Addressing the Semantic Gap Between Video Sensors and Applications

Wu-chi Feng, Khanh Nguyen, Feng Liu
Portland State University
Department of Computer Science
P.O. Box 751
Portland, OR 97207
{wuchi, fliu}@cs.pdx.edu

Thanh Dang
Washington State University - Vancouver
School of Engineering and Computer Science
Vancouver, WA 98686
thanh.dang@wsu.edu

ABSTRACT
In this paper, we propose a framework to support the bridging of applications and computer-vision based sensor networks. We argue that the semantic gap, the difference between the data collected in a sensor network and the information needed by the application, in video-based sensor networks can only be addressed by providing systems support in such a way that allows users and computing systems to meet in the middle. We first outline the vision of the system that we are working towards. We then describe initial experiments that we have conducted using a functional component of the system applied to real-world video data that is being collected by intelligent transportation systems researchers.

Categories and Subject Descriptors
J.7 [**Computer Applications**]: Computers Systems

General Terms
Algorithms, Design, Human Factors.

Keywords
Video Sensors, Data Fusion, User Interfaces.

1. INTRODUCTION
Sensor networking technologies are becoming advanced enough to support many real applications that can benefit society. Examples of such applications include irrigation monitoring and structural monitoring (bridges / buildings). While the number of sensor networking applications continues to grow, many applications exist that have a significant gap between the *data* a sensor network or set of sensor networks can deliver and the *information* that is ultimately useful to the application. We call this gap the *semantic gap.*

The semantic gap in such systems varies heavily depending upon the complexity of the application. For example, simpler applications like counting cars on a highway on-ramp have little or no semantic gap because the in-road loop detectors can provide the necessary information. That is, the necessary information is directly derivable from the loop detectors. Sensor networks with video components tend to exhibit larger semantic gaps. An example of such an application is creating actions from video for health care or video surveillance where the data needs to be converted / interpreted to find the information needed.

Figure 1: Crosswalk video example

Consider the crosswalk shown in Figure 1. The crosswalk is located in the City of Portland and has large pedestrian flows at certain times of the day. The crosswalk is at an intersection without a signal light for either vehicles or pedestrians. By law, pedestrians have the right of way and priority at intersections; hence, vehicles are required to yield to pedestrians. Intelligent transportation systems (ITS) researchers are interested in studying pedestrian / vehicle conflicts where vehicles either fail to yield to pedestrians or stop too close to the pedestrian. This, in turn, will allow them to try different mitigation strategies such as additional upstream traffic signaling or adaptive lighting in order to improve overall safety and efficiency.

The semantic gap here exists between the video images (data) and the information they require. Examples of information they would like to derive include:

- What is the frequency of pedestrian / vehicle conflict?
- Which pedestrians are runners, handicapped, or elderly? What is their frequency?
- What percentage of the pedestrians on the far sidewalk cross the crosswalk?

NOSSDAV '13, February 26 – March 1, 2013, Oslo, Norway.

The underlying images from sensor networks, however, are a sea of pixels. While computer-vision algorithms exist that can perform rudimentary functions like find all humans in a video or detect all cars in a video, separating out actions is a much more difficult task. That is, assigning semantics to these actions is where the hard work really is. For this example, the computer vision algorithms would need to distinguish between pedestrians in the crosswalk and other pedestrians on the sidewalks.

In this paper, we propose a visually interactive interface for scalar, audio, and video-based sensor networks, where the human and computing system meet somewhere in the middle of the semantic gap. The basic idea is to separate out the application-specific semantics from the underlying computer vision algorithms. Thus, users provide some of the "meaning" of data within a video, allowing the underlying computer vision algorithms to be greatly simplified as well as targeted in their execution. We show the design principles of our technique and provide the beginning experimentation for an example application of counting pedestrians crossing a street. We believe that through simple interactions (i.e., humans providing basic semantic information), computer-vision based sensor networks can be extended to provide much more useful information to the user in a more scalable way.

2. A FRAMEWORK TO SUPPORT BRIDGING SEMANTIC GAPS

Our approach is based upon two observations. First, *creating a generic computer-vision based system that can be used for a large number of different deployments is extremely difficult.* In particular, the amount of complexity and code necessary to do relatively simple tasks like locating a sidewalk in an image, or determining scale can become arbitrarily complex. Second, *in many cases the video sensor is relatively static and humans could easily provide such information if an appropriate interface was available to quickly give the system this information.* In such cases, with simple input from the user, we believe the sensor network can deliver more useful data to the user. In short, we believe that we need to bring the users into the computation through a concept we term *semantic bridges*.

The goals of our framework include:

- Providing a *simple* interface to allow users to instruct computer vision algorithms on what to do. In effect, we want to remove application-specific programming from the system, but, at the same time, provide users a way to provide input into computer vision algorithms.
- Providing a *visual* programming interface to allow users to construct multi-modal sensor systems. The idea is to provide abstract components with inputs and outputs that allow users to connect system components in a data-centric way. This is similar to visual programming paradigms found in systems like Lego Mindstorms [11].
- Providing a *refinable* system where users can iteratively refine the semantic information on collected data. This would enable users have more interactive discovery.
- Providing a *replayable* system that allows users to retry different inputs in the interface. Furthermore, having replay would also help systems programmers and computer vision programmers refine their algorithms.

We believe that the system would then provide the benefits of being (i) much more tailored to the application, (ii) faster to implement, (iii) more generic and reusable, and (iv) easier to re-deploy (e.g., moving implemented system to a new crosswalk).

2.1 Semantic Bridge Components

We envision a number of components in our system including data sources, processing functions, and operators. Recall, we distinguish between *data*, which come from the sensor network, and *information,* which is the higher-level knowledge the user is attempting to extract.

Sources: Sources generate data that will be used within the system. For scalar sensor networks, we assume that the entire sensor network is abstracted or that groups of scalar sensors are abstracted into a single component in the system. Because the data from scalar sensors is first order, abstracting it at a higher-level will make it less complicated for the user. For audio and video sensors, a stream of time-stamped data (frames or sets of audio samples) will be created. Finally, we also assume sources can be from live or stored sources (for refining the system with the same data). The latter introduces some issues that will be discussed in the Section 5.

Processing Components: To make the construction of applications easier on the user, we expect that there will be a number of processing components, each with very simple interactions but provide application-centric processing of data. The guiding principle of each processing component is to keep as much of the application layer knowledge out of the processing component. For data generated from a video sensor, we envision a number of simple components the user can select from. There will eventually be more, but we want to give the flavor of the level for which processing will be done.

- Removing video with no movement: removes video where no movement is detected.
- Restrict processing to a sub-region: allows users to restrict regions where the computer vision algorithms run. For example, for crosswalk monitoring, this will free the computer vision algorithm from having to distinguish between humans on a nearby sidewalk and the crosswalk without having to implement such knowledge in the program itself.
- Line triggered processing: allows a user to overlay lines onto a preview telling the computer vision algorithm to either pass on the video or count the number of objects that go between the two lines.

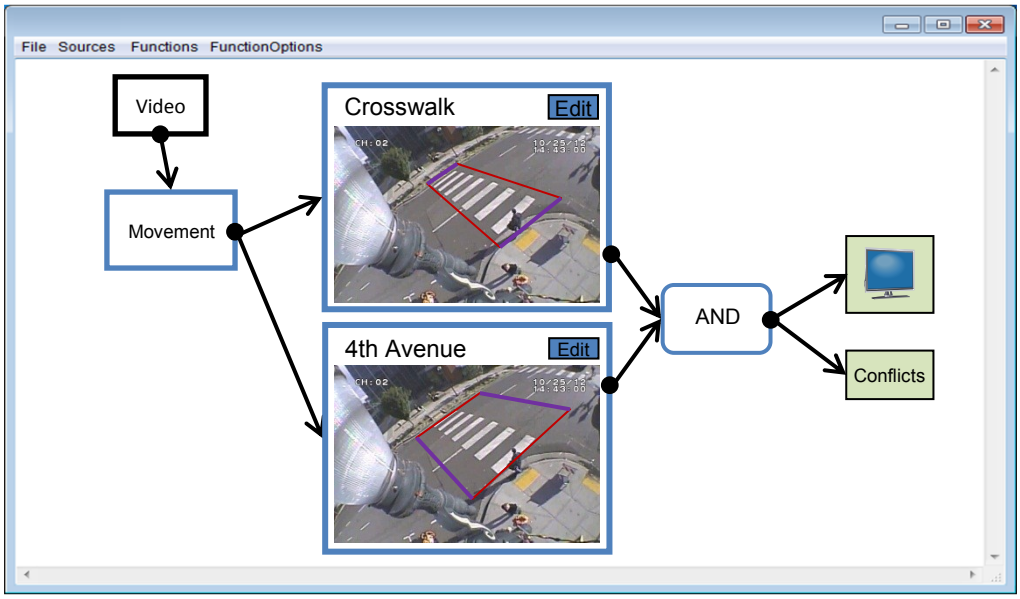

Figure 2: Mock-up of Envisioned Semantic Bridge Application Example.

- Object size selection: allows a user to tell the computer vision algorithm what the object "size" of interest is. Thus, for the crosswalk monitoring example, humans and cars can be distinguished by their size without having to establish scale within the video, assuming the camera placement does not have significant amounts of perspective skew.

As new components are developed, we believe it is important to have each as simple and generic as possible.

Operators: Because the components are relatively simple in design, there will be a need to provide basic operators. We envision that the system will have a component that allows basic operators such as AND, OR, NOT. These functions will then allow the user to build more application-centric, higher-layer tailoring with additional generic building blocks.

We note here that feedback loops are not currently envisioned. Each component represents an *if* statement because it only passes on video that has met the criteria of the user (i.e., if object in video meets criteria, pass it on). Because the processing is expected to distill the data into information that is more useful for the application rather than controlling the video sensor network, feedback upstream is most likely not needed.

2.2 An Example Application

In Figure 2, we have shown a mocked up example of the proposed system. In this figure, there is one video source (black box on the left), three processing nodes (Movement, Crosswalk, and 4th Avenue), one operator (AND), and two outputs (computer display and conflicts). The four blue outlined boxes "implement" the application and are briefly described here:

- *Movement* removes video with no motion detected. This marks frames that do not need to be processed by downstream nodes to limit the amount of heavy-weight computer vision algorithms that need to be run.
- Crosswalk – detects the presence of pedestrians in the crosswalk. This is accomplished by editing the node, which lets the user annotate a sample image used for the computer vision algorithm. Here, the user can specify the crosswalk, or area of interest, shown with a red box. Further, the user instructs the system that it is only interested in objects that cross the purple lines at the end of the crosswalks; this has the effect of removing cars. Finally, the user can input a size of object, if needed.
- 4th Avenue – detects cars. Similar to the crosswalk, the user annotates a red box to indicate area of interest with purple lines to denote the interest in only objects that are along the flow of traffic. Thus, the net effect is that pedestrians are not processed in this node.
- AND – implements the "vehicle / pedestrian conflict" algorithm by intersecting the detected cars and the detected pedestrians. The output of this gets replicated to the computer display for the user and saved to a file for future reference.

As demonstrated by this example, bringing the user into the sensor network makes the system more accurate for the user as well as easier to implement. Specifically, there are a number of advantages to the proposed approach. First, application semantics are separated out from the underlying computer vision algorithms. In this example, the cars and pedestrians are handled separately and each box represents the semantics of cars and pedestrians. Second, situational awareness (e.g., where the crosswalk is) no longer need to be implemented, removing the code required to generically

find crosswalks from the code. Yet, because of the visual interface, a user can quickly define the situation parameters necessary for the computer vision algorithms to work. Third, algorithms to determine scale, albeit with appropriately placed cameras, can be removed assuming the user has taken some care to have the scale of objects relatively constant. Here, the user can specify a box that is the relative size of the object of interest (i.e., the pedestrian). Finally, once defined for a particular scenario, the system can be redeployed to another with the user having to re-specify situational parameters but not the functional interaction between components.

In the remainder of this paper, we will describe our experiences with building the video processing component.

2.3 Video Processing Component v.1.0

We have begun the implementation of the video processing component that has sub-region selection and start / stop lines. The user is able to enter these via the visual interface and is then used to guide the computer vision algorithm.

The underlying computer vision system we have implemented consists of three main steps: moving pixel detection, moving object detection and object tracking. We first use the background subtraction method described in [1] to detect moving objects. Specifically, we use the Mixture of Gaussians method to model the static background and then take the difference between the background and each video frame to detect moving pixel maps. Rather than re-implementing the algorithm, we used the library written by Laurence Bender that is available via sourceforge [8]. The Mixture of Gaussians approach has a number of advantages. First, it quickly adapts to the background so it can be done after turning the camera on and not doing anything special with respect to objects in the image. Second, it works well for outdoor environments where the light can be constantly changing due to daylight or weather. Finally, of some of the systems we tried, it is relatively fast. We have found through experimentation that it typically takes 100 frames to initialize the background model. The main disadvantage we have found with it (which will also apply to most of other algorithms) is that if an object is stationary during learning it becomes part of the background although not intended. This comes up, for example, if the learning algorithm is started while a car is at a red light. Once we have the moving pixel maps, we group the moving pixels into objects. We then track each object through video frames using a Kalman filter implemented in OpenCV and resolve the occlusion problem using a simplified version of the solution described in [2]. The tracked objects that are returned can then be displayed to the user.

3. Experimentation

For experimentation, we have obtained several hours of video footage from the Intelligent Transportation Systems (ITS) Lab at Portland State University. One of the interests of the lab is in vehicular, bicyclist, and pedestrian safety. One particular problem of interest is pedestrian / vehicle conflicts, which are defined as cars not yielding to pedestrians as is required by Law. They have deployed a camera system outside the Engineering Building to collect data. For their research, it is fairly typical to have paid undergraduate students either sit at an intersection or sit in front of a computer monitor and manually count and annotate such events. An example image from the deployed camera is shown in Figure 1. We note here that this is a fairly typical deployment used by the ITS researchers. The researchers also use similar images for bicycle adherence studies; the City of Portland has special bicycle markings with green boxes for cyclist safety and the researchers are interested in evaluating the effectiveness of different solutions.

Figure 3: User annotations for the pedestrian crossing experiments.

For the purposes of our studies, we have selected two sequences, each 10 minutes in length. Both sequences are from the same camera but under different lighting conditions. The easier data set, which we call *overcast,* was taken on an overcast day (shown in Figure 3). The second data set, which we call *sunny*, was taken when the sun was out (shown in Figure 1). The big difference between these two sample videos is that *overcast* has very little shadows on the cars and people while *sunny* has shadows that are both large and very well-defined. As we will show, this has significant impact on the algorithms' performance.

3.1 Testing Pedestrian Crossing

To test the efficacy of the video component, we set out to test the counting of pedestrian crossings from the video set. We first set the area of interest in the video and then demarcated the two ends. Any object that was in the crosswalk and tracked across between the start and finish lines caused a count event to be created. The annotations from the user drawn for the camera are shown in Figure 3. The blue line indicates the sub-region selection around the crosswalk, while the green and purple lines at the ends of the crosswalk signify the end lines.

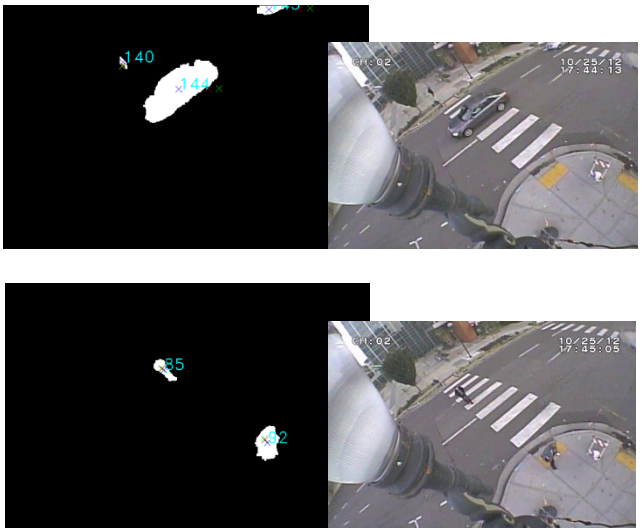

Figure 4: Object tracking examples

	Manually Counted	Algo Counted
Overcast	12	11
Sunny	15	39

Table 1: Pedestrian crossing results

Figure 4 shows two examples of objects that were tracked during the running of the algorithm. The left images show the object number that the system is currently tracking, while the right images show the video frame corresponding to the left images.

For the sequences, the number of pedestrians counted by hand and by the video component is shown in Table 1. As shown in the table, the object tracking on the *overcast* data set worked fairly well while the results for the *sunny* data set were not that great. The errors from the *overcast* data set were from two people being counted as one (-1), a car's object mask covering a pedestrians (-1), and a person's body mask being separated into two (+1). We should note that the object mask from the car covering the pedestrian (i.e., a failed yield) is of great interest to the ITS researchers. For the *sunny* data set, the computer vision algorithm had difficulty in two main areas: occlusions and shadows. For the occlusions, because an object and its corresponding shadow were larger, keeping track when occlusions occurred in their masks caused issues. For the shadows, the computer vision algorithms had trouble managing its mask. Sometimes, pedestrians would be counted as two objects (person and shadow) instead of a single object. This just demonstrates some of the difficulties that the computer vision community faces. *The most important part, however, is that no cars were counted as pedestrians*, even under difficult lighting.

We expect that for situations where the lighting and movement is a little bit more controlled that the system will be more useful. Nevertheless, we found that the implemented system can help bring semantics into the processing; in this case, being able to separate cars and pedestrians. Thus, using fairly generic computer vision algorithms and a simple interface, we can provide the semantics that are necessary for the application.

4. Related Work

We very briefly touch on some of the related work to ours.

A number of abstractions have been designed and implemented for sensor networks with abstractions occurring at differing levels. For applications, construction kits have been proposed in SenseTK [14] and SNACK [7] for video and scalar sensor networks, respectively. In these systems, interaction is managed through programmatic interfaces; thus, users are still required to have significant programming skills to construct applications. In addition, these kits focus more on reusability where prewritten software components can be reused to construct applications. There is no support for representing as well as expressing semantics in sensor data.

Many software development environments provide visual interactive interfaces for building applications or controlling devices. Lego Mindstorms, for example, has basic blocks that users can select and connect them to build an application [11]. Support for semantic expression, however, is very limited. Similar to Lego Mindstorms, systems like Supervisory Control and Data Acquisition (SCADA) also provide block-based application construction [5]. They focus more on expressing logic and control. Our work somewhat intersects with work in this area. We, however, address unique challenges in bridging semantics in multi-modal sensor data itself.

Finally, several programming abstractions have been proposed for sensor networks such as [4][12][15][16]. These abstractions hide the details of networking and storage interactions and expose users to only necessary interfaces for data collection. Among the first was Directed Diffusion which provides a framework for distributed event detection and propagation [9]. In TinyDB, a sensor network is viewed as a database. Users query data declaratively by providing SQL-like queries through an interface without worrying about low-level detail operations of individual nodes [12]. Hood [16] and Logical Neighborhoods [13] provide a notion of neighborhood as programming primitives. A neighborhood is identified using constructs based on application criteria and to share state with them. While these approaches are successful in providing communication and programming abstractions, they do not provide support for non-expert users to construct applications for sensor networks.

5. Discussion

There are a number of issues that arise when using our framework that we briefly describe here.

Limitations of scale: The user needs to be aware of perspective scale if the user is going to specify an object

size of interest. In the crosswalk example, given the size differential between cars and walker is quite large, using size as a distinguishing trait is not affected as much by perspective scale.

Bringing real-time into computer vision: The framework is currently data driven (i.e., not real-time). As such, it cannot be used in a real-time processing application. To move towards real-time processing, we need to understand the trade-off in accuracy versus processing time. For example, one could use a lower-resolution image but would then have less accurate results. Similarly, one could use a lower frame rate but it would heavily impact tracking.

Synchronization: In conjunction with real-time, we currently have not designed explicit synchronization into the system. Applications that could benefit are ones with triggers; for example, using audio to listen for the presence of an event to trigger the video.

Parallel processing: Given that each processing box has a relatively small amount of work, a system with multiple components will end up repeating some work (i.e., tracking). It may be possible that our paradigm might be amenable to mapping onto multi-core systems.

Control structures: The system also does not support "control" structures like if /then statements, for loops, or while loops. Inclusion of such structures would be possible at the expense of simplicity for the user. Because of potential feedback loops, users would then have to manage time and what feedback means within the application.

6. Conclusion and Future Work

We have described a framework for helping to enable building more complex computer-vision based sensor systems that can allow users to introduce semantic based processing into the system. Through visual programming, our framework can alleviate the requirement of the user being a computer programmer. We have implemented and tested a single video component of our system in a pedestrian crossing scenario. While some of the limitations of computer vision algorithms manifest themselves, we believe that using a system like ours can make it more scalable and less complex from a programming standpoint. We are currently working toward implementing the interconnections for the components in the framework and understanding the impact of real-time computer vision algorithms on accuracy.

7. REFERENCES

[1] T. Bouwmans, F. El Baf, B. Vachon, "Background Modeling using Mixture of Gaussians for Foreground Detection - A Survey", *Recent Patents on Computer Science*, Volume 1, No 3, pages 219-237, Nov. 2008.

[2] D. Conte, P. Foggia, G. Percannella, M. Vento, "Performance Evaluation of a People Tracking System on PETS2009 Database", in *Proceedings of the 2010 Seventh IEEE International Conference on Advanced Video and Signal Based Surveillance*, Fisciano, Italy, pp 119-126, August, 2010. DOI=http://doi.acm.org/ 10.1109/AVSS.2010.87

[3] C. Cotton, D. Ellis, A. Loui, "Soundtrack Classification by Transient Events", *ICASSP*, pp 473-476, 2011 http://dx.doi.org/10.1109/ICASSP.2011.5946443

[4] T. Dang, W. Feng, N. Bulusu, H. Tran, "Demo: Zoom: a Multi-resolution Tasking Framework for Crowd-sourced Geo-spatial Sensing", in *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems* (SenSys '11). ACM, New York, NY, USA, 391-392.

[5] S. Dawson-Haggerty, X. Jiang, G. Tolle, J. Ortiz, D. Culler, "sMAP: a Simple Measurement and Actuation Profile for Physical Information", in *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems* (SenSys '10). ACM, New York, NY, USA, 197-210.

[6] J. Fernandez, A. Fernandez, "SCADA Systems: Vulnerabilities and Remediation", *J. Comput. Small Coll.* 20, 4 (April 2005), 160-168.

[7] B. Greenstein, E. Kohler, D. Estrin, "A Sensor Network Application Construction Kit (SNACK)", in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems* (SenSys '04). ACM, New York, NY, 69-80.

[8] http://scene.sourceforge.net/models.html

[9] C. Intanagonwiwat, R. Govindan, D. Estrin, "Directed Diffusion: a Scalable and Robust Communication Paradigm for Sensor Networks", in *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking* (MobiCom '00). ACM, New York, NY, 56-67.

[10] K. Lee, D. Ellis, "Audio-based Semantic Concept Classification for Consumer Video. *Trans. Audio, Speech and Lang. Proc.* 18, 6 (August 2010), 1406-1416. DOI=10.1109/TASL.2009.2034776

[11] Lego MINDSTORMS, http://mindstorms.lego.com/en-us/Software/Default.aspx

[12] S. Madden, M. Franklin, J. Hellerstein, W. Hong, "TinyDB: an Acquisitional Query Processing System for Sensor Networks", *ACM Trans. Database Syst.* 30, 1 (March 2005), 122-173. DOI=10.1145/1061318.1061322 http://doi.acm.org/10.1145/1061318.1061322

[13] L. Mottola, G. Picco, "Programming Wireless Sensor Networks with Logical Neighborhoods", in *Proceedings of the First International Conference on Integrated Internet Ad hoc and Sensor Networks* (InterSense '06), New York, NY, USA, Article 8.

[14] P. Sitbon, W. Feng, N. Bulusu, T. Dang, "SenseTK: a Multimodal, Multimedia Sensor Networking Toolkit", in *Proceedings of Multimedia and Computing Networking 2007*, San Jose CA, January 2007.

[15] M. Welsh, G. Mainland, "Programming Sensor Networks using Abstract Regions", in *Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation - Volume 1 (NSDI'04)*, Vol. 1. USENIX Association, Berkeley, CA, USA.

[16] K. Whitehouse, C. Sharp, E. Brewer, D. Culler, "Hood: a Neighborhood Abstraction for Sensor Networks", in *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services* (MobiSys '04). ACM, New York, NY, USA, 99-110. DOI=10.1145/990064.990079 http://doi.acm.org/10.1145/990064.990079

Columns on Last Page Should Be Made As Close As Possible to Equal Length