Detection and Segmentation of Approximate Repetitive Patterns in Relief Images

Harshit Agrawal*
harshit.agrawal@research.iiit.ac.in

Anoop M. Namboodiri
anoop@iiit.ac.in

Center for Visual Information Technology, IIIT-Hyderabad

ABSTRACT

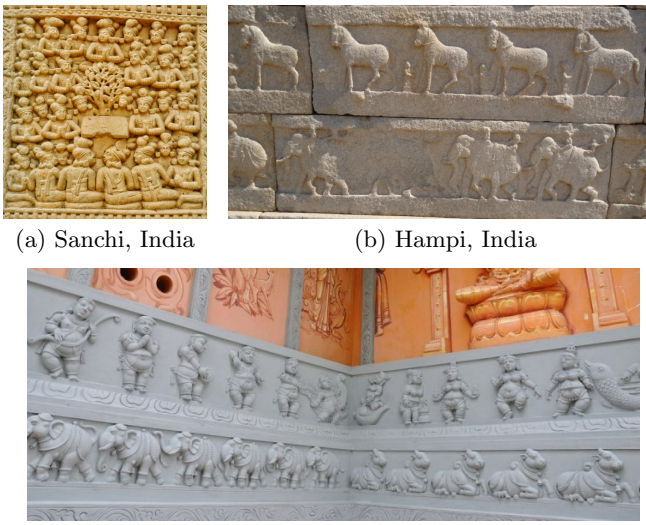
Algorithms for detection of repeating patterns in images often assume that the repetition is regular and highly similar across the instances. Approximate repetitions are also of interest in many domains such as hand carved sculptures, wall decorations, groups of natural objects, etc. Detection of such repetitive structures can help in applications such as image retrieval, image inpainting, 3D reconstruction, etc. In this work, we look at a specific class of approximate repetitions: those in images of hand carved relief structures. We present a robust hierarchical method for detecting such repetitions. Given a single image with reliefs, our algorithm finds dense matches of local features across the image at various scales. The matching features are then grouped based on their geometric configuration to find repeating elements. We also propose a method to group the repeating elements to segment the repetitive patterns in an image. In relief images, foreground and background have nearly the same texture, and matching of a single feature would not provide reliable evidence of repetition. Our grouping algorithm integrates evidences of repetition to reliably find repeating patterns. Input image is processed on a scale-space pyramid to effectively detect all possible repetitions at different scales. Our method has been tested on images with large varieties of complex repetitive patterns and the qualitative results show the robustness of our approach.

Keywords

Reliefs, Spatial Configuration, Repetitive Patterns, Scale-Space Pyramid, Patch Matching

1. INTRODUCTION

Repetitive patterns are present in various structures and shapes of the world at many different scales and forms. In man-made environments, structures with repetitive patterns or elements are often employed as they are aesthetically pleasing. In reliefs, structures often appear in different repetitive patterns. Repetitions in themselves have redundancy as all the structures have some common properties like shape, style, texture, etc. The information extracted by detecting repetitions can be used as prior in segmentation and reconstruction of elements of the structures. It can also provide valuable information in case of partial occlusions. Reliefs are a common form of decoration and medium of depicting incidents and stories in man made structures from ancient times. With time many of these structures get weathered down and parts of the reliefs may get broken or damaged. Repetitive patterns can also assist in single view reconstruction of the scene [19].



(a) Sanchi, India (b) Hampi, India

(c) Shri Senpaga Vinayagar Temple, Singapore

Figure 1: Example reliefs with approximate repetitive patterns.

Robust representation of repetitive patterns can be used as an invariant descriptor to identify semantically relevant objects to match across images [5]. Many algorithms have been developed for detecting repetitions and translational symmetry [7, 8, 9, 10, 12], but a fully automatic method for detecting approximate repetitions in complex real images like reliefs is still a very challenging task. We propose a robust method to detect the approximately repeating structures on reliefs. The hierarchical method proceeds from lower to higher level features, thus robustly detects structures with partial repetitions.

*Corresponding author

ICVGIP '12, December 16-19, 2012, Mumbai, India

We consider a generic notion of repetitive pattern as repetition of a structure or an object. A repetitive pattern can have both uniformly spaced repetitions or irregularly spaced repetitions. In urban facades, the structural elements like windows and doors usually have regularity in repetition. The regularity can always be used as a reliable cue for detecting such repetitive patterns. Different algorithms [6, 12, 14] have been developed that exploits the regularity of such repetitions. Unlike facades, reliefs have irregular repetitions with changes in the appearances. Perspective view of facade images can be rectified using vanishing points to get the fronto-parallel view [12]. It is often difficult to extract robust vanishing points in reliefs due to absence of linear structures (see Fig. 1(a)). In our approach, we do not constrain the input image to have a fronto-parallel view.

Humans inherently recognize symmetries and repetitive patterns in objects and images. Detection of repetitions for humans happen at multiple levels of detail. At a coarse level, we may use the overall texture of a scene or part of it to find possible repetitions. We then analyze the parts of objects within it and their arrangement to find objects that repeat. If those pieces or objects are found in a similar configuration elsewhere, we identify this object as a repetition. The motivation for the design of our algorithm is the same. We begin by finding reliable matches for individual components in an image. These individual matches are then verified and grouped together to get regions with possible repetitions. These grouped matches are then used to detect different repetitive patterns and elements.

We have collected images with various types of repetitive patterns from different sources. It includes images with identical and approximate repetitive patterns. We also test our algorithm for facade images and images from PSU-Near Regular Texture database. The images in the collection were manually annotated to get the ground truth for quantitative evaluation of the algorithm.

1.1 Related Work

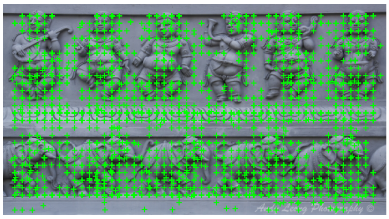
Repetition detection in images is a long standing problem and till date researchers in computer vision have made significant progress towards detection of symmetries and repetitions not only in images but also in 3D data. Leung and Malik [7] proposed a simple window matching scheme followed by grouping of patterns for finding repeating scene elements. Schaffalitzky and Zisserman [15] proposed a RANSAC-based grouping method for imaged scenes, which repeat on a plane in a scene. Park *et al.* [14] developed an algorithm using an efficient Mean-shift belief propagation for 2D lattice detection on near-regular textures. These approaches give good result on grid-like repetitive patterns. However, it is very rare to find a grid-like repeating pattern in hand carved reliefs and are generally limited to one-dimensional repetitions. Hence we cannot use the above approaches for detecting repetitions in our problem. The proposed algorithm does not require the presence of a grid-like repetitive pattern and can detect the repetitions even if an element is repeated only once at two arbitrary locations in the image.

Wenzel *et al.* [17] have proposed a method for detecting repeated structures in facade images. They begin by detecting the dominant symmetries and then use clustering of feature pairs to detect the repeating structures in the image. Wu *et al.*[18] developed an approach to detect large repetitive structures with salient boundaries. They assume reflective symmetry in the architectural structures and use this to localize vertical boundaries between repeating elements. Their algorithm depends on accurate vanishing point detection for rectifying the image to a fronto-parallel view. Their algorithm does not work if the repetition interval is larger than the width of the repetitive structure. In reliefs, we can not assume reflective symmetry, accurate vanishing point detection, or regularity in intervals between repetitions. Our algorithm does not pose any of the above constraints on the input image. However, our algorithm does not output complete repetitions for significantly large perspective distortions.

Zhao and Quan [20] describe a robust and efficient method for detecting translational symmetries in fronto-parallel view of facade images using joint spatial and transformation space detection. Their algorithm outputs incorrect results if the lattice is incomplete or the repetitions are non-uniformly spaced. Recently, Zhao *et al.* [21] developed a robust and dense per-pixel translational symmetry detection and segmentation in facade images. Although their algorithm considers most of the limitations of earlier approaches they cannot be applied directly to detect repetitions in reliefs. They create translational map in horizontal and vertical directions only. Moreover, for segmentation of a repeating element, they have used a learning based approach to classify each pixel as either a wall pixel or a non-wall pixel. Such a classification based segmentation is impractical in relief carvings due to the high textural similarity between foreground and background. Our approach is different from all the previous approaches as we do not have any pre-assumptions about the repetitive pattern or the repetitive element type. We also compute pixel level correspondences of the repetitive elements after merging the results from all the scales in the scale space pyramid. Cai and Basui [2] proposed a region growing image segmentation algorithm for detecting and grouping higher level repetitive patterns. Their algorithm begins with manual marking of a region of repetition and then they iterate between growing and refinement steps.

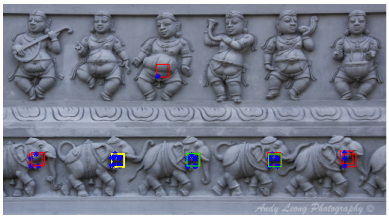
We present a robust algorithm for repetition detection and segmentation in reliefs with fewer assumptions about the nature of repetition compared to existing approaches. Our hierarchical pairwise matching approach is closely related to the work of Carneiro and Jepson [4]. They have developed semilocal image features to improve the robustness of feature matching between a pair of images with possible rigid or non-rigid deformations. The hierarchy of matching levels in our approach can also be compared to levels in a deep learning network, where higher level features and concepts are defined in terms of lower-level ones. We propose a robust pairwise matching approach that detects pairwise matches with high confidence scores. In image matching, interest point groups are matched using pairwise spatial constraints [13]. We have also used pairwise spatial constraints to enhance the confidence of a match. Barnes *et al.* [3] finds dense pairwise matches using k-NN search.

To improve the correspondences they use the offset values of the neighboring pixels. In contrast, we start with confident pairwise matches, found independently and then impose the neighboring constraint while doing hierarchical grouping. The hierarchical approach reduces the search space considerably, and is critical to establish matches with approximate repetitions. Our approach is different from image retrieval in the sense that we are not using a Bag-of-words

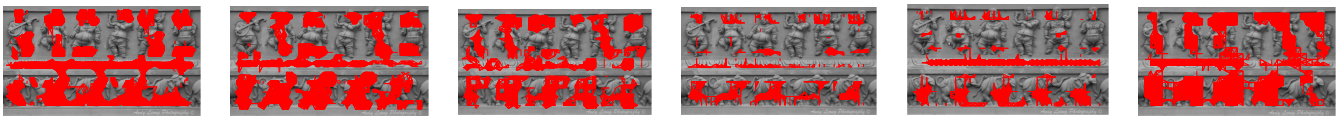


(a) Dense Feature Extraction

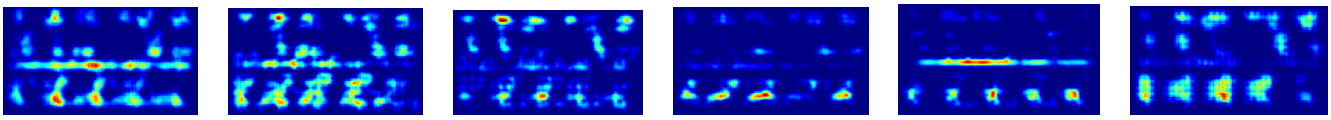
(b) Pairwise Feature Matching



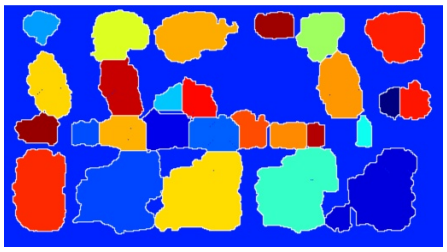
(c) Pairwise Patch Matching



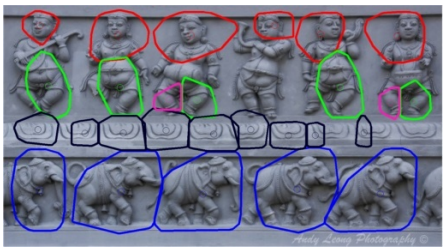
(d) Connectivity graph at different scales



(e) Score images for corresponding connectivity graphs (blue with lower scores and red with higher scores).



(f) Segmented image regions.



(g) Detected repetitive patterns.

Figure 2: (a) Dense sift features after removal of false matches (b) Pairwise sift matching, blue features are pairwise matches of yellow (c) Pairwise patch matching, green patches are correct match of yellow patch and red patches are false matches (d) Connectivity graph at various scales in the scale-space pyramid (e) Score Image for the corresponding connectivity graph in (d). (f) Segmented image regions after merging all score images with distinct labels for each region. (g) These labels are merged and grouped to produce the final output of our algorithm with color-coded convex hulls of repetitive patterns.

representation of features. We would like to have strong matches between features and in a bag-of-words representation, many features will be denoted by a representative visual word which will increase the number of false matches among the features.

2. OBSERVATIONS AND ASSUMPTIONS

Repetitive patterns in reliefs have certain interesting characteristics that are different from the typical patterns found in the urban facades. This section lists some of the observations and assumptions that lead to the design of our repetition detection algorithm.

1. In general, repetitions in reliefs do not occur in a regular grid or at regular interval. Repetitive patterns often appear non-uniformly in an image (see Fig. 1(b)). Our algorithm is independent of factors including the number of repetitions, repetition interval and repetition direction.

2. Occasionally, in reliefs it is difficult to define repetitive patterns where each object is a single unit. Repeating elements generally have some variance in appearance from other repetitions (see Fig. 1(c)). Elements can have partial repetitions and we achieve robust detections for such reliefs with our algorithm.

3. It is often difficult to apply traditional rectification techniques on reliefs as detecting the vanishing line is not robust enough in images such as Fig. 1(a). Our algorithm is not constrained to any requirement of input image to be fronto-parallel or rectified. Detection is robust to significant image skews.

3. PAIRWISE MATCHING

As a brief overview, our algorithm initiates by building a scale-space pyramid of the given input image. Fig. 2 shows the complete pipeline of our algorithm. We process each scale independently. At each scale, we extract feature descriptors and a list of matching features are found for each feature. We then consider the features in image patches and search for similar patches using our similarity score function. These pairwise patch-matches give us a confidence score for matching image patches. Confidence scores are then used to identify the regions that are repeating using watershed segmentation of the confidence score image. We begin by

presenting our framework for finding pairwise patch correspondences, then describe the detection and segmentation of repetitive patterns.

3.1 Pairwise Feature Matching

In order to cover the entire image, we have extracted SIFT descriptors using sift interest points and dense SIFT descriptors with a fixed step size (10 pixels, in our experiments). We convert the SIFT descriptors to ROOT SIFT. This has been shown to work well on many computer vision tasks [1]. We denote the set of SIFT descriptors $\boldsymbol{S}_n$ for the image at current level n. For each $s_i \in \boldsymbol{S}_n$, k nearest neighbors are found using efficient implementation of kd-tree data structure. We consider s_j as a reliable match for s_i if their scales and orientations are very similar. For a pair of sift descriptors, we define a similarity score as:

$$SS(s_i, s_j) = 1 - \frac{sd(s_i, s_j) + od(s_i, s_j) + dd(s_i, s_j)}{3}, \quad (1)$$

where $sd(s_i,s_j)$ is absolute difference between the scales, $od(s_i,s_j)$ is absolute difference between the orientations and $dd(s_i,s_j)$ is the L1 norm of difference vector of their descriptors, each of them normalized between 0 and 1. We consider the sift pair as a strong match if $SS(s_i,s_j)$ is more than a threshold *sim_thresh* (= 0.7 in our experiments). We discard the pair if any of *sd*, *od*, *dd* exceeds the individual threshold (nearly *sim_thresh/3*).

Remove false matches: Trivial repetitions in images like the background plane or repetitions with very small interval are considered insignificant. To remove false matches for sift feature, we find its *num_neighbor* nearest neighbors in sift descriptor space. We use the following false match removal algorithm to filter the set of SIFT descriptors. We remove a feature s_i from $\boldsymbol{S}_n$ if:

- No nearest neighbor has similarity score greater than *sim_thresh*, else check the following conditions.
- At least two nearest neighbor has spatial euclidean distance less than the minimum repetition interval allowed or,
- At least one nearest neighbor are spatially very close to s_i.

The minimum repetition interval is chosen as the patch size (section 4.2) of the current scale. This step removes a large number of insignificant features, which increases the efficiency of the algorithm. All the remaining features in $\boldsymbol{S}_n$ have strong and reliable matches along with their corresponding similarity scores. We denote the j^{th} strong match of s_i as $sift\_match_{ij}$ and score with $sift\_score_{ij}$.

Images with reliefs are noisy due to their textures. Although strict thresholds are used for similarity scores, we further want to increase the reliability of matches. We use a higher level matching after sift feature matching to further remove the remaining false matches. In next step of our algorithm, we introduce a higher level feature matching technique that boosts the robustness and reliability of the feature matches found in this step.

3.2 Pairwise Patch Matching

Matching a set of features has proved to be more reliable then matching a single feature. Next higher level of SIFT matching is matching a set of SIFT features spatially close to each other in the image space. We consider a set of overlapping image patches $\mathbf{P} = \{p_c\}$, where each patch p_c is of size $\tau_n \times \tau_n (\tau_1 \sim 15)$ is centered at a regular interval in the image. Let $\mathbf{s}_c = \{s_i\}$ be the set of SIFT descriptors lying within the image patch p_c. We present our patch-match algorithm for finding matches for each patch in $\mathbf{P}$.

For each image patch $p_c \in \mathbf{P}$, find all the patches that can possibly match p_c. To find the possible patches we use $sift\_match_{ij}$ for all $s_i \in \mathbf{s}_c$. Each possible patch is centered such that the spatial position of s_i and s_j are same in their corresponding patches. We define $matching\_score(p_i, p_j)$ using the spatial configuration of matching sift features in p_i and p_j. To provide flexibility, we divide each patch in 2×2 cells and use soft-binning for each matching sift feature. We find the distance of each matching sift feature in the patch to the centers of the 4 cells and then give a weight $w_i = \frac{1}{1+d_i}$, where d_i is spatial euclidean distance from the center of i^{th} cell. After considering all the matching sift features, we will get a pair of vectors v_i and v_j of size 4×1. For a pair of patches, the *matching_distance* is defined as the L1 norm of the difference vector of v_i and v_j. So the $matching\_score = 1 - matching\_distance$. Two patches have a strong match if their matching score is greater than a threshold (0.7 in our experiments).

Algorithm 1 Patch-Match Algorithm

```
patch_match ← φ
for all p_c ∈ P do
    patch_match(p_c) ← φ
    pos_match(p_c) ← φ
    for all s_i ∈ s_c do
        p ← patch corresponding to sift_match_ij
        pos_match(p_c) ← pos_match(p_c) + p
    end for
    for all pos_match(p_i) ∈ pos_match(p_c) do
        score(p_c) ← matching_score(p_c, pos_match(p_i))
        if score(p_c) ≥ threshold then
            patch_match(p_c) ← patch_match(p_c)+
                                    pos_match(p_i)
        end if
    end for
    patch_match ← patch_match + patch_match(p_c)
end for
```

After this step, $patch\_match(p_c)$ stores position of all the matching patches of p_c along with their matching scores. The patch-match algorithm proposed above will further remove features that are not matched in groups. We remove from $\mathbf{P}$ all the patches that do not have any strongly matching patch. The correctness of retained features are increased after the patch-match algorithm.

4. GROUPING PATCHES

The pairwise patch matches found in the previous section must be grouped together to accomplish our goal of identifying repetitive patterns. Before grouping patches, we want to remove overlapping patches that have strong matches to the same patch. Repetitions in reliefs are not identical and hence there could be multiple places at which the same matching patch is centered. To find a single patch out of multiple overlapping patches, we follow a variant of the non-maximum suppression technique. For each patch match we have a

matching score that is the confidence of the match between the two patches. We say that two patches are overlapping if the Euclidean distance between their centers is less than the patch width τ_n, where n is the current level in the scale-space pyramid. For all overlapping matching patches we only retain the patch with maximum *matching_score* and discard the rest of the overlapping patches. We have also experimented by taking the average position of all the overlapping patches but the resulting patch often gets misplaced as the average patch does not correspond to a valid patch. Hence we have used the former approach.

We consider grouping of patches as the next higher level of patch matching. In this section, we propose a grouping algorithm that tries to group neighboring patches based on the configuration and confidence of their individual matching patches. If a region is repetitive at two places then its sub-regions should also repeat in the same spatial configuration in both the places. Following the above intuition we design the grouping algorithm as follows:

We consider two patches to be neighboring patches if the euclidean distance between their centers is less than $\alpha \times \tau_n$ ($\alpha = 1.2$, in all experiments). It is similar to 8-neighborhood in pixel space. For grouping patches, we find neighboring patches $neighbor(p_c)$ for each patch $p_c \in \mathbf{P}$. From the patch-match algorithm (Algorithm 1) we know that both p_c and the neighboring patch $p_i \in neighbor(p_c)$ have some matching patches with strong matching scores. If a pair of matching patches of p_c and p_i have the similar neighborhood property as p_c and p_i then the confidence of the matches are further increased. As the spatial distance between p_c and p_i is small there is a high possibility of both the patches belonging to a single repetitive unit. We have exploited this fact to design the patch grouping algorithm (Algorithm 2).

Algorithm 2 Patch Grouping Algorithm

```
// All considered patches ∈ P
for all p_c ∈ P do
    neighbor(p_c) ← neighboring patches of p_c
    for all p_i ∈ neighbor(p_c) do
        for all pair ∈
                (patch_match(p_c), patch_match(p_i)) do
            if config_matches({p_c, p_i}, {pair}) then
                join centers of p_c and p_i
                join centers of patches in pair
            end if
        end for
    end for
    if p_c is not joined to any patch then
        patch_match ← patch_match − p_c
    end if
end for
```

In reliefs, the individual repetitive unit can have small variations in the shape and structure. So the neighborhood property is defined to have the trade-off between robustness and accuracy. Consider two pairs of patches (p_a, p_b) and (p_{am},p_{bm}) where p_a and p_b are neighboring patches, p_{am} and p_{bm} are their matching patches respectively. We consider two criteria for the pairs of patches to have a matching configuration. First, spatial distance between the neighboring patches and second, relative spatial arrangement of the neighboring patches in the image space. We have kept a relaxed threshold for both the criteria providing robustness to the grouping, where as the correctness is already been verified at each level before this step. If the absolute difference of the dist(p_a, p_b) and dist(p_{am}, p_{bm}) (dist($\cdot$,$\cdot$) is euclidean distance) is less than a threshold($\sim$ 5 pixels) then it satisfy the first criterion. For spatial orientation, we find the angle made by the line joining the centers of each patch with the horizontal axes. If the absolute difference in the angles is less then a threshold ($\sim 35^\circ$ to 45°), then it satisfy the second criterion. We say that $config\_matches(\{p_a, p_b\}, \{p_{am}, p_{bm}\})$ is true if both the criteria are satisfied. After joining the center of the grouped patches, we get a connectivity graph (Figure 2(d)) in the image space where nodes are the patch centers and the edges denotes that patches have high chance of belonging to a single repetitive unit.

Merging Results of all Scales: A region of the graph with high connectivity among the neighboring patches corresponds to a strongly repeating region but we cannot guarantee that a region with low or no connectivity is not a repeating unit. All the computation is done at each scale of scale-space pyramid. Each scale could possibly detect different patches. To ensure completeness, we need to merge the output of all the scales. We have outputs in the form of matching patches and the connectivity graphs at each scale.

To merge the results from different scales, we create a score image from the connectivity graph at each scale. Score image has a score (between 0 and 1) at each pixel and the score denotes the confidence of that pixel belonging to a repetitive pattern. While joining the patches in Algorithm 2, we removed all the patches that are not joined to any neighboring patch. So all the patches in $patch\_match(p_c)$ is joined with valid neighboring patch. We create a $score\_patch(p_c)$ of size $\tau_n \times \tau_n$ with each value as the maximum score among all the matches $patch\_match(p_c)$ of a patch p_c. A Gaussian mask with $\sigma_n = \frac{\tau_n}{3}$ is applied to give more weight to the center of the patch. The $score\_patch(p_c)$ for p_c is added in the score image at the corresponding place. We add the score patches for all $p_c \in \mathbf{P}$ to the score image at the current level. After repeating this for all scales, we will have a score image at all the scales. Then, each image is mapped to the image at the highest scale of the pyramid. For each pixel, we take the maximum score among all the score images. We also merge $patch\_match(p_c)$ for all patches at all scales to the $patch\_match(p_c)$ of the highest scale. After merging the results at all scales, we create a repetition field rep_field in the image space where $rep\_field(x, y)$ stores the $patch\_match(p_c)$ information along with the matching scores for patch p_c which is centered at (x, y).

5. DETECTION AND SEGMENTATION OF REPETITIVE PATTERN

After grouping patches in the above section, we have the following information available for the given input image. Score image in which each pixel gives the confidence of belonging to a repetitive pattern. For each pixel in the image space, rep_field stores all the matching pixel positions and scores. To detect the repetitive patterns we segment the score image using watershed segmentation technique [11]. We have no prior knowledge about the number of repetitive elements in the image, hence we have to use an unsupervised approach. Watershed segmentation is an automatic segmentation that considers a topographic representation of the image intensity. Intuitively, an infinite source of water

(a) Original (b) Ground Truth (c) Our Output

Figure 3: Example of an approximately repeating structure.

is kept at the lowest basin and then watersheds or dams are formed to prevent water flow from one catchment basin to another.

In our score image, the matches with high matching score will have high intensity values and an absence of a match with zero intensity value. If we invert the score image then region with high matching score corresponds to catchment basins and hence watershed algorithm can be effectively used. Watershed algorithm often gives an over-segmented output. Inverted score image will form large number of catchment basins as an evident of max operation while merging all the scales. Pre-processing of the image with smoothing filters were also over-segmenting the image. So we first converted the gray-scale score image into a binary image by keeping a low threshold (between 0.01 to 0.15) and then convert it back to a gray scale image using euclidean distance transform of the inverted image. This reduces the over-segmentation to large extent and outputs larger segments with unique labels corresponding to possible repetitive regions. We develop a label merging algorithm to get the correct repetitive regions. Each watershed pixel separates two regions with labels l_1 and l_2. Given the repetition field rep_field, we can say that if no pixel in l_1 has a repeating pixel in l_2, then both l_1 and l_2 must constitute the same label. If more than β ($\beta = 10$, in our experiments) watershed pixels satisfy the constraint then we merge l_1 and l_2 into one label.

After appropriate merging of the regions, we get possible repetitive regions associated with unique labels. To detect the repetitive pattern, we need to find correspondences between the repeating element. For each pixel in a label, we find the labels corresponding to its matching pixels using rep_field. Large number of correspondences between two labels implies that both the label belongs to same repetitive pattern. After finding the pairwise correspondences, we group the regions to get the correct repetitive patterns with the repetitive elements.

6. RESULTS AND DISCUSSIONS

We have tested our method on a PC with 2 GHz CPU and 4GB RAM. The Matlab implementation of our method has 3.8min average run time for a typical 500x500 image. We tested our algorithm on a collection of various images. It includes images of repeating reliefs, urban facades, Normal-NRT images. The relief images includes bas-reliefs or low-reliefs, high reliefs and sunken reliefs. We have used some of the facade images from ZuBuD database [16]. The regular texture images are taken from PSU Normal-near regular texture images. The collection also includes images from flickr. We have shown results on representative images from the collection of reliefs, facades and NRT images.

Our algorithm gives correct detection results for reliefs with highly similar repetitions in almost all the images. When the repetition is approximate (see Fig. 3), the algorithm robustly detects the repetitive pattern. In reliefs, only particular parts of the object may repeat. In those cases, algorithm properly segments parts that belong to the repetitive pattern such as in Figures 4.1 and 4.5, where the heads are approximately repeating. Figures 4.2 and 4.3 show the robustness of our approach for irregular repetitions. In Fig. 4.2, the partial elephant is grouped with the horse pattern due to the matching back, which has same appearance to horse's back. Fig. 4.4 shows detection of multiple irregular patterns. The red patterns are a result of matches between sky patches. In Fig. 4.6, our algorithm detected parts of the repetitive element as different pattern because of the absence of matching patches in those regions.

Table 1: Repetition detection and segmentation performance of our algorithm

Image Type	#Images	Avg. Accuracy	Avg. Recall
Reliefs	53	89.66%	79.77%
facades	22	85.3%	80.1%
normal-NRT	13	88.1%	58.3%

We prepared the ground truth results for all the images by drawing an approximate border around each repetitive elements. We evaluate our detection and segmentation algorithm using accuracy and recall measures. The segmentation performance is explained mainly by accuracy and the detection performance is explained by recall measure. We call any detected region a true positive (TP) if it belongs to the correct repetitive pattern. If a non-repeating region is assigned to a repetitive pattern, then we call it a false positive (FP), similarly a segmented region is false negative (FN) if it does not belong to correct repetitive pattern. Similar evaluation criteria is used by Basui and Cai [2]. Table 1 shows the detection and segmentation performance of our algorithm on the collection of images. Figure 7 shows result of our algorithm on facade and NRT images.

Limitations - Our algorithm has some limitations for robust repetition detections. Our algorithm is not robust to occlusions and large projective skews because of insufficient pairwise matches. In Fig. 6(d), the top-left window is partially detected due to occlusion by a tree. In Fig. 6, robust repetitions are detected, but for relatively large projective skew like in Fig. 6(a), our algorithm fails to find reliable matches. In images where the repetitive patterns are very close to each other for example Fig. 6(b) and 6(c), our segmentation algorithm incorrectly merge multiple elements or segments one element into two parts but still we can get satisfactory segmentations by tuning the threshold values.

7. CONCLUSION AND FUTURE WORK

We proposed a robust and efficient method for detecting approximate repetitions in relief images. Our algorithm outputs labeled segmentation of the repetitive patterns by computing a convex hull of the repeating elements. We have evaluated our algorithm on images with various types of repetitions. The robustness of the algorithm is also tested on facade and near-regular-texture images. Our algorithm outputs good results for repetitions with large texture variations. We allows small changes in scale and shapes to be matched for the same repetitive pattern. Our algorithm

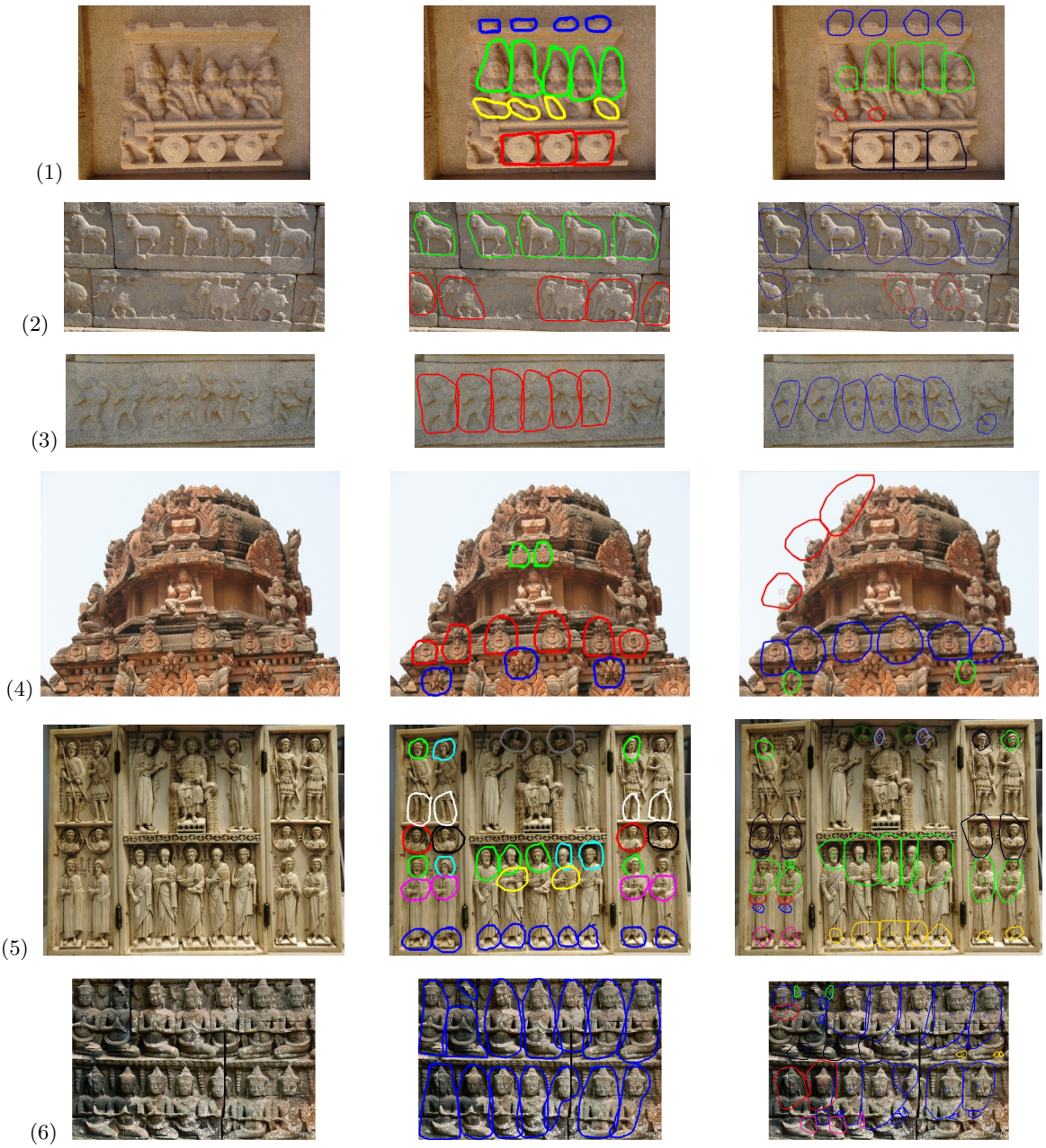


Figure 4: Each row shows the result of detection and segmentation on varieties of reliefs images. (1) shows detection in case of approximate repetitions. (2) Elephant's back is wrongly detected as a horse's repetition. (3) is an accurate detection. (4) Red pattern is a false positive due to repeating sky. (5) Detected multiple repetitions with irregular intervals. (6) As parts of the statue is repeating, they are identified as different repeating elements.

works well for irregular and low count repetitions.

In future, we would like to exploit the pairwise correspondences detected by our algorithm. We believe that the detected repetition can be used in improving the 3D reconstruction from a single image. The repetitions can also help in reconstructing partially damaged structures using region growing and graphical model techniques. The detected objects can possibly be used to describe and retrieve similar objects from a large database of images. We would also like to work and develop a general repetition detection for multiple types of images.

8. ACKNOWLEDGMENTS

This work was partly supported by the India Digital Heritage(IDH) project of Department of Science & Technology, Govt. of India.

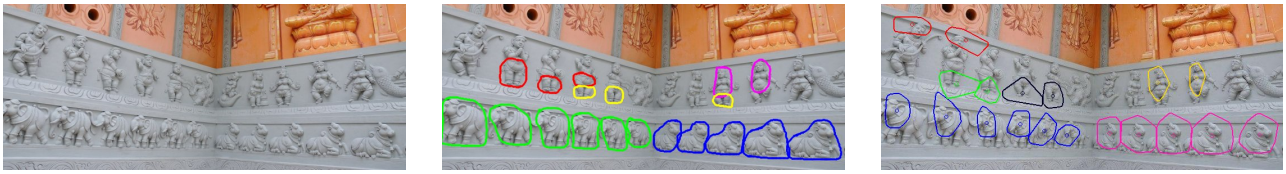


Figure 5: Example of robustness of our detection and segmentation algorithm in presence of multiple repetitive patterns and significant projective skews. Original image(left), Annotated image(center), Our Result(right).

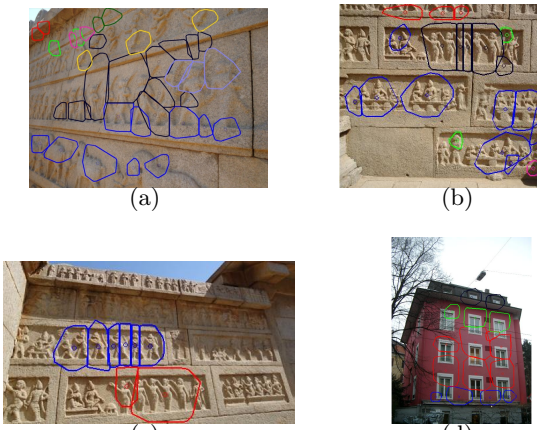


Figure 6: Example failure cases (a) Large projective skews, (b),(c) close repetitive patterns, and (d) partial detection due to occlusions.

Figure 7: Example of our detection performance on Facades and NRT images.

9. REFERENCES

[1] R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[2] G. Baciu and Y. Cai. Higher level segmentation: Detecting and grouping of invariant repetitive patterns. *CVPR*, 2012.

[3] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein. The generalized PatchMatch correspondence algorithm. In *ECCV*, 2010.

[4] G. Carneiro and A. Jepson. Flexible spatial configuration of local image features. *PAMI*, 2007.

[5] P. Doubek, J. Matas, M. Perdoch, and O. Chum. Image matching and retrieval by repetitive patterns. In *ICPR*, 2010.

[6] T. Korah and C. Rasmussen. 2d lattice extraction from structured environments. In *ICIP*, 2007.

[7] T. K. Leung and J. Malik. Detecting, localizing and grouping repeated scene elements from an image. In *ECCV*, 1996.

[8] H.-C. Lin, L.-L. Wang, and S.-N. Yang. Extracting periodicity of a regular texture based on autocorrelation functions. *Pattern Recognition Letters*, 1997.

[9] Y. Liu, R. Collins, and Y. Tsin. A computational model for periodic pattern perception based on frieze and wallpaper groups. *PAMI*, 2004.

[10] G. Loy and J.-O. Eklundh. Detecting symmetry and symmetric constellations of features. In *ECCV*, 2006.

[11] F. Meyer. Topographic distance and watershed lines. *Signal Process.*, 1994.

[12] P. Müller, G. Zeng, P. Wonka, and L. Van Gool. Image-based procedural modeling of facades. *ACM Trans. Graph.*, 2007.

[13] E. Ng and N. Kingsbury. Matching of interest point groups with pairwise spatial constraints. In *ICIP*, 2010.

[14] M. Park, K. Brocklehurst, R. T. Collins, and Y. Liu. Translation-symmetry-based perceptual grouping with applications to urban scenes. ACCV, 2011.

[15] F. Schaffalitzky and A. Zisserman. Geometric grouping of repeated elements within images. In *BMVC*, 1998.

[16] T. S. H. Shao and L. V. Gool. Zubud-zurich buildings database for image based recognition. Technical report, Swiss Federal Institute of Technology, 2004.

[17] S. Wenzel, M. Drauschke, and W. Förstner. Detection and description of repeated structures in rectified facade images. *Photogrammetrie, Fernerkundung, Geoinformation (PFG)*, 2007.

[18] C. Wu, J.-M. Frahm, and M. Pollefeys. Detecting large repetitive structures with salient boundaries. ECCV, 2010.

[19] C. Wu, J. M. Frahm, and M. Pollefeys. Repetition-based dense single-view reconstruction. In *CVPR*, 2011.

[20] P. Zhao and L. Quan. Translation symmetry detection in a fronto-parallel view. In *CVPR*, 2011.

[21] P. Zhao, L. Yang, H. Zhang, and L. Quan. Per-pixel translational symmetry detection, optimization, and segmentation. In *CVPR*, 2012.