A Multithreaded Architecture on Android to Efficiently Cater Integral Projections based Face Tracking Algorithm to Handheld Devices

Suman Kumar
Centre of Excellence Division,
Frog, 6th Floor, Sigma Tech
Park, Bangalore-560 066

Vijay Anand
Centre of Excellence Division,
Frog, 6th Floor, Sigma Tech
Park, Bangalore-560 066

ABSTRACT
Integral Projections based face tracking algorithm provides high accuracy and computational efficiency to accurately track human faces. However, the inherent complexity of having three dimensions and the usage of projection model with each dimension has posed a challenge to deploy on resource constrained handheld devices.

Motivated by these observations, we propose a multi-threaded architecture on Android to address this issue. First, we discuss few related works to investigate and analyze the Integral Projection face tracking algorithm and detail the constraints associated to identify face tracking accurately. Second, we discuss few characteristics of android architecture and detail constraints associated with middleware sub-system to cater face tracking efficiently. Consequently, we describe the proposed architecture and the rationale of functionality based module classification to address inherent challenges like reducing false positive rate on resource constrained devices. In particular, we point out the benefits of the Face Algorithm Agent to ensure smooth and swift portability onto different flavors of android. To validate our approach, trial experiments were performed using the said architecture on Zoom2 hardware platform based on Android. Experimental results lead us to conclude the architecture is efficient at reducing the false positive rates significantly. Finally, future research directions based on the current architecture results are pointed out.

Keywords
Face Tracking, Android, Architecture, QoE.

1. INTRODUCTION
Literature survey reveals that a wide range of approaches have been proposed to deal with the problem of human face tracking. Most of these algorithms are based on skin-like color detection [1]–[8], which is a well-known robust method to track human heads and hands. Color detection is usually followed by a location of individual facial features, for example, using PCA [2], splines [3] or integral projections [4], [5], [6]. However, in most of the existing research, projections are processed in a rather heuristic way. Other kinds of approaches are adaptations of face detectors to tracking, e.g. based on contour modeling [9], [10], eigenspaces [11] and texture maps [12]. These methods are computationally expensive and sensitive to facial expressions, so efficiency is usually achieved at the expense of reducing location accuracy. On one hand, most of research efforts have been concentrated on improving the face tracking rate and reducing the dimensionality to cater more efficiently. Latest advancements include Integral Projection Based Face Tracking algorithm that can not only be used to track human faces more accurately, but the dimensionality reduction involved in projection yielding substantial improvements in computational efficiency and location accuracy. Such an integral projection based face tracking model has been implemented using Intel IPL and OpenCV libraries [13], for desktop. However, its inherent computational complexity doesn't suit best on resource constrained hand held devices.

On other hand, Android, a software platform from Google has pioneered to cater Value Added Services and innovative applications on handheld devices [16]. However, the packet video multimedia framework employed within android significantly contributes to the overall system overhead while catering such computation intensive algorithms like Integral projection based face tracking. Also, in an effort for easier portability across platforms, android has a modular architecture which supports codecs as OpenMax compliant components. These abstractions further add up significant delay to the overall system performance. This constraint in android, multimedia framework in particular has resulted in in-effective usage of the underlying hardware platform which has a dedicated Digital Signal Processor (DSP) for such high computation operations. This shortcoming in the android middleware sub-system has motivated us to design a novel robust multi-threaded architecture to cater projection based face tracking algorithm effectively.

In this paper, we propose a multithreaded architecture on Android to address these constraints. The said architecture ensures that complex computations are carried out on DSP and the computation results are used by General Purpose Processor (GPP) for further post-processing. We point out the need for a middleware component, Face Algorithm Agent which not only utilizes the multimedia framework efficiently but also reduces the false positive rate which might occur in a multi-threaded environment. A proof-of-concept architecture extending the Google's android architecture is developed to verify the feasibility of the approach. To conclude, we summarize the results of the trial experiments conducted.

2. DESIGN APPROACH
An integral projection is a one-dimensional pattern, obtained through the sum of a given set of pixels along a given direction. Let i(x; y) be an image and R(i) a region in it, the vertical integral projection of R(i), denoted by PV R(i) is given by: PV R(i)(y) = i(x; y); 8(x; y) 2 R(i). The horizontal projection of R (i), denoted by PHR (i), can be defined in a similar way. The

first step above all, is to detect the face in the first frame. The face detection technique described in [9] is used as initialization. Then, the integral projection based face tracking algorithm is applied on these co-ordinates of the detected face. The tracking algorithm is part of an iterative process, which recalculates the location of the face and facial features in a sequence of images. For each face being tracked, a bounding ellipse and the location of eyes and mouth are computed. The input to the tracker is a face model, the state of tracking in the previous image, it¡1, and a new image it. The face model consists of a set of projection models of the whole face and some parts in it, where the locations of facial features are known. This model is computed using the first image of the sequence, where eyes and mouth locations are given by the face detector. However, computation of face model takes more processing time than the rate at which the frames are received especially in real-time scenarios like live capture. This constraint has resulted with high false-positive face tracking rate. Consequently, we propose a multi-threaded architecture on android to achieve overall high detection rate and low false-positive rate. The goal is to construct a novel architecture on android to cater to both face detection and face tracking algorithm efficiently. In this approach, each algorithm runs as a different task i.e., computation of face model and face tracker runs as two different tasks respectively. Section 3 details more on the design of such an efficient architecture, its benefits and the associated constraints. Section 4 summarizes the results of trial experiments conducted.

3. THE PROPOSED MODEL

The prerequisite for any face tracking algorithm is the face model. The face model is the computed result of face detection algorithm applied on the first video frame. However, computational complexities involved in implementing any existing robust face detection algorithm takes around 1-2 seconds on hand held devices. Two main design issues surface immediately. First, computation of face tracking algorithm on the result of detected face further contributes to the delay by which time the object (human face) in focus might have moved beyond the searchable window area. Second, the availability of results of face detection may not necessarily coincide with the scheduling of the face tracking algorithm on the correct video sequence. This has not only resulted in degraded performance, but has also resulted in high false-positive and low-predictable face tracking rate, especially on resource constrained handheld devices. Consequently, in the proposed model, we have addressed these two critical issues as shown in fig 1. The diagram describes the codec-sub system architecture of Zoom2 on android. It has two building blocks: The ARM side core and the DSP side core of OMAP chipset. The face detection algorithm which possesses high computational complexity is executed as a new task (FDT) to execute first video sequence on Digital Signal Processor (DSP) co-processor of the underlying platform for faster execution. This resolves the first issue of the design constraint. Subsequent reception of video frames continues using General Purpose Processor (GPP). This addresses the other issue thereby gathering the data set for face tracking algorithm while results from FDT is awaited.

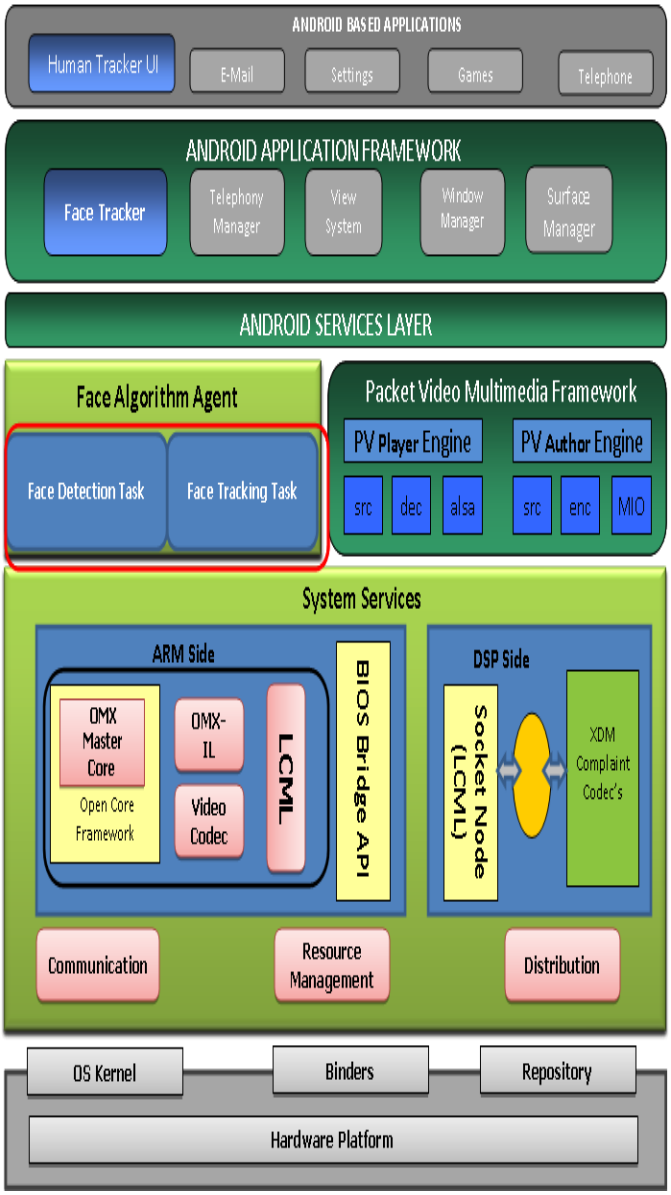


Fig 1: System Architecture View

The design of android architecture has two notable advantages: First, the communication among components in a layered architecture is asynchronous in nature. Second, the functionality based layer classification to offer their services seamlessly to the applications. These architectural goals are primarily leveraged by the defined android middleware component called Face Algorithm Agent (FAA). This component is responsible to schedule, co-ordinate between Face Detection Task (FDT) and Face Tracking Task (FTT). It also acts as an interface with android middleware sub-system. On reception of first video sequence, the FAA is the first middleware component that gets executed. This component schedules the face detection algorithm task as a DSP component for faster execution. In parallel, subsequent reception of video sequences continues on

GPP and gets stored in a ring buffer. Upon completion of the face detection task, the resulting face model is given as input to the face tracking task (FTT). The FTT uses integral projection based algorithm which operates on the face model and the training data which was gathered and stored in the buffer. This model minimizes the false positive output reasonably well.

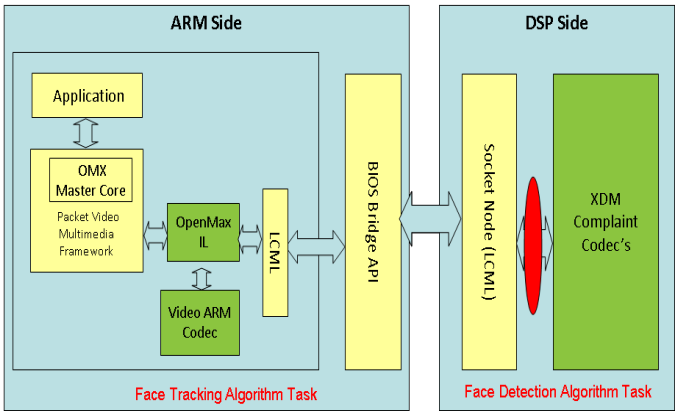


Fig 2: Meta Architecture View

3.1 Face Algorithm Agent Execution Mechanism

The flow chart shown below (in fig. 3) details the execution mechanism of the FAA component. This component schedules the incoming video sequence appropriately so as to reduce the false positive rate detection of human face. On reception of first video sequence, a new task (FDT) is created and this video buffer is provided as input to this task to compute the co-ordinates of face. This FDT task is scheduled to run on the DSP processor for better system throughput. Subsequent video frames are buffered by the GPP which acts as training set for the FTT. Upon successful completion of FDT, the output co-ordinates are fed to the FTT to track the movement of human face in the video sequence. However, if the video is received at a faster rate, say at 30 frames per second (30 fps), then by the time the output of FDT is available there is a high probability that the position of human face would have changed. In such as case, processing of FTT will result in false positive detection as the FTT will be operating on the result of first frame of previous second.

3.2 Reduction of high false-positive detection rate

Our approach solves this problem at two stages. Firstly, it runs the FDT on high computing DSP processor which yields result faster than GPP. This reduces the overall time taken in computing FDT. Secondly, an auto co-relation function is used at the output of FTT which determines if the output is valid at that instance in time or not. This combined approach minimizes the false positive rate significantly. So, in this model, to compensate for the erroneous result due to delay contributed by Face Detection Task, we apply auto co-relation function (ACF) on the latest video sequence and the face model output of FDT. Assuming, at a reception rate of 30 fps and computation time of FDT as 1 sec, we take ACF between the 30th frame and the first frame to identify similarities in the output obtained. This ensures an additional level of check to confirm if the detected face is still in the same location or has moved beyond the kernel of 32 * 32. In this way, we are able to reduce the false positive rate by nearly 10 %.

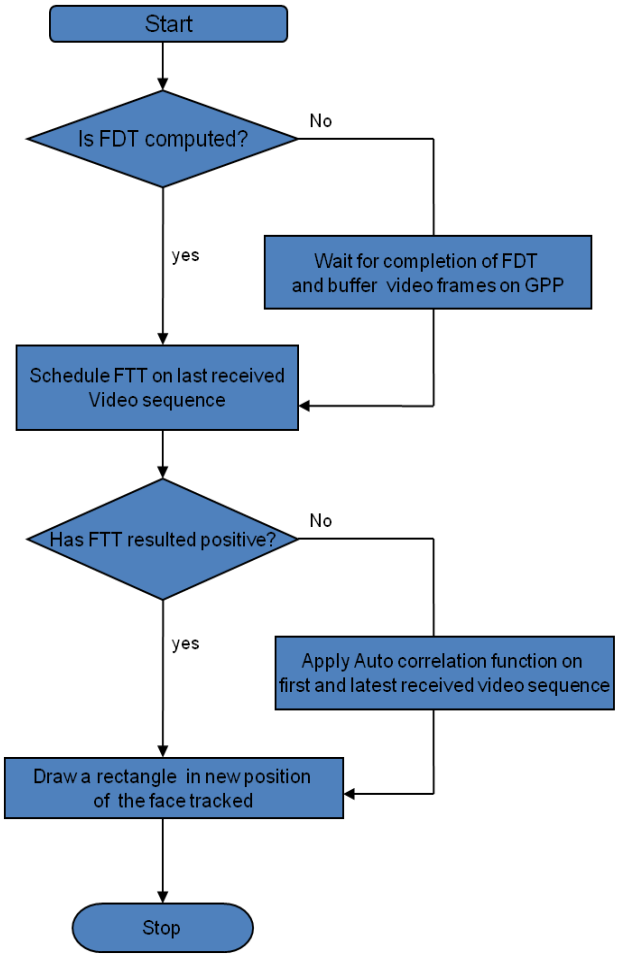


Fig 3: Face Algorithm Agent Execution Mechanism

3.3 Message Sequence Chart

The diagram shown below is the message sequence chart of the execution of the architecture.

Step 1: User selects the Face Tracking option from the UI.

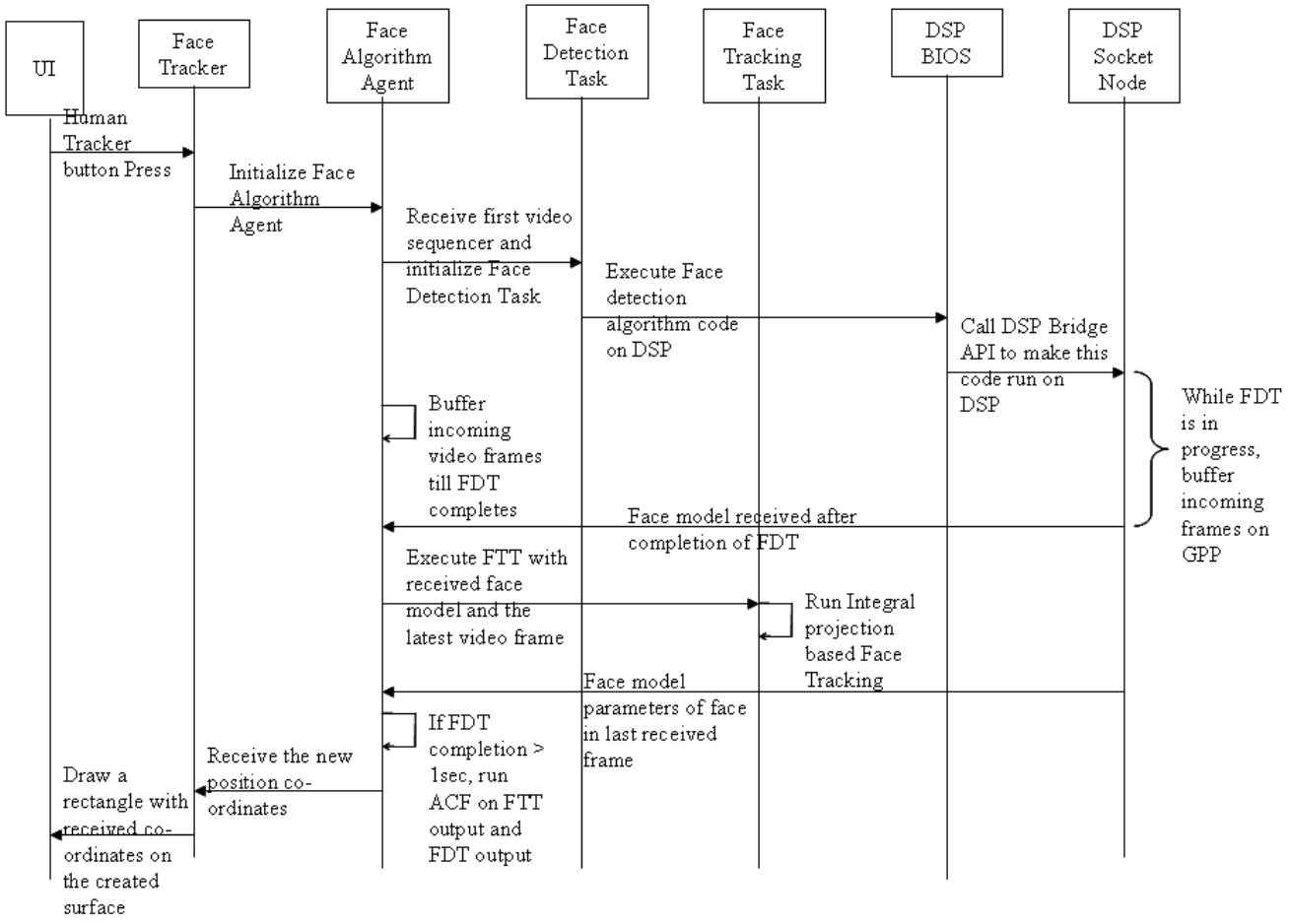
Step 2: The Face Tracker application component gets invoked. This component in-turn creates the services component of android to interact with middleware.

Step 3: The services component creates the Face Algorithm Agent component which is responsible to invoke FDT and FTT.

Step 4: When first video frame is received, the FAA creates a new task (FDT). This FDT operates on the received video frame to compute the co-ordinates of the human face.

Step 5: While FDT is in-progress, create the training data set by accumulating the incoming video frames on the GPP side.

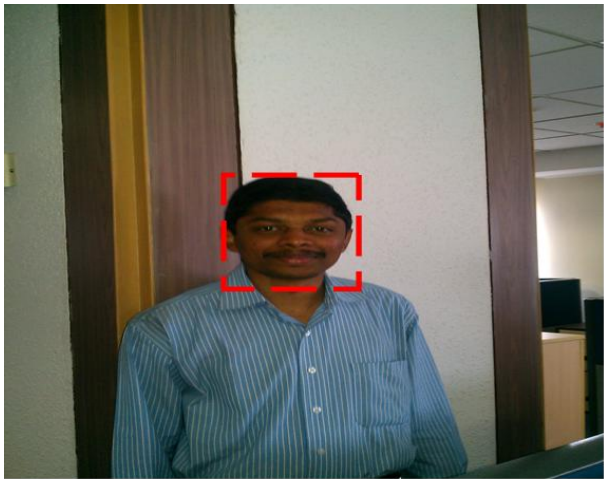
Step 6: On completion, User selects the Face Tracking option from the UI.



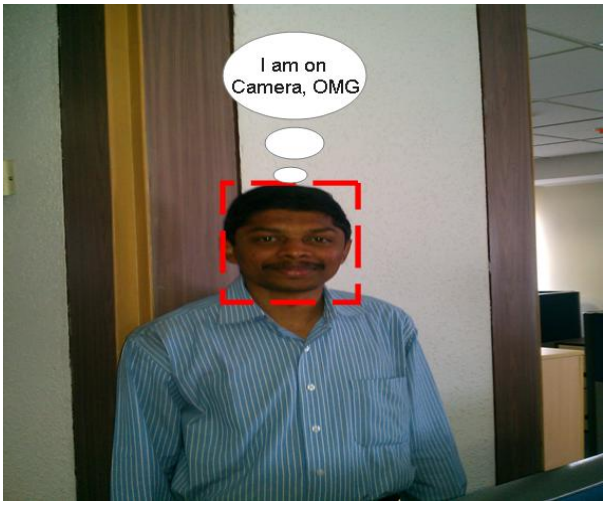
4. EXPERIMENTATION

To evaluate our approach and to demonstrate its strength, different processor load factors and varying frontal faces conditions were simulated. In this section, we present results of experiments to characterize the adaptation of our approach.

The architecture also supports tailor-able adaptation for multiple multimedia framework dimensions. The experimentation results show that our approach is effective at reducing the false positive rates significantly.



Face Detected, Display Caption and Track



New Face Detected, Add caption and Track

Both the Faces Detected, Tracked Accurately

5. CONCLUSION AND FUTURE WORKS

In this paper, we have presented a novel multithreaded architecture to efficiently cater Face Tracking Algorithm to android based devices. Though integral projection based face tracking algorithm has vast potential to accurately track human faces, the high computational complexities of face detection algorithm and existing constraints has led to high false-positive detection rate, especially on resource constrained handheld devices. Consequently, we detail the proposed architecture which envisions reducing false-positive detection rate. A proof-of-concept extending android platform was illustrated to evaluate the feasibility of this architecture to translate into product deployment. Our experiments have proved its feasibility in achieving low false positive detection rate. A related aim is to inform the scientific community about design in such new environments by providing MSC (Message State Chart) and Flow diagrams.

Future work includes using new models instead of ACF to increase accurate face tracking rates.

6. REFERENCES

[1] Refining Face Tracking with Integral Projections Gin´es Garc´ıa Mateos Dept. de Inform´atica y Sistemas, Universidad de Murcia 30.170 Espinardo, Murcia (Spain).

[2] Bradski, G. R.: Computer Vision Face Tracking For Use in a Perceptual User Interface. Intel Technology Journal Q2'98 (1998)

[3] Spors, S., Rabenstein, R.: A Real-Time Face Tracker for Color Video. IEEE Intl. Conference on Acoustics, Speech, and Signal Processing, Utah, USA (2001)

[4] Kaucic, R., Blake, A.: Accurate, Real-Time, Unadorned Lip Tracking. Proc. of 6th Intl. Conference on Computer Vision (1998) 370–375

[5] Sobottka, K., Pitas, I.: Segmentation and Tracking of Faces in Color Images. Proc. of 2nd Intl. Conf. on Aut. Face and Gesture Recognition (1996) 236–241

[6] Stiefelhagen, R., Yang, J.,Waibel, A.: A Model-Based Gaze Tracking System. Proc. of IEEE Intl. Symposia on Intelligence and Systems (1996) 304–310

[7] Pahor, V., Carrato, S.: A Fuzzy Approach to Mouth Corner Detection. Proc. Of ICIP-99, Kobe, Japan (1999) I-667–I-671

[8] Schwerdt, K., Crowley, J.L.: Robust Face Tracking Using Color. Proc. of 4th Intl. Conf. on Aut. Face and Gesture Recognition, Grenoble, France (2000) 90–95

[9] Garc´ıa-Mateos, G., Ruiz, A., L´opez-de-Teruel, P.E.: Face Detection Using Integral Projection Models. Proc. of IAPR Intl. Workshops S+SSPR'2002, Windsor,Canada (2002) 644–653

[10] Isard, M., Blake, A.: Contour Tracking by Stochastic Propagation of Conditional Density. Proc. 4th Eur. Conf. on Computer Vision, Cambridge, UK (1996) 343–356

[11] Vieren, C., Cabestaing, F., Postaire, J.: Catching Moving Objects with Snakes for Motion Tracking. Pattern Recognition Letters, 16 (1995) 679–685

[12] Pentland, A., Moghaddam, B., Starner, T.: View-Based and Modular Eigenspaces for Face Recognition. Proc. CVPR'94, Seattle, Washington, USA (1994) 84–91

[13] La Cascia, M., Sclaroff, S., Athitsos, V.: Fast, Reliable Head Tracking Under Varying Illumination: An

Approach Based on Registration of Texture-mapped 3D Models. IEEE PAMI, 22(4), (2000) 322–336

[14] Intel Corporation. IPL and OpenCV: Intel Open Source Computer Vision Library. http://www.intel.com/research/ mrl/research/opencv/

[15] Kaijian Xu, Manli Zhu, Daqing Zhang, Tao Gu: "Context-Aware Content Filtering & Presentation for Pervasive & Mobile Information Systems"; Proceedings of the 1st international conference on Ambient media and systems, Quebec, Canada, 2008.