Automated Workflow Formation for IoT Analytics: A Case Study

Tanushyam Chattopadhyay(✉), Avik Ghose, Arijit Mukherjee, Santa Maiti, and Arpan Pal

Innovation Labs, Tata Consultancy Services, Kolkata, India
{t.chattopadhyay,avik.ghose,mukherjee.arijit,
santa.maiti,arpan.pal}@tcs.com

Abstract. The rapid deployment of sensors across the world in various sectors has fuelled a growing demand of smart applications and services that can leverage this boom of Internet of Things (IoT). However, developing analytical applications for IoT is a difficult process as applications tend to be cross-domain and there are close relationships with the physical world. It is unreasonable to imagine that the application developers will possess all relevant skills and knowledge related to the domain, physical world, signal processing and deployment infrastructure. This paper presents an method that assists the IoT application developer by (i) providing an annotated repository of algorithms, (ii) recommending algorithms depending on the signal type to reduce the effort required from a signal processing expert, and (iii) providing a framework to execute the IoT application thereby reducing the development cost and time by capturing the knowledge of experts in models. We have evaluated our method by comparing the accuracy for a typical IoT application obtained by using the algorithms used by signal processing experts against the algorithms recommended by our method.

Keywords: Model driven development · IoT analytics · Automated work flow creation

1 Introduction

The importance of *not-so-distant-future* sensor-based intelligent and ubiquitous systems has been apparent during the past few years when global technology giants have all underlined the potential of the "Internet of Things (IoT)". The technology is capable of creating a vast network of smart devices and connected things to an unprecedented extent and enhance solutions and services in multiple domains such as healthcare, energy & utilities, transportation etc. and also delve into the development of cross-domain solutions and services. However, developing IoT applications is inherently difficult due to the requirement of cross domain and cross technology knowledge. This poses requirements for technical experts who charge heavily for their time, thereby increasing product costs. The primary motivation of this work is to reduce the dependency on the availability of such

B. Mandler et al. (Eds.): IoT 360° 2015, Part II, LNICST 170, pp. 36–43, 2016.
DOI: 10.1007/978-3-319-47075-7_5

niche skilled workers and automate the process of application development by using knowledge models covering the aforesaid aspects. Working towards this vision, we have created a framework that is capable of recommending suitable algorithms for sensor signal processing once the application goal is defined. We have also constructed a knowledge-base and associated libraries for extracting a super set of features that can be used in any signal processing work-flow in order to automatically select the most suitable set of features for classification. We make an hypothesis that feature extraction and selection methods depend on the sensor signal class and the classification goal. This is the step we have automated at the first level, this is because this step is intutive and knowledge driven, requiring maximum time of the domain and algorithm experts.

Traditionally IoT applications [1,2] involves four steps namely (i) sensor integration, (ii) sensor data collection, (iii) data storage, and (iv) data analysis. Existing platform as services (PaaS) for IoT like Google App Engine, Heroku doesn't support any specialized services for IoT application development. [3,4] proposes an integrated application development platform but no work flow automation tool is provided. We found only two prior works that tries to bring in automation. In [5], authors have used a database that stores prior performance and complexity of different algorithms. They have initially converted raw data into a set of features that can be mined and then used in automatic selection of appropriate algorithms based on the problem data set. However, the proposed method has no means to capture the various levels of knowledge. Also they do not provide means to define the goal or the sensors to be used. In [6], authors have presented a survey which provides useful insights regarding automation of the outlier removal process if the data type can be properly classified.

We illustrate our work by automating an example work-flow for computing heart-rate (HR) using Photo Plethysmograph (PPG). This is a well researched topic and we have chosen to compare our results against [11]. We study the features used by this approach and then try to reason out the same using knowledge driven method, this is illustrated in Sect. 2.

The proposed system consists of a designer interface, an algorithm repository and a execution framework. The designer interface allows users to author application work-flows using a web-based interactive Graphical User Interface (GUI). The algorithm repository is a crowd-sourced repository of algorithms which are designed and implemented by researchers, domain experts and analysts using a common set of languages and a common interface. The execution framework provides the spinal cord for such design. The framework is detailed in Sect. 2.2.

2 Proposed Method

The generic signal processing work flow involves the steps as shown in Fig. 1.

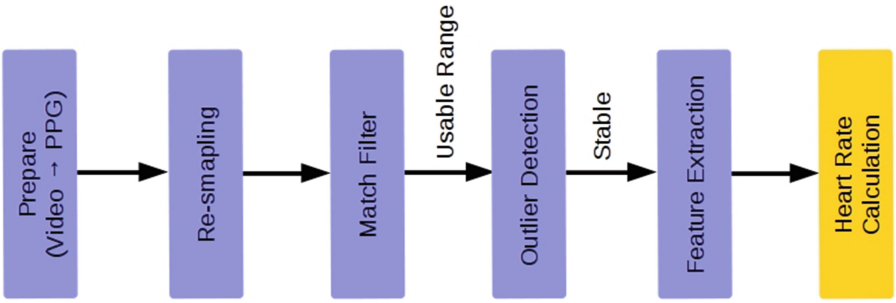


Fig. 1. High level workflow for IoT signal processing

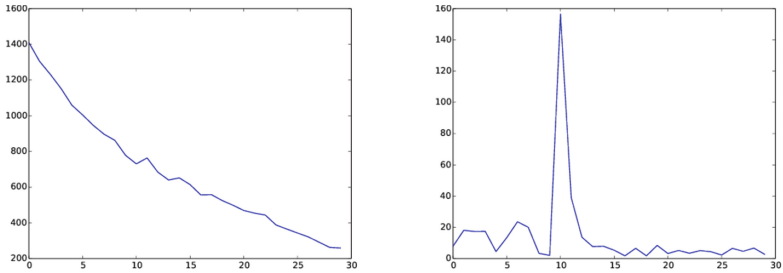
2.1 Work-Flow Design

Prepare: The first phase of our work flow is prepare. In this phase the input video is converted into PPG. As discussed, we have used the method described in [10]. This paper takes only the red component of the RGB video obtained from mobile camera. Then the video is cropped from the central region and an average of 'R' component for all the pixels within the region of interest is computed. This is reported as a single sample of the PPG time-series.

Re-sampling: This module is required when the input sample rate of the input varies over time or varies for different devices. This is also required for predicting the result more precisely. For example if we capture video at 30 Frames per Second (FPS), each sample represents almost 2 heart beats per minute on the other hand if we sample at 15 FPS, each sample point is equivalent to 4 heart beats per minute. Thus sample rate is also indicative for the precision level of the outcome. We capture this expected sample rate information from the question like "what is the expected accuracy level for this application". Once the precision requirement is finalized by the developer, we translate it to the required sample rate. Simple re-sampling algorithms are used to re-sample the input signal to the required sample rate.

Matched Filter: Match filter is required to truncate the signal using the domain knowledge. This parameters are obtained from the question "What is the possible range of heart rate?" to be answered by the developer. Any one having the domain knowledge gives this input to be within the range of 50 to 180 (normal). In this use case the specific range of heart beat allows to truncate the frequency range and also to estimate the periodicity of the time domain signal.

Outlier Detection: Outlier detection is an important module in this workflow as it helps to remove the undesired part of the signal and thus reduce the execution time. Moreover the frequency response of the signal after removing the



(a) FFT of the signal before noise cleaning (b) FFT of the signal after noise cleaning

Fig. 2. Comparison of FFT with and without noise cleaning

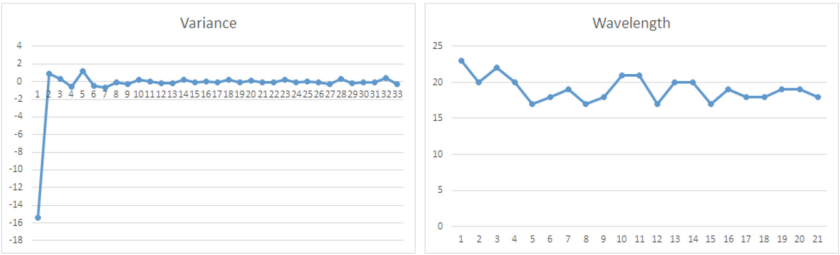
outlier is more informative. In the Fig. 2a shows the frequency response of the signal before outlier removal and the Fig. 2b represents the FFT (Fast Fourier Transform) of the input signal after removing the outlier. Our outlier detection module takes the knowledge that there should be a consistency of the signal over time as there is a periodicity in the signal. Outlier removal involves two sub-steps namely (i) estimation of window length, and (ii) outlier removal.

In the proposed method the pseudo code for window estimation is like:

- Compute the valid range of the periodicity or wavelength from the input given in match filter
- Vary the window size within this range (say here it is 10 to 45 for a 30 FPS video)
- Compute the KL distance (d_i) between the time domain signature of two consecutive i and $(i+1)^{th}$ windows
- Find the i for which d_i = minimum of $d_i \forall i$
- Repeat this for all samples in the signal and get the different window size for the signal and store it in a buffer

The next step is to compute the statistical moment variance for each window. We are not considering mean as mean value of the signal over a window may vary over time. This is a property of PPG signal. So we have considered only the difference between the variance of the time domain values over windows. We also applied a threshold based approach to remove the outlier. In the Fig. 3a we have shown the first order difference of variance over time.

Feature Extraction: We have constructed a super set of features those are usually used in signal processing. Our feature set considers the different features used in signal processing. We have initially reduces the feature set depending on the type of the signal as shown in Table 1. As we know that the input signal for this use case is stationary and periodic in nature we have used FFT as the frequency domain feature and the time domain features are described below: We have already defined the method of window selection.



(a) Difference of Variance over cycles (b) Time Domain Detected cycles

Fig. 3. Variance over cycles and periods

Table 1. Recommended features depending on the signal type

Type	Type	Recommended feature
Periodic	Stationary	FFT and time domain peaks
Periodic	Non-stationary	DWT, and envelope
Aperiodic	Linear	STFT and DWT
Aperiodic	Non-linear	Time delay embedding and recurrence plot

- Compute the k number of local peaks with top k amplitudes for each window.
- Let $px_{i,j}$ and $py_{i,j}$ represents the sample number and the corresponding amplitude respectively for the j^{th} peak of the i^{th} window.
- Compute the ratio of $px_{i,j}$ and $px_{i+1,j}$ for all i and j values and let it be feature f_1.
- Compute the ratio of $py_{i,j}$ and $py_{i+1,j}$ for all i and j values and let it be feature f_2.
- Compute the difference of $px_{i,j}$ and $px_{i+1,j}$ for all i and j values and let it be feature f_3.
- Compute the difference of $py_{i,j}$ and $py_{i+1,j}$ for all i and j values and let it be feature f_4.

Now we observe that the for PPG signal f_3 is consistent over time. From the ontology as we have defined earlier, it is a mandatory feature for this use case. So we have not considered other features in this use case and used only f_3 for heart rate computation. Similarly we get the heart rate from the FFT of the signal, too. In our experiment we have used k = 3. Thus we get 3 time domain values and one frequency domain value as expected heart rate. Finally we have taken an average of these four values to compute the final heart rate.

2.2 Execution Framework

The execution framework provides a method to build and execute the code generated using the work-flow execution. The execution framework performs the following tasks.

– Build the algorithm code for target platform
– Generate models for target platform code. These can be statistical or mathematical models as standard model files
– Generate the application code for target platform using work-flow logic
– Deploy the code on target platform using Open Systems Gateways Interconnect (OSGI) standards
– Run the code and generate results on user request

The execution model for portable platforms has been discussed in [12], which our work re-uses to implement the execution model and takes the framework beyond phones to other embedded gateway platforms. The execution framework is details in Fig. 4.

The framework provides skeleton for parameter passing, returning results and function sequencing using a simple construct of data structures and function references. All functions in-take a generic set of parameters which are converted to specifics inside the implementations. The functions can be chained or branched. So, the framework interface is a composite design pattern which combines both facade, strategy and adapter design pattern. The function flow follows a hybrid between pipe design pattern and the decorator design pattern, allowing both stage-wise refinement as well as same step improvement. The code for the final application is generated by following this framework by the code-generation engine which picks algorithms from the design work-flow and stitches them together into the framework.

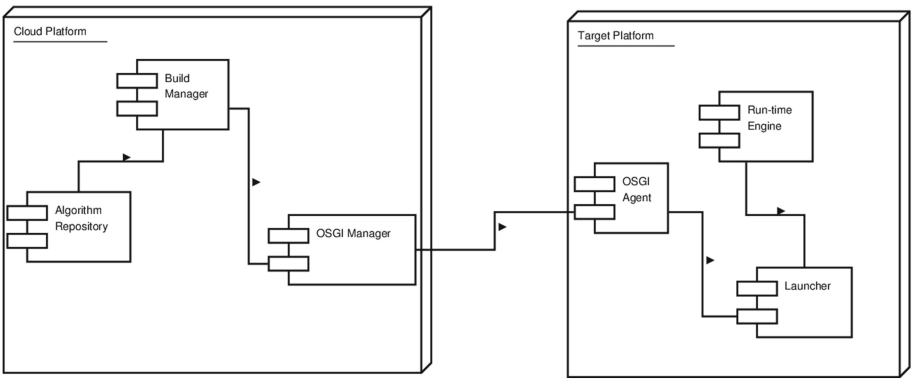


Fig. 4. Execution framework overview

3 Results and Discussions

We have used the feature recommended by our work-flow and also used our recommended windowing method to generate the code for [10]. The code was generated using our execution framework for Android phones and the generated C code was integrated into the phone using Java Native Interface (JNI).

We tested the accuracy against our captured data that comprises of 316 PPG readings captured using different mobile devices like Samsung S-Duos, iPhone 4, Intel Xolo, Sony Xperia, and LG Nexus 4. We are going to describe the method in terms of two aspects (i) accuracy of the proposed automated work flow generation and (ii) reduction in development time using our proposed work. We have evaluated our result against the state of the art method described in [10] and we have found that our proposed method which uses automatically generated feature set produces almost same result. The comparison with ground truth and state of the art work as reported in [10] is presented in Table 2. It is evident from the table that the predicted feature set works with almost same accuracy.

Table 2. Comparison of the accuracy obtained by purposed tool generated algorithm workflow and manual human programmed workflow

User	1	2	3	4	5	6	7	8
Actual heart rate	54	66	84	106	80	105	105	80
Heart rate in [10]	53	63	84	98	88	102	104	81
Heart rate in present	54	65	84	98	89	101	104	80

The results described in Table 2 shows that our result is quite compliant with the ground truth. Here all heart rates reported are the rates per minute. The noise in some signal that were not detected in our automated outlier generation method is the reason behind error with respect to ground truth. The features we have used in this work is a superset of the features used in [10] and thus the result is also close with this work. The development time for the [10] work was three months to develop the algorithm chain. On the other hand the similar workflow was generated using our tool is 3 days only.

4 Conclusion

In this paper we have presented a method that can assist the application developer to write any signal processing application without details knowledge on signal processing or algorithm. He needs to have some domain knowledge which is captured to populate an ontology in a question and answer manner. In this paper we are also claiming that the feature set to be used does not depend on the type of sensor and instead it is dependent on the type of the signal. We have also proposed a super set of possible features that are to be processed for any stationary signal. Dimensionality of the feature can be reduced either by giving the ground truth as the input for feature optimizer like MIC. Otherwise the domain knowledge is captured by question and answer based manner from the developer as we did it in this use case. Finally we have shown that the results obtained by automatic feature selection is producing the result which is at per with the manually selected features. So this work will help the future researchers

of signal processing application developers in IoT. We have also shown this on a simple use case but can try on complicated problems like [13,14].

References

1. Gubbi, J., et al.: Inter-net of Things (IoT): a vision, architectural elements, and future directions. Elsevier J. Future Gener. Comput. Syst. **29**, 1645–1660 (2013)
2. Balamurali, P., et al.: Software platforms for internet of things M2M. J. Indian Inst. Sci. Multidiscip. Rev. J. **93**(3), 487–498 (2013). ISSN 0970-4140 Coden-JIISAD
3. Pal, A., et al.: Model driven development for internet of things: towards easingthe concerns of application developers. In: International Conference on IoT as a Service, IoT360 Summit, Rome (2014)
4. Misra, P., et al.: A computing platform for development and deployment of sensor data based applications and services. Patent No. WO2013072925 A2
5. Automatic mapping from data TP preprocessing algorithms. US Patent No WO2002073529
6. Gupta, M., Gao, J., Aggarwal, C.C., Han, J.: Outlier detection for temporal data: a surve. IEEE Trans. Knowl. Data Eng. **25**(1), 1–20 (2014)
7. Sackner, M.A., Inman, D.M.: Method and system for extracting cardiac parameters from plethysmographic signals. US Patent Number: 6783498, 31 August 2004
8. Wieringa, F.P., et al.: Contactless multiple wavelength photoplethysmographic imaging: a first step towards SpO2 camera technology. Ann. Biomed. Eng. **33**(8), 1034–1041 (2005)
9. Humphreys, K., et al.: Noncontact simultaneous dual wavelength photoplethysmography: a further step toward noncontact pulse oximetry. Rev. Sci. Instrum. **78**(4), 044304 (2007)
10. Pal, A., et al.: A robust heart rate detection using smart-phone video. In: Proceedings of 3rd ACM MobiHoc Workshop on Pervasive Wireless Healthcare. ACM (2013)
11. Banerjee, R., et al.: PhotoECG: Photoplethysmography to estimate ECG parameters. In: ICASSP 2014
12. Ghose, A., et al.: Design insights for a mobile based sensor application framework: for aiding platform independent algorithm design. In: IPSN 2015
13. Chattopadhyay, T., et al.: Recognition of channel logos from streamed videos for value added services in connected TV. In: IEEE International Conference on Consumer Electronics (ICCE) (2011)
14. Ghosh, H., et al.: Multimodal indexing of multilingual news video. Int. J. Digit. Multimedia Broadcast. (2010)