ICMIEE22-091

Indoor water on floor detection using monocular camera based on self-supervised segmentation

Mohammad Khairul Hasan[1,*]
[1] R&D Center, Duksan Mecasys, Daejeon- 34139, REPUBLIC OF KOREA

ABSTRACT
In this paper, we consider the problem of detecting wet area on the floor inside a room using a monocular camera mounted near the ceiling. We use a set of Gabor filters on a floor image and apply nonlinear transformations on the filtered images in order to create feature images. Based on the feature images we create a feature vector for each pixel in the input image. Then we use *K*-mean clustering with *K*=2 on the normalized feature vectors and process the resulting clusters to create wet floor segmentations. Our experimental results show that more than 50 percent of wet area of the entire floor can be detected given that the camera has a limited amount of motion (e.g., pan and tilt motions). This problem appears as a sub-problem in many unmanned monitoring systems. Our method results a fast, low-cost and robust solution for this problem.

Keywords: water detection, Gabor filter, K-mean clustering, unmanned monitoring system.

1. Introduction

We consider the problem of detecting water on floor in an indoor environment using a monocular camera mounted near the ceiling of a room where the camera can be moved in pan and tilt directions to cover the entire floor area of the room. We assume that the system is unmanned, and the main objective of this system is to raise alarm at an early stage of possible flooding. Indoor water detection from color images is a difficult task. The main visible changes due to a layer of water on a floor are (a) some portion of the wet floor reflects light beam in a particular direction (specular reflection) and as a result that portion looks highly bright if the camera is placed at the direction of the reflected light beam and (b) if the camera does not receive reflected light beam from a wet surface, then that surface appears slightly darker as compared to neighboring dry surface. Since our objective is to detect water at a very early stage, we cannot wait for the specular reflection which may or may not come to the camera before water crosses dangerous level. In this paper, we propose a robust solution that uses the fact that water presence changes the texture appearance of the dry floor. Our algorithm in general does not depend on any specular reflection from the water covered surface. Since it is difficult to gather huge amount of image data consisting of water covered floor, we have used a self-supervised approach.

2. Related Works

Teshima *et al.* worked on a similar problem. They use a moving camera to detect water based on the specular reflection reflected from wet surface [1]. Our case is different because we use a camera mounted in a fixed place with little amount of motion in pan and tilt directions. Since our system covers a wide area of floor we cannot depend only on specular reflection. Achar *et al.* proposed a self-supervised algorithm for outdoor river scene detection [2]. Our problem is very different from that of Achar *et al.* Their work separates the river surface from the surrounding environment in an outdoor environment whereas in our case, we need to separate water covered surface from dry floor surface in an indoor environment. Rankin and Matthies presented a daytime water detection solution using color variation [3]. Iqbal *et al.* published a nice review paper that gives a survey on outdoor water hazard detections [4]. Santana *et al.* proposed a water detection technique based on water motion in an outdoor environment [5].

To the best of our knowledge, there is no previous solution for this problem under our settings. In fact, this problem is a sub-problem of a commercial application that we are currently working on. We searched thoroughly but failed to get a suitable solution. The closest solution has been given by Teshima *et al.* [1]. However, their solution uses a moving camera which is not feasible for our target area (establishment of the mechanism for moving a camera just below the ceiling of a big room is expensive). For completeness, the summary of our results for 16 images and the results of Teshima *et al.* have been shown in Table 1.

Table 1 Comparison of our results with results of Teshima *et al.*

	Precision rate (%)			Recall rate (%)		
Results of Teshima *et al.* [1]	81.1			34.1		
Our Results	Max	Min	Avg	Max	Min	Avg
	100	0	82.7	100	0	86.2

3. Working Environment

The main objective of our work is to detect water layer on the floor area in a room. We assume that our target environment has these following properties:

1. The environment is a large room with no window.
2. There are just a few light sources on the ceiling and a camera is mounted in a corner of the room near the ceiling.

* Corresponding author. Tel.: +82-10-3463-1978
E-mail addresses: hasan@dsmecasys.com

3. The camera can be moved 120 deg in the pan direction and 90 deg in the tilt direction, where zero-degree tilt means that the camera is directed to the horizontal direction and 90-degree tilt means that the camera is directed to the downward direction.
4. The camera has a high focal length (more than 70 mm) and as a result, from most of the directions the images contain only floor area.
5. For some directions images contain the wall or some other furniture but using suitable segmentation algorithm we can easily separate the floor part from the adjacent wall or furniture parts. More precisely, since we know the pan and tilt angle associated to each image, we can determine (using some pre-processing technique) which part of that image contains floor area.
6. The working area of our algorithm is not necessarily of rectangular shape. Although each image is rectangular, we can use suitable masking to work on almost any shape as we like.
7. We split each input image into non overlapping sub images of size 640×480 each and ignore the remaining part (input image size is 6720×4480). We call each sub-image a zone image or a zone in short. Our algorithm works on all the zones one by one.
8. We assume that at the beginning, the floor corresponding to each zone is completely dry and at that time the algorithm captures the information related to the dry floor and use this information later to detect water presence (hence it is self-supervised). Since in the room there is no window, we assume that Lambertian component of dry floor surface radiance is constant [6] and the specular component of dry floor surface radiance is negligible as compared to the specular component of water covered floor surface radiance.
9. We assume that if floor is covered by a layer of water surface, then a major portion of the specular reflection of each light source goes to a particular direction. If this direction hits the camera sensor, the image contains much bright area (see Fig.1). This area can easily be detected in the image by using proper thresholding. However, since the specular reflection depends heavily on the surface normal, corresponding light source position, camera position and camera orientation, only a small portion of the entire floor area may reflect a light beam to the camera as specular reflection. After thresholding, we consider each connected component of specular reflection one by one and categorize into two groups (a) slim: the component does not contain a circle of radius ρ and (b) fat: the component contains a circle of radius ρ, where ρ is a parameter, whose value is decided empirically. We consider that each fat component contains water layer, and each slim component is either a noise or a water droplet. We compute the area of each fat component and remove them from the image by masking and apply our algorithm on the remaining part of the image. We can combine water layer under each fat component with the result returned from our algorithm and compute approximately the amount of water in each zone. So, from now on, we consider that each image does not contain any fat component.

Fig.1 Specular reflection from water surface. The red rectangle contains a fat component, and the green rectangle contains a slim component

4. Outline of Algorithm

We implement our algorithm on each zone independently. In this section we will explain the overall algorithm on a particular zone which is a grayscale image.

Assuming that the values of D, d, and δ will be decided later, the high-level idea of our algorithm applied on each zone is given as follows:

1. In the preprocessing step, we collect $D\times D$ pixel values from around the center of the zone. In this step, the whole zone is dry. Let *D-dry* be the set of $D\times D$ pixels. For each zone we save corresponding *D-dry* as a bitmap image.
2. We apply the algorithm explained in the next section to create a feature vector for each pixel in the zone.
3. Using K-mean (K=2) Clustering algorithm we cluster the pixels into two groups group-A and group-B. Let $\{A_1, A_2,\ldots A_n\}$ and $\{B_1, B_2,\ldots B_m\}$ be sets of connected components of group-A and group-B respectively, where each component contains a $d\times d$ square inside it (we ignore any component that is too small to contain a $d\times d$ square). Let us assume that

$A_1, A_2, \ldots, A_n$ and $B_1, B_2, \ldots, B_m$ are sorted according to their sizes in non-increasing order.

4. Let A_1^d and B_1^d be two $d \times d$ boxes inside A_1 and B_1 found by detecting the largest circle inside A_1 and B_1 respectively. We can use algorithms explained in [7] to get these boxes. Let $d_a^1, d_a^2, d_a^3 \ldots$ are Mahalanobis distances from each element of A_1^d to the set *D-dry* and let a_d-*dry* be the average of these distances. Similarly, we calculate b_d-*dry* for set B_1^d. If $A_1(B_1)$ is empty, then let us consider that a_d-*dry*(b_d-*dry*) is zero. Without loss of generality let us assume that a_d-*dry* <= b_d-*dry*. Depending on the values of a_d-*dry* and b_d-*dry* we follow the following rules:
    a. Both a_d-*dry* < δ and b_d-*dry* < δ: the whole zone is dry
    b. Both a_d-*dry* >= δ and b_d-*dry* >= δ: the whole zone is covered by water
    c. a_d-*dry* < δ and b_d-*dry* >= δ: first initialize W = $\{B_1\}$ and then for $B_2, \ldots, B_m$ we do the following - for each B_i $2 <= i <= m$, we find a $d \times d$ box inside B_i and find b_i-*dry* as the average of Mahalanobis distances from elements in this $d \times d$ box to *D-dry*. If b_i-*dry* >= δ, we update W ← W∪$\{B_i\}$. Finally, we consider that the components of W indicate the wet segments in the zone.

5. Implementation Detail

The backbone our algorithm is an unsupervised texture segmentation which follows closely the idea given in [8]. We capture the features from the zone image through a set of filters, keep a subset of these filtered images based on some greedy algorithm and then apply a nonlinear function on these filtered images. Finally, we apply *K*-mean Clustering for *K*=2 and apply the algorithm explained in the previous section to detect water on floor.

We use Gabor filter to get filtered images. A Gabor filter in spatial domain can be given by:

$$t = g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi \frac{x'}{\lambda} + \psi\right)$$

with

$$x' = x\cos\theta + y\sin\theta \text{ and } y' = -x\sin\theta + y\cos\theta$$

where λ is the wavelength of the sinusoidal factor, θ represents the orientation, ψ is the phase offset, σ is the standard deviation and Υ is the spatial aspect ratio.

Each zone has dimension (640×480). We applied 28 filters for each θ in {0, 45, 90, 135} and λ = {image-width/radial-frequency} where radial-frequency ∈ {4√2, 8√2, 16√2, 32√2, 64√2, 128√2, (image-width/4)√2}. For each filter we use a kernel of size = 17 with ψ = 1, σ = 7 and ψ = 0. In a similar way as explained in [8], we use following steps in order to get a feature vector for each pixel in the zone which is a grayscale image.

1. We apply the above-mentioned Gabor filters on the zone image to get a set of filtered images.
2. We select a subset of filtered images in a greedy way so that the following condition holds: Let $s(x,y)$ and $s^1(x,y)$ be reconstruction of the zone images by adding all filtered image and the selected subset of filtered images respectively. Then,

$$SSE = \sum_{x,y} [s^1(x,y) - s(x,y)]^2$$

$$SSTOZT = \sum_{x,y} s(x,y)^2$$

$$R^2 = 1 - \frac{SSE}{SSTOT} \geq 0.95$$

   We select filtered images one by one in a greedy way until R^2 becomes at least 0.95. The greedy algorithm is same to the corresponding algorithm explained in [8] so we skip the detail here.
3. We apply a nonlinear transformation to each selected filtered image. More precisely, for each selected filtered image, we apply normalization so that the minimum and maximum are -8 and 8 respectively. On each pixel of the filtered image, we apply the following activation function and take the absolute value with α = 0.25:

$$\varphi(t) = \tanh(\alpha t) = \frac{1 - e^{\alpha t}}{1 + e^{\alpha t}}$$

   We then apply a gaussian filter of size 35×35 with σ = 0.5×image-width/radial-frequency and call the resulting image a feature image.
4. For each pixel (*row*, *col*) of the zone image, we create a feature vector like this: first we take pixel values of all feature images at position (*row*, *col*) and combine them to create a feature vector and then we insert *row*, and *col* at the top of this feature vector. These *row* and *col* values in each feature vector help to combine neighboring pixels while clustering.
5. Finally, we normalize the feature vectors across all but top two dimensions (*row*, and *col*). This step is not that crucial. It just helps the *K*-mean Clustering algorithm work better.

For the entire algorithm explained in the previous section and in this section, we choose $D = 200$, $d = 50$, $\rho = 30$. The value of threshold δ has been decided empirically as 2.8. This threshold depends mainly on the dry floor texture and can be decided in the preprocessing stage.

6. Experiments and Results

We apply our algorithm on 16 zone-images with size of 640×480 each, where 6 of them are completely dry and remaining are partially wet. Fig. 2 shows a zone image and one filter and two feature images of this zone image. One can observe that although wet area is not that distinctively visible in the original image, that wet area is quite visible (as bright part) in one of the feature images.

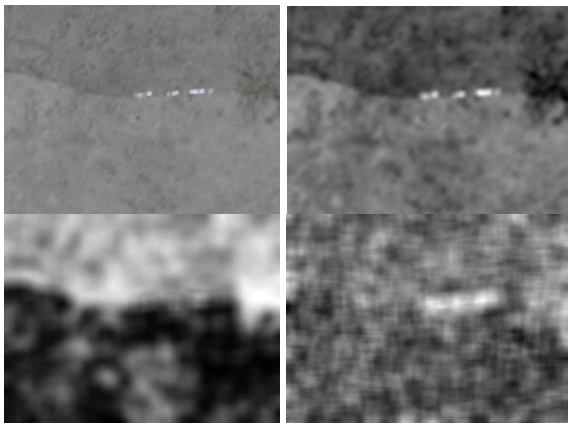

Fig.2 Top left: zone image, Top right: filtered image, Bottom: two feature images

For each zone, first we compute the segmentation that gives dry and wet floor areas. And then we compute intersection over union IoU score for the wet area of our segmentation result and ground truth. Our results show that 4 zones give score over 90% each, 2 zones give score over 60% each 1 zone gives score over 50% and all 6 dry floor zones has been classified successfully. One false negative result (result shows dry but there is water on the floor) has been found and one result shows much more water than the actual amount of water (score = 19%). Fig.3 shows two samples where water has been detected with sufficient accuracies (94% IoU and 96% IoU respectively). Fig. 4 shows the sample where accuracy is very low (19% IoU) and the sample that gives false negative result. Let W_s and W_g be water area from our algorithm and from ground truth respectively. In Fig. 3 and Fig. 4, yellow part is $W_s \cap W_g$, red part is $W_g - (W_s \cap W_g)$ and green part is $W_s - (W_s \cap W_g)$. For each zone we also compute Precision and Recall rates. Detailed results of our experiments on 16 images are shown in Table 2.

It should be noted that in both the cases in Fig. 4, the amount of water is very low. Although both have similar amount of water, in the false negative case the floor is bit darker than that of low accuracy case. As a result, in the false negative case, the algorithm cannot differentiate the dry floor part from the wet floor part. These two zone images have been shown in Fig. 5.

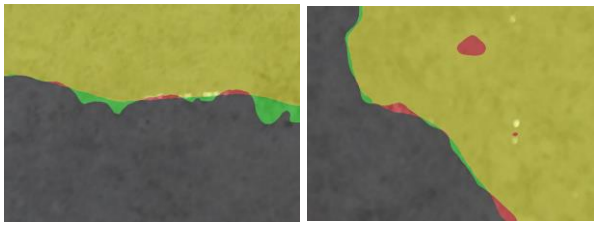

Fig.3 Overlapping of segmentation results and ground truths for two zones

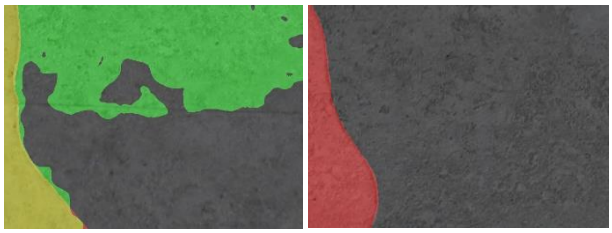

Fig.4 Left: segmentation result accuracy is very low (19 %), Right: false negative result.

Fig.5 Left: zone image for the case with low accuracy (19 %), Right: zone image for the false negative case.

Table 2 Detailed results of our experiments

	IoU	Precision (%)	Recall (%)	Remark
Image 01	0.941	94.9	99.1	
Image 02	0.964	98.7	97.7	
Image 03	0.773	77.3	100	
Image 04	0.696	70	99.1	
Image 05	0.917	99.8	91.9	
Image 06	0.938	99.8	93.9	
Image 07	0.54	97	54.9	
Image 08	1	100	100	Dry floor
Image 09	1	100	100	Dry floor
Image 10	1	100	100	Dry floor
Image 11	1	100	100	Dry floor
Image 12	1	100	100	Dry floor
Image 13	1	100	100	Dry floor
Image 14	0.19	19.1	99.4	
Image 15	0.355	66.5	43.3	
Image 16	0	0	0	False negative
Average	0.77	82.7	86.2	

c

5. Discussion

Water detection on floor is a difficult task specially when the camera is monocular, and the position of light sources are such that light beam after specular reflection does not go to the camera sensor directly. We present a self-supervised and efficient algorithm to detect water on floor. Our algorithm gives good results (in terms of IoU, Precision rate and Recall rate) for most of the cases. In the future it would be interesting to improve the performance of this algorithm.

8. References

[1] Teshima, T., Saito, H., Shimizu, M., & Taguchi, A. (2009). Classification of Wet/Dry Area Based on the Mahalanobis Distance of Feature from Time Space Image Analysis. In MVA (pp. 467–470).

[2] Achar, S., Sankaran, B., Nuske, S., Scherer, S., & Singh, S. (2011). Self-supervised segmentation of river scenes. In 2011 IEEE International Conference on Robotics and Automation (pp. 6227-6232).

[3] Rankin, A., & Matthies, L. (2010). Daytime water detection based on color variation. In 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (pp. 215–221).

[4] Iqbal, M., Morel, M., & Meriaudeau, F. (2009). A survey on outdoor water hazard detection. Skripsi Program Studi Siste Informasi..

[5] Santana, P., Mendonça, R., & Barata, J. (2012). Water detection with segmentation guided dynamic texture recognition. In 2012 IEEE international conference on robotics and biomimetics (ROBIO) (pp. 1836–1841).

[6] Ikeuchi, K., & Sato, K. (1991). Determining reflectance properties of an object using range and brightness images. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(11), 1139-1153.

[7] R. C. Gonzalez and R. E. Woods. Digital Image Processing. Prentice-Hall, 2002

[8] Jain, A., & Farrokhnia, F. (1991). Unsupervised texture segmentation using Gabor filters. Pattern recognition, 24(12), 1167–1186.