LOCATING BINARY FEATURES FOR KEYPOINT RECOGNITION USING NONCOOPERATIVE GAMES *

Victor Fragoso *Matthew Turk* *João Hespanha*

University of California, Santa Barbara
{vfragoso@cs, mturk@cs, hespanha@ece.}ucsb.edu

ABSTRACT

Many applications in computer vision rely on determining the correspondence between two images that share an overlapping region. One way to establish this correspondence is by matching binary keypoint descriptors from both images. Although, these descriptors are efficiently computed with bits produced by an arrangement of binary features (pattern), their matching performance falls short in comparison with other more elaborated descriptors such as SIFT. We present an approach based on noncooperative game theory for computing the locations of every binary feature in a pattern, improving the performance of binary-feature-based matchers. We propose a simultaneous two-player zero-sum game in which a maximizer wants to increase a payoff by selecting the possible locations for the features; a minimizer wants to decrease the payoff by selecting a pair of keypoints to confuse the maximizer; and the payoff matrix is computed from the pixel intensities across the pixel neighborhood of the keypoints. We use the best locations from the obtained maximizer's optimal policy for locating every binary feature in the pattern. Our evaluation of this approach coupled with Ferns shows an improvement in matching keypoints, in particular those with similar texture. Moreover, our approach improves the matching performance when fewer bits are required.

1. INTRODUCTION

Determining the correspondence between two images that share an overlapping region is an important task in computer vision. Many applications rely on this correspondence, e.g., image stitching [1], tracking by detection [2], and others.

One way to establish this correspondence is by matching keypoints on both images, where every keypoint is represented by a descriptor that captures information of the keypoint's surrounding pixel neighborhood (patch). A keypoint then is matched by determining the closest or similar keypoint using their descriptors.

Many applications that demand real-time processing, e.g., [3, 4], require a fast keypoint description and matching. SIFT [5] has shown a great performance in describing keypoints for matching, however its computational cost is high. To alleviate this cost, Lepetit et al. [6] proposed the use of binary features

$$f(I, p_1, p_2) = \begin{cases} 1 & \text{if } I(p_1) > I(p_2) \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where p_1 and p_2 are two pixel locations, and $I(p_1)$ and $I(p_2)$ are their pixel intensities. We can efficiently describe and match an image patch by concatenating the bits produced by several binary features in a pattern [6, 7].

Ideally, a unique binary descriptor per patch is desired, as it will guarantee a low matching error rate. Although these features are easy to compute and match, they cannot capture enough discriminative information for matching keypoints accurately, and we are interested in improving them.

One way to increase their performance is to select the best locations of every binary feature in the pattern for capturing more discriminative information. In other words, to make the matching more accurate, we need to select p_1 and p_2 of every feature in the pattern to produce disjoint keypoint descriptions.

Our problem of selecting the best pixel locations for the pattern can be formulated as an optimization problem. In particular, the best pixel locations can be seen as an optimal game strategy for a player that wants to recognize keypoints given their image patches. Therefore, we can use the mathematical tools that game theory provides to find such optimal strategy.

In this work, we present an approach based on noncooperative games for finding the locations of every binary feature used in a pattern for keypoint recognition. We designed a game that enforces the use of informative pixel pairs in a pattern. We show that the feature locations found improve the recognition of keypoints across certain image distortions.

2. RELATED WORK

Recent literature shows three different manners of locating binary features in a pattern used by a keypoint matcher: randomly, heuristically, and generatively. The available locations for the features are constrained by the size of the patch and the number of bits to use, i.e., the number of binary features in the pattern.

Lepetit et al. [6] and Ozuysal et al. [7] used a pattern containing binary features randomly located. Both approaches train a classifier that learns to recognize keypoints by using the bits produced after evaluating the pattern with the training patches. Generating the pattern randomly is easy and fast. However, the pattern does not guarantee an arrangement of features that capture the most discriminative information.

Calonder et al. [8] presented experiments that evaluated several patterns formed heuristically. The best pattern that they found was produced by locating the features following an isotropic Gaussian distribution. Their pattern is also easy and fast to generate. However, the paper did not justify the nature of such pattern, leaving questions open about their optimality.

Leutenegger et al. [9] presented a radial-symmetrical pattern. The pattern has spaced single pixel locations that are on concentric circles centered on the keypoint. The locations used are determined by computing those pixel pairs whose distance is less than a threshold. Nonetheless, Leutenegger and colleagues did not provide a derivation of the constraints that determine the topology of the pattern.

Rublee et al. [10] presented a methodology that learns the best pixel locations that present high variability and means close to 0.5

*THIS IS A PREPRINT VERSION PUBLISHED BY THE AUTHORS.

across a large training set. However, the authors used those locations as the optimal pattern for any keypoint matching problem. This implies a risk of a bad generalization that can decrease the matching performance. These motivated us to create a different method, based on noncooperative games, to compute an optimal pattern.

3. THE GAME

3.1. Review of Zero-Sum matrix games

In these games two players confront each other. Each player possesses a finite set of actions or an action space Γ. The outcome of a game is quantified by a function that takes in the actions played. This function can be represented by a matrix $A = [a_{ij}]$, where the i-th row corresponds to the i-th action taken by one player, and the j-th column corresponds to the j-th action played by the opponent.

Each player in the game has an objective: one player wants to minimize the outcome by selecting rows, whereas the opponent wants to maximize it by selecting columns. The game can be played simultaneously which means that each player decides which action to play without knowing the other player's action.

A mixed policy is a probability distribution over the actions of a player, describing a game strategy that a player follows, ideally, according to his objective. It is assumed that both players play independently. Therefore, we can compute the expected outcome of the game

$$J = \boldsymbol{y}^{\boldsymbol{T}} A \boldsymbol{z} = \sum_{i=1}^{|\Gamma_1|} \sum_{j=1}^{|\Gamma_2|} a_{ij} y_i z_j \tag{2}$$

where y_i is the probability that the minimizer selects action i and z_j is the probability that the maximizer selects action j. A pure policy is a game strategy that suggests to execute a single action k at each step. Therefore, a pure policy can be described with a mixed policy by indicating to perform action k with probability of 1.

Mixed saddle-point equilibrium policies $(\boldsymbol{y}^\star, \boldsymbol{z}^\star)$ determine "optimal" strategies yielding on average a good course for both players throughout the game. A mixed saddle-point equilibrium is achieved when both players play optimally, and any deviation from the optimal strategy brings a worse expected outcome: the outcome increases when the minimizer plays with a non optimal strategy $\boldsymbol{y}$, and it decreases when the maximizer plays with a non optimal strategy $\boldsymbol{z}$. This can be stated more formally as follows:

$$\boldsymbol{y}^{\star \boldsymbol{T}} A \boldsymbol{z} \leq \boldsymbol{y}^{\star \boldsymbol{T}} A \boldsymbol{z}^\star \leq \boldsymbol{y}^{\boldsymbol{T}} A \boldsymbol{z}^\star \tag{3}$$

Therefore, we are interested in computing mixed saddle-point equilibrium policies for our problem, which can be found by solving appropriate linear programs [11].

3.2. Locating Binary Features

To guarantee a better keypoint recognition rate we must create patterns that produce disjoint binary descriptors given a set of patches to describe, i.e., produce a unique descriptor per patch as much as possible. Hence, we must analyze every patch in order to find the best locations for every binary feature in the pattern. An ideal pair of pixels for locating a binary feature present a high intensity variation across all patches. This intensity variation implies a high likelihood of producing disjoint descriptors when we concatenate the generated bits.

Therefore, we can define a simultaneous zero-sum matrix game in which a player wants to maximize his payoff by selecting pixel pairs that present high variation across patches, while the opponent wants to reduce such a payoff by selecting patch pairs. Hence, we need to design a payoff function that returns a high payoff if a pixel pair has high intensity variation across patches, and a low payoff if the variation is low. We are interested in obtaining the optimal policy for the maximizer as it determines the pixel pairs that we can use in a pattern for improving the keypoint recognition rate.

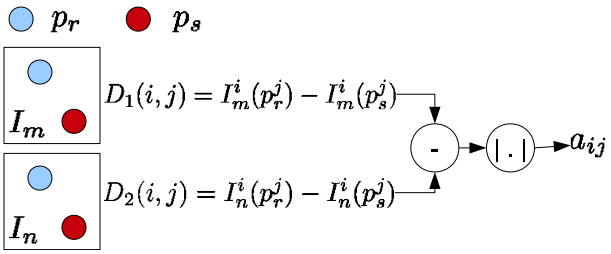


Fig. 1. Payoff matrix computation. The entry a_{ij} is computed using the actions $(I_m, I_n)_i$ and $(p_r, p_s)_j$. A high payoff is returned when the two pixel locations provide a high variation across patches I_m and I_n, and a low payoff is returned when the variation is low.

We define the structure of this game more formally by defining the action spaces and the payoff function. The minimizer has the 2-combination set of the patches set $\mathcal{I}$ as his action space, i.e., a set formed of pairwise combinations of patches

$$\Gamma_1 = \binom{\mathcal{I}}{2} \tag{4}$$

The maximizer has the 2-combination set of the pixel locations set $\mathcal{P}$ as his action space, i.e., a set formed of pairwise combinations of pixel locations

$$\Gamma_2 = \binom{\mathcal{P}}{2} \tag{5}$$

The payoff matrix therefore is built as follows

$$\begin{aligned} a_{ij} &= |D_1(i,j) - D_2(i,j)| \\ &= |I_m^i(p_r^j) - I_m^i(p_s^j) - I_n^i(p_r^j) + I_n^i(p_s^j)| \end{aligned} \tag{6}$$

where $(I_m, I_n)_i \in \Gamma_1$ and $(p_r, p_s)_j \in \Gamma_2$. The terms $D_1(i,j)$ and $D_2(i,j)$ evaluate the difference of the pixel intensities at a particular patch, and by subtracting them, we measure the variation across patches (see Fig. 1). This payoff function enforces the desired objectives: the maximizer will select pixels that are different within the patch and vary across patches.

We are interested in computing the best strategy for the maximizer, i.e., $\boldsymbol{z}^\star$, a mixed saddle-point equilibrium policy. Our primary goal is to obtain the best feature locations from the solution. After solving the game, we sort the elements in the computed policy using their probabilities, and pick the required number of locations. We generate the pattern randomly when no solution or a pure policy, i.e., a single location with high variation, is found.

The players in the described game can be interpreted in a very intuitive manner. The minimizer can be pictured as an agent that controls the nature of the patches in order to confuse the opponent, while the maximizer can be pictured as an agent that wants to select the pixel pairs carefully to recognize the keypoints better. The mixed policies found to this game assume that each participant plays rationally, which is not entirely true in the minimizer case.

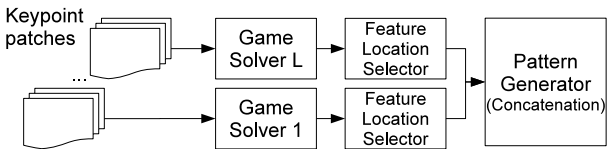


Fig. 2. The keypoint patches are splitted into chunks, and they are introduced to a game solver. After finding the mixed policy for each chunk, a selector extracts the best locations for the features. Subsequently, the pattern is formed.

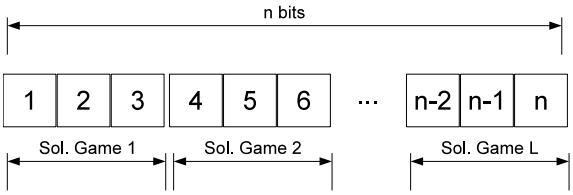


Fig. 3. Each game produces a bit region specialized in recognizing a subset of the keypoints. We concatenate the L solutions of each game to form the final arrangement of n bits. In this figure, only 3 bits were computed per game.

3.2.1. Computational issues and their solutions

Computing the action spaces Γ_1 and Γ_2 is computationally intractable as the memory required to store them can be high. Therefore, by reducing the sets $\mathcal{P}$ and $\mathcal{I}$ we can produce action spaces with a lower cardinality, and that are therefore memory efficient.

We reduce the memory complexity by partitioning the problem into subproblems: we can form a pattern by coupling the solutions of several games, whose solutions provide the best binary features specialized in recognizing a subset of the patches (see Fig.2 for an overview). We partition the set of patches into L subsets, i.e., $\mathcal{I} = \cup_{l=1}^{L} \mathcal{I}_l$, and we formulate L games, each generating a pattern for a specific subset $\mathcal{I}_l$. Hence, Γ_1 is now defined over the pixel pairs constrained by $\mathcal{I}_l$ for each game. To reduce Γ_2 we add a distance constraint on the pixel pairs for each game, i.e.,

$$||p_r - p_s|| \leq \epsilon \tag{7}$$

where ϵ is a threshold in pixels.

The solutions to these new games are still valid, because the arrangements of the binary features are bit concatenations. The final concatenation of bits therefore contains sections that are specialized in recognizing a subset of the keypoints (see Fig. 3). Furthermore, the proposed architecture (see Fig. 2) allows to solve the games and select the features in parallel which can speed up the process.

Mixed saddle-point equilibrium policies usually are sparse. However, in order to fulfill the fixed number of bits required for each game, we trim the found policy by ranking the actions with its probabilities and select the best actions accordingly. When a pure policy or no solution is found, then the best strategy is to generate the locations randomly.

4. EXPERIMENTAL RESULTS

We used the ferns-matcher [7] for evaluating our approach. This matcher trains a classifier that learns to recognize keypoints after observing the generated binary codes produced by using a randomly computed pattern and a training set of patches. We integrated our approach to generate a pattern before training the classifier, and we compared its performance with another classifier trained with a randomly generated pattern.

Fig. 4. Graffiti dataset (top row) and Wall dataset (bottom row). Left column holds the reference images while right column holds a distorted image. These datasets contain five images of the reference image varying the viewpoint.

The ferns-matcher divides the generated binary codes into s chunks of bits (fern) for training, and there are k ferns that produce $n = k \times s$ bits total. We adapted the solutions discussed in Sec. 3.2.1 as follows: a single game is solved to find the feature locations per fern, and Γ_1 is formed after partitioning the training keypoint set in $L = k$ subsets and Γ_2 is formed after adding the distance constraint, eq. (7) with $\epsilon = 5$.

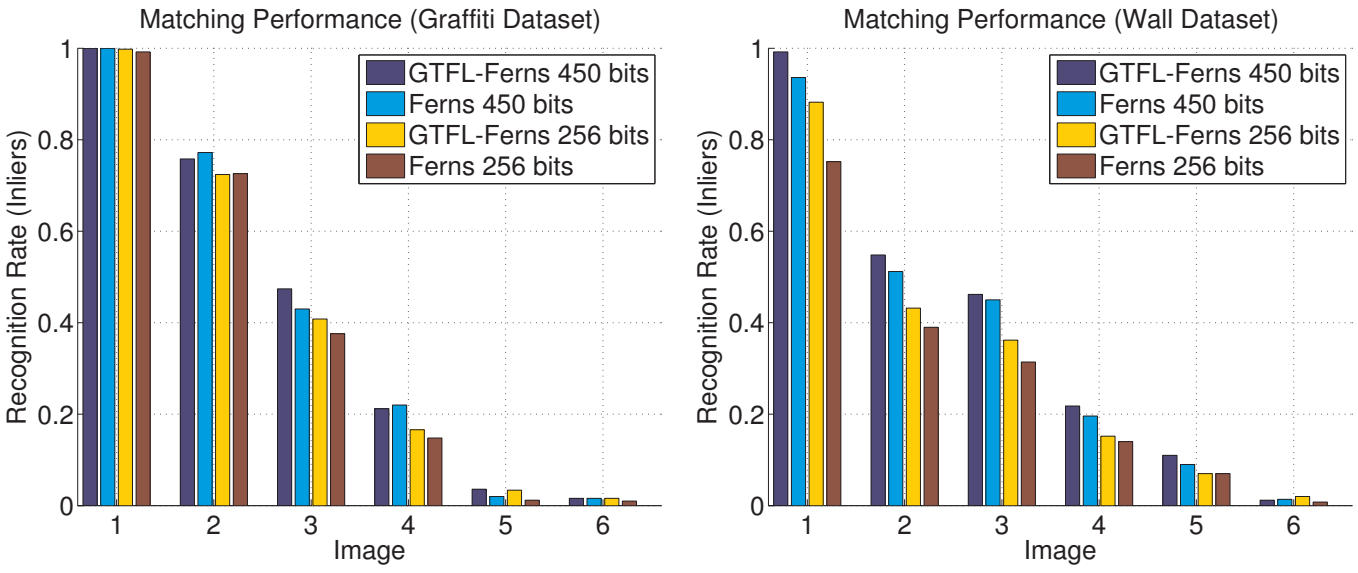
The experiments were implemented in C++ with the use of the libraries OpenCV 2.3 for vision tasks and GLPK 4.47 for solving the linear programs that find the mixed saddle-point equilibrium policy. We extended the OpenCV class that matches keypoints using ferns by including our approach to find the pattern before training.

We used the Wall and Graffiti datasets (see Fig. 4) from the widely used Affine Covariant Features dataset [12] for our experiments. These two datasets contain a reference image and five images that show different viewpoints of the same scene. The used datasets provide the transformations (homographies) that map the reference image to the distorted images.

To detect keypoints we used the OpenCV implementation of the ORB keypoint detector [10]. We matched the 500 keypoints with several matchers and determined a successful match when $||x_D - Hx_R|| < \delta$, where x_D is a keypoint detected on a distorted image; H is the homography; and x_R is the reference keypoint. We used $\delta = 5$ pixels to tolerate the pixel error implied in the transformation, and the default patch size from OpenCV (31×31 pixels).

Fig. 4 shows a performance comparison between GTFL-Ferns, a ferns-matcher coupled with our approach, and a regular ferns-matcher with different bit sizes ($450 = 50$fern $\times$ 9bits/fern and $256 = 32$fern $\times$ 8bits/fern bits). The x-axis shows the image indices ranging from 1 to 6, where image 1 is the reference image (used for training) and the remaining images are sorted denoting an increase in the viewpoint. The y-axis shows the fraction of keypoints correctly matched.

We can observe in Fig. 5(a) that GTFL-Ferns performed competitively when 450 bits were used, and better when 256 bits were used. We can notice from 5(b) an improvement for 450 and 256 bits when the patches to recognize present a similar texture. Furthermore, our approach improved the recognition considerably even on the training image (see Image 1 in Fig. 4), which confirms the generation of more disjoint binary descriptors. Moreover, we can see an overall gain in the tolerance of an increasing viewpoint angle even though we never accounted for that distortion.



(a) (b)

Fig. 5. Performance evaluation in matching keypoints with increasing viewpoint using Ferns and GTFL-Ferns. Although the computed mixed policy is not fully applied (see 3.2.1), GTFL-Ferns presents an overall improvement, in particular when the patches have a very similar texture.

5. CONCLUSIONS AND FUTURE WORK

We presented a simultaneous zero-sum matrix game that models an interaction between the nature of the image patches enclosing a keypoint and a player that wants to find good locations for computing binary features. The game, which is built and solved automatically, is designed for computing binary feature locations that capture more discriminative information for better recognizing keypoints. We analyzed the computational issues and described the solutions that this approach presents. Moreover, our proposed solution can be parallelized for computing the binary feature locations faster.

We showed an evaluation of our approach coupled with a ferns-matcher [7]. Our experiments showed an improvement on the recognition rate (keypoints correctly matched), and specially an improvement when fewer bits are required to recognize keypoints and when the patches to describe present a similar texture. In addition, our results also indicate that there is a gain in the tolerance of an increasing viewpoint angle even though we never accounted for that distortion.

Although, binary features are simple and efficient to match, these features have a lower capacity of capturing representative information for recognizing keypoints accurately. A more significant improvement can be achieved if more information is included in the game, e.g., patch appearance after rotation and scale changes.

We plan to extend our evaluation of our algorithm on patches that are previously rectified using the main keypoint orientation, and compare the results with recent approaches that do not train a classifier and use the binary code produced by the pattern itself, such as ORB [10] and BRISK [9].

Acknowledgments: Victor Fragoso would like to thank UC MEXUS-CONACYT for the funding (Fellowship 212913).

6. REFERENCES

[1] M. Brown, R. Szeliski, and S. Winder, "Multi-image matching using multi-scale oriented patches," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 2005, vol. 1, pp. 510–517.

[2] M. Ozuysal, V. Lepetit, F. Fleuret, and P. Fua, "Feature harvesting for tracking-by-detection," in *European Conf. on Computer Vision*, pp. 592–605. Springer Berlin / Heidelberg, 2006.

[3] G. Klein and D. Murray, "Parallel tracking and mapping on a camera phone," in *Proc. IEEE Intl. Symposium on Mixed and Augmented Reality*, Oct. 2009, pp. 83–86.

[4] D. Wagner, A. Mulloni, T. Langlotz, and D. Schmalstieg, "Real-time panoramic mapping and tracking on mobile phones," in *Proc. IEEE Virtual Reality Conference*, March 2010, pp. 211–218.

[5] David G. Lowe, "Distinctive image features from scale-invariant keypoints," *Intl. Journal of Computer Vision*, vol. 60, no. 2, pp. 91, Nov. 2004.

[6] V. Lepetit, P. Lagger, and P. Fua, "Randomized trees for real-time keypoint recognition," in *Proc. IEEE Computer Vision and Pattern Recognition*, 2005, vol. 2, pp. 775–781.

[7] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua, "Fast keypoint recognition using random ferns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, pp. 448–461, March 2010.

[8] M. Calonder, V. Lepetit, and P. Fua, "Brief: Binary robust independent elementary features," in *European Conf. on Computer Vision*, vol. 6314, pp. 778–792. Springer Berlin / Heidelberg, 2010.

[9] S. Leutenegger, M. Chli, and R. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Proc. IEEE Intl. Conf. on Computer Vision*, 2011.

[10] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: an efficient alternative to sift or surf," in *Proc. IEEE Intl. Conf. on Computer Vision*, 2011.

[11] João P. Hespanha, "An introductory course in noncooperative game theory," Available at `http://www.ece.ucsb.edu/~hespanha/published`, 2011.

[12] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *Intl. Journal of Computer Vision*, vol. 60, pp. 63–86, October 2004.