Data-Driven Surgical Workflow Detection: Technical Report for M2CAI 2016 Surgical Workflow Challenge

Olga Dergachyova[1,2], David Bouget[1,2] Arnaud Huaulmé[1,2], Xavier Morandi[1,2,3], and Pierre Jannin[1,2]

[1] INSERM, U1099, Rennes, F-35000, France,
[2] Université de Rennes 1, LTSI, Rennes, F-35000, France
[3] CHU Rennes, Département de Neurochirurgie, Rennes, F-35000, France
e-mail: olga.dergachyova@univ-rennes1.fr

1 Methodology

The current method was initially described in [1]. This report is partial extract from the full-version paper including some additional implementation details, please see the reference for more information. The method was designed as generic meaning that its processing pipeline does not depend on the surgical domain of the provided data. The surgical workflow detection of our approach consists in four stages, as shown in Figure 1. Firstly, a Surgical Process Model is constructed as described in section 1.1. The next stage is to create descriptions of video frames (see section 1.2). Section 1.3 presents intermediate AdaBoost classification done in the third stage. In the last stage, detailed in section 1.4, Hidden semi-Markov Model provides final classification results.

The method is able to perform in real-time if required frequency is not higher than one prediction per second. Thus, only the first frame of each one-second period is analysed, and the prediction obtained from it is assigned to all frames of the same period.

1.1 Surgical Process Modelling

The main idea of our approach is to remain generic and independent towards the data and apply the same algorithm to different datasets. To implement this idea we rely on Surgical Process Modelling [2]. A surgical procedure can be represented with an individual Surgical Process Model (iSPM) describing its order of surgical task execution. The union of several iSPMs gives a generic model called gSPM showing different possible ways to perform the same procedure. In addition, it is possible to extract from the gSPM some useful information such as a list of all tasks and their durations.

First we transform frame-per-frame training data annotations of the form $A_i = \{t_1(p_1), t_2(p_1), ..., t_j(p_2), ...\}$ into iSPMs in form $P_i = \{\rho_1 = p_1(t_{beg}, t_{end}, d), \rho_2 = p_2(t_{beg}, t_{end}, d), ..., \rho_j = p_n(t_{beg}, t_{end}, d)\}$, where t is time (local time stamp), d is duration, ρ is procedure state and p is surgical phase. In this work the gSPM

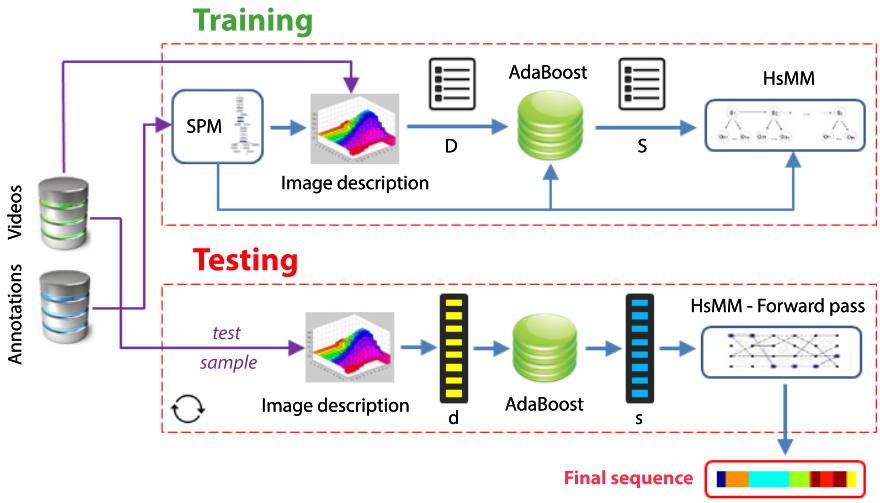

Fig. 1. Scheme of the proposed four-stage approach for surgical workflow detection, outlining processing at training and testing time. D stands for descriptions of all training samples, S for signatures of all training samples, d for test sample description, and s for test sample signature.

is represented in the form of a directed weighted graph automatically computed from iSPMs. They are parsed in order to extract all unique surgical phases which play the role of vertices of the graph. Then, we derive all edges, meaning transitions between phases. When the structure of the graph is in place, we assign an attribute to each vertex and edge. The vertex attributes indicate phase durations. Each vertex stocks the information about the durations of the assigned phase in all training sequences, as well as their mean, minimal and maximal value. The edge attributes define the probability of continuing by performing the phase pointed out by the edge. Initially this attribute is represented by the total number of transitions made by the edge in all training sequences, after what it is normalized by the sum of all transitions of the edges going out from the same vertex. We use the model and its statistics to manage the following detection process in all stages.

1.2 Image description

The second stage of the approach is to describe input video frames. In order to maintain the generic aspect of the method, the images from the videos are described with no attachment to particular areas or objects and no assumption on surgery type. Only standard global visual cues are used for that purpose. This choice is driven by two reasons: standard image descriptors are generally computationally light, and they allow successful classification of all kinds of images unlike other more complex descriptors designed for specific goals. Three main

image aspects are examined: color, shape and texture. The color is represented in form of histograms from RGB, HSV and LUV color-spaces. The shape is described by Discrete Cosine Transform (DCT) and Histograms of Oriented Gradients (HOG). The texture is transmitted through Local Binary Patterns (LBP) histograms. The image description is computed as follows. The RGB histogram of the entire image contains 16 bins for each color component. The sums of all bins are also computed for each color component. The same is done for the HSV, except Hue component having 18 bins. For the LUV color-space we extract only L component in 10 bins, and we also take their sum. For the DCT representation only 25 values, corresponding to the highest frequencies, are taken. We compute 6 HOG of 9 bins on 6 different areas of the image. Four histograms are computed from 4 rectangular sectors of equal size going from the image center to its corners. The fifth sector of the same size is focused on the center of the frame. The last sector contains the entire image. For the LBP histogram we use 58 uniform patterns only. All computed values are concatenated into a visual description vector of total 252 values ($3*16+3*1+18+2*16+3*1+10+1+25+6*9+58 = 252$). It serves as input for the following AdaBoost classification.

1.3 AdaBoost classification

In order to classify each analysed sample into surgical phases, the third stage of our method relies on AdaBoost classifier using image descriptions from section 1.2 as input. The boosting approach, underlying concept of AdaBoost, is a machine learning technique using a great number of weak classifiers assembled into a cascade to form one strong classifier. The AdaBoost algorithm [3] finds features that separate positive data samples from negatives the best. The advantage of AdaBoost applied to our problem is its capacity to analyse each data aspect separately to find the most discriminant ones unlike SVM, for example, which takes all features in a scope. To be more concrete, visually each surgical phase can possibly differ from all others only by one or couple of features (e.g., specific color component or gradient direction), so it can have its own particularities. Thereby, in our case each surgical phase needs a proper classifier of type one-vs-all distinguishing it from all others.

During training time, for each surgical phase, we create a set of positive (i.e. belonging to this phase) and negative (i.e. belonging to any other phase) samples represented by description vectors. The sizes of datasets are balanced by random selection of representative samples. Then the AdaBoost cascade specific to this phase is trained on the built set. Decisional trees of two level depth are used as week classifiers. The depth of the cascade, meaning the number of training rounds, is empirically set to 100. Usually, when the number of input features is large, the AdaBoost algorithm performs a random pooling to select a subset of features. Knowing that we have only 252 features, we prefer to examine them all. Thus, on each round $t \in T$ we analyse features one-by-one to find a combination (root node and leaves of the decisional tree) that allows the best separation, i.e. minimizing the error (1).

$$\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i], \tag{1}$$

where $h_t(x_i)$ is the decision of the week classifier on round t taking input sample $x_i \in X$ having label $y_i \in Y = \{-1, +1\}$, and D_t is a distribution (i.e. set of weights expressing importance of each sample) over the training set on round t.

First we determine a feature for the root and then for two leaves. Once the week classifier h_t is set, it obtains a parameter $\alpha_t \in \mathbb{R}$ that measures its importance in cascade, which is computed as follows:

$$\alpha_t = \frac{1}{2}\, ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \tag{2}$$

Initially, all the weights in the distribution are equal, i.e. $D_1(i) = 1/m$, where m is the number of positive and negative samples in total. But on each round, the weights of misclassified samples are increased in order to pay more attention on hard examples on the next round:

$$D_{t+1}(i) = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}, \tag{3}$$

where Z_t is a normalization factor (chosen so that D_{t+1} be a distribution).

At the end of the training we have N AdaBoost cascades, where N is the number of surgical phases ($N = 8$ for the challenge dataset), it is defined by the computed gSPM.

At testing time, we compute AdaBoost cascade decision as majority vote:

$$H(x) = sign(O(x)) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right), \tag{4}$$

where $O(x)$ is a raw output of the cascade classifier.

We transform the raw output into a confidence scores from 0 to 100, indicating classifier certainty, as follows:

$$S(x) = abs\left(\frac{O(x) \cdot 100}{\sum_{t=1}^{T} \alpha_t}\right) \tag{5}$$

For the sake of speed gain in testing time AdaBoost cascade may be transformed into "soft" one by checking on each round if the confidence about the prediction exceeds a given threshold in order to stop the cascade descend. But for the challenge, we used "rigid" cascade meaning that all weak classifiers were consulted to obtain a prediction.

Suchwise, each sample passing through all cascades obtains a signature of $2 * N$ length consisting of N positive/negative responses (1 or -1) and N confidence scores. Each i_{th} response shows if the i_{th} classifier recognizes the sample as belonging to its phase or not. The values of the signature containing confidence scores are divided into k intervals. Here k is empirically set to 20. This is needed to obtain a discrete signature which is used as input for the next stage of our method.

1.4 Hidden semi-Markov Model

In this final stage we construct a predictive model taking into account the temporal aspect of the procedure to improve the detection capacity of our method. Hidden Markov Model is a powerful tool for modelling time series. Based on observed data, the model seeks to recover the sequence of hidden states that the process comes through. Applicable to surgical workflow segmentation, the surgical phases would be the hidden states we want to discover and the data samples (signatures from the last stage in our case) would be the observations. An HMM is formalised as $\lambda = (S, O, A, B, \pi)$, where S is a finite set of states, O is a finite set of observations (sometimes called vocabulary), A is a set of probabilities of transition from one state to another, B is a probability distribution of observation emissions by states, π is a probability distribution over initial states (see [4] for more details).

Classical HMM has a major drawback: exponential state duration density due to auto-transitions. This is poorly suited for modelling of physical signals and processes with various state durations. An extension of the classical model exists, it is called explicit-duration HMM or Hidden semi-Markov Model (HsMM). A state duration probability distribution P is added to the model $\lambda = (S, O, A, B, \pi, P)$ and the probabilities of all auto-transitions in A are set to zero.

The last stage of our algorithm is HsMM training on signatures made of AdaBoost responses used as observations. First of all, following model parameters are initialized thanks to gSPM:

- vector S of N length, N being the number of phases; set by the vertices of the graph
- matrix $A(N \times N)$; set by edge attributes of the graph
- vector π of N length; set by edge attributes of the graph
- matrix $P(N \times D)$, where D is the maximal duration of any phase; set by vertex attributes of the graph

Then, a finite observation vocabulary O^M is built from all unique signatures found in the training data, where M is the number of unique signatures. B is initially computed by counting the number of occurrences of all signatures from O in each phase. In theory, B is a $(N \times M)$ matrix, but in practice B is $(N \times M{+}1)$ matrix. It is done to be able to process signatures not existing in the vocabulary.

Theoretically, some the cells in the matrices A (e.g. diagonal showing auto-transitions), B (e.g. observations not seen in certain phases, as well as the additional column), π (e.g. phases that never indicate the beginning of the procedure) and P (e.g. moments $d \in D$ with no transition) should be set to 0. But in order to avoid divisions by zero and some other technical issues, we set them to $eps = 1e - 30$. Also, this will enable possible transitions between phases that have not been observed in the training dataset. The model is then refined using

the forward-backward algorithm from [5][1] computing the marginal likelihoods of a sequence of states.

At testing time, the sequence of signatures representing the surgical procedure is decoded one-by-one with the forward pass (slight modification of [5]). When the first signature of the sequence is available, the initialization shown in algorithm 1 is done. Matrix α, forward variable, is saved for later use during following iterations. The index of the B matrix corresponding to observed signature O_t at time t is denoted $v(O_t)$. The phase is then computed with algorithm 3. As soon as next signature becomes available, we execute one round of iteration (see algorithm 2) operating with previously saved α. After this iteration the state estimation is performed again to update the current phase if needed. The above operation is dynamically repeated till the end of the sequence. For the theoretical explanation of the applied method please refer to the source [5].

We assume that the combination of forward and backward passes would give better results. But respecting challenge condition stating that detection should be made in online mode, we omit backward pass and perform state estimation procedure just after the latest forward iteration.

Algorithm 1 Initialization

$\alpha \leftarrow \text{zeros}(N, D+1)$
for each state S_n **do**
  for $d \leftarrow 1, 2, ..., D$ **do**
    $\alpha[n,d] \leftarrow \pi[n] \cdot B[n, v(O_1)] \cdot P[n,d]$
  end for
end for

Algorithm 2 Iteration

$X \leftarrow (\alpha.\text{col}(1)' \cdot A)'$
for each state S_n **do**
  for $d \leftarrow 1, 2, ..., D$ **do**
    $\alpha[n,d] \leftarrow \alpha[n,d+1] \cdot B[n, v(O_t)] + X[n] \cdot B[n, v(O_t)] \cdot P[n,d]$
  end for
end for
$\alpha \leftarrow \alpha \cdot 1/\text{sum}(\alpha)$ ▷ scaling to avoid possible underflows

Algorithm 3 State estimation

for each state S_n **do**
  $Prob[n] \leftarrow sum(\alpha.\text{row}(n))$
end for
$Label \leftarrow \underset{N}{argmax}(Prob[n])$

[1] Matlab source code: http://sdcs.sysu.edu.cn/space/005315/SourceCode.html

2 Results

The training dataset provided for the M2CAI 2016 Surgical Workflow Challenge contained 27 videos of cholecystectomy surgery with phase annotations, whereas the testing dataset consisted of 15 videos.

As stated by the challenge organizers after evaluation performed on submitted resulting detections, our approach yields in 51.5 for Jaccard index and reaches 70.7% accuracy after taking into account the 10-second relaxed boundary.

References

1. O. Dergachyova, D. Bouget, A. Huaulmé, X. Morandi, P. Jannin: Automatic data-driven real-time segmentation and recognition of surgical workflow. IJCARS 11(6), 1081–1089 (2016)
2. P. Jannin, X. Morandi: Surgical models for computer-assisted neurosurgery. NeuroImage 37(3), 783–791 (2007)
3. R. Schapire: The boosting approach to machine learning: An overview. Nonlinear estimation and classification, 149–171 (2003)
4. L. Rabiner: A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
5. S. Yu, H. Kobayashi: An efficient forward-backward algorithm for an explicit-duration hidden Markov model. IEEE signal processing letters 10(1), 11–14 (2003)