Simple Real-Time Pattern Recognition for Industrial Automation

Vikram Voleti
GreyOrange India Pte Ltd
Gurgaon, Haryana, India
vikram.v@greyorange.sg

Prakhar Mohan
GreyOrange India Pte Ltd
Gurgaon, Haryana, India
prakhar.m@greyorange.sg

Saket Gupta
GreyOrange India Pte Ltd
Gurgaon, Haryana, India
saket.g@greyorange.sg

Javed Iqbal
GreyOrange India Pte Ltd
Gurgaon, Haryana, India
javed.i@greyorange.sg

ABSTRACT

Industrial automation often involves time-intensive operations on real-time inputs, and accuracy is critical to the functioning of the full industrial pipeline. While older methods are simplistic but inaccurate, newer methods are more accurate but often complex and time-consuming. In this paper, we achieve 100% accuracy using simple methods on an image classification task. We take advantage of the electronic systems involved in the industrial pipeline, and offer a methodology that is applicable to other tasks as well. This paper describes our method, and details all the technicalities considered while designing it. The timing results and edge cases are also summarized.

CCS CONCEPTS

• **Computing methodologies → Vision for robotics**; *Appearance and texture representations*; *Object identification*; • **Information systems →** Electronic data interchange;

KEYWORDS

Industrial automation, pattern recognition, image processing

ACM Reference Format:
Vikram Voleti, Prakhar Mohan, Saket Gupta, and Javed Iqbal. 2017. Simple Real-Time Pattern Recognition for Industrial Automation. In *Proceedings of 2017 Int. Conference on Industrial Design Engineering (ICIDE 2017)*. ACM, New York, NY, USA, Article 4, 6 pages. https://doi.org/00.000/000_0

1 INTRODUCTION

Industrial automation often involves time-intensive operations on real-time inputs. Different electronic and mechanical systems work on specific tasks at multiple points in an industrial pipeline. Timing is extremely important to synchronize their tasks. Often the first module is a computer vision system that processes a video input to delegate tasks to the different systems via signals based on the item in the video.

For example, in a warehouse automation system, a sequence of items runs on caskets on a conveyor belt. They are to be sorted into different chutes based on certain attributes, like size or type of item, or destination in the case of e-commerce warehouses. Often, a computer vision module first interprets the attributes of each item from its images in the video feed. It then sends this information, along with each casket's identity, to the warehouse management system's server. The server then computes the identity of the chute from the item's attributes, and sends a signal allocating the casket identity to the computed chute's identity. This signal is used by the mechanical system to appropriately activate relays that push the item on the casket into the computed chute.

As can be expected, it is critical that the computer vision module performs its tasks within enough time to be able to communicate the relevant information to the mechanical relays well before the casket on the conveyor belt reaches the chute's opening. The time for communication of this information is limited by the speed of the conveyor belt, as well as the distance between the computer vision module and the chutes. In optimizing for warehouse productivity, the speed of the conveyor belt is often high. In optimizing for space, often the chutes and electronic modules are placed close together. Hence, it is important that the computer vision module completes its tasks in as minimal time as possible.

Accuracy is equally critical to the functioning of the full pipeline. In case the vision module interprets item attributes wrongly, items would be sorted into wrong chutes. To correct the mistakes, the entire pipeline might have to be stopped, which would significantly drop the warehouse's productivity. Often, warehouses process hundreds of packets per minute. So every minute wasted would lead to significant decrease in the margin return on warehouse investment.

Moreover, even 99% accuracy is not enough in an industrial setting. As mentioned, the turnaround of a warehouse is usually in the order of hundreds per minute. Even if one of those is faulty, it potentially leads to hundreds of errors per day. Obviously this is not a good scenario for the correct functioning of a warehouse.

In this paper, we focus on designing the process of item attribute recognition that takes the least time and yet gives maximum accuracy. The items on the caskets of the conveyor belt could be of any color, shape, size, subject to constraints posed by the warehouse. To this effort, we experiment with the simplest of algorithms for pattern recognition inn images. We also specify the trick used to boost accuracy up too 100% even though the algorithms by themselves are inaccurate.

ICIDE 2017, December 2017, Dubai, UAE
© 2026 Copyright held by the owner/author(s).
ACM ISBN 000-0000-00-000/00/00…$15.00
https://doi.org/00.000/000_0

2 RELATED WORK

Several papers describe object detection in industrial settings, and summarize [6] the progression of various methods used and improved upon to detect object features. Different feature descriptors such as Histogram-of-Gradients [5], Scale-Invariant Feature Transform [13, 14, 17], Gabor coefficients [3], have been expanded upon. These methods calculate local features in an image and provide various advantages towards applications such as object recognition and localization tasks. [19] explores an unsupervised method of detecting scale-invariant features in images.

It is well known that a cascading of weak object features is better than one strong local feature. Such methods of detecting several weak features and boosting them has been looked into with success [9, 18]. All these methods have been extensively compared for their utility in various tasks [11, 16, 20].

As mentioned, these pre-deep learning methods are highly accurate in a research environment, but often unusable in an industrial setting. Real implementations of good algorithms often have suffer from lack of robustness due to newness of the real environment, and often require time-intensive computations that are unaffordable in a real-time scenario.

More recently, deep learning methods like convolutional neural networks [1, 4, 12, 21] are being used extensively in computer vision applications with great success. However, this has been largely dependent on the availability of large amount of data, labelled or unlabelled, and the design of larger architectures [22]. For more specific tasks like the one discussed in this paper, of detecting an item on a casket, data is considerably more scarce than that required for using deep neural networks effectively.

Other methods [2, 8, 15] offer more complicated means of solving this problem. We, however, focus on approaching the problem in a much simpler way, that is applicable to an industrial setting rather than a research setting.

3 OUR APPROACH

The problem we target to be solved by the computer vision module is to detect whether a casket has an item (a packet) on it or not, as shown in Figure 1. It is complicated by the fact that the packets could be of any shape, size, form, color (subject to the constraints of the warehouse). We use relatively simpler pre-deep learning algorithms to determine from the features computed from a frame whether the frame is that of a casket containing a packet or not. Although these often do not provide high accuracies in real scenarios, we manipulate them to effectively obtain 100% accuracy.

The recognition of a packet on a casket is a very complex problem involving the determination of complex features and further processing that might not meet real-time constraints, because the packets are not homogeneous. Instead of recognizing a packet, we process the images to figure out whether they resemble an empty casket or not. This way, dependency on the properties of the packet on the casket is completely nullified. We then take a majority vote [7, 10] on these decisions to determine if that casket has an item on it or not.

The environment we work under is described in section 4. The first part of our method involves localizing the casket position. We use the help of an external sensor to determine whether the camera is facing a casket region or not. This is described in section 5. We then process every image of casket obtained from a camera to compute features that could be used to determine whether it contains a packet or not. The features experimented with are detailed in section 6.

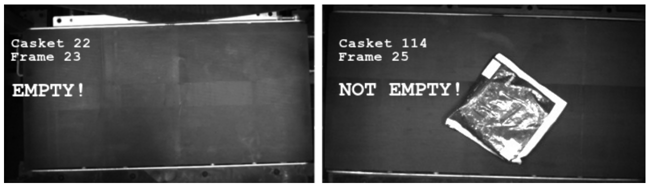

Figure 1: Left: Casket without any packets, right: Casket with a packet

The features obtained for each new image from the camera are then compared with those of a model of an empty casket. If they are close enough to the features from the model, the new image is said to be that of an empty casket. Accordingly, the program outputs a 1 or a 0. The model used and its comparison are covered in the section 7. Section 8 describes the edge cases of our method.

4 WORK ENVIRONMENT

The setup of the system is as shown in Figure 2. The computer vision module consists of a monochrome camera connected to a computing module. The camera is mounted at a fixed position above the conveyor belt, facing down on it. As a convention, the camera is oriented such that the width and height of the images captured are aligned with the width of a casket (perpendicular to the direction of movement of the belt) and length of a casket (in the direction of movement of the belt). The height of the camera above the belt is fixed such that the entire width of the casket is visible in the images captured. This is done because a packet may be positioned anywhere along a casket's width, so we cannot afford to leave out any portion of the width.

The height of the image captured, however, is reduced to exclude a fraction of the length of the casket. This is done so that only a portion of the casket is visible in each image captured. In case the entire height of the casket is visible in an image, the problem of matching the image with a template image (or parameters thereof) shall be significantly tougher. On the other hand, by reducing the image's height, there shall be multiple images of the central portion of the casket as it passes through the camera's field of view, which is not a disadvantage in itself.

Certain measures have been taken so as to reduce the number of variables in the pattern matching problem. The images are captured in an enclosed environment with constant lighting. This means that there are no variable illumination effects to be handled. Also, there is not much change in the perspective angle of caskets being captured, since the position and orientation of the camera are fixed. Had the problem been more general, it is concluded that the real-time requirement might not be achievable.

The frame rate of the camera used has been set to maximum (125fps), subject to the resolution of the image captured. Since we are taking a majority vote on the per-frame decisions, it is imperative that we collect as many images as possible so that our

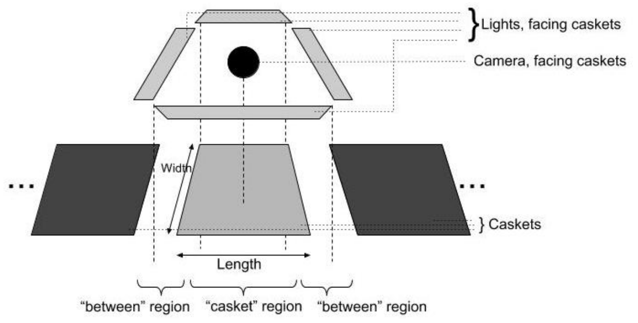

Figure 2: ECDS Setup

decision is robust. One must remember that the computer vision module is connected to other modules that are dependent on its timely outputs for their own functioning. The entire system of modules runs at according to a common clock. As seen earlier, we require the mechanical system gets the computer vision module's output well before the casket reaches it. Thus, to keep the processing of the computer vision module fast, we set the frame rate of the camera as high as possible, and try to perform our computations in the time between the capture of two frames.

This means, if the camera's frame rate of 125fps, the total processing time of an image must not exceed 8 milliseconds. One could argue that using a high-specification computer could enable us to lower processing time. But it does not make good business sense to employ an advanced and costly system for this task. This is why we try to solve our problem with more constraints - a monochrome camera instead of color, and a small CPU instead of a GPU.

5 CASKET LOCALIZATION

In the following sub-sections, the method to determine the central portion of the casket has been described. For convenience, the sample space of images of the top view of the conveyor belt is split into a "casket" region - when a casket passes in front of the camera, and a "between" region - when the portion in between two caskets passes in front of the camera. Of course, there is no real distinct division between these two regions, the images transit from one region to the other continuously as the belt runs.

5.1 Reliance on Tracker Data

It is imperative to find out when the central portion of every casket is in front of our camera. For this purpose, an electronic encoder, or tracker, is placed in the belt drive to find out the current position of the belt. This, in turn, is used to track the ID of the casket present in front of the camera at any given time. However, in our case, this encoder only transmits the position of the belt about every 100ms, while we require the position of the belt about every 8ms, since our camera works at 125fps.

It is possible to extrapolate the current position of the belt up to two decimal points from the speed of the belt drive, which also can be obtained from the encoder in the belt drive.

This way, by only noting the output of the program for every image within the central portion of the casket, and taking a majority vote on these outputs, it is possible to determine whether the corresponding casket is empty or not.

However, extrapolating from the encoder data is not very reliable, since the order of the time considered is in milliseconds, and at such orders the precision of the data could be inaccurate enough for the central portion to be mapped wrongly. For example, assuming that the central portion is between 30% and 50% of the casket's total length, if the encoder data is shifted by up to 10%, it shall result in an apparent central area between 40% and 70% of the casket's length. This means that only 40-60% of the actual central portion shall be noted, and 60-70% of the casket length, which corresponds to between_casket region, shall be wrongly considered.

5.2 Independent of Tracker Data

To stay independent of tracker inaccuracies, data (1s and 0s) along the full length of every casket can be considered, instead of just the central portion. Then, instead of taking a majority vote, the number of 1s in the series of outputs can be counted. If the number of 1s exceeds a threshold, it can be said that the corresponding casket is empty. Else, the casket contains a packet.

5.3 Bargain

Instead of considering the full length, a subset of it that is a little more than the exact central portion is considered. It is wasteful to consider the areas corresponding to the *between_casket* region, but care should be taken to avoid narrowing the region of interest too much. Hence, only 25% to 55% of the casket length is considered to find out whether the corresponding casket is empty or not.

6 SIMPLE APPROACHES TO IMAGE PROCESSING

Since time is crucial in an industrial setting, to target a real-time scenario and scale it up through parallel processing, we have tried to utilize simpler algorithms. Taking a clue from cascade classifiers, we make the simpler algorithms robust by applying them over multiple images. We take advantage of similarity in images, and rely on multiple not-completely-reliable data to make up one absolutely reliable decision. To solve the image processing problem of finding out whether a casket has a packet on it or not, five approaches have been devised and implemented, each with increasing accuracy.

In order to check the accuracy of each approach, many images have been recorded of caskets of different categories - empty caskets, caskets with big packets, caskets with small packets. Small packets here refer to those that are just under the minimum packet size determined by a warehouse. Each method has been checked of each type of casket+packet combination. In real cases, caskets are very noisy, the noise being from dust accumulated, wear-and-tear, random artifacts on caskets, etc.

The challenging part is to classify a packet-less noisy casket as "empty", but classify a casket with even a small packet as "non-empty". It was observed that when the thresholds were adjusted to accommodate more noise on an empty casket, more caskets with small packets were wrongly classified as empty. Finding a balance between these two aspects was challenging.

6.1 Using Entropy

The idea is that the entropy of the image of an empty casket is far less than that of a casket with a packet on it. To implement this, the range of the entropy value of an image of an empty casket can be calculated from a basis set of images of an *"empty_casket"*, and the minimum and maximum values of this range could be set as thresholds for any new image. If the calculated entropy of a new image lies within this threshold, it is deemed to be an image of an empty casket, else not.

While this could work in case of packets that do not match the casket's color (grayscale value), it is not necessarily true for all types of packets. Furthermore, the thresholds calculated are so dependent on the basis set of empty casket images that any small change in the basis set could lead to a significant change in the threshold values. It can be seen that this method shall lead to several false positives and false negatives.

6.2 Using Power Spectrum

The idea is to calculate the power spectrum of the image of an empty casket, and use it as a template to compare with the power spectrum of any new image. However, a similar problem to the previous method arises - which image to choose to make a template? A random image from the list of *empty_casket* images was selected to represent the template of an empty casket. Its power spectrum was calculated up to a certain degree, and stored as the model of an empty casket. Every new image captured was made to undergo the same process of calculating the power spectrum to the same degree. This power spectrum was compared with the model power spectrum using a variety of methods - sum of absolute differences, sum of squared differences, correlation, etc. However, this method was also fraught with the same weaknesses as the previous method. The false positives and false negatives obtained severely pulled down the accuracy of this method.

6.3 Using Global Feature Descriptors

The idea is to calculate global feature descriptors such as Gabor filter coefficients [3], Histogram of Gradients [5], cascaded features [18], and classify each image based on the feature values calculated. While these are good features in themselves, they were not suitable for the problem at hand. Since global features describe the entire image, they do not discern small changes in an image, such as a small packet on a casket. The difference in the values found for an empty casket, and a casket with a packet on it are significant to distinguish such cases. But the difference in values for an empty casket and a casket with a small packet on it would not be enough to distinguish them.

It is also to be noted that sufficient leeway needs to be made to account for variations on an empty casket itself. In an industrial setting, wear and tear of a casket is not only expected but assumed. Bar codes from packets could get stuck on a casket, the casket's belt might tear while sliding a packet onto it or off it. In order to account for these changes, the thresholds over the features calculated from every image are expanded. In doing so, when working with global feature descriptors, it is not possible to reliably distinguish empty caskets from caskets with small packets on them.

6.4 Using Edges

It is observed that the most useful property to distinguish an image of an empty casket from that of a casket with a packet on it is the collection of edges on the image. Wherever there is a packet on the casket, that area would be surrounded by stronger edges, while an empty casket would have very weak edges on it. As shall be seen later, this property is used to our advantage in designing an "Empty Casket Detection System".

Owing to a wide margin of pixel values of an empty casket, as well as the requirement for a packet being of any shape, size and color, the edges determined by us need to be robust to this variability. Simple edges cannot take care of all of these cases, or the edge cases arising out of them. So, some morphological operations have been combined with sharpening and thresholding operations before suitable edges are calculated.

6.5 Using Blobs

In this method, the edges from an image are filled using a blob-filling algorithm, some morpholigical operations are applied, and a threshold is applied on the subsequent binary image. This method gives 100% accuracy, clearly classifying caskets with small packets as "non-empty" but noisy caskets as "empty".

However, this method takes longer time than that required by the system. Although the mean time taken by the algorithm was well within limits, the maximum possible time taken was much over the limit. Working in an industrial environment, it was decided that we could not take the risk of employing this method, lest the time taken exceeds the maximum time within which it needs to perform.

Table 1 lists the mean and maximum times taken by three approaches. The time has been computed on an Intel NUC with a core i3 processor and a 4GB RAM.

Table 1: Time and accuracy of different features

Approach	Mean Time	Maximum Time	Accuracy
Entropy	0.2 ms	1 ms	70%
Edges	0.4 us	2 ms	100%
Blobs	4 ms	15 ms	100%

7 EMPTY CASKET DETECTION ALGORITHM

As mentioned previously, our approach to detecting an empty casket is to make a model of an empty casket, and compare every new image to this model. The probability of the new image fitting this model is calculated. This probability is used to determine whether the new image is that of an empty casket or not. Once this binary decision has been captured for multiple images of a casket, a vote is taken on the decisions of those images belonging to the central portion of the casket.

Using majority voting methods described in [7, 10], if sufficient number of images in the central portion are determined to belong to those of an empty casket, then the casket is deemed to be empty. However, if only a few images within the central portion are probabilistically empty, then the casket is deemed to be not empty. How many images are sufficient to deem a casket to be empty is a point of contention, and can be empirically fixed for a particular speed of the running belt.

As the conveyor belt is running, whenever an empty portion of a casket passes in front of the camera, the output of this program shall be 1. In the other cases, viz. when the area between caskets is in front of the camera, or when a casket with a packet on it is in front of the camera, the program shall output 0.

Since the height of the image captured is reduced from the full height of a casket, there shall be multiple images pertaining to the casket passing in front of the camera (indeed, of any portion of the belt). Thus, in case of an empty casket, there shall be multiple frames whose output is 0, corresponding to the *between_casket* region preceding this casket, followed by a series of 1's corresponding to the empty portion of the *casket* region, followed again by a series of 0's corresponding to the *between_casket* region after the casket.

So, by checking the output of the program for the central portion of the casket, it can be determined whether that casket is empty or whether it carries a packet on it, based on a majority vote.

7.1 Model

Our model of an empty casket consists of the intensity values of such an image, and also the edge values within that image. Strictly speaking, the model consists of features calculated from the image. In our case, the features are the intensity value and the edge value at every pixel. It is observed that calculating more complex features would increase the time taken to take a decision on each image. So, a combination of these simple features has been considered for our model.

It is assumed that the intensity and edge values of an empty casket are drawn from a Gaussian distribution. This assumption is especially true of a down-scaled image of an empty casket. Figure 3 shows an example of the histogram of values over the range 0-255 at a pixel, across images of empty caskets, and the Gaussian distribution made via the mean and standard deviations of the values. In the example, while values mostly lie near 0 (since the casket is empty), the intensity values very accurately follow a Gaussian distribution.

Hence, the recordings of multiple images of an empty casket are slightly down-scaled in size (effectively blurred), and their edge values are determined. Then, the mean and standard deviation of these values at every pixel are determined. This forms our model - the mean and standard deviation of the intensity and edge values at every pixel.

7.2 Comparison with model

When running, the camera captures an image of the portion of the casket in front of it. This image is down-scaled in size, and its edge values are determined. For every pixel, these values of intensity and edge are compared with the respective mean and standard deviation values in the model at the same pixel. Z-scores are thus calculated at every pixel using equations (1) and (2).

$$z_i = (x_i - \mu_i)/\sigma_i \tag{1}$$

$$z_e = (x_e - \mu_e)/\sigma_e \tag{2}$$

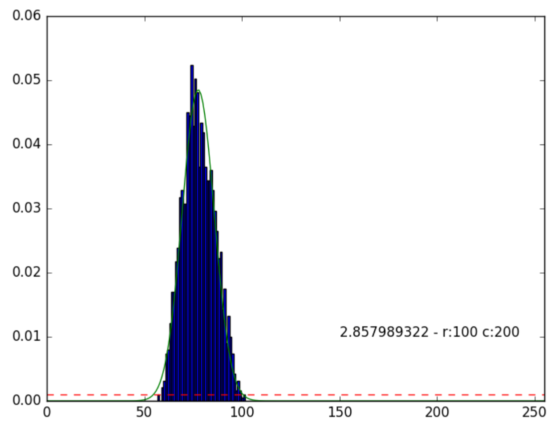

Figure 3: Distribution of values of one pixel across images of an empty casket

Our model is the mean intensity value μ_i, corresponding standard deviation σ_i, mean edge value μ_e, and corresponding standard deviation σ_e, for every pixel. The z-score thus calculated for a pixel represents how likely that pixel belongs to the model.

If both the z-score values for a pixel of a new image lie within a threshold, then that pixel is said to belong to the model, i.e. that of an empty casket. After determining whether each pixel in the new image belongs to the model or not, the number of pixels that belong to the model are counted. If this number is above a threshold, the image is determined to be that of an empty casket.

7.3 Thresholds

In our experiments, a good threshold for z-score was determined to be 2.7. This value was arrived at after several experiments with fitting the training set itself with the model made from it. In effect, we are assuming that a pixel shall only be considered to be from an empty casket if its intensity and edge values lie within $\mu - 2.7\sigma$ and $\mu + 2.7\sigma$, where μ and σ values have been recorded in our model.

It is to be noted that in a real-world scenario, it is not necessary for every pixel in an image of an empty casket to have its intensity and edge values to lie within this range. This is because our model is only limited to representing our training set of *empty_casket* images, while a new image of an empty casket could have some pixels beyond the z-score range determined by the model.

This is not a hindrance in determining whether the overall image is that of an empty casket or not, since we then set a threshold on the number of pixels not belonging to the model. This threshold is set so as to account for changes in the shape and structure of a casket. Thus, even though our model was built from a limited set of images, we have generalized our method to capture all sorts of changes in our target set of images of an empty casket.

As seen before, a similar strategy was used to determine whether a casket is empty or not, from decisions of individual images of the same casket determined to be those of an empty casket or not.

8 EDGE CASES

8.1 Packets with color similar to casket

The method above could fail in those cases where the packet has a color very similar to that of the casket. As described above, a lot of care has been taken in determining such edges that are robust to this kind of failure case. It has been observed that caskets with packets whose cross-sectional area exceeds $10cm \times 10cm$ under a casket area of around $100cm \times 70cm$, are determined to be non-empty in 100% of the cases. The cross-sectional area mentioned are standard minimum industrial packet sizes.

8.2 Small packets

Another factor that might lead to the failure of our method is the size of the packet. Caskets with packets with cross-sectional area below $10cm \times 10cm$ are not guaranteed to be classified as non-empty caskets. This is because certain leeway has been set in the form of thresholds to account for variability in the shape and structure of the caskets. But small packets, especially when placed on the edges of caskets, could lead to calculated parameters that lie within the thresholds, classifying the casket as empty. To tackle this problem, more complex features need to be calculated, which might not be able to satisfy the real-time requirement of the problem.

9 CONCLUSION

Through excessive experimentation with color, shape, size of packets, we achieve 100% classification of whether a casket is empty or whether it carries a packet, in real time, under the constraints of a warehouse. We achieved such a high level of classification using simple image processing algorithms by performing the classification over multiple frames and taking a majority vote. In this paper, we discussed the details of implementing a computer vision module to do this, including the image processing algorithm used, and the comparison to the model of an empty casket. We also discussed all the edge cases of our method.

10 FUTURE SCOPE OF WORK

As seen before, better methods of detecting packets on caskets are available (using blobs, at the very least), but timing has been a constraint. It is possible to circumvent the problem of time by performing the required operations using a Graphical Processing Unit (GPU). Since GPUs perform tasks in parallel, all image processing operations shall be completed within a fraction of the time currently taken, which shall then translate to better prediction with minimal errors.

Having sped up the image processing part of the algorithm, it is possible to then use a camera with a higher frame rate to capture even more images in the same time. This shall help make a highly robust decision about whether a casket is empty or not. In addition, a depth camera could be used to detect whether a packet is lying on the casket or not. This shall solve other extraneous problems, such as those of classifying a noisy casket as empty. It is to be noted that thinner packets that do not cross the minimum resolution of the depth sensor used shall not be detected using depth cameras.

ACKNOWLEDGMENT

The authors would like to thank Gaurav Kejriwal and Srijib Maiti from GreyOrange India, an Indian company that builds robotic systems for warehouse automation, for the opportunity to collaborate with them on this work.

REFERENCES

[1] I. Sutskever A. Krizhevsky and G. E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *In NIPS*. 1106–1114.

[2] A. Hoogs G. Brooksby W. Hu A. Perera, C. Srinivas. 2006. Multi-object tracking through simultaneous long occlusions and split-merge conditions. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (2006)*.

[3] M. Bastiaans. 1980. Gabors expansion of a signal into gaussian elementary signals. In *Proceedings of the IEEE*, Vol. 68. Issue 4.

[4] A. Toshev D. Erhan, C. Szegedy and D. Anguelov. 2014. Scalable object detection using deep neural networks. In *In CVPR*.

[5] N. Dalal and B. Triggs. 2005. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition*. IEEE Computer Society Conference on. Vol. 1. IEEE,.

[6] J. Ponce et al. 2006. Toward Category-Level Object Recognition. *LNCS* 4170 (2006), 49–64.

[7] Y. Freund. [n. d.]. Boosting a weak learning algorithm by majority. *Information and Computation* 121 ([n. d.]). Issue 2.

[8] H. Li J. Su H. Zhu, J. Zhou and X. Li. 2002. D barcode preprocessing scheme based on image recognition. In *Proc. SPIE 4875, Second International Conference on Image and Graphics*, Vol. 651.

[9] Z. Xie J. Gao and X. Wu. [n. d.]. Generic object recognition with regional statistical models and layer joint boosting. *Pattern Recognition Letters* 28 ([n. d.]). Issue 16.

[10] G. James. 1998. Majority vote classifiers: theory and applications. *PhD thesis, Department of Statistics - Stanford University, Stanford, CA* (1998).

[11] L. Juan and O. Gwun. [n. d.]. A Comparison of SIFT, PCA-SIFT and SURF. *International Journal of Image Processing* 3 ([n. d.]). Issue 4.

[12] S. Ren K. He, X. Zhang and J. Sun. 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *In ECCV*.

[13] Y. Ke and R. Sukthankar. 2004. PCA-SIFT: A More Distinctive Representation for Local Image Descriptors. In *Proc. Conf. Computer Vision and Pattern Recognition*. 511–517.

[14] D. G. Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision* (2004).

[15] A. Zisserman M. Varma. [n. d.]. Classifying images of materials: Achieving viewpoint and illumination independence. In *In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS*, Vol. 2352. 255–271.

[16] K. Mikolajczyk and C. Schmid. 2005. A performance evaluation of local descriptors. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27. 1615–1630. Issue 10.

[17] J. M. Morel and G. Yu. [n. d.]. ASIFT: A New Framework for Fully Affine Invariant Image Comparison. *SIAM Journal on Imaging Sciences* 2 ([n. d.]). Issue 2.

[18] M. Jones P. Viola. 2001. Rapid object detection using a boosted cascade of simple features. In *Proceedings of IEEE Int. Conf. on Computer Vision and Pattern Recognition*. 511–518.

[19] P. Perona R. Fergus, A. Zisserman. 2003. Object class recognition by unsupervised scale-invariant learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2003)*.

[20] X. Ren and J. Malik. 2003. Learning a classification model for segmentation. In *Proc. IEEE International Conference on Computer Vision (2003)*.

[21] K. Simonyan and A. Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *In ICLR*.

[22] D. Ramanan X. Zhu, C. Vondrick and C. Fowlkes. 2012. Do we need more training data or better models for object detection?. In *In BMVC*.