

Implémentation du jeu Hex via la Programmation Orientée Objet et stratégies de décision en C++

1st Jair Anderson Vasquez Torres
Ingénieur Degree Programme STIC
ENSTA Paris
Paris, France
jair-anderson.vasquez@ensta.fr

2nd Santiago Florido Gomez
Ingénieur Degree Programme STIC
ENSTA Paris
Paris, France
santiago.florido@ensta-paris.fr

Abstract—Ce projet implémente en C++ un système complet pour le jeu Hex, conçu principalement comme un exercice de Programmation Orientée Objet. La solution organise le domaine du jeu au moyen de classes qui encapsulent le plateau, l'état, les coordonnées et les règles, permettant la génération de coups légaux et la vérification de victoire au moyen d'un parcours BFS sur les connexions des pions. Sur cette base s'intègrent des stratégies de décision (p. ex., Negamax avec hachage et table de transposition) et une évaluation de positions découplée du moteur, qui peut être heuristique ou s'appuyer sur un modèle neuronal de valeur intégré à l'exécutable (exportable en TorchScript), sans lier la conception à une architecture spécifique. De plus, une interface graphique (GUI) en SFML a été développée pour faciliter l'interaction, la visualisation du plateau et les tests du comportement des agents.

Index Terms—Hex game, C++, Object-Oriented Programming, Negamax, Transposition Table, TorchScript, SFML.

I. DESCRIPTORS AND PAIRING

In computer vision, a descriptor is understood as a numerical representation, generally vectorial, that is used for the summarized representation in features of the neighborhood of an entity in an image, which is specifically conceived in order to be able to perform matching or comparison operations of those same entities with other images even when there are events such as changes of scale, rotations, illuminations, or noise that can interfere with the normal matching process [1]; a descriptor can then be understood, therefore, as the result of mapping into a vector a local neighborhood in an image, guaranteeing robustness understood as stability under image changes, viewpoints, geometric distortion and occlusions, the discrimination of different objects given the vector, and efficiency, in such a way that it will be easy to compute and compact in memory, that is, it is a feature vector corresponding to the key points.

On the other hand, a detector is an algorithm whose function is to find in the image a set of points, or keypoints, that are repeatable and well localizable [2]; generally, detectors operate from the definition of a measure $r(x, y)$ or $R(x, y, \omega)$ that is used for the detection of corners, blobs, or stable regions in the domain mainly of local multi-scale characterization, seeking local maxima or minima of these functions and filtering unstable candidates, focusing on repeatability, localization precision of these keypoints in the figure, and stability in detection [3].

A. Oriented FAST and Rotated BRIEF

ORB is a local feature method that outputs keypoints and a binary descriptor; it can be conceptually understood as a pipeline that combines a FAST-family detector and a BRIEF-family descriptor. Considering that since FAST is not scale-invariant, ORB detects FAST keypoints on multiple rescaled versions of the image, on an image pyramid [4].

The FAST method works by considering a circle of generally 16 pixels in the neighborhood of a candidate pixel p and classifies the pixel p as a corner in the case where there exists a group of pixels on that circle (the neighborhood of p) that have sufficient intensity in comparison with p using a threshold value t , as defined in Eq. (1):

$$S_{p \rightarrow x} = \begin{cases} d, & I_{p \rightarrow x} \leq I_p - t \\ s, & I_p - t < I_{p \rightarrow x} < I_p + t \\ b, & I_p + t \leq I_{p \rightarrow x} \end{cases} \quad (1)$$

where $S_{p \rightarrow x}$ is the “label” of pixel x on the circle.

- d (*dark*): the circle pixel is at least t darker than the center.
- b (*bright*): the circle pixel is at least t brighter than the center.
- s (*similar*): it is inside the band $(I_p - t, I_p + t)$, that is, it does not differ enough.

FAST *declares* that p is a corner if there exists a set of contiguous pixels on the circle such that all of them are *bright* or all of them are *dark*. It is a very fast method because it can be implemented via process optimizations such as the use of a learned decision tree to select which positions to evaluate first [5].

By itself, the FAST segment test constitutes only a binary classification; however, FAST introduces a score value so that non-maximum suppression can be performed and only local maxima are kept [5]. One (efficient) definition given is in Eq. (2):

$$V = \max \left(\sum_{x \in S_{\text{bright}}} (|I_{p \rightarrow x} - I_p| - t), \sum_{x \in S_{\text{dark}}} (|I_p - I_{p \rightarrow x}| - t) \right). \quad (2)$$

In ORB, the FAST threshold is set sufficiently low so as to obtain more than N candidates that can then be evaluated using a Harris corner measure, and to keep the top N values [4].

A standard Harris measure uses the second-moment (structure tensor) matrix H and response in Eq. (3):

$$C = |H| - k(\text{trace}(H))^2. \quad (3)$$

The selection between a score type, more stable with Harris, or faster but slightly less stable with FAST_SCORE, is made available by OpenCV [6].

Given the nature of FAST, it does not naturally produce an orientation; this is why ORB needs to add an orientation step, and for that it implements the centroid idea. ORB improves stability by computing moments only within a circular region of radius r , with raw moments defined in Eq. (4).

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \quad (4)$$

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \quad (5)$$

Then the orientation is the angle of the vector from patch center O to centroid, as in Eq. (6):

$$\theta = \text{atan2}(m_{01}, m_{10}) \quad (6)$$

On the other hand, BRIEF is responsible for representing a patch p by means of a bitstring that is produced from simple intensity comparisons, as shown in Eq. (7).

$$\tau(p; x, y) = \begin{cases} 1, & p(x) < p(y) \\ 0, & p(x) \geq p(y) \end{cases} \quad (7)$$

An n -bit descriptor can then be built as in Eq. (8).

$$f_n(p) = \sum_{i=1}^n 2^{i-1} \tau(p; x_i, y_i) \quad (8)$$

It focuses on binary strings given the inherent ease of comparing them using Hamming distance rather than L_2 distances in vectors [4]. Plain BRIEF is very sensitive to rotation; in response to this it suggests the concept of steered BRIEF, which rotates the sampling pattern as a function of a discretized angle θ . Let the test locations be encoded in a matrix as in Eq. (9).

$$S = \begin{pmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \end{pmatrix} \quad (9)$$

These locations are rotated according to Eq. (10).

$$S_\theta = R_\theta S \quad (10)$$

Finally, the descriptor is computed using the rotated test coordinates, as expressed in Eq. (11).

$$g_n(p, \theta) := f_n(p)|_{(x_i, y_i) \in S_\theta} \quad (11)$$

ORB then analyzes a subtle but critical issue: when you orient BRIEF consistently, the bit statistics change—the means move away from 0.5 and the tests become less discriminative and more correlated.

The final descriptor used in ORB is an rBRIEF constructed by generating tests with a mean close to 0.5 and high variance, which also preserve low correlation with the tests already selected. The procedure can be summarized as follows: (i) enumerate all pairs of subwindows (then remove overlapping tests), yielding candidate tests; (ii) run each test over all training patches; (iii) sort tests by the distance of their mean from 0.5 (best first); and (iv) apply a greedy selection, keeping a test only if its absolute correlation with all selected tests is below a threshold. This produces a final descriptor that remains binary and fast (Hamming), but with bits that are more informative and less redundant than a naive “steered BRIEF”.

For binary descriptors $d_1, d_2 \in \{0, 1\}^{256}$, matching uses the Hamming distance, as defined in Eq. (12):

$$\text{Ham}(d_1, d_2) = \sum_{i=1}^{256} \mathbf{1}[d_{1,i} \neq d_{2,i}]. \quad (12)$$

This can be computed efficiently as in Eq. (13):

$$\text{Ham}(d_1, d_2) = \text{popcount}(d_1 \oplus d_2). \quad (13)$$

BRIEF emphasizes this efficiency, and ORB notes SSE popcount optimizations in their matching implementation [7].

Finally, in ORB an image pyramid of scales is constructed and, for each scale, FAST corners are detected using a test threshold and are assigned a score using FAST_SCORE or Harris; the strongest features are retained, generally using Harris; the orientation θ is computed via the intensity centroid moments; and finally a descriptor is computed using a smoothed patch, a rotated sampling pattern, and an rBRIEF learned test set, in order to be able to perform descriptor matching using Hamming distance computed via popcount. It is worth highlighting that the way ORB is constructed allows it to handle in-plane rotation, which is addressed through the centroid-based orientation and the steered sampling pattern. Additionally, it can handle image scale by implementing pyramidal detection; however, it remains not fully affine-invariant to viewpoint changes, because projective changes can still break patch appearance.

B. KAZE

As in the case of ORB, KAZE is an algorithm for the detection and description of keypoints, but unlike ORB it constructs the scale space with nonlinear, edge-preserving diffusion; it detects points with a Hessian-type detector and describes them with a (M-)SURF-type descriptor over that nonlinear scale space [8].

KAZE starts with the construction of the nonlinear scale space; for that, a family of images $L(x, y, t)$ is defined, where t plays the role of scale or diffusion time, and it is modeled using the PDE given in Eq. (14):

$$\frac{\partial L}{\partial t} = \text{div}(c(x, y, t) \nabla L). \quad (14)$$

Here, ∇L points toward where the image changes the fastest (edges = large gradient). The diffusion “flow” can be seen as Eq. (15):

$$\mathbf{J} = -c \nabla L, \quad (15)$$

then $\text{div}(\mathbf{J})$ measures how much flow “accumulates” or “leaves” a point, resulting in a definition of brightness propagation analogous to heat propagation, but with a conductivity c that is a controllable parameter [8]. This is precisely the key, because c depends on the gradient, as expressed in Eq. (16):

$$c(x, y, t) = g(\|\nabla L(x, y, t)\|). \quad (16)$$

It should be noted that this gradient is not the raw gradient of the image, but rather the gradient computed on a Gaussian-smoothed version.

If $|\nabla L_\sigma|$ is small, it corresponds to a flat region in the image, and what is sought is to increase diffusion; however, in the opposite case, if $|\nabla L_\sigma|$ is large, it corresponds to a strong edge in the image where the main intention is not to cross that edge [9]. This is ensured by the two typical forms of g given in Eq. (17) and Eq. (18):

$$g_1(s) = \exp\left(-\frac{s^2}{k^2}\right), \quad (17)$$

$$g_2(s) = \frac{1}{1 + \frac{s^2}{k^2}}. \quad (18)$$

In both definitions, a fundamental element that emerges is k , which is a contrast threshold used as a separation between variations that are considered small for the image, such as noise or textures, and those that are considered large, such as an edge. When k is small, many gradients are considered as edges, which implies that diffusion is stopped in many parts of the image, that is, it is smoothed less; whereas if k is large, only very large gradients are considered as edges, so there is more diffusion and the image is smoothed more.

Since this problem does not have a closed-form analytic solution, KAZE uses numerical schemes that are semi-implicit and that use additive operator splitting in order to construct the scale space with stability [8].

At each level or scale of KAZE, the Hessian response is computed through its determinant and maxima are searched both in position and in scale, as given in Eq. (19):

$$L_{\text{Hessian}} = \sigma^2 (L_{xx}L_{yy} - L_{xy}^2). \quad (19)$$

Here, L_{xx} , L_{yy} , and L_{xy} are the second derivatives (local curvature) measured on L , and the factor σ^2 is a normalization so that the response is comparable across scales (because derivatives “shrink” as scale increases).

Then, KAZE searches for extrema in spatial neighborhoods and across scales, and estimates with precision the position

of the maximum that was found with the Hessian response in Eq. (19). KAZE also computes a SURF-type orientation: it takes first-order derivatives in a circular neighborhood with a radius proportional to ω , weights them with a Gaussian, and searches—through a sliding angular window—for a dominant angle.

For the descriptor, KAZE makes use of an M-SURF descriptor, as already mentioned, adapted to the nonlinear scale space, integrating gradient-type responses over sub-patches; the objective of this type of descriptor is to capture how intensity changes around the point in a way that is robust to noise [10].

For a keypoint at scale σ_i , it computes derivatives L_x and L_y at that scale. It builds a 4×4 grid of subregions around the keypoint [8]. In each subregion it sums a vector of the form shown in Eq. (20):

$$d_v = \left(\sum L_x, \sum L_y, \sum |L_x|, \sum |L_y| \right). \quad (20)$$

It then concatenates all subregion vectors to obtain a typical 64-dimensional descriptor, and finally normalizes it. If orientation is used, the sampling is rotated and the derivatives are also computed in that orientation.

This allows, finally, each KAZE keypoint to be described as in Eq. (21), together with a descriptor (64D or extended depending on the implementation).

$$(x, y, \sigma, \theta) \quad (21)$$

Given the implementation of extrema detection in nonlinear scale spaces constructed by diffusion, the detector is scale-invariant. The computation of the dominant orientation of the keypoint in order to build the descriptor makes it rotation-invariant. If the upright mode of OpenCV is used, although the algorithm becomes faster, it loses that invariance property by not computing the dominant orientation. It can be said that, due to the normalization inherent to the M-SURF descriptor, this algorithm also obtains descriptors that are approximately contrast-invariant. Finally, KAZE typically behaves well for moderate viewpoints, but it is not “affine-invariant” in the strong sense.

REFERENCES

- [1] A. Huamán, “Feature Description,” *OpenCV Documentation* (OpenCV 4.14.0-pre), accessed Feb. 6, 2026. [Online]. Available: https://docs.opencv.org/4.x/d5/dde/tutorial_feature_description.html
- [2] A. Huamán, “Feature Detection,” *OpenCV Documentation* (OpenCV 4.14.0-pre), accessed Feb. 6, 2026. [Online]. Available: https://docs.opencv.org/4.x/d7/d66/tutorial_feature_detection.html
- [3] T. Tuytelaars and K. Mikolajczyk, “Local Invariant Feature Detectors: A Survey,” *Foundations and Trends in Computer Graphics and Vision*, vol. 3, no. 3, pp. 177–280, 2007, accessed Feb. 6, 2026. [Online]. Available: <https://livelho.impa.br/ip08/reading/features.pdf>
- [4] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: an efficient alternative to SIFT or SURF,” in *Proc. IEEE International Conference on Computer Vision (ICCV)*, Nov. 2011, doi: 10.1109/ICCV.2011.6126544, accessed Feb. 6, 2026. [Online]. Available: https://sites.cc.gatech.edu/classes/AY2024/cs4475_summer/images/ORB_an_efficient_alternative_to_SIFT_or_SURF.pdf

- [5] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision – ECCV 2006* (Lecture Notes in Computer Science), vol. 3951, pp. 430–443, 2006, accessed Feb. 6, 2026. [Online]. Available: https://www.edwardrosten.com/work/rosten_2006_machine.pdf
- [6] OpenCV, "cv::ORB Class Reference," *OpenCV Documentation* (OpenCV 3.4), accessed Feb. 6, 2026. [Online]. Available: https://docs.opencv.org/3.4/db/d95/classcv_1_1ORB.html
- [7] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in *Proc. European Conference on Computer Vision (ECCV)*, 2010, accessed Feb. 6, 2026. [Online]. Available: https://www.cs.ubc.ca/~lowe/525/papers/calonder_eccv10.pdf
- [8] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "KAZE Features," in *Proc. European Conference on Computer Vision (ECCV)*, 2012, accessed Feb. 6, 2026. [Online]. Available: https://www.doc.ic.ac.uk/~ajd/Publications/alcantarilla_et_al_eccv2012.pdf
- [9] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, 1990, accessed Feb. 6, 2026. [Online]. Available: <https://www.sci.utah.edu/~gerig/CS7960-S2010/materials/Perona-Malik/PeronaMalik-PAMI-1990.pdf>
- [10] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "SURF: Speeded Up Robust Features," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008, accessed Feb. 6, 2026. [Online]. Available: <https://www.cs.jhu.edu/~misha/ReadingSeminar/Papers/Bay08.pdf>