; ad-hoc algorithms for maximum-a-posteriori inference are devised based on the resulting symbolic expressions. The iLang framework aims at enabling, under the integrated modeling paradigm, the construction of algorithms that incorporate image formation, motion correction, registration, classification, de-noising and other basic imaging tasks. The iLang framework addresses imaging as probabilistic reasoning; it includes a mechanism for the description of the model, i.e. a modeling language, a mechanism for the definition of inference queries, i.e. an inference language, and an inference engine that utilizes the data structures produced by the interpreter of the modeling language to perform inference. By using a formal modeling language, the computer gains the concept of a probabilistic model. Endowing the numerical representations of the probabilistic models with graph structures, then, enables the automated synthesis of efficient inference algorithms. The modeling language of iLang is based on language primitives designed for the construction of directed and indirected probabilistic graphical models. The inference engine of iLang addresses, without lack of generality, maximum probability and posterior sampling inference queries. The design of the language and of the data structures for the representation of the models yields a graph-based message-passing system that supports algorithms for maximum probability and posterior sampling. We describe two algorithms currently implemented in iLang: an algorithm for maximum probability estimation based on the Alternating Direction Method of Multipliers (ADMM) and an algorithm for posterior sampling of high dimensional models based on Hamiltonian Markov Chain Monte Carlo.

2 Methods

Imaging problems are not different, at the abstract level of probabilistic reasoning, from other computational problems that arise in artificial intelligence and machine learning, although imaging problems have two salient characteristics:

1. Imaging data is often very high dimensional;
2. An underlying structure, in computational problems related to imaging, arises from the spatial organization of the imaging data.

The high dimensionality may prohibit the use of certain classes of algorithms such as posterior sampling techniques. In imaging applications, the system matrices are often too large to be explicitly evaluated and stored in memory. The underlying structure, however, often can be exploited to evaluate efficiently matrix-vector multiplications on the fly and to increase the performance of the inference algorithms. A design challenge arises: abstracting imaging problems into the framework of probabilistic reasoning, therefore enabling the use of general purpose inference algorithms, while exploiting the underlying structure that arises from the spatial organization of the imaging data.

In iLang, the modeling language, the inference engine, and the inference query language are implemented as modules for the Python programming language. Language primitives constitute the units for the definition of probabilistic graphical models. As explained in the next section, the language primitives are based on a library of high performance geometric primitives which incapsulate the computations that emerge from the spatial structure of the imaging data (see Fig. 1).

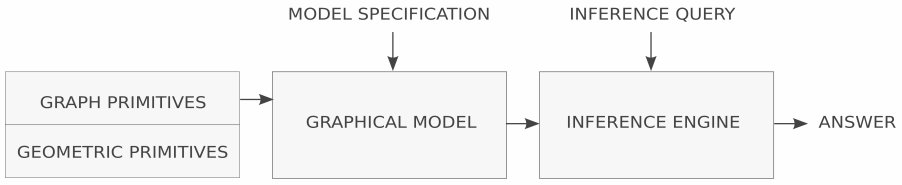

Fig. 1. The iLang probabilistic reasoning framework. The modeling language enables the definition of probabilistic graphical models using simple graph primitives. The inference engine infers the state of variables of a probabilistic graphical model. Imaging specific graph primitives are based on a library of high performance geometric primitives.

2.1 The Modeling Language

Models in iLang are constructed by defining a set of variables and specifying their interaction by means of a set of graph primitives. The following sections explain the rationale of the design of the iLang modeling language (section 2.1.1), describe the graph primitive construct (section 2.1.2), the model specification mechanism (section 2.1.3) and the geometric primitives that underly the graph primitives (section 2.1.4).

2.1.1 Set-Of-Rules on a Graph A probabilistic model expresses the joint probability distribution associated to a set of variables. One question that arises when designing a software framework for probabilistic reasoning is how to define a numerical representation of a probabilistic model and whether such representation enables the construction of efficient inference algorithms. Let us consider the following approaches to the numerical representation of probabilistic models:

1. *tabulation:* A table expresses the probability of each possible state of the variables (in case of discrete variables).
2. *symbolic representation:* The joint probability distribution of the variables is expressed by a mathematical formula.
3. *set-of-rules:* A computer program returns the joint probability for a given configuration of the variables.

Tabulation is only viable for small problems and for discrete variables. Symbolic representation of probabilistic models is an active research topic [8], enabling low memory representations of models and the automated computation of the derivatives of the probability functions via symbolic expression manipulation. While the symbolic approaches yield efficient and flexible tools for probabilistic reasoning [10], their use is currently limited to small problems where the system matrices can be explicitly evaluated. The third approach consists in writing a computer program that evaluates the probability associated to a given configuration of the variables and eventually the log-conditional probability functions associated to subsets of the variables and the derivatives of the log-probability functions. The most common approach to probabilistic modeling in medical imaging is based on the *set-of-rules* representation. One describes the model in symbolic form with pen-and-paper and manipulates the symbolic expressions to obtain expressions of the required conditionals, marginals and derivatives. A computer program that evaluates such expressions is then crafted. The conditional independencies of a probabilistic model can be represented by means of a graph (i.e. a probabilistic graphical model). The set of conditional independencies corresponds to the factorization of the joint probability distribution associated to the variables of the model. The explicit representation of the set of conditional independencies via a graph provides insight of the probabilistic model. The graph is often utilized, therefore, in order to aid the pen-and-paper symbolic expression manipulation and crafting of the computer programs: through the properties of the graph, one can tell which variables (Markov blanket) and factors contribute to the conditional probability distribution of a subset of the variables. The iLang framework adopts the model representation approach 3, in conjunction with a data structure based on the graph of the probabilistic model. Informing the computer software of the conditional independence structure of the model introduces many advantages. The combination of the set-of-rules approach and the graph, while allowing maximum flexibility, simplifies the model specification process, provides a mechanism for code encapsulation and provides a data structure suitable for the automated synthesis of inference algorithms. The core data-structure representing an iLang model is a graph, defined by a set of graph primitives. A graph primitive defines the interaction between a set of variables in terms of sets of rules for the computation of log-conditional probabilities and their derivatives, as described in the next section. Pen-and-paper symbolic manipulation is still part of the model definition process, however occurring only at the stage of designing a graph primitive.

2.1.2 Graph Primitives A graph primitive of the iLang modeling language expresses the interaction between a set of variables. Variables associated to a graph primitive are objects with a *name* property and a *value* property. The graph primitive object exposes, for each of the internal variables, one to four methods that return 1) the log conditional probability; 2) the gradient of the log conditional probability; 3) the Hessian of the log conditional probability; 4) the proximity map of the log conditional probability. This is depicted in Fig. 2. A

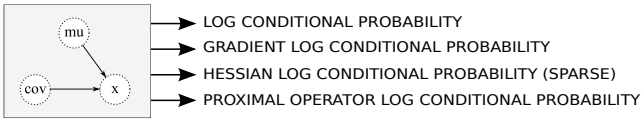

Fig. 2. Interface of a graph primitive: a graph primitive encodes the dependence amongst variables by specifying methods to compute the log conditional probabilities of each variable. Optionally, the graph primitive exposes methods to compute the first and second derivatives and the proximal operator of the log conditional probabilitties.

graph primitive is defined by subclassing a base object of type *GraphPrimitive*; defining a dictionary with the names of the variables; a dictionary that specifies the directed or indirected graph structure; and by implementing interface methods according to a simple predefined naming convention. The example that follows specifies a graph primitive that encodes a multivariate Gaussian probability distribution $p(x|mu, cov) = \mathcal{N}(x; mu, cov)$:

```
class MultivariateGaussian(GraphPrimitive):
    variables  = {'x':'continuous','mu':'continuous','cov':'continuous'}
    dependencies = [['mu','x','directed'],['cov','x','directed']]
    preferred_samplers = {'x':['HamiltonianMCMC']}

    # graph primitive interface
    def log_conditional_probability_x(self,x):
        hessian = self._compute_hessian()
        mu = self.get_value('mu')
        return -.5*numpy.dot(numpy.dot((x-mu),hessian),(x-mu).T)

    def log_conditional_probability_gradient_x(self,x):
        hessian = self._compute_hessian()
        mu = self.get_value('mu')
        return -.5*numpy.dot((x-mu),hessian+hessian.T)

    def log_conditional_probability_hessian_x(self,x):
        hessian = self._compute_hessian()
        return hessian

    # utility:
    def _compute_hessian(self):
        cov     = self.get_value('cov')
        self._hessian = numpy.linalg.inv(cov)
        return self._hessian
```

Note, in the example, that variables are defined as discrete or continuous. Further classes of the iLang variables will be added in future implementations, such as symmetric, positive definite and chordal matrices. The graph primitives currently implemented in iLang are reported in Fig. 3.

2.1.3 Model Specification A model is specified by instantiating an object of type *GraphicalModel* and naming the variables of the model. The dependence between the variables is specified by connecting the variables by means of graph primitives, as in the example that follows.

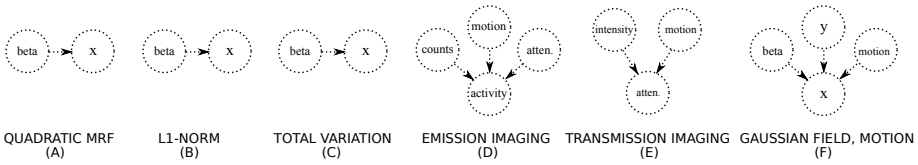

Fig. 3. Graph primitives currently implemented in iLang.

```
graph = ilang.GraphicalModel()
graph.add_variables(['var1','var2','var3'])
graph.add_model(MultivariateGaussian,{'var1':'x','var2':'mu','var3':'cov'})
graph.set_given({'var2':numpy.zeros([1,5]),'var3':numpy.eye(5)})
```

In this example, the object *graph* represents a probabilistic graphical model with 3 variables: *var1*, *var2*, *var3*; *var2* and *var3* have given values and *var1* is a 5-dimensional random variable with probability distribution $p(var1|var2, var3) = \mathcal{N}(var1; var2, var3)$. Note that the correspondence between the variables of the graph and the inner variables of the graph primitive has been specified in the *add_model* function call. The *graph* object exposes the methods that are required to perform inference; the internal machinery of the *graph* object translates the names of the variables, calling the methods of the graph primitives as required.

2.1.4 Geometric Primitives The graph primitives for imaging make use, internally, of efficient GPU-accelerated routines that perform common image processing tasks. Currently:

- Rigid spatial transformations
- Ray-tracing
- Image re-sampling
- FFT-IFFT
- Finite difference operator

Such geometric primitives enable a wide range of models and algorithms. The experiments section highlights how the spatial transformation, resampling and finite difference geometric primitives come into play to define a graph primitive that enables spatio-temporal tomography.

2.2 The Inference Engine

The probabilistic graph object provides all the methods required to perform inference. These include (proxy) methods to compute the log conditional probability of each of the variables and their first and second derivatives; methods to compute properties of the graph, such as the global Markov properties; and methods for the manipulation of the graph. Currently, the iLang framework implements an algorithm based on the Alternating Direction Method of Multipliers (ADMM) for maximum probability inference and an algorithm based on Hamiltonian Markov Chain Monte Carlo for posterior sampling.

2.2.1 Maximum Probability The algorithm for maximum probability estimation is based on a combination of the Iterated Conditional Modes (ICM) [7] algorithm and the ADMM algorithm [13]. The value of the variables of the model that maximizes the joint probability is computed by maximizing, in turn, the conditional probability distribution of each of the variables (ICM). The optimization of each conditional probability is performed by means of the ADMM algorithm. ADMM, developed in the context of convex optimization [13], has the advantage of enabling the use of non-differentiable factors, such as the models (B) and (C) in Fig. 3, using the proximity operators of the factors in place of the first derivatives. In order to apply ADMM to optimize the log conditional probability for each of the variables, the inference engine performs a transformation of the graph, consisting in extracting the Markov blanket of the variable and transforming it into a Forney-style augmented factor graph [14], as exemplified in Fig. 4 for variable x. The ADMM algorithm then consists in a message passing algorithm over the factor graph. The Forney-style factor graph represents the augmented Lagrangian of the local optimization problem expressed in consensus form [13, 14]. The Forney-style factor graph is a bipartite graph obtained by placing on the right side one node for each factor of the cost function (3 nodes in the example, corresponding to 3 graph primitives) and one node on the left side, encoding an equality constraint. The factor graph in this form expresses the augmentation of the optimization problem with variables x_1, x_2, x_3 (see Fig. 4-right). The point of the augmentation is that edge variables attached to the same equality constraint must ultimately equal each other, but they can temporarily be unequal while they separately try to satisfy different cost functions on the left. Finally, the problem is augmented with one variable for each edge connecting the two sides of the bipartite factor graph: the Lagrangian multipliers y_1, y_2, y_3. The ADMM algorithm consists in the exchange of the following

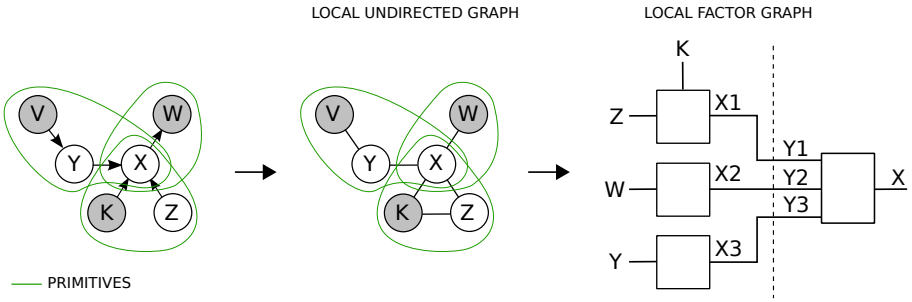

Fig. 4. Transformations of a probabilistic graphical model. Left: a directed probabilistic graphical model; center: moralized undirected gaph; right: Forney-style factor graph utilized by the inference engine of iLang.

messages (see [13] for the derivation of the messages):

$$x_k^{n+1} := \arg\min_v f_k(v) + \frac{\rho}{2}\|v - y_k^n + x^n\|_2^2 \tag{1}$$

$$x^{n+1} := \frac{1}{N}\sum_{k=1}^{N} x_k^{n+1} + \frac{1}{\rho N}\sum_{k=1}^{N} y_k^n \tag{2}$$

$$y_k^{n+1} := y_k^n + x_k^{n+1} - x^{n+1}, \tag{3}$$

where f_k is the k-th factor (with $k = \{1, 2, 3\}$ in the example) and ρ is the augmented Lagrangian regularization parameter (the default value is $\rho = 0.1$, see [13] for a discussion on the selection and adaptation of ρ). The splitting introduced by data augmentation enables the use of non-smooth factors. If f_k is smooth (the graph primitive corresponding to factor k exposes a method to compute the gradient of the log conditional probability), the inference engine performs the minimization using, by default, the L-BFGS Quasi-Newton algorithm, or the Newton algorithm if the graph primitive exposes a method to compute the Hessian of the log conditional probability. If the factor is non-smooth, the inference engine sets x_k^{n+1} by evaluating the proximity operator of f_k, calling the proximity operator method of the underlying graph primitive.

2.2.2 Posterior Sampling The posterior sampling algorithm currently implemented in iLang is based on Markov Chain Monte Carlo. Each of the variables of the graph are sampled in turn by sampling from their conditional probability distributions (Gibbs sampling). The samples from each of the conditional probability distributions are obtained by means of various MCMC techniques, depending on the methods exposed by the factors of each conditional probability distribution. A local factor graph analogous to Fig. 4-right is constructed; if all the graph primitives connected to variable x expose methods to compute the gradient of the conditional probability of x, the MCMC algorithm uses Hamiltonian dynamics [15] with gradient equal to the sum of the gradients returned by each primitive. The local set of variables is augmented with momentum variable q and each new sample of x is obtained by sampling a candidate of q from a normal probability distribution and then by sampling x conditionally to q as follows (see [15]):

$$q^{n+1}|x^n \approx p(q^{n+1}|x^n) = p(q^{n+1}) = \mathcal{N}\left(q^{n+1}|0, M\right) \tag{4}$$

$$x^{n+1}|q^{n+1} \approx p(x^{n+1}|q^{n+1}), \tag{5}$$

samples of x^{n+1} from $p(x^{n+1}|q^{n+1})$ are obtained by integrating the Hamiltonian dynamics over fictitious time τ from the initial values q^{n+1} and x^n. The integration is performed using the leapfrog method:

$$q(\tau + \frac{\epsilon}{2}) = q(\tau) + \frac{\epsilon}{2}\nabla_x f(x(\tau)) \tag{6}$$

$$x(\tau + \epsilon) = x(\tau) + \epsilon M q\left(\tau + \frac{\epsilon}{2}\right) \tag{7}$$

$$q(\tau + \epsilon) = q(\tau + \frac{\epsilon}{2}) + \frac{\epsilon}{2}\nabla_x f(x(\tau + \epsilon)), \tag{8}$$

with a certain number of steps, with step size ϵ, to give proposed moves x^* and q^* and accepting or rejecting according to the Metropolis Hastings criterion, as specified in [15]. Here ∇_x denotes the gradient of factor f and M is a weight matrix. The weight matrix, by default, it set to the identity, unless all the factors expose methods to compute the Hessian, in which case it is set to the sum of the Hessian terms, producing a piecewise constant Riemannian Manifold Hamiltonian MCMC algorithm [15]. Although the choice of the parameter ϵ is in general critical in order to obtain high acceptance ratios, especially in high dimensions, setting M to the Hessian of the factor f, as discussed in [15], enabling the algorithm to scale to high dimensions and relaxes the choice of ϵ. This is the default mode of iLang if all the factors expose the Hessian method. The default value of ϵ is 0.1.

3 Motion-aware Positron Emission Tomography

In PET imaging, the low number of photon counts per unit time imposes long acquisition times (several minutes). During the acquisition, the subject moves, determining blurring and ghosting effects in the reconstructed images. Although attempts have been made to measure the motion of the subject during the acquisition of the PET data by using motion detection devices, the problem is still largely unsolved. The problem can be formulated in the probabilistic framework as follows, in the case, applicable to brain imaging, of rigid motion. Although the activity in the imaging volume changes over time due to motion, let us assume, disregarding pharmacokinetics in this first instance of spatio-temporal model, that the rate of emission in the frame of reference that moves rigidly with the head of the patient is constant. Assuming that the only source of uncertainty associated to the measurements is the inherent uncertainty due to photon counting, the conditional probability distribution associated to the photon counts at time t, given the motion parameters at time t and the activity in the reference frame, is a Poisson distribution. Let us denote with $q_d^{[t]}$ the photon counts along line of response (LOR) d at time t; with $z^{[t]} = \{z_1^{[t]}, z_2^{[t]}, \ldots, z_d^{[t]}\}$ the vector of the photon counts at time t; with $A = \{a_{bd}\}$ the matrix of the probabilities that an event emitted in voxel b is detected in LOR d; with $R_{\gamma^{[t]}}$ the rigid transformation at time t, parameterized by parameters $\gamma^{[t]}$ and with $\mathcal{P}$ the Poisson distribution:

$$p(z^{[t]}|\lambda, R_{\gamma^{[t]}}) = \prod_d \mathcal{P}(\sum_b a_{bd}[R_{\gamma^{[t]}}\lambda]_b, z_d^{[t]}) \tag{9}$$

Let us assume a sparsifying total-variation prior probability distribution for the activity:

$$p(\lambda|\beta) \propto e^{-\beta\|\nabla\lambda\|_1} \tag{10}$$

Denoting by $\bar{1}$ the vector of 1's, the gradient of eq. (9) is given by (see [5]):

$$\frac{\partial}{\partial\lambda_b}\log p(\lambda|z_{[t]}, R_{\gamma^{[t]}}) = -\sum_t R_{\gamma^{[t]}}^T A^T\bar{1} + \sum_t R_{\gamma^{[t]}}^T A^T \frac{z^{[t]}}{AR_{\gamma^{[t]}}\lambda} \tag{11}$$

Let us assume that the motion parameters $\gamma^{[t]}$ are unpredictable, i.e. that the motion parameter $\gamma^{[t]}$ is a priori independent from the motion parameter $\gamma^{[t']}$, $t \neq t'$. By the chain rule of differentiation, the derivative of the log conditional probability of the i-th motion parameter at time t is given by:

$$\frac{\partial \log p(\gamma^{[t]}|z^{[t]},\lambda)}{\partial \gamma_i^{[t]}} = \sum_d - \left[A\left[\frac{\partial R^T_{\gamma^{[t]}}\lambda}{\partial \gamma_i^{[t]}}\right]\right]_d + z_d^{[t]} \frac{\left[A\left[\frac{\partial R^T_{\gamma^{[t]}}\lambda}{\partial \gamma_i^{[t]}}\right]\right]_d}{\left[AR^T_{\gamma^{[t]}}\lambda\right]_d} \qquad (12)$$

Optimization of the joint probability with respect to the model parameters is not trivial due to the non-differentiability of the prior and to the non-negativity constraint (here not expressed explicitly) of λ. In iLang, the calculations of eq. (11) and (12) are encapsulated in the graph primitive (D) of Fig. 3 and the graph primitive (C) of Fig. 3 implements the proximity operator for the total variation prior (i.e. soft thresholding of the image gradient - see [13]). The model is encoded in iLang as follows:

```
graph = ilang.GraphicalModel()
graph.add_variables('lambda')
for t in range(Nt):
    graph.add_variable('z'+str(t))
    graph.add_variables('gamma'+str(t))
    graph.add_model( ilang.Primitives.PET_rigid_motion, ..
      {'lambda':'activity','gamma'+str(t):'motion','z'+str(t):'counts'})
    graph.set_given('z'+str(t))
    graph.set_value('z'+str(t), sinograms[t])
graph.add_variable{'beta'}
graph.add_model(ilang.Primitives.TotalVariation, ..
  {'lambda':'x','beta':'sparsity'})
graph.set_given('beta')
graph.set_value('beta',0.1)
```

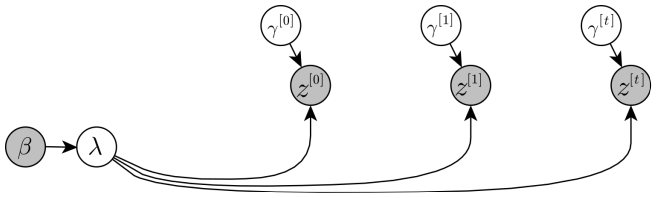

Fig. 5. Graph generated by iLang for motion-aware Positron Emission Tomography.

where `sinogram` is a list of N_t sinogram arrays. This produces the graph of Fig. 5. Inference is performed as follows:

```
sampler = ilang.Sampler(graph)
sampler.maximum_probability(max_iterations=100)
activity_estimate = sampler.get_last_sample('lambda')
```

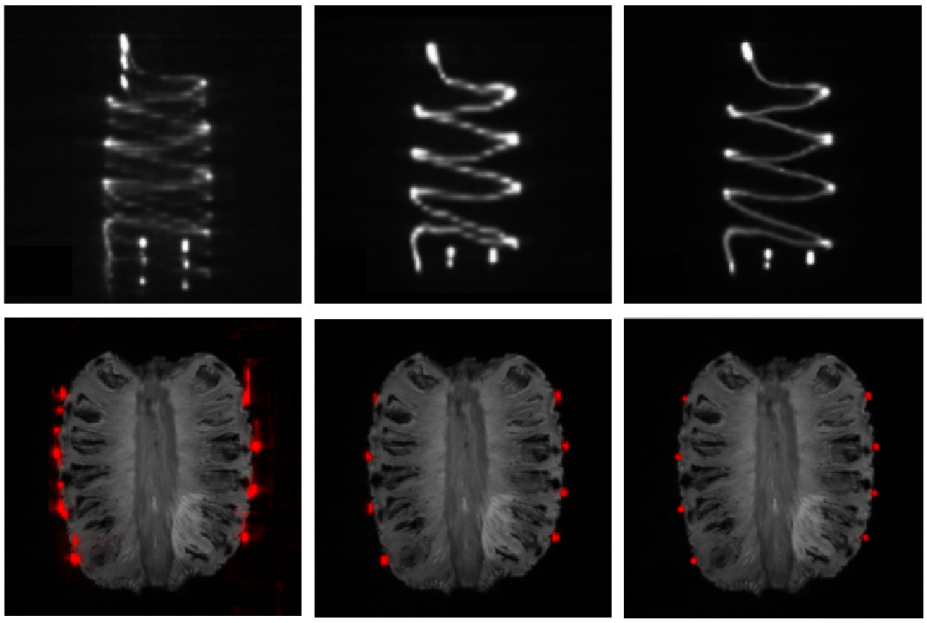

Fig. 6. Motion-Aware PET: Reconstructions obtained by imaging an FDG-filled capillary source wrapped around a pineapple. From left to right: no motion correction; motion estimation; motion estimation and total-varation prior. Top row: volume rendering; bottom row: representative slice - overlay of PET and MR.

4 Conclusion

The iLang software framework enables probabilistic reasoning in volumetric imaging, simplifying the definition of complex imaging models. Endowing the numerical representation of probabilistic models with a graph enables the automated synthesis of efficient inference algorithms. The inference engine of iLang enables the definition of non-smooth constraints such as non-negativity and sparsity. The iLang software constitutes a unified framework for multi-modal imaging that enables the integration of image formation, registration, de-noising and other image processing tasks. The application reported in section 3 constitutes a novel powerful imaging paradigm, where motion is considered a nuisance variable and estimated from the PET emission data under the assumption of sparsity.

5 Download

The iLang software is distributed with a permissive open source license at `http://ilang.github.io`

6 Acknowledgements

This research was supported by the NIH NCRR (P41-RR14075), the NIH NIBIB (R01EB013565), and TEKES (ComBrain).

The authors would like to thank Paulina Golland and the MIT EECS/CSAIL journal club for the useful introduction to the ADMM algorithm.

References

1. Leemput, K.V. Maes, F. Vandermeulen, D. Suetens, P.: Automated model-based tissue classification of MR images of the brain. In: In: IEEE Trans Med Imaging. Oct. 1999, 18(10), 897–908.
2. Ashburner, J., Friston, K.: Unified segmentation. In: Neuroimage, 26(3), 839–51, 2005.
3. Fischl, B. Salat, D.H. Van Der Kouwe, A. Makris, N. Segonne, F. Quinn, B.T. Dale, A.M.: Sequence-independent segmentation of magnetic resonance images. In: Neuroimage, 23, 69–84, 2004.
4. Venkataraman, A. Rathi, Y. Kubicki, M. Westin, C.F. Golland, P. Joint modeling of anatomical and functional connectivity for population studies. In: IEEE Trans Med Imaging. Feb. 2012, 31(2), 164–82.
5. Pedemonte, S. Bousse, A. Hutton, B.F. Arridge, S. Ourselin, S.: 4-D generative model for PET/MRI reconstruction. In: MICCAI 2011, 14(Pt 1), 581–8.
6. Menze, B.H. Leemput K.V., Lashkari, D. Weber, M.A. Ayache, N. Golland, P.: A Generative Model for Brain Tumor Segmentation in Multi-Modal Images. In: Jiang, T., Navab, N., Pluim, J.P.W., Viergever, M.A. (Eds.) MICCAI 2010, Part II, LNCS 6362, pp. 151–159. Springer-Verlag Heidelberg 2010.
7. Besag, J.: On the Statistical Analysis of Dirty Pictures. In: J. of the Royal Stat. Soc. Series B (Methodological), 48(3), 259-302, 1986.
8. Bergstra, J. Breuleux, O. Bastien, F. Lamblin, P. Pascanu, R. Desjardins, G. Turian, J. Warde-Farley D. and Bengio, Y.: Theano: A CPU and GPU Math Expression Compiler. In: Proceedings of the Python for Scientific Computing Conference (SciPy) 2010. Austin, TX.
9. Goodman, Noah, et al.: Church: a language for generative models. In: arXiv preprint arXiv:1206.3255, 2012.
10. Patil, A. Huard, D. and Fonnesbeck, C.J.: PyMC: Bayesian Stochastic Modelling in Python. In: J. Stat. Softw. Jul 2010; 35(4): 1–81.
11. Stan Modeling Language Users Guide and Reference Manual, 2014. http://mc-stan.org/
12. Lunn, D.J. Thomas, A. Best, N. Spiegelhalter, D.: WinBUGS – a Bayesian modelling framework: concepts, structure, and extensibility. In: Statistics and Computing, 10:325–337.
13. Boyd, S. Parikh, N. Chu, E. Peleato, B. and Eckstein, J.: Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers In: Foundations and Trends in Machine Learning, Vol. 3, No. 1 (2010) 1–122.
14. Forney Jr, G.D.: Codes on graphs: Normal realizations. In: IEEE Transactions on Information Theory, 47(2):520548, 2001.
15. Girolami, M. and Calderhead, B.: Riemann Manifold Langevin and Hamiltonian Monte Carlo Methods. In: Journal of the Royal Statistical Society: Series B (Statistical Methodology) 73.2, 123–214, 2014.