Processing Multiple Image Streams for Real-Time Monitoring of Parking Lots

Yu-Hsn Liu[1*], Kok-Leong Ong[2], Vincent C.S. Lee[3], Yi-Ping Phoebe Chen[1]

ABSTRACT

We present a system to detect parked vehicles in a typical parking complex using multiple streams of images captured through IP connected devices. Compared to traditional object detection techniques and machine learning methods, our approach is significantly faster in detection speed in the presence of multiple image streams. It is also capable of comparable accuracy when put to test against existing methods. And this is achieved without the need to train the system that machine learning methods require. Our approach uses a combination of psychological insights obtained from human detection and an algorithm replicating the outcomes of a SVM learner but without the noise that compromises accuracy in the normal learning process. Performance enhancements are made on the algorithm so that it operates well in the context of multiple image streams. The result is faster detection with comparable accuracy. Our experiments on images captured from a local test site shows very promising results for an implementation that is not only effective and low cost but also opens doors to new parking applications when combined with other technologies.

I. INTRODUCTION

The motivation behind the work in this paper is the desire for a parking system that aims to reduce frustration for drivers in their attempt to hunt for a free parking lot. Especially under heavily utilised conditions, navigating a parking site and competing with other drivers for a free spot is often a time consuming and frustrating task. Current advanced parking systems at various sites in Australia implements a "sensor-to-lot" approach with signages near the site to assist drivers. Although such an implementation provided assistance to drivers, there are many drawbacks.

By using a sensor for each parking lot, a large parking site becomes costly to implement when the costs of fitting sensors and wiring them to the signage is considered. Consequently, the implementation is kept simple to contain the costs. As a result, the implementation failed to take advantage of the collective information provided by the sensors. In situations where the site is heavily utilised, drivers quickly face frustrations because (i) signage information become inaccurate; (ii) sensor lights (that indicated free lots) become difficult to spot; and (iii) the effectiveness of light indicators are limited to a small range due to the "line of sight" approach. This a "local optimal" solution since drivers in a busy parking site can only depend on available information in the vicinity rather than the collective information provided by the sensors.

In search for a better car park system than the commonly used "sensor-to-lot" approach, we discovered that many research do not address the problem of informing drivers about free parking lots, and using that information effectively to reduce the frustration of drivers. A different solution is thus called for that started this investigation. As smart phones connected to the Internet via 3G networks become ubiquitous, we foresee that they may present the answer to ease, if not, end a driver's car park hunting nightmare.

Our premise is that if drivers are informed in advanced about the situation of a parking site, it will enable decisions to be made to avoid the frustrations of not been able to secure a free parking lot. Extending this idea, it would become possible to use the technology in ways such as enabling guidance to parking lots on a large parking site, directing drivers to alternate parking sites under busy situations, and so on.

For such a system, the sensor in each parking lot needs to be wired to a server so that they can be mashed up with information on the Web to create the applications we envisioned. Doing so however will significantly increase infrastructure costs. The solution is to replace multiple sensors with a single IP-enabled camera. By reducing the number of input points, we lower costs but now require a method to detect the presence of a parked car. Our proposal is novel in terms of marrying image processing technologies and machine learning concepts to deliver a cost effective and accurate solution. Such a solution will have a good degree of accuracy and in the presence of multiple image streams, requires fast processing capacity on a cost-effective computer.

We begin with a discussion of related works in Section II before moving on to the discussion of our solution in Section III along with experimental results in Section IV. From this base algorithm, we improve performance of the algorithm in the context of processing multiple image streams on a single compute device. The performance enhancements were described in Section V. Lastly, conclusions of our work is found in Section VI.

Paper submitted 08/15/11; revised 11/15/11; accepted 12/30/11.
** Corresponding Author: Yu-Hsn Liu (E-mail: yliu65@students.latrobe.edu.au)*
[1] Department of Computer Science and Computer Engineering LaTrobe University, Bundoora, Victoria 3086
[2]School of Information Technology Deakin University, Burwood, Victoria, Australia
[3]Faculty of Information Technology, Monash University Victoria, Australia

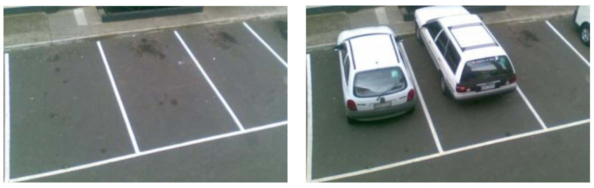


Fig. 1. (a) A reference image where the parking lot is empty. The same set of filter is applied to the reference image as well as the incoming image stream represented by (b). For images streams where the vehicle colour is light (as in (b)), it is rather easy to obtain a high detection accuracy.

II. RELATED WORKS

Our work comprises of two areas: (i) the design of smart car parks and (ii) the detection of free parking lots in images. On the design of car park systems, which is not the main discussion point of this paper but relevant, our survey revealed a focus on a number of key areas. Many address problems in aspects of parking such as smart payment systems [1]-[3] transit-based information [4], [5], automated parking [1], [6]. In these areas, the problems and objectives addressed are different from our motivation.

Two areas of car park design research are however of interest to us. The first is e-Parking systems such as those in [7], [8]. Most e-Parking systems are Web-based where drivers look for available parking spots around points of interest. A driver can also book a parking space in advanced using either SMS or via the Web. In a way, our proposal to use modern smart phone is the evolution from the works reported in this area. While this paper attempts to report the detection mechanism, its design is motivated by the desire to eventually create such a system where smart phone can be used to book car park spaces, get car park information as well as other applications that use the collective information generated from the cameras. Another difference from e-Parking systems in our case is the shift from the focus of Web-based systems to the focus on applications that run on the smart phone utilising the information made available on a Web server.

The other area of car park design research is Parking Guidance and Information Systems (PGIS) represented by the works of [9]-[12]. The key premise of PGIS is the ability to provide guidance to drivers in finding a park. While similar in motivation, our work differs from such systems in terms of the proposed implementation. Generally, PGIS is deployed using signages around the car park facility and in limited cases, encompassing the entire city area. These signages emit collated information from vehicle detection sensors to provide drivers information about free parking lots. In that sense, parking information is not personalised and is only available when the driver is near the physical facility. With smartphones and 3G networks, we envision a system that will publish car park information down to its available lot details providing individual smartphones the capability to filter information to the needs of the individual driver. Additionally, such a model opens up various possibilities for developers to create novel applications arising out of the published information. Yet to achieve all the above, a detection system that can work with such new technologies are required. Hence, the motivation of the work reported in this paper.

On the issue of detection, one approach is the use of machine learning techniques, where labeled images are used to train a classifier that will be deployed to detect the presence of a car. Here, different classifier technologies, methods of training and the structure of classifiers were explored. For example, [13] proposed a 8-class Support Vector Machines (SVM) classifier with probabilistic outputs while in [14], a simple (linear hyperplane) classifier was used to achieve above 90% accuracy by optimising the information of individual features in an image. Others used image processing techniques to achieve similar results. Interpretation of image sequences using visual surveillance techniques was proposed by [15] while [16] and [17] tracks movement of vehicles as the basis for determining free parking lots.

For [14], the major drawback is in the scalability of its solution. For an 8-class SVM classifier, the system is only capable of dealing with 3 parking lots in a single image. If a single camera captures 4 parking lots, a 16-class SVM classifier is needed. As the effort and computation requirements double with every additional parking lot, the solution's practical significance is limited. In the area of image processing techniques such as [15]-[17], the detection mechanism requires incurs either high computational costs or
large memory space. In comparison, our approach is far more scalable than the proposal in [14] and requires less computing resources than those proposed in [15]-[17].

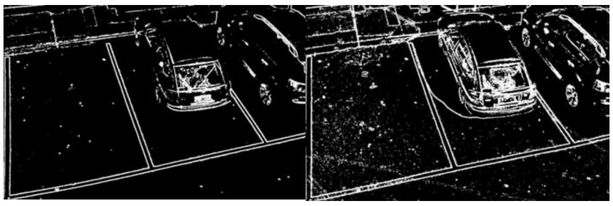


Fig. 2. The only difference in the two images is the position of the sun. As a result, the intensity level affected the filters' output as seen in (a) and (b) where edge detection and binarisation filters are applied using default parameter values. Clearly this impacted detection accuracy, especially in the case of the first parking lot. In our algorithm, we used a simple statistical method to adjust the filter's parameters on-the-fly so that it can be compared to the reference image accurately.

III. OUR APPROACH

In the context of image processing, which this system now depends on, the problem is a classic case of object detection [18], [19]. The challenges of object detection are the high variability in appearances of objects in a given class (in our case, parked cars) and the added variability between instances of the same object due to alternate viewing angles and/or conditions (e.g., the same car viewed from the front, side, or back). As images are taken as specific time intervals from different cameras, we now have multiple image streams to be processed. The problem thus calls for real-time efficiency and accuracy.

To achieve this, we first address the question of accurate detection. As we learnt in Section II, a popular approach is to use a classifier to distinguish between an image belonging to a target class (i.e., car present in lot) and one that doesn't (i.e., car *not* present in lot). Usually a set of vectors, with each representing an image, is used in the training of classifiers to find discriminating features that separates two or more classes. In the case of the SVM [20], well-known for its binary classification accuracy under small data samples and high dimensionality, the set of discriminating features are identified by the hyperplane.

Simply put, the concept of a hyperplane is a cut that best separates the feature spaces into two distinct classes. In SVM, this cut is determined via the learning process using a series of vectors and its associated class label. The issue with this process is the dependency on the learning algorithm, which itself is dependent on the data, to find the best cut that defines the hyperplane. To increase accuracy, much of the work focuses on the training data either by stripping the vector down to key feature spaces and/or increasing learning instances. In any case, the idea is to reduce the noise in the training instances to allow a "cleaner cut" and hence, a better classification accuracy. We were however inspired by a different approach.

1. Recreating the Ideal "Hyperplan"

We asked if we can define the hyperplane directly. If we can do so, we will be able to eliminate the noise from the training instances giving rise to a significant increase in accuracy. In pursuit of our ideal hyperplane, we begin by learning how the most accurate classification machine, i.e., the human subject, determines the presence of a free parking lot. Since we are no psychologist, we turned to existing literature for some guidance. Fortunately for us, Zhao and Nevatia [21] reported such an experiment with some useful findings. In the test they conducted, the factors most people mentioned about knowing the presence of a car are **(i)** their rather rectangular shapes; **(ii)** the visibility of front and rear windshields; **(iii)** evidence of a parking lot; and **(iv)** environmental conditions such as shadows or light.'

During classification, these factors are the discriminating feature spaces to be used for detecting the presence of a car in an image. And in the specific case of the SVM, they will be the hyperplane we are seeking when feeding the learning instances to the SVM learner. While conceptually this is easy to explain, trying to implement this within the SVM isn't as straightforward. After all, the algorithm was designed to learn about the cut rather than to be told of the cut. While it is possible to process images of the noise to get close to the ideal hyperplane, it is not possible to automate this under multiple streams of images. This led us to consider an alternative.

In our opinion, these factors are clearly the key feature spaces to use in determining the presence of a car when given an image. In other words, the human subject would filtered other information focusing on the key features to arrive at the conclusion. In lingo of SVM, this would be the hyperplane we seek when feeding the learning instances to the SVM learner.

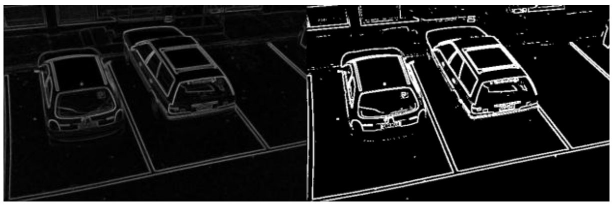

(a) (b)

Fig. 3. Final image used to detect presence or absence of car: (a) before applying the binarisation filter; (b) after applying the binarisation filter, which improves accuracy. Clearly, the reason for the improved accuracy is the wider margin between the two intensities after the binarisation filter.

While conceptually this is easy to explain, trying to implement this within the SVM isn't as straightforward. After all, the SVM was designed to learn about the discriminating features rather than been told what they are.

The immediate and apparent solution is to produce learning instances containing only these feature spaces. Instead of going with this option, we toyed with an alternative approach: *why not explicitly code the classifier instead of feeding our psychological observations into the SVM learner?* Clearly, the benefit of doing so is performance, i.e., a custom classifier exploiting the psychological observations will result in real-time processing capabilities that the application needs. The idea of an accurate and fast classifier is very attractive to us. Hence, our decision to implement these findings using image processing techniques.

We first convert the colour images into 8-bit grey scale images allowing each pixel to be represented exactly in a byte for the ease of implementation. For now, we restrict ourselves to the analysis of a single parking lot taken in an image. Our solution will easily and directly scale to multiple parking lots. With the first factor been the shape, our intuition is to begin by applying an edge filter on the image. As shown in Figure 1(b) and Figure 2(a), the edge filter strips the noise in an image by dropping texture and tonal details but leaving behind structural properties to allow an object to be determined.

The next factor is crucial to the speed and accuracy of our proposal. The experiments at our test site revealed that the key determinant of an unavailable park lie in the visibility of the windshields. The windshields are glass surfaces that reflect light giving off a higher intensity within the parking lot relative to its environment. Even in dim multi-story parks, this remained the case even after the image was stripped off its details by the edge detection filter.

Our algorithm uses this key observation, which will be discussed in Section III-B.

The third human consideration is to look for evidence of a parking lot. In our case, this factor is built into the algorithm as we deal with a per parking lot basis during detection. For an image taken, we will predefine the boundary coordinates for each parking space in the image. In fact, the boundary coordinates defined an area smaller than the parking lot. In our experiments, we find that this gave rise to better efficiency *and* accuracy when the area is concentrate around the spot(s) where the windshields, i.e., factor (ii), are likely to appear.

The last human observation is by far the most challenging. Car colour and size, and varying lighting (or weather) conditions can cause false positives (or negatives) in the detection outcomes. For a dark car, there may be insufficient light from the windshield to conclude the presence of a car (i.e., false negatives). Likewise, a small car will give bigger variation in terms of where they can park within a lot space. And with windshields a key determinant

in our algorithm, over variation in the position of the windshield will increase false negatives. Interestingly, dealing with lighting conditions was much easier than dealing with car colour and size. The issue with lighting conditions is mainly constrained to open parking spaces with natural lighting. As weather conditions (e.g., position of the sun) vary, the detection accuracy also fluctuates.

On weather conditions, we found rain to be an issue when our test camera was not properly sheltered. This caused significant problems when we applied the edge filter leading to false positives. This was easily overcame by mounting a shelter on the camera. Unlike fixed light sources in sheltered parking lots, the movement of sunlight throws varying light intensity (i.e., shadows) on the same parking lot resulting in both false positives and negatives. As Figure~2 showed, the movement of sunlight caused presence of noise when passing images through the edge detection filter. Our approach is to use multiple reference images to compensate the varying levels of light due to the sun's movement. Instead of a single reference image (such as Figure 1(a) taken at time$\mathcal{t}$), we used an array of reference images taken throughout the day to allow variation in the threshold thereby minimising the errors. With this intuition, we discuss the algorithm in the next section.

2. Algorithm

Let $\mathcal{C} = \{c_1, c_1, \dots, c_n\}$ be the set of cameras in a car park facility. For any camerac_i, we define a tuple $\langle c_i, \mathcal{P}_i = \{p_1, p_2, \dots, p_k\}\rangle$ such that $p \in \mathcal{P}_i$ is a parking lot monitored byc_i. We also define a tuple $\langle c_i, \mathcal{R}\rangle$ such that $\mathcal{R} = \{r_1, r_2, \dots, r_j\}$ is a set of reference images taken by c_i when the lots in $\mathcal{P}$ are unoccupied and

$r \in \mathcal{R}$ is a reference image taken at some time period. For any $p \in \mathcal{P}$ the rectangular detection zone $\mathcal{Z}(p) = \langle (x_1, y_1), (x_2, y_2)\rangle$ marks the area where the light intensity is measured in $\mathcal{R}$ and also the image stream $\mathcal{I}$ captured byc_i, which we define as a tuple $\langle c_i, \mathcal{I} = \{r'_1, r'_2, \dots\}\rangle$.

In defining $\mathcal{Z}(p)$, the coordinates are usually within the boundary defined by the parking lines *and* located approximately where the windshields are likely to appear for a given camera angle. As seen in Figure 3, after edge detection and binarisation, the edges of the windshields become a means to identify a change in the intensity reading in the `middle' of the parking lot thus, suggesting the presence of a car. While it is possible to work on Figure 3(a), we find better accuracy after the binarisation filter as the margin of error is significantly increased $-$ as shown in Figure 3(b).

The detection is made by comparing the intensity reading between r and r' for a given $p \in \mathcal{P} \in \langle c_i, \mathcal{P}_i\rangle$ such that the intensity difference in the area defined by $\mathcal{Z}(p)$ on $r \in \mathcal{R} \in \langle \mathcal{P}, \mathcal{R}\rangle$ and $r' \in \mathcal{I} \in \langle c_i, \mathcal{I}\rangle$ is aboveε.In determining ε, the light intensity threshold that suggests the presence of a car, some calibration will be expected. This calibration is made with respect to the site condition and we believe is acceptable for a system of this nature. At our test site, we set this at a value of 15%. In other words, if the light intensity measured from the reference image r on the area determined by $\mathcal{Z}(p)$ is 1, then the light intensity measured from r'on the same area must be > 1.15 to conclude the presence of a car. The calculation to achieve this is given in Algorithm 1.

Since our images are grey scales, each pixel carries a value in the range of 0 … 255, where 255 is a white that indicates the highest light intensity on the pixel. For any given $\mathcal{Z}(p)$, we are interested in the average intensity of light defined by $\mathcal{Z}(p)$ on $\mathcal{R}_{ci}$ and $r' \in \mathcal{I}_{ci}$ respectively. We then compute the ratio to determine if the intensity in r' is ε higher than r . Equation 1 summarises this calculation.

$$\varphi(r, r') = \frac{1}{\mathcal{Z}(p)} \sum \frac{r'_{(x,y)}}{r_{(x,y)}} = \begin{cases} > \varepsilon & \text{Occupied} \\ \leq \varepsilon & \text{Free} \end{cases} \qquad (1)$$

As mentioned and is the case with any threshold, this needs to be adjusted to suit individual cases. Once camera positions are fixed, reference images may be taken and the image streams can be used to empirically work out the best value of ε for a given camera. Once set, any variation in the environment is compensated using a different $r \in \mathcal{R}$ instead. Thus $\mathcal{R}$ is critical in cancelling out any noise that may impede accurate detection. We note that this calibration process is far more efficient than some machine learning approach, where training and verification can take longer.

We also apply an additional binarisation filter to eliminate pixel noise to achieve a cleaner wire frame of a car. We find that doing this will improve accuracy further when pixel values are cleaned up to either a value of 0 or 255. The challenge of using this filter is the need to provide a thresholdℓ, where a pixel value > ℓ will result in a white (and black otherwise). It is tempting to simply go for the mid-value of the intensity range, i.e., setting ℓ = 127. However, doing so will not allow for varying light intensity in different photos and can, as our experiments show, result in a poorer detection accuracy. To determine the rightℓ, we first find the average intensity in the image. Next, we adjust this mean value by adding 1-standard deviation to the threshold to derive ℓ, which is the basis for the binarisation filter. We find that by adding 1-standard deviation to the mean intensity of the image, the results are more accurate as image noise are removed.

Algorithm 1 DetecCars $(\{\langle c_i, \mathcal{P}_i, \mathcal{R}_i, \mathcal{I}_i\rangle, \ldots\})$

```
    for all $p \in \mathcal{P}_i$ do
        $r \leftarrow \sigma_{\text{Current time window}}(\mathcal{R}_i)$
        $r' \leftarrow \sigma_{\text{Most recent image}}(\mathcal{I}_i)$

$\mathcal{F} = \{\text{Greyscale, Edge Detect, Binarisation}\}$
        $r \leftarrow \text{ApplyFilter}(r, \mathcal{F})$
        $r' \leftarrow \text{ApplyFilter}(r', \mathcal{F})$

        if $\varphi(r, r') > \varepsilon$ then
            print Occupied for $c_i.p$
        else
            print free for $c_i.p$
        end if
    end for
```

IV. EXPERIMENTAL RESULTS

In determining the effectiveness of our approach, we will benchmark our technique against that reported in [13]. Their proposal uses the SVM, where multiple classifiers were built to determine the availability of 3 parking lots. Whenever possible, we replicate the empirical conditions used in [13] so as to give an accurate comparison.

In our setup, the same number of samples, i.e., 300, were taken on 3 parking lots as shown in Figure 1. Like our benchmark, the samples were taken over a day from the same position accounting for lighting conditions, changes in the colour intensity, and movement of vehicles in and out of the parking lots. We also experienced rain conditions that wasn't in the plan but nevertheless provided additional consideration in the design of such a parking system. Unlike the benchmark however, we do not require prior training. Instead, we define $\mathcal{Z}(p_1)$, $\mathcal{Z}(p_2)$, and $\mathcal{Z}(p_3)$, indicating where the windshields are likely to be. We also spent another day taking images for $\mathcal{R}$ at 4 interval periods: early morning, late morning, early afternoon and late afternoon. We then recreate the 8-class SVM reported in the benchmark by replicating the training process. In doing so, the immediate difference is the amount of overheads required in the preparation of the benchmark method. For 3 parking lots, 2400 patches (300 for each class) of the image is needed. Acquiring these patches proved a very time consuming process that is unattractive when scaling up to large parking sites. Compared to our approach, there is no need to involve the mammoth task of training, which of course is an immediate benefit.

In [13], a range of classification accuracies were reported using different number of training samples. In this paper, we work directly with the highest number of samples so as to yield the most accurate version of their classifier. We then compare this accuracy level against our work using the same test images.

	Approach by [13] using 2400 samples	Proposed Technique (w/o training samples)
against SVM (3 spaces)	85%	93%
against SVM (3 spaces)	93.52%	93%
against SVM (1 spaces)	83%	97%

Fig. 4. A comparison of detection accuracy using highest level of training samples and the Markov Random Field (MRF) reported in~\cite{WHW+07} against the proposed technique, where such machine learning training is non-existent. Instead, encoding the human heuristics into simple image processing actions and time-related calibration provided comparable accuracy and significant reduction in training time.

On our tests (Figure 4), our approach achieved 93% in classification accuracy for 3 parking spaces and a very high 97% on a single parking space. This is on par with the reported 93.52% accuracy in [13], when a high level of training samples is used with the Markov Random Field (MRF) correction (for 3 spaces). When training samples are dropped, our technique becomes immediately attractive when the high cost of training is eliminated. In our case, the inaccuracies were a result of small cars giving rise to a bigger variation of the windshield positions within the parking lot. In such a situation, the light intensity gathered by our algorithm was too low to trigger detection, i.e., false negatives.

In [13], the average conflict rate was also measured. This measure reflects the error as a result of camera angles capturing the presence of other cars beside the lot of interest, e.g., Figure 3(b) when camera angle is positioned on the left of the image. Again on this measure, the benchmark performed better only when there are sufficiently high training samples. In many cases, we can improve on this measure by reconsidering the camera positions. In positions where the overlap is minimal, we can improve on this measure without changing any part of the algorithm. We do recognise that this suggestion may increase the cost of the system but it will really depend on the decision maker to decide the balance to strike with accuracy on this matter and the costs.

The final measure is on the level of false positives. The challenge of false positives arises out of changing light conditions. Primarily due to the movement of sunlight in open spaces and the colour of the car in sheltered parking sites, our approach has a rate of 3.86% on average. While this is higher than the proposed SVM and MRF correction method in [13] (1.25%), it performs better than the benchmark without the MRF correction (4.39%). We intend to improve on this measure as part of our future work. Instead of just increasing the number of reference images in $\mathcal{R}$, we will consider similar correction techniques like the MRF used in the benchmark. On the performance of the false negatives though, our performance does not seem to differ greatly from the techniques evaluated (as reported in Figure 5).

V. PERFORMANCE ENHANCEMENTS

So far, the discussion dealt with a single image stream $\mathcal{I}$ from a single camera $c_i \in \mathcal{C}$. Typically, a large parking facility will require multiple cameras (i.e., $\mathcal{C}$) to capture all available lots $\langle c_i, \mathcal{P}_i \rangle$, which means that multiple image streams $(\mathcal{I}_1, \mathcal{I}_2, \ldots)$ are generated. To process every frame (or image) on the server will incur expensive computational costs that are undesirable when (i) the number of image streams are high and (ii) the rate at which each camera captures images are also high. As such, to avoid a spike in computational load, we need to further investigate how multiple image streams can be computed efficiently. As noted in Section IV, as we reduce the amount of pixels, we sacrifice on accuracy suggesting that we maintain each image at a reasonable size. Likewise if the capture rate is low, then the timeliness of updating the facility information will be delayed. Since it is undesirable to compromise on either the image size or capture rate, we need an alternative way to ensure performance do not take a hit in the presence of multiple image streams.

A crucial observation is that the high capture rate of each camera also meant that we experience a high rate of "similar situation". That is, between two consecutive frames (or images) from the same camera, the probability that a parking lot is 'free' or 'occupied' remain very high. In other words, there is no need to compute $\varphi(r, r')$ (in Algorithm 1) on every image in the streams $-$ since after each expensive compute, the outcome has a high probability of being the same as the previous frame. This led us to ask if we can devise another algorithm that is highly compute efficient at detecting a change between two consecutive images? If we can develop such an algorithm, then we only need to compute $\varphi(\,)$ when we think there is a change in the situation. As such, the algorithm need not be absolutely accurate (i.e., we can accept a degree of error) as long as it is highly efficient at telling us if we need to apply Algorithm 1. This led to the question of whether we can develop an efficient *change detection* algorithm.

Conventional change detection algorithms use different methods to detect changes between two consecutive images. The most basic method is to take an image stream $\mathcal{I}$ as the input and for each image $r' \in \mathcal{I}$, a binary image called a "change mask [22]-[24] is generated by computing each pixel x_i of r'_i and x_{i-1} of r'_{i-1} using $\mathcal{B}(x_i, x_{i+1})$, where $\mathcal{B}(\,) = 1$ if $x \in r'$ has changed significantly from $x \in r'_{i-1}$. Otherwise, $\mathcal{B}(\,) = 0$. Computing the change mask is straightforward but pose two crucial issues. First, the method to compute a change mask has a compute complexity that i proportional to the size of the image resulting in an equivalent complexity as $\varphi(\,)$. Second, detecting a change is not easily from the change mask as it has been found to be highly application specific [25]-[27].

Of course there are other techniques proposed to improve change detection between two images and the problem has a long history with many other computer problems such as image registration, object segmentation and tracking, etc., all of which bear a substantial literature of their own.

	False Accept Rate	False Reject Rate
against SVM (3 spaces)	4.39%	8.73%
against SVM (3 spaces) + MRF	1.25%	3.56%
against SVM (1 spaces)	4.85%	8.12%
against proposed technique	3.86%	5.34%

Fig. 5. A comparison of false positives (or false accept), i.e., wrongly detected the presence of a car in a parking lot and false negatives (or false reject), i.e., wrongly detected the absence of a car in a parking lot, using the techniques in [13] against the proposed technique.

The biggest challenge in change detection lie in the lack of an agreed uniform definition of what constitutes to a "significant change" between two images. Can "significant change" be determined by geometric changes [28], [29], such as a difference in the pixel intensity [29], [30]? Or do we require more sophisticated techniques that use hypothesis tests that often require complex compute on the images, such as comparing the histogram of two images and computing the statistical likelihood of a change [31], [32]? Regardless of the accuracy or compute efficiency of these mechanisms, we note that our problem do *not* require the presence of such complex algorithms. Again this argument is driven by our need for more compute efficient methods than $\varphi(\,)$. Since we can afford false positives, i.e., wrongly detecting that a change in situation has happened, we can opt for a very simplistic approach. This method, when combined with the right probability distributive function, actually result in a significant reduction of compute costs.

1. Algorithm

The key idea behind Algorithm 1 is the encoding of human observation as a way to classify the presence or absence of a car within a park. This approach has the advantage of not requiring training but instead uses reference images and calibration to determine if the light intensity suggests the presence of a car in a parking lot. In order to achieve the accuracy required in Algorithm 1's $\varphi(\,)$, (i) three image filters were used and (ii) the intensity of all pixels within the specific area of the parking lot was determined.

One way to be faster than Algorithm 1 is to reduce the number of filters used and the number of pixels computed in $\varphi(\,)$. With the image filters, hardware acceleration can be used by acquiring cameras that take grey scale images for example. We therefore focus on the number of pixels to process. Our observation is that if a situation change is detected in image $\mathcal{r}_i$, then the likelihood of a new situation arising in $\mathcal{r}_{i+1}$ is very low. In other words, the probability of a situation change $\mathcal{P}(\,)$ following a detected situation change in $\mathcal{r}_i$ will only increase with subsequent shots, i.e.,$\mathcal{P}(\mathcal{r}_{i+1}|\,\mathcal{r}_i) \le \mathcal{P}(\mathcal{r}_{i+2}|\,\mathcal{r}_{i+1}) \le \ldots$. This general observation allows us to apply an exponential distribution $f(x)$ to the number of pixels assessed. Recall in computing $\varphi(\,)$, an area $\mathcal{Z}(\mathcal{p})$ marked within the parking lot is considered. Since $\mathcal{P}(\mathcal{r}_{i+1}|\,\mathcal{r}_i) \approxeq 0$ (in most cases), we only need to sample a small number of pixels in $\mathcal{Z}(\mathcal{p})$ that $\varphi(\,)$ checks, i.e,. $f(i)\cdot|\mathcal{Z}(\mathcal{p})|$, where $|\mathcal{Z}|$ is the number of pixels defined in the detection zone. Overtime and guided by the statistical distribution, we progressively increase the amount of pixels checked in $|\mathcal{Z}(\mathcal{p})|$ resulting in a net reduction of pixels computed between two situational changes. As with the use of reference images $\mathcal{R}$ to compensate for differences in the environment, $f(\,)$ needs to be calibrated according to the traffic flow of the parking facility. Such calibration can be achieve by adjusting two parameters (μ, β) in f where their relationship is given as

$$f(x) = 1 - \frac{1}{\mu} e^{-x+\frac{\beta}{\mu}} \tag{2}$$

In our case, β is the location parameter in f that determines, when the exponential distribution applies to $\mathcal{Z}(\mathcal{p})$. If it should start after the 10th image (i.e., at $\mathcal{r}_{i+10}$ following a situation change in $\mathcal{r}_i$), then $\beta = 10$. The scale parameter μ equates to the average time a car remains in a parking lot. So if a car remains in the parking lot on average of 3 hours, then $\mu = 3$ or $\mu = 180$ (minutes) depending on the granularity of the exponential scale used.

In terms of implementation, a crucial step is in the choice of the pixels to select from $\mathcal{Z}(\mathcal{p})$ for such quick assessment. Clearly if only a small number of pixels are selected, then those pixels should not be constrained to a specific area within $\mathcal{Z}(\mathcal{p})$. Otherwise, the degree of error can be high. To minimise the occurrence of such errors, the selected pixels should continue to test the maximum area possible in $\mathcal{Z}(\mathcal{p})$. We achieve this by aligning the number of pixels from $f(i)\cdot|\mathcal{Z}(\mathcal{p})|$ on an imaginary line across $\mathcal{Z}(\mathcal{p})$. The pixels are equally spaced out on this imaginary line as points to test intensity readings. The readings taken are then compared to the pixel readings in the reference image. When the difference exceeds ε, we trigger a full compute using $\varphi(\mathcal{r}, \mathcal{r}')$. Figure 6 elaborates this process in detail while Algorithm 2 incorporates this process in $\varphi(\mathcal{r}, \mathcal{r}', f)$ as shown.

The pixels that lie on this imaginary line become points in $\mathcal{Z}(\mathcal{p})$ where readings are sampled. No doubt such readings will not achieve a high accuracy but since we already have prior knowledge of the existence of a car (or its absence), we only use the result to decide if there may be a change in situation. As such, the results will only lead to false positives that trigger the compute of $\varphi(\,)$ on all pixels in $\mathcal{Z}(\mathcal{p})$ to verify if a situation change has occurred. These

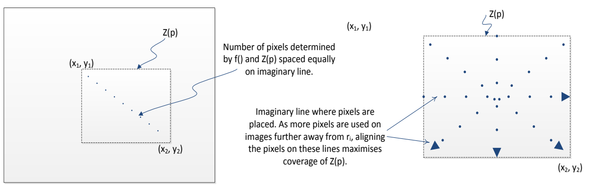

Z(p)
(x1, y1)
(x2, y2)
Number of pixels determined by f() and Z(p) spaced equally on imaginary line.
(x1, y1)
Z(p)
Imaginary line where pixels are placed. As more pixels are used on images further away from ri, aligning the pixels on these lines maximises coverage of Z(p).
(x2, y2)

Fig. 6. A visual explanation of how $\varphi(\mathcal{r}, \mathcal{r}', f)$ is computed. As described in Section V-A, we first determine the number of pixels to select by computing $f(i)\cdot|\mathcal{Z}(\mathcal{p})|$, where i is the i-th image since Algorithm 1 detected a situation change. So if $f(i)\cdot|\mathcal{Z}(\mathcal{p})|$ = 9, then we will select 9 pixels along the imaginary line that stretched across $\mathcal{Z}(\mathcal{p})$ as shown. These 9 pixels are where the intensity readings are taken on $\mathcal{r}$ and the reference image $\mathcal{r}'$. The magnitude of the average intensity is then taken and compared against ε and it the threshold crosses, we invoke the more computational intense but accurate $\varphi(\mathcal{r}, \mathcal{r}')$. Note again that with less pixels used, it means that the i-th frame is more recent to the frame that had a situation change that we modelled as $\mathcal{P}(\,)$.

false positives will not affect the accuracy of the system except to waste some compute time. Nevertheless, the overall result of such fast compute reduces the number of triggers on a full compute with $\varphi(\,)$ and therefore, we continue to improve overall performance. Algorithm 2 shows the changes made to Algorithm 1, where $\varphi(\mathcal{r}, \mathcal{r}', f)$ and ε capture the essence of our discussion in this section and we present performance results of this system next.

```
Algorithm 2 DetecCars ($\{\langle c_i, \mathcal{P}_i, \mathcal{R}_i, \mathcal{I}_i\rangle, \ldots\}$)
    for all $\mathcal{p} \in \mathcal{P}_i$ do
        $\mathcal{r} \leftarrow \sigma_{\text{Current time window}}(\mathcal{R}_i)$
        $\mathcal{r}' \leftarrow \sigma_{\text{Most recent image}}(\mathcal{I}_i)$
```

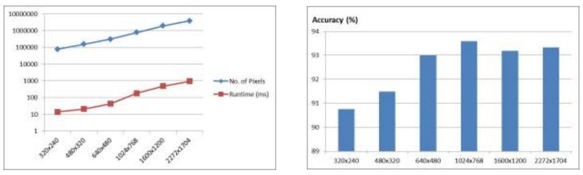
```
$\mathcal{F} = \{\text{Greyscale}, \text{Edge Detect}, \text{Binarisation}\}$
            $r \leftarrow \text{ApplyFilter}(r,\ \mathcal{F})$
            $r' \leftarrow \text{ApplyFilter}(r',\ \mathcal{F})$

            // sample some pixels as determined by
            // distribution $f(\,)$ and $\mathcal{Z}(p)$
            **if** $\varphi(r,\ r', f) > \varepsilon$ **then**
                // initiate full compute as difference
                // is above threshold, $\varepsilon$
                **if** $\varphi(r,\ r') > \varepsilon$ **then**
                    **print** Occupied for $c_i.p$
                **else**
                    **print** free for $c_i.p$
                **end if**
            **end if**
        **end for**
```

2. Experimental Results

Our earlier empirical evaluation in Section IV focused on the accuracy of the proposed solution. While the solution achieved good accuracy, Algorithm 1 alone failed to provide decent real-time operation on a single computer. This led to the performance enhancements that result in Algorithm 2. In this section, we report our performance evaluations of these algorithms.

Our implementation uses C# and the .NET's built-in graphics library for bitmap manipulation. The image filters were drawn from an Open Source library called AForge.NET (http://www.aforgenet.com) to allow us to quickly implement the algorithm to test the viability of our proposal. Our first test is to validate the performance of Algorithm 1. Figure 7(a) shows the number of pixels for each image profile and its corresponding average runtime of 50 images. As the Y-axis is a log scale, we can see that as the number of pixels double from one image profile to another (e.g., 640x480 to 1024x768), we see the runtime quadruples. As such, although high resolution images provide better accuracy for Algorithm 1, we are constrained by the runtime costs. Figure 7(b) shows the average accuracy achieved over 50 images at different sizes. A significant improvement in accuracy is seen when the image size goes to 640x480. Recall that the size of $\mathcal{Z}(p)$ is a region of the profile and not the entire image, it is clear that this profile provided the minimum number of pixels required in $\mathcal{Z}(p)$to get an accurate reading. Although the image size of 1024x768 gave further accuracy, its runtime performance isn't ideal. Interestingly, as the image scales beyond 1027x768, accuracy drops suggesting the impact of noise on higher resolution images. Our attempt to recalibrate ε did not result in any significant improvements. However this should not be a concern given that the focus of such a system is to be cost-efficient and therefore, we expect to work with images of lower resolution to begin with. The performance and the error considered, we conclude that the image size at 640x480 yield a good result. Any image larger than 640x480 yield noticeable 'pauses' in intensity computation while using images smaller than 640x480 lowers accuracy. Even with variation to ε, our empirical evaluation indicates that accuracy and size is optimally balanced at 640x480.

The next performance evaluation takes into account of 120 images with 3 situation changes. We then measure the runtime to complete processing all 120 images. We further



(a) (b)

Fig. 7. Runtime performance of computing different image sizes using Algorithm 1: (a) average runtime of 50 images over different sizes along with the number of pixels in each image profile - y-axis is log scale; (b) average detection accuracy over 50 images at different sizes. Comparing (a) and (b), we conclude 640x480 as a good size on our hardware.

note the number of times $\varphi(r, r')$ was invoked in Algorithm 2 so as to determine the false positive percentage. For this test, $f(x)$ was configure with $\beta = 0$ and $\mu = 10$. Setting β to 0 means we start sampling $\mathcal{Z}(p)$ immediately following a situation change. Further we expect a car to leave or occupy a parking lot every 10 minutes (i.e., μ). The sequence of 120 images (i.e., a shot taken every minute) hence correspond to 2 hours of image shots taken on the same 3 parking lot space depicted in Figure 1. We then vary μ over a number of repeats on the same set of images recording the runtime performance in Figure 8.

Figure 8(a) shows the runtime performance of Algorithm 1 against Algorithm 2 (which uses the performance enhancements outlined in Section V-A). Note that $\mu = 10$ applies only to Algorithm 2 since Algorithm 1 does not use any optimisation. By avoiding a full compute in Algorithm 2, we see significant 36% improvement in performance (from 5.03 seconds to 3.19 seconds). As we increase μ, further performance improvements is seen in Figure 8(b) but at the expense of lower accuracy. We have not reported this accuracy since it depends entirely on when situation changes take place against the setting of μ. Nevertheless it is suffice to note that as μ increases, we will compromise on accuracy. Further to conclude the experiments, it is important to appreciate that the absolute runtime improvements quickly add up when multiple image streams are to be processed.

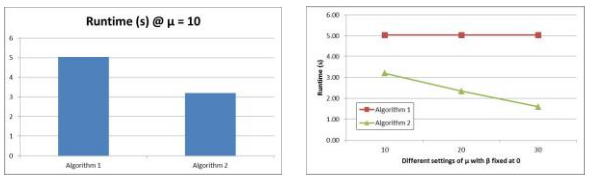
VI. CONCLUSION

Despite advances in parking systems, we continue to face frustrations at heavily utilised parking sites. Current systems fail because drivers have no prior access to information until arrival. And upon arrival, much of the search for a free park is ad-hoc based on information from signages and light indicators within the driver's line of sight. Our proposed system will ease driver frustrations through a system that integrates Internet-enabled smart phones. In Australia and many parts of the world, the ubiquitous adoption of such devices has made it feasible for drivers to access live parking information prior to arrival. When combined with other technologies, this opened up possibilities of a parking system that could inform drivers before arrival at site, direct drivers to parking lots, and thus regulating traffic in the surroundings. Our immediate future work is therefore to build applications on smart phones to demonstrate these ideas coming off a "camera-to-server" approach. Critical to achieving this is the development of a detection mechanism to fit the "camera-to-server" model, which is cost effective and technically viable. As argued earlier, existing systems and current experimental projects do not consider aspects of this problem. The work reported in this paper thus fills this gap.

The research contribution is a method to enable a "camera-to-server" implementation by balancing the costs against the features needed to deliver the parking system. Unique characteristics of our approach include the applied insights of human detection (as reported in the psychological test conducted by Zhao and Nevatia [21]) in our algorithm, and the explicit coding of a detection behaviour based on the learning characteristics of a SVM learner. By explicitly coding the classification behaviour of a SVM learner, we eliminated the cost of training. At the same time, we eliminated noise that would otherwise be embedded in the hyperplane through the normal training process. This gives us the improvement in detection accuracy. The performance enhancements that follow reinforce the practical utility of this method by reducing the number of full compute through effective use of an appropriate statistical distribution. The final result is a detection mechanism that supports the "camera-to-server" approach with high efficiency and accuracy in an environment with multiple images streams.

ACKNOWLEDGEMENT

Sections of this invited paper appeared in an early version published under the 3rd Int. Conf. Networked Digital Technologies, Proc. Communications in Computer and Information Science, Springer, 2011.



(a) (b)

Fig. 8. Runtime performance comparison between Algorithm 1 and Algorithm 2: (a) runtime of a single image stream containing 120 images with $\mu = 10$ and 3 situation changes; (b) the same image stream used in (a) but with different μ settings. Note that since Algorithm 1 does not use μ, its runtime remains consistent while Algorithm 2 shows improved performance enhancements at the cost of a lower detection accuracy. We have not reported this accuracy as it will vary according to when situation changes occur against the μ values.

REFERENCES

[1] J. Chinrungrueng, U. Sunantachaikul, and S. Triamlumlerd, "Smart Park- ing: An Application of Optical Wireless Sensor Network," Applications and the Internet Workshops, IEEE/IPSJ International Symposium on, vol. 0, p. 66, 2007.

[2] W. Jones, "Parking 2.0: Meters Go High-Tech," IEEE Spectrum, p. 20, 2006.

[3] K. C. Mouskos and N. A. P. Maria Boile, "Technical Solutions to Overcrowded Park and Ride Facilities," University Transport Research Centre (Region 2), http://tris.trb.org/view.aspx?id=814921, Technical Report: FHWA-NJ-2007-011, 2007.

[4] B. Farhan and A. T. Murray, "Siting park-and-ride facilities using a multi-objective spatial optimization model," Computers and Operations Research, vol. 35, no. 2, pp. 445–456, 2008.

[5] C. J. Rodier, S. A. Shaheen, and C. Kemmerer, "Smart Parking Management Field Test: A Bay Area Rapid Transit (BART) District Parking Demonstration," Institute of Transportation Studies, Univer- sity of California, Davis, http://pubs.its.ucdavis.edu/publication de- tail.php?id=1237, Research Report: UCD-ITS-RR-08-32, 2008.

[6] A. Mathijssen and A. Pretorius, "Verified Design of an Automated Parking Garage," in Formal Methods: Applications and Technology, ser. Lecture Notes in Computer Science. Springer Berlin/Heidelberg, 2007, vol. 4346, pp. 165–180.

[7] T. Hodel-Widmer and S. Cong, "PSOS: Parking Space Optimization Ser- vice," in 4th Swiss Transport Research Conference, Monte Verit/Ascona, March 2004, pp. 1–22.

[8] K. Inaba, M. Shibui, T. Naganawa, M. Ogiwara, and N. Yoshikai, "Intelligent Parking Reservation Service on the Internet," in Symposium on Applications and the Internet-Workshops, San Diego, CA, USA., 2001, pp. 159–164.

[9] Q. Liu, H. Lu, B. Zou, and Q. Li, "Design and Development of Parking Guidance Information System based on Web and GIS Technology," in 6th International Conference on ITS Telecommunications, Chengdu, China, 2006,

pp. 1263–1266.

[10] Y. Li, R. Ma, and L. Wang, "Intelligent Parking Negotiation Based on Agent Technology," in Information Engineering, WASE International Conference on, vol. 2, jul 2009, pp. 265–268.

[11] Y. Mo and Y. Su, "Design of Parking Guidance and Information System in Shenzhen City," in Computing, Communication, Control, and Management, ISECS International Colloquium on, vol. 4, aug. 2009, pp.37–40.

[12] H. Zhong, J. Xu, Y. Tu, Y. Hu, and J. Sun, "The Research of Parking Guidance and Information System based on Dedicated Short Range Communication," in Intelligent Transportation Systems, Proceedings. IEEE, vol. 2, oct. 2003, pp. 1183–1186.

[13] Q. Wu, C. Huang, S. Wang, W. Chiu, and T. Chen, "Robust Parking Space Detection Considering Inter-Space Correlation," in Multimedia and Expo, IEEE International Conference on, jul 2007, pp. 659–662.

[14] M. Vidal-Naquet and S. Ullman, "Object Recognition with Informative Features and Linear Classification," in Computer Vision, IEEE Interna-tional Conference on, Nice, France, 2003, pp. 281–288.

[15] G. L. Foresti, C. Micheloni, and L. Snidaro, "Event Classification for Automatic Visual-based Surveillance of Parking Lots," Pattern Recog-nition, International Conference on, vol. 3, pp. 314–317, 2004.

[16] C. H. Lee, M. G. Wen, C. C. Han, and D. C. Kou, "An Automatic Monitoring Approach for Unsupervised Parking Lots in Outdoors," in Security Technology, International Carnahan Conference on, oct. 2005, pp. 271–274.

[17] I. Masaki, "Machine-Vision Systems for Intelligent Transportation Sys- tems," Intelligent Systems and their Applications, IEEE, vol. 13, no. 6, pp. 24–31, nov. 1998.

[18] A. Mohan, C. Papageorgiou, and T. Poggio, "Example-Based Object Detection in Images by Components," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 23, no. 4, pp. 349–361, 2001.

[19] H. Schneiderman and T. Kanade, "A Statistical Method for 3D Object Detection Applied to Faces and Cars," in Computer Vision and Pattern Recognition, International Conference on, Hilton Head, SC, USA, 2000, pp. 1746–1759.

[20] E. Osuna, R. Freund, and F. Girosi, "Support Vector Machines: Training and Applications," Massachusetts Institute of Technology, Cambridge, MA, USA, Tech. Rep., 1997.

[21] T. Zhao and R. Nevatia, "Car Detection in Low Resolution Aerial Images," Image and Vision Computing, pp. 710–717, 2001.

[22] P. Rosin, "Thresholding for change detection," in Computer Vision, 1998. Sixth International Conference on, jan 1998, pp. 274 –279.

[23] ——, "Thresholding for change detection," Computer Vision and Image Understanding, vol. 86, no. 2, pp. 79 – 95, 2002. [Online]. Available:http://www.sciencedirect.com/science/article/pii/S1077314202909604

[24] P. Smits and A. Annoni, "Toward specification-driven change detection," Geoscience and Remote Sensing, IEEE Transactions on, vol. 38, no. 3, pp. 1484 –1488, may 2000.

[25] L. Di Stefano, S. Mattoccia, and M. Mola, "A change-detection algo- rithm based on structure and colour," in Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance, 2003., july 2003, pp. 252 – 259.

[26] M. J. Black, D. J. Fleet, and Y. Yacoob, "Robustly estimating changes in image appearance," Computer Vision and Image Understanding, vol. 78, pp. 8–31, 2000.

[27] Y.G. Leclerc, Q.-T. Luong, and P. Fua,"Self-consistency and mdl: A paradigm for evaluating point-correspondence algorithms, and its application to detecting changes in surface elevation," International Journal of Computer Vision, vol. 51, pp. 63-83,2003,10.1023/A:1020940807324.[Online]. Available:http://dx.doi.org/10.1023/A:1020940807324

[28] A. Can, C. Stewart, B. Roysam, and H. Tanenbaum, "A feature-based, robust, hierarchical algorithm for registering pairs of images of the curved human retina," Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 24, no. 3, pp. 347 –364, mar 2002.

[29] C. Stewart, C.-L. Tsai, and B. Roysam, "The dual-bootstrap iterative closest point algorithm with application to retinal image registration," Medical Imaging, IEEE Transactions on, vol. 22, no. 11, pp. 1379 –1394, nov. 2003.

[30] D.G. Lowe, "Distinctive image features from scale-invariant keypoints," Int. J. Comput. Vision, vol. 60, pp. 91–110, November 2004.[Online].Available:http://dl.acm.org/citation.cfm?id=993451.996342

[31] S.-S. Ho and H. Wechsler, "Detecting changes in unlabeled data streams using martingale," in Proceedings of the 20th international joint conference on Artifical intelligence, ser. IJCAI'07. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, pp.1912–1917.[Online].Available:http://dl.acm.org/citation.cfm?id=1625275.1625584

[32] N. Mozafari, S. Hashemi, and A. Hamzeh, "A precise statistical approach for concept change detection in unlabeled data streams." Computers & Mathematics with Applications, pp. 1655–1669, 2011.