Image Segmentation of Bricks in Masonry Wall Using a Fusion of Machine Learning Algorithms

Kajatin, Roland ; Nalpantidis, Lazaros

Published in:
Proceedings of ICPR 2020 workshop on Pattern Recognition in Construction and the Built Environment

Link to article, DOI:
10.1007/978-3-030-68787-8_33

Publication date:
2021

Document Version
Peer reviewed version

Image Segmentation of Bricks in Masonry Wall Using a Fusion of Machine Learning Algorithms

Roland Kajatin[1,2] and Lazaros Nalpantidis[1,3]

[1] Department of Electrical Engineering, Technical University of Denmark, Elektrovej 326, 2800 Kgs. Lyngby, Denmark
[2] roland.kajatin@gmail.com
[3] lanalpa@elektro.dtu.dk

Abstract. Autonomous mortar raking requires a computer vision system which is able to provide accurate segmentation masks of close-range images of brick walls. The goal is to detect and ultimately remove the mortar, leaving the bricks intact, thus automating this construction-related task. This paper proposes such a vision system based on the combination of machine learning algorithms. The proposed system fuses the individual segmentation outputs of eight classifiers by means of a weighted voting scheme and then performing a threshold operation to generate the final binary segmentation. A novel feature of this approach is the fusion of several segmentations using a low-cost commercial off-the-shelf hardware setup. The close-range brick wall segmentation capabilities of the system are demonstrated on a total of about 9 million data points.

Keywords: Image segmentation · Construction robotics · Machine learning · Deep learning.

1 Introduction

The work presented in this paper focuses on the construction industry domain, the automation of which has gained an increased attention in recent years. The application of computer vision systems at different phases of the lifecycle of civil assets were analyzed in [26]. Such systems are applied during construction, operation, as well as maintenance, and perform operations such as defect detection and condition assessment [26, 13]. From a safety perspective, [14] analyzed the feasibility of a drone based hazard detection system at construction sites. The application of augmented reality in the construction industry is yet another field of interest [25, 27]. Building information modeling has also gained new advances in recent years [15, 2, 1].

This paper focuses on the renovation aspect of the construction industry; more specifically on the renovation of brick walls. During construction of the wall, mortar is used to bind bricks together. Over time, the integrity of the wall might become compromised due to the degradation of the mortar (and bricks). This paper proposes a computer vision system for masonry wall segmentation in

Fig. 1: An example setup of the autonomous mortar raking robot (designed by Robot At Work [18]). It is composed of a modular rail system to which electric motors are attached (blue housing in the image). The end tool is a milling device used to remove the mortar from between the bricks.

order to enable a robotic platform—affixed to the surface of the wall, as shown in Fig. 1—to automatically mill away the mortar from between the bricks without damaging them. By replacing the old mortar with fresh one the integrity of the wall is enhanced, which prolongs the lifetime of the building.

Without the robot, this operation would be carried out by a craftsman who would hold the tool in hand and progressively mill away the mortar. This operation requires the construction of some kind of scaffolding, ensuring the safety of the operator. Furthermore, this process takes time and requires the operator to pay close attention to the edges of the bricks. During the renovation process, it is imperative that only the mortar is milled and the bricks are left unscathed. All these constraints put a burden on the operator. Furthermore, according to [4], prolonged exposure to such mechanical vibrations may lead to disorders in the vascular and neurological systems of the upper limb, called hand-arm vibration syndrome, posing a health risk to operators.

The robotic solution mitigates all of the above problems and harmful effects towards the operators. However, it now becomes a challenge for the robot to *know* where the mortar is. The brick wall has presumably deformed over the years, or was inherently unstructured—depending on the type and uniformity of bricks used. This makes the mortar detection a more general problem. Therefore, the contribution of this paper to enable a robotic platform to perform autonomous mortar raking is summarized as follows:

1. First, a number of image segmentation algorithms are considered and evaluated for the segmentation of close-range images of brick walls using color information and depth from stereo vision.

2. Secondly, a process to fuse the individual segmentation masks is proposed based on the weighted sum and empirically determined threshold operation of said masks.

2 Related Work

Although the application of computer vision systems in the construction industry in general is abundant, the literature on the specific application of masonry (brick) wall segmentation is scarce. Some applications employ laser technology (such as terrestrial laser scanners) to sample the environment [23, 22, 19], while others rely on 24-bit color images [11, 16].

Valero et al. [23] present a technology for automatic fault detection in ashlar masonry walls. They argue that surveying of buildings for condition done by human experts is often unreliable and dependent on the expertise and experience of the operators. Therefore, their proposal uses terrestrial laser scan (TLS) data (precise 3D point cloud), contrary to the our approach, together with color information gathered by a DSLR camera. They overlay the color information by generating a color-cloud using structure from motion. Their segmentation of the masonry wall is based on the 2D continuous wavelet transform documented in [22], contrary to our machine learning based classifiers.

Riveiro et al. propose an algorithm for processing laser scanning data using an improved, marker-controlled watershed segmentation in [19]. They used a Riegl terrestrial laser scanner in their case study to sample a point cloud, contrary to our stereo vision based depth calculation. They reduce the dimensionality of the data by using a raster model to lattice the (planar) point cloud of the masonry wall. They fit a plane to the point cloud, and then take the orthogonal projection for each lattice to form a 2D pixel. The intensity value of each pixel is dependent on all the 3D points in the cloud inside the corresponding lattice. While our approach uses ML classifiers, their segmentation of the bricks is done using markers to perform the watershed based segmentation. The markers are derived from geometrical constraints under the assumption that the bricks follow a horizontal course. Using image gradients, the horizontal and vertical mortar channels are found, and connected to form a rough segmentation of the bricks – they call it a wireframe. The inner areas of the found segments in the wireframe serve as the markers for the watershed algorithm, which provides a more accurate segmentation along the edges.

In [16], Oses et al. proposes an automatic, image-based delineation method used in connection with built heritage assessment. Their goal is to assist and speed up the process of determining the degree of protection required for a certain built heritage. Their method converts the input image into a single-channel grayscale one, and then slices it into a number of smaller region of interests (ROIs). These cover the entire image, but are processed independently. Outliers (i.e. intensity values with a frequency lower than a certain threshold) are removed by inpainting. For each ROI, three modes (i.e. intensity values) are selected based on the histogram of the image. Using an arbitrary neighborhood

around these modes, the image is binarized. The blobs in the three binary image are thinned by removing their inner parts. Afterwards, they use the probabilistic Hough transform to detect straight lines in the three images. The detected lines are then fused: colinear lines that are close to each other are connected, and short ones are removed. While our approach uses the color- and depth information to train the models, they construct a series of features using the line segments, which are used to train several classifiers (e.g. SVM, naive Bayes, decision trees, etc.). These classifiers then provide the final segmentation of the bricks.

Similarly to our system, Ibrahim et al. present a machine learning based segmentation algorithm in [11], however, they only use a single model. Their goal is to segment individual brick instances in a 2D image input, both for modern brick walls as well as ancient archaeological sites. Their method can be divided into two steps. During the first step, a convolutional neural network (UNet) is used to generate a grayscale prediction image from the input. Their network is trained on the RGB values of the input image. The second step aims to extract an accurate outline of the bricks using the delineation map obtained in the previous step by applying the inverted distance transform and the H-minima transform. Finally, the watershed algorithm is applied using the H-minima basins as markers for starting points. This last step is helpful in separating touching bricks from each other and correctly drawing an edge in between them. However, in our case, this is not a concern, since the close-range images of the brick wall do not contain touching bricks.

The work presented in this paper aims to add to the body of literature within brick wall segmentation to provide the necessary segmentation for autonomous mortar raking. Contrary to [23, 22, 19], our approach is based on color and depth information (from stereo vision) using low-cost imaging hardware. Similarly to [11, 16], we use machine learning models, however, we take the pixel-wise classification further and propose an algorithm to combine the segmentation masks of several models.

3 Proposed System

The proposed system focuses on providing an accurate segmentation of close-range images of masonry walls. The algorithm involves a number of steps outlined in Fig. 2. Contrary to the approaches of related work presented in Section 2, this system is designed to work with close-range (i.e. 100–250 mm from the wall) and cost-effective imaging of small areas of the wall at a time. Therefore, suitable hardware is selected (see Section 3.1) and the rest of the system is tailored to work with the available data (see Section 3.2).

3.1 Hardware

The selected hardware is the Intel RealSense D435 camera [12], which is a compact device ($90 \times 25 \times 25$ mm and 72 g) that houses a depth module and an extra RGB module. The depth module uses assisted stereo vision to calculate the

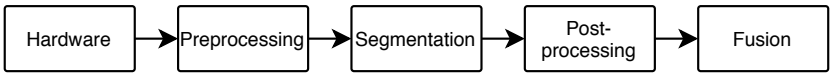

Fig. 2: Outline of the proposed system for the segmentation of close-range images of masonry walls.

depth. There are two grayscale imagers and an extra infrared projector which projects a static IR pattern to help aid the depth calculation in scenes with inadequate texture. Based on the characteristics of the device presented in [12] and visible dimensions of 54×228 mm of the bricks in the considered use cases, the D435 is able to capture an entire brick along the vertical and horizontal axes at a minimum distance of $r_{\min} = 67.14$ mm and $r_{\min} = 159.46$ mm respectively, using:

$$r_{\min} = d_{\text{target}} \frac{f}{d_{\text{CCD}}} \ , \tag{1}$$

where d_{target} denotes the dimensions of the brick, f is the focal length of the device, and d_{CCD} denotes the dimensions of the sensor. Furthermore, the single-pixel resolution is expected to be less than 1 mm up to a maximum distance of $r_{\max} = 197.7$ mm from the wall based on:

$$r_{\max} = d_{\text{target}} \frac{f}{p_{\text{res}} d_{\text{CCD}}} \ , \tag{2}$$

where $p_{\text{res}} = 3$ is the minimum distance between two centroids in pixels (i.e. the minimum distance required to have a well-defined separation between two points in the image).

Depth information is recovered from stereo vision using the epipolar geometry [9]. The two cameras are offset from each other by a baseline of $b = 50.27$ mm. The D435 uses the left imager as the reference. Since the left-most part of the left image is not seen by the right imager, the depth map contains a so-called invalid depth band on the left side. This is given in [12] as:

$$\frac{b}{2z \tan\left(\frac{\text{HFOV}}{2}\right)} \ , \tag{3}$$

where HFOV is the horizontal field of view of the imagers, and z is the distance from the wall. In the aforementioned working range of the device, (3) yields around 10% – 15% of the depth map being invalid. According to [9], the depth is recovered as:

$$z = \frac{bf}{x - x'} \ , \tag{4}$$

where the term $x - x'$ is called the disparity, and z is the depth value. As the depth is inversely proportional to the disparity, the theoretical minimum depth is bound by the maximum disparity value. By the introduction of an artificial disparity shift (i.e. an artificial decrease in the denominator in (4)), the range of the valid depth calculations can be modified. Using a shift of 86, the depth

range is moved to be within approximately 99.83 mm and 246.08 mm, which corresponds to the desired working range of the device.

3.2 Software

The acquired depth map and color image are further processed as outlined in Fig. 2.

Preprocessing The preprocessing steps convert the raw data from the device into the required format used by each algorithm (detailed in Section 3.2). There are four main groups of preprocessing methods used: filtering of the raw depth map, alignment of the the depth map with the color frame, color space conversion, and image transformations (e.g. resizing and slicing).

There are a number of filters used to make up for various shortcomings of the depth map, which in turn enhance its overall quality. The spatial noise is corrected with a filter that smooths out variations in the depth map while keeping the integrity of actual edges. This filter is based on a 1D exponential moving average (EMA) calculation controlled by an extra parameter to preserve the integrity of the edges in the depth map. The same filtering principle is also applied in the time domain across consecutive depth maps to reduce temporal noise, which manifests in small variations in the depth values for a static scene. Finally, missing depth values (i.e. holes) in the depth map are filled in using the neighboring left depth value.

Since both the color and the depth information is used by the segmentation algorithms, it is important that the depth values be aligned with the color pixels. The alignment is based on the inverse transformation of pixel coordinates to camera coordinates using the intrinsic parameters of the cameras. Then, the two frames are aligned using their measured extrinsic relation. Regarding the color frames, three input domains are considered for the algorithms: RGB, HSV, and a color and texture based [24] feature sets.

Segmentation Following the description in Section 1, the proposed system is concerned with the accurate segmentation of close-range images of brick walls. The task is interpreted as a pixel-wise classification problem using two distinct classes: mortar and brick.

In total, there are eight classifiers investigated, of which the first one is the k-nearest neighbor (kNN) algorithm [3]. This classifier assigns class allocation based on the majority-class of k nearest neighbors of the data point. The second classifier is the naive Bayes algorithm [28], which applies Bayes' theorem with a (naive) strong assumption of conditional independence between the features. Another classifier which uses Bayes' theorem is the quadratic discriminant analysis (QDA) [21], which applies a quadratic decision surface to discriminate the different classes. The fourth classifier is the support vector machine (SVM) [7], which separates the classes based on the maximum margin decision boundary solution. The maximum margin refers to the fact that the decision boundary is

selected such that it is as far away as possible from the classes. The fifth one is the decision tree classifier [6], which recursively partitions the input space and model each region locally. These regions are connected in a recursive manner. An extension of the decision tree classifiers is the random forest model [5], which fits a number of decision trees and combines their individual outputs using averaging, thus improving the predictive accuracy. Another ensemble classifier is the AdaBoost [10], which iteratively fits classifiers on the data, adjusting the weights of incorrectly classified points at each iteration (i.e. forcing the subsequent classifiers to focus more on difficult examples). Finally, the last classifier is the UNet deep learning model [20], which is a convolutional deep neural network that outputs pixel-wise classification.

Each of these classifiers produces a binary segmentation (i.e. pixel-wise classification) of each preprocessed (as outlined in Section 3.2) input image individually. During training, the outputs are used to adjust the parameters of the models, while during inference, the masks are post-processed (see Section 3.2) and then combined (see Section 3.2) to produce the final binary segmentation mask.

Post-processing There are two types of post-processing methods applied to each binary mask individually: morphological image processing and connected components analysis, both of which aim to reduce the noise in the output and produce homogeneous (complete) areas for the bricks.

The first method relies on applying morphological operations on the binary mask. The applied morphological operation is the closing operation, which performs a dilation and an erosion in this particular order on the input image. This post-processing step closes the holes inside the brick regions, as well as removes small brick-labeled objects from the frame (which are assumed to be noise due to their small pixel size).

The second method is based on the connected components analysis of the binary mask. Similarly to the morphology, the goal is to remove small patches of foreground objects under the assumption that the bricks are expected to occupy a large portion of the image. The labeling of distinct regions is based on a decision tree approach presented in [8]. The analysis uses 8-connectivity, and produces an output image where connected components are uniquely labeled. The size of each of these components is determined simply by the amount of pixels they occupy. If this area is smaller than a certain threshold, then the corresponding pixels are set to 0 (i.e. background) in the mask. Since the connected components analysis only considers foreground pixels, the same procedure is performed on the inverted mask as well. Although the morphological transformation fills in most of the holes inside the bricks, there could still be some left at this point. By inverting the image, these small holes are now labeled by this algorithm, and removed due to their area being below the threshold.

Fusion At the final step, the post-processed masks produced by all considered classifiers are combined to form the final segmentation. The combination logic is

based on a weighted voting scheme. First, each classification algorithm is assigned a specific weight based on their performance (see Section 4). The weights are assigned by taking the softmax function of the selected performance metrics (average of F1-score and accuracy) of the classifiers.

Let s_i denote the score of the i^{th} classifier, then the weights are given by the softmax function as

$$w_i = \frac{\exp(s_i)}{\sum_j \exp(s_j)} \ , \tag{5}$$

where the denominator is the sum of the exponentials of all the scores. Using the softmax function ensures that the weights sum up to 1, and that their relative importance is guided by the performance of the corresponding classifiers. The binary masks are multiplied by the corresponding weights and then summed up to generate a single mask. Then, a threshold operation is performed to binarize the combined mask and yield the final segmentation.

The value of the threshold is experimentally determined. If the threshold is too large, the final mask will have a high confidence for the labeled brick areas, but these will be smaller in size and less accurate around the edges. If the threshold is too small, then the output will have brick areas with less confidence around the edges. Therefore, it is important to select a threshold which retains the accuracy around the edges, but still has a high confidence on the output brick regions.

4 Experimental Evaluation

The evaluation of the proposed system outlined in Section 3 is first performed for each algorithm individually. Once these performance scores are evaluated, the weights are calculated as shown in (5), and the performance of the entire system, using the combined mask, is again evaluated.

The evaluation is based on standard classification metrics. The binary classification problem at hand produces either a positive or a negative output (depending on the class) for each pixel, which means there are four possible scenarios: true negative (TN), false negative (FN), true positive (TP), and false positive (FP). These outcomes are encapsulated into the following metrics which are used for the evaluation: recall, precision, specificity, negative predictive value, F1 score, and accuracy [17].

4.1 Dataset

In order to train and test the machine learning classifiers, a dataset is created in the following manner. The D435 is used to record four video streams of brick walls under various environmental conditions, but only for reddish bricks. In total, 27 static frames are extracted from both the color and the depth streams. The depth frame is preprocessed, according to the description in Section 3.2, therefore, it is only necessary to apply the same preprocessing steps during inference, and not during the training. The color and preprocessed depth frames constitute

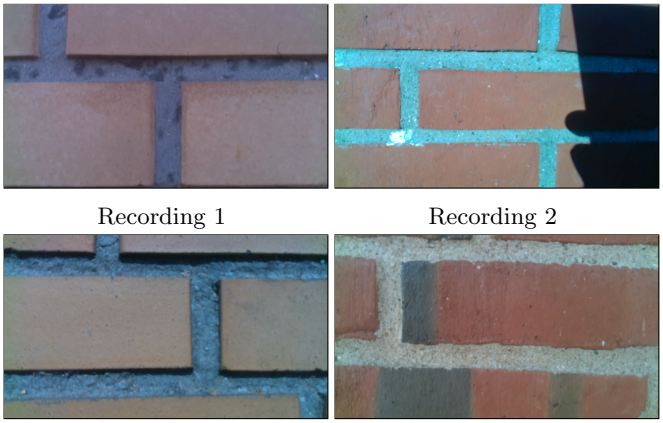

Fig. 4: Example color frames from each of the four recordings. Such color frames, together with the corresponding depth information, constitute the dataset. Note that the four recordings are made under different lighting conditions. In (b), the recording contains a shadow in the right side, while in (d), the bricks have multiple colors.

the entire dataset (i.e. four data points for each pixel location). A color frame example from each recording is shown in Fig. 4.

The dataset is further diversified by three data augmentation processes: flipping the image (i.e. mirroring along either the vertical or horizontal axis), changing the contrast, or adjusting the brightness. These modifications introduce some variance, and noise into the frames, which in turn enable the algorithms to learn parameters which can deal with these factors. These processes also mimic the possible effects of the environmental conditions (e.g. lighting condition), as well as the relative orientation of the camera with respect to the wall. Thanks to the data augmentation step, the number of samples in the dataset is increased to 108 in total. Since each frame has a resolution of 848×480 pixels, the dataset contains almost 44 million data points, each of which is a four-element vector with the corresponding color and depth information.

The ground truth segmentation masks for each training sample are manually created in the next step using Photoshop. These masks represent the perfect segmentations, which are regarded as the 100% accurate outputs. These are used both during the training of the algorithms, as well as during the evaluation to compare the segmentation output with these perfect masks.

The dataset is split into training and test sets at a ratio of four to one. Each split contains examples from each of the four recordings. Therefore, the final training set contains 86 examples, and there are 22 in the test set. The training set is used to train all of the machine learning classifiers, while the test set is used to evaluate them.

Table 1: Evaluation Results of the Chosen Classifiers

model	domain	recall	precision	specificity	NPV	F1	accuracy
kNN	HSV	0.97	0.96	0.87	0.91	0.93	0.94
Naive Bayes	HSV	0.98	0.89	0.65	0.92	0.85	0.90
QDA	RGB	0.98	0.93	0.78	0.93	0.90	0.93
SVM	HSV	0.98	0.95	0.83	0.94	0.92	0.94
Decision Tree	HSV	0.97	0.95	0.85	0.91	0.92	0.94
Random Forest	HSV	0.98	0.95	0.86	0.93	0.93	0.95
AdaBoost	RGB	0.96	0.94	0.83	0.87	0.90	0.93
UNet	RGB	0.98	0.95	0.84	0.93	0.92	0.94
Fusion		0.98	0.96	0.87	0.94	0.94	0.95

4.2 Results

The eight machine learning classifiers are evaluated individually in the first step. The metrics are summarized in Table 1. As mentioned above, the F1-score and accuracy metrics are used to determine the weight for each algorithm during the fusion step.

The kNN was set up to use 3 neighbors, uniform weights for each neighborhood, and the Euclidean distance measure. This classifier achieved one of the best results in multiple metrics using the HSV color space. It outperforms all other models in both precision and specificity. High precision means that the kNN tends not to make false positive predictions compared to true positives, while the high specificity score indicates that the model accurately classifies most of the true negatives. Overall, this classifier achieved an F1-score of 0.93 and an accuracy of 0.94 on the test set.

The second model evaluated was the naive Bayes, which performed worst among the eight classifiers. Although it handled the true positive cases well, it struggled with the negatives. A low specificity value of 0.65 indicates that this model made a large amount of false positive predictions compared to the true negatives. It achieved an F1-score of 0.85 and accuracy of 0.9 using the HSV color space. Due to the positioning of the camera close to the wall, a large portion of the image is occupied by the positive class (i.e. bricks). Therefore, even the naive Bayes classifier could achieve acceptable F1-score and accuracy, even though it handled the negative class poorly.

Similarly to the naive Bayes, the QDA classifier shows the same results in terms of predictive capabilities regarding the positive and negative classes, however, it performed better using the RGB color space. This is not surprising, considering that the QDA also uses Bayes' rule internally. Regardless, the overall performance of this classifiers is superior to that of the naive Bayes, with an F1-score of 0.90 and accuracy of 0.93.

The support vector machine classifier was set up with a squared-exponential kernel (i.e. *RBF*), and a regularization parameter of 1, using the HSV color space. Although the SVM achieved high evaluation scores, with an F1-score of 0.92 and accuracy of 0.94, the downside of using it is that there are 5240 support

vectors in the trained exponential model for each class, so in total more than 10 thousand. Thus, the model takes a long time to make predictions, since there are a great amount of cross products to be calculated for each input (and each image has more than 400 thousand pixels).

The decision tree classifier, on the other hand, performs similarly to the SVM (using the HSV color space, the F1-scores and accuracy metrics are the same), yet this model is much faster than the SVM model. The decision tree, using the HSV domain, has 875 nodes in total, however, since there are two classes, each split only has two possible outcomes, and the maximum depth is set to 10 during training. So at most, there are 10 decision rules that each input goes through. Thus, the decision tree classifier beats all of the previous classifiers, with the exception of the kNN, simply because it is faster with better F1-score and accuracy.

The random forest model adds a layer of complexity on the decision trees by training 10 of them and having them vote for the most likely outcome. Naturally, there are approximately 10 times as many nodes (i.e. 8598), however the maximum depth of each tree is still 10. So each input pixel goes through 100 decisions altogether. It also makes this classifier slower than the decision tree, yet it achieves a slightly better performance. It has an F1-score of 0.93 and an accuracy of 0.95. Another insight into the random forest is the relative importance of the features calculated as the normalized total reduction of the Gini index by the given features in the whole model. In the case of the HSV-D color space, the importances are: 0.307, 0.500, 0.111, and 0.082. The saturation channel alone corresponds to half of the predictive value of this model. This is a consequence of having a very low saturated mortar as one class and a somewhat saturated brick class. The second most important feature is the hue channel. Although not tested, this value is expected to drop on a multi-colored brick dataset, simply because in the current case, the model is expecting a red brick as input. Finally, not much emphasis is put on the depth input, which might be a result of having the lowest variance within the depth values (i.e. the two planes are close to each other).

The AdaBoost model was set up with a maximum number of 50 estimators, a decision tree-based classifier, and a learning rate of 1. Although this classifier handled the positive cases well, it has the lowest negative predictive value of 0.87. It means that AdaBoost predicted a significant amount of false negatives compared to the true negatives. Overall, it achieved an F1-score of 0.90 and accuracy of 0.93 (same as the QDA) using the RGB color space.

Finally, the training of the deep learning UNet model is slightly different from the other classifiers. The same training set was used, however, this model required a longer training process. The learning rate was set to 0.005 initially, and it was reduced by 20% after every four epochs. The reduction of the learning rate should allow the model to gradually close in on the optimal set of weights. Another tweak of the training process has to do with the disproportionate representation of the two classes. Since there are almost three times as many brick examples than mortar, during the evaluation of the loss, if the classes are not represented

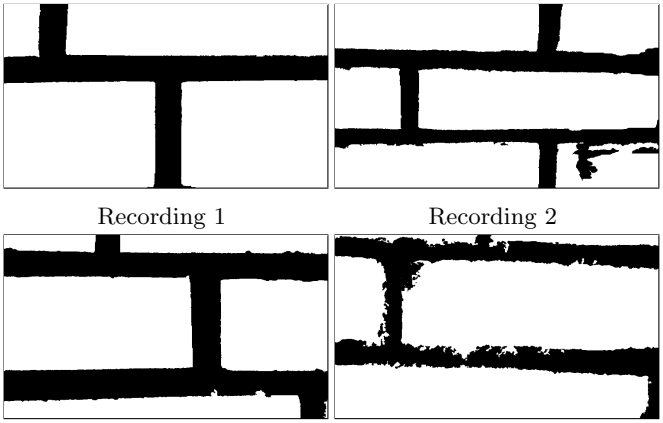

Fig. 6: Binary segmentation masks for the inputs in Fig. 4 created with the final computer vision system, which combines the individual outputs of the eight machine learning based classifiers (see Table 1).

equally, the model could learn to disregard the errors caused by misclassification of mortar. Therefore, during the calculation of the loss, the contribution of the brick pixels is reduced by 65%, such that it would match the weight of the mortar contribution, which improves the performance of the UNet model. The model was trained over 280 epochs with a batch size of 1 under 168 minutes. The accuracy of the model using the RGB color space is 0.94 with and F1-score of 0.92, thus it is slightly worse than the random forest, and on par with the kNN classifier.

In order to combine the individual masks, the following weights are assigned to each classifier shown in Table 1 from top to bottom, calculated according to (5): 0.127, 0.119, 0.124, 0.126, 0.126, 0.127, 0.124, and 0.126. The kNN and the random forest classifiers received the highest weights, while the naive Bayes received the lowest weight (0.119). The threshold value used to binarize the combined segmentation mask was determined experimentally by looking at the performance of the entire vision system on the test set. Finally, a value of 140 was used as the threshold, and the combined vision system achieved an F1-score of 0.94 and accuracy of 0.95 (see Table 1). The system has equally high recall and precision scores, which means that there are not many misclassified pixels in the output. In total, there were almost 9 million data points in the test set. Approximately 3.25% of the predictions were false positive, and 1.44% were false negative. The false positives often have an effect on the accuracy around the edges of the bricks, however, having false positives would only cause the robotic platform to be less thorough during the milling. False negatives, on the other hand, indicate to the robot that there is mortar to be milled, where in reality it is actually brick.

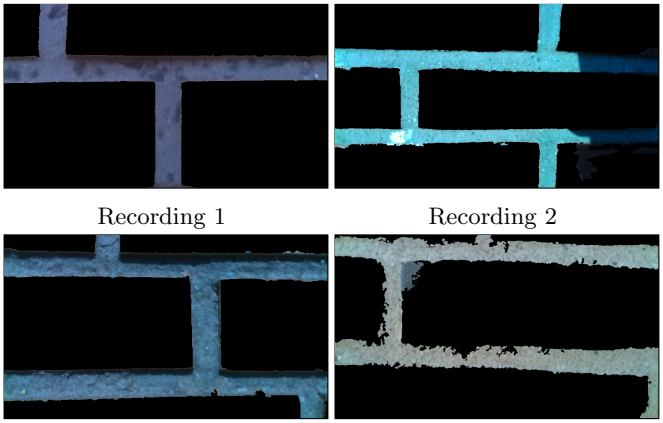

Fig. 8: Overlay of the final segmentation masks shown in Fig. 6 on the inputs shown in Fig. 4. The output masks were inverted to mask out the brick areas, so that the accuracy of the system around the edges of the bricks would be visible.

The final segmentations of the four example inputs in Fig. 4 are shown in Fig. 6. Using the combination of classifiers produces the best overall segmentation in all four recordings at the same time. The first and third masks are the most accurate. The system is able to handle the shadows in Fig. 5b, however, it struggles most with the fourth example in Fig. 5d.

The masks are overlaid on the example inputs in Fig. 8. These images intend to show the accuracy of the masks around the edges of the bricks. The masks in Fig. 7a and Fig. 7c have a high accuracy at the edges, which would allow the robot to mill most of the mortar away without damaging any of the bricks. The mask in Fig. 7b is also accurate in most areas of the image, except for the lower right side in the shadow. The mask in Fig. 5b has a hole in the brick area at the bottom right corner, however, the opening at the top edge of this brick is narrow. Since the hole does not go through the entire brick area, it is not a mortar channel. Finally, the segmentation overlay in Fig. 7d is the least accurate one. The small protruding false brick classifications could be ignored to some extent by reducing the shape to a more regular rectangular one (assuming the bricks have rectangular shapes). However, the most serious issue is the upper left corner of the large brick in the middle, which is not classified as brick. This would indicate to the robot that there is mortar there, which would mill away the corner of the brick.

5 Conclusion and Discussion

This paper has presented a computer vision system used to produce accurate binary segmentation masks of close-range images of brick walls. The advantages

of the presented approach are cost effectiveness (it requires only cheap hardware with stereo vision capabilities, e.g. the Intel RealSense D435), accuracy (achieved by combining the results from a number of classifiers), and flexibility (the system can easily be expanded with additional classifiers). The performance of vision system was evaluated on around 9 million data points with a final F1-score of 0.94 and accuracy of 0.95. The segmentations shown in Fig. 6 are accurate in most cases, with minor issues for areas in the shadow (Fig. 5b) and multi-colored bricks (Fig. 5d).

In order to further enhance the final segmentation, the threshold operation could be modified to be more dynamic. For instance, one could try to find brick shapes (i.e. rectangles) at different gray-levels in the combined mask. It is equivalent to localizing the threshold, instead of using a single global threshold value. In this case, one could train classifiers that work well, for instance, on shadowy areas, and then use a local threshold value (or even a higher local weight) for such parts of the image. Thus, the final mask would be more accurate, using local information for the binarisation process.

Acknowledgment

The authors would like to thank the company Robot At Work for offering their collaboration to solve the mortar raking problem. Furthermore, we thank Rune Hansen, Finn Christensen, and Kasper Laursen from Robot At Work for their support and contribution to the project.

References

1. Adán, A., Quintana, B., Prieto, S., Bosché, F.: An autonomous robotic platform for automatic extraction of detailed semantic models of buildings. Automation in Construction **109**, 102963 (2020)
2. Adán, A., Quintana, B., Prieto, S.A., Bosché, F.: Scan-to-bim for secondarybuilding components. Advanced Engineering Informatics **37**, 119–138 (2018)
3. Altman, N.S.: An introduction to kernel and nearest-neighbor nonparametric regression. The American Statistician **46**(3), 175–185 (1992)
4. Bovenzi, M.: Health effects of mechanical vibration. G Ital Med Lav Ergon **27**(1), 58–64 (2005)
5. Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)
6. Breiman, L., Friedman, J., Stone, C.J., Olshen, R.A.: Classification and regression trees. CRC press (1984)
7. Cortes, C., Vapnik, V.: Support-Vector Networks. Machine learning **20**(3), 273–297 (1995)
8. Grana, C., Borghesani, D., Cucchiara, R.: Optimized Block-Based Connected Components Labeling With Decision Trees. IEEE Transactions on Image Processing **19**(6), 1596–1609 (2010)
9. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2003)
10. Hastie, T., Rosset, S., Zhu, J., Zou, H.: Multi-class adaboost. Statistics and its Interface **2**(3), 349–360 (2009)

11. Ibrahim, Y., Nagy, B., Benedek, C.: CNN-Based Watershed Marker Extraction for Brick Segmentation in Masonry Walls. In: International Conference on Image Analysis and Recognition. pp. 332–344. Springer (2019)
12. Intel Corporation: Intel RealSense D400 Series Product Family Datasheet (2020), https://dev.intelrealsense.com/docs/intel-realsense-d400-series-product-family-datasheet
13. Kapoor, M., Katsanos, E., Thöns, S., Nalpantidis, L., Winkler, J.: Structural integrity management with unmanned aerial vehicles: State-of-the-art review and outlook. In: Sixth International Symposium on Life-Cycle Civil Engineering (IALCCE 2018). Ghent, Belgium (2018)
14. Kim, D., Yin, K., Liu, M., Lee, S., Kamat, V.: Feasibility of a Drone-Based On-Site Proximity Detection in an Outdoor Construction Site. In: Computing in Civil Engineering 2017, pp. 392–400 (2017). https://doi.org/10.1061/9780784480847.049
15. Lu, Y., Wu, Z., Chang, R., Li, Y.: Building information modeling (bim) for green buildings: A critical review and future directions. Automation in Construction **83**, 134–148 (2017)
16. Oses, N., Dornaika, F., Moujahid, A.: Image-based delineation and classification of built heritage masonry. Remote Sensing **6**(3), 1863–1889 (2014)
17. Powers, D.M.: Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. Journal of Machine Learning Technologies **2**, 37–63 (2011)
18. RAW: About Robot At Work (2019), https://robotatwork.com/about-raw/
19. Riveiro, B., Lourenço, P.B., Oliveira, D.V., González-Jorge, H., Arias, P.: Automatic morphologic analysis of quasi-periodic masonry walls from LiDAR. Computer-Aided Civil and Infrastructure Engineering **31**(4), 305–319 (2016)
20. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015)
21. Srivastava, S., Gupta, M.R., Frigyik, B.A.: Bayesian quadratic discriminant analysis. Journal of Machine Learning Research **8**(Jun), 1277–1305 (2007)
22. Valero, E., Bosché, F., Forster, A.: Automatic segmentation of 3D point clouds of rubble masonry walls, and its application to building surveying, repair and maintenance. Automation in Construction **96**, 29–39 (2018)
23. Valero, E., Forster, A., Bosché, F., Hyslop, E., Wilson, L., Turmel, A.: Automated defect detection and classification in ashlar masonry walls using machine learning. Automation in Construction **106**, 102846 (2019)
24. Wang, X.Y., Wang, T., Bu, J.: Color image segmentation using pixel wise support vector machine classification. Pattern Recognition **44**(4), 777–787 (2011)
25. Webster, A., Feiner, S., MacIntyre, B., Massie, W., Krueger, T.: Augmented reality in architectural construction, inspection and renovation. In: Proc. ASCE Third Congress on Computing in Civil Engineering. vol. 1, p. 996 (1996)
26. Xu, S., Wang, J., Wang, X., Shou, W.: Computer vision techniques in construction, operation and maintenance phases of civil assets: A critical review. In: ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction. vol. 36, pp. 672–679. IAARC Publications (2019)
27. Zaher, M., Greenwood, D., Marzouk, M.: Mobile augmented reality applications for construction projects. Construction Innovation (2018)
28. Zhang, H.: Exploring conditions for the optimality of naive bayes. International Journal of Pattern Recognition and Artificial Intelligence **19**(02), 183–198 (2005)