

Reconnaissance informatique de lettres de l'alphabet en langue des signes en temps réel

Matthias Picard - Romain Senhadji

16 AVRIL 2024

Abstract

Nous présentons ici une approche de vision par ordinateur pour la reconnaissance de la langue des signes en temps réel. La méthode proposée ne repose pas sur l'apprentissage profond, mais utilise plutôt des descripteurs SIFT pour comparer les images de mains capturées avec des images de référence. Nous nous assurons au préalable que les points d'intérêts extraits se trouvent uniquement sur la main du sujet, en appliquant au préalable un masque de peau. Nous montrons des screenshots et une vidéo de démonstration de notre solution, qui démontrent la rapidité de notre algorithme ainsi que sa relative efficacité. Nous discutons également de nos tentatives et des améliorations potentielles de notre méthode, comme la diversification du contexte d'acquisition des images.

I Introduction

Une manière classique d'aborder un problème de classification en computer vision est de construire un vocabulaire de descripteur d'une image puis de construire un classificateur linéaire qui déterminera sa classe. Ou bien, il est également possible de directement comparer les descripteurs avec ceux d'images de référence pour la trouver. Depuis l'avènement du Deep Learning, les méthodes les plus efficaces pour une tâche de classification se sont révélées être celles qui apprenaient également ce vocabulaire. Mais ces méthodes demandent un nombre très conséquent de donnée et une puissance de calcul non-négligeable pour être entraînées. L'objectif de ce projet est de s'intéresser à une tâche classique de classification en computer vision en essayant d'y répondre sans Deep Learning. La tâche en question est un problème de reconnaissance en direct d'un signe de main issue de l'alphabet en langage des signes américains. À partir d'une capture vidéo, nous souhaitons être capables d'extraire les descripteurs visuels extraits d'une main avec des algorithmes comme SIFT[1], puis de les comparer avec d'autres descripteurs issus d'images de mains de référence. L'image de référence ayant le plus de correspondance avec celle issue de la capture sera considérée comme la prédiction de notre algorithme.

I.i Cadrage du problème

Dans un cadre général, les principaux challenges de la reconnaissance de signes de main en temps réel sont la nécessité d'avoir un algorithme très rapide et capable de détecter un mouvement, en plus bien sûr des problématiques de performances de classification. Nous avons tenté d'adapter nos ambitions à ce que nous pensions être capables de produire avec nos connaissances du cours. Pour notre problème de classification de l'alphabet en langage des signes, nous avons décidé de ne considérer que des images statiques. On ne cherche donc pas à détecter les signes des lettres de l'alphabet correspondant aux lettres J et Z (elles nécessitent un mouvement). Cette simplification implique aussi que nous devons récupérer à intervalle de temps régulier des captures vidéos pour pouvoir effectuer le traitement statique dessus. Finalement, notre objectif est de classer une image de main parmi 24 classes ??, correspondant chacune à une lettre de l'alphabet (pas de J et de Z donc). Notre solution doit toujours rester le plus rapide possible afin de garder une illusion de temps réel. Notons enfin que nous nous focalisons sur la détection des signes effectués par la main droite.

II Détails de l'algorithme

Notre code a été implémenté en python et grâce à la librairie openCV[2]. Il est disponible ici : <https://github.com/MatthiasPicard/projet-vision-CS>

II.i Constitution du dataset de référence

Pour créer nos images de références, nous avons pris en photo la main droite d'un des membres du groupe effectuant chacun des 24 signes sur un arrière-plan homogène sans autres objets apparaissant dans le champ 4. Les images sont de taille 480 x 640, donc de qualité relativement ordinaire. Bien que nous pourrions espérer des meilleures performances avec des images de qualités supérieures, le traitement que cela aurait demandé aurait été trop long pour répondre à notre problématique de temps réel. Pour pouvoir appliquer nos algorithmes de matching uniquement sur la main (et donc éviter que les autres éléments de l'image perturbent les prédictions), notre méthode nécessite également de calculer les masques de mains de chacune des images.



Figure 1: Ensemble des signes de mains représentant une lettre de l'alphabet dans le langage des signes américains. Notez que nous ne cherchons pas à prédire le J et le Z (il faut effectuer un mouvement, et nous voulons rester sur un cadre statique)

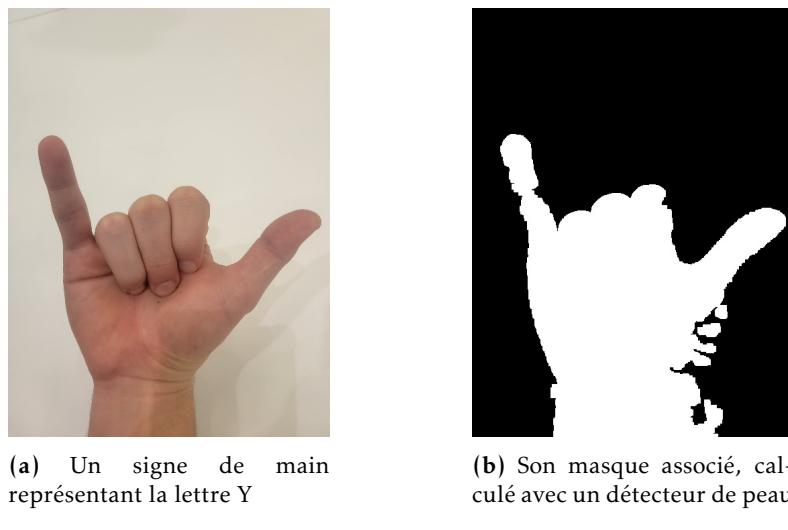


Figure 2: Exemple d'une des images de référence et de son masque de peau associé

de référence. Nous les stockons donc également pour éviter d'avoir à perdre du temps à les recalculer à chaque prédiction.

II.ii Détection de la peau

Une problématique majeure rencontrée lors de l'identification des points d'intérêt résidait dans le fait que notre algorithme détectait un nombre excessif de points autour de la main étudiée. Cette surdétection réduisait la précision de nos résultats, car l'algorithme associait des points qui n'étaient pas pertinents pour notre étude spécifique (c'est-à-dire, des points n'impliquant pas directement la main analysée). Pour résoudre ce problème, nous avons développé un filtre basé sur la détection de la couleur de la peau. Ce filtre a pour objectif de restreindre l'activité de notre algorithme aux zones correspondant uniquement à la main ciblée, améliorant ainsi la précision dans la recherche de ses points d'intérêts.

Nous avons exploré diverses techniques pour améliorer la détection de la peau à travers des approches de filtrage basées sur la couleur. La première méthode que nous avons testée, bien que basique, s'est avérée initialement insuffisante pour répondre à nos besoins. Cette méthode, décrite dans l'article A Survey on Pixel-Based Skin Color Detection Techniques [3], implique l'utilisation de règles simples de classification basées sur les composantes RGB des pixels. Selon cette approche, un pixel est classifié comme appartenant à la peau si les conditions suivantes sont remplies : $R > 95$, $G > 40$, $B > 20$, la différence entre le maximum et le minimum des trois valeurs RGB est supérieure à 15, la différence absolue entre R et G est plus grande que 15, et R est supérieur à G ainsi qu'à B. Nous avons appliqué cette méthode sur nos images prises en condition réelle et que nous utilisons pour évaluer nos algorithmes (voir 3). Nous n'étions pas satisfaits du résultat et avons donc

abordé une autre méthode pour segmenter avec la peau.

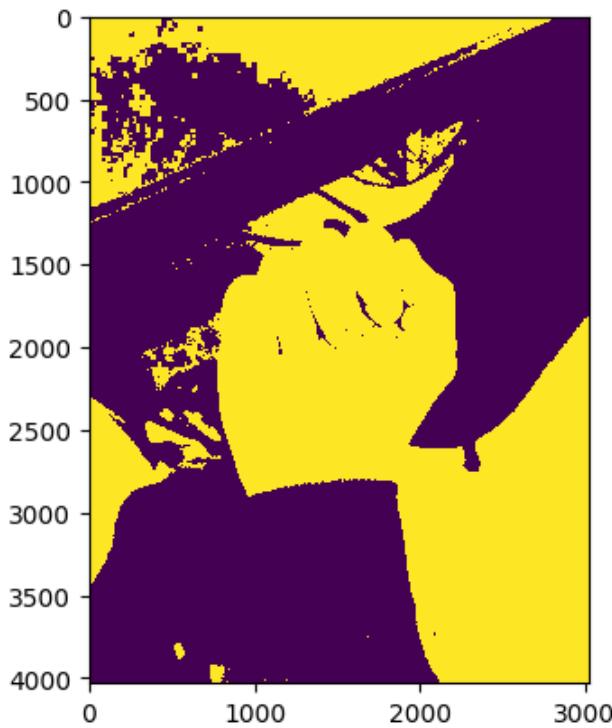


Figure 3: Illustration de la surdéttection de points d'intérêt et l'application de notre filtre de peau.

L'autre méthode que nous avons utilisée finalement est une approche non-paramétrique. Cette approche utilise des histogrammes pour construire un modèle de peau et ainsi comparer pixel par pixel dans l'espace de couleur choisi (HSV, Lab, BGR...) avec notre modèle.

Nous avons donc exploré la détection de zones de peau dans des images avec comme espace de couleur principalement Lab où seules les composantes de couleur a et b sont prises en compte puisque nous travaillions avec des histogrammes en 2D. Le processus impliquait la conversion des espaces couleur via la fonction cvtColor d'OpenCV et la construction d'histogrammes 2D pour les modèles de peau et de non-peau en utilisant la fonction calcHist. Ces histogrammes ont été calculés en utilisant les images du dataset Pratheepan Dataset [4]. Chaque histogramme était normalisé en divisant les comptages de pixels par le nombre total de pixels pour ces classes. Pour classer les pixels des images testées, nous avons calculé la probabilité de chaque pixel appartenant à la peau ou à la non-peau basée sur ces histogrammes. Afin d'améliorer la précision de notre masque de segmentation de peau, nous avons appliqué des opérations de post-traitement utilisant les fonctions morphologyEx et GaussianBlur de cv2, ce qui a permis de réduire le bruit et d'affiner les résultats de segmentation. Cette approche est illustrée dans les figures ci-dessous, montrant l'image originale, son masque de peau associé, et le masque après traitement pour élimination du bruit. Cependant, nous nous rendrons compte plus tard que finalement, le masque original avait de meilleurs résultats que cette version améliorée.

II.iii Matching de descripteurs

Pour pouvoir comparer la main capturée avec les images de références, nous extrayons des descripteurs en utilisant l'algorithme SIFT. Notre choix s'est porté sur ce dernier de par sa facilité d'implémentation avec OpenCV, et le fait qu'il puisse à la fois détecter des points d'intérêts et construire des descripteurs à partir de ces derniers. Nous convertissons d'abord les images en noir et blanc et mettons les images de références à l'échelle de l'image capturée. Puis une fois que nous avons récupéré les descripteurs de l'image capturée et ceux des images de références (en appliquant le masque de peau). Nous comparons les features en utilisant un Brute-Force Matcher [5], qui se contente de matcher les descripteurs des deux images qui sont les plus proches selon une norme L2 (en cross-match, c'est-à-dire qu'on ne considère qu'il y a match que si les deux descripteurs se matchent mutuellement). C'est ainsi qu'on arrive à définir des points de similarités entre les deux images. Notre prédition est donc la lettre associée à l'image de référence qui a le plus grand nombre de matching avec la main capturée.

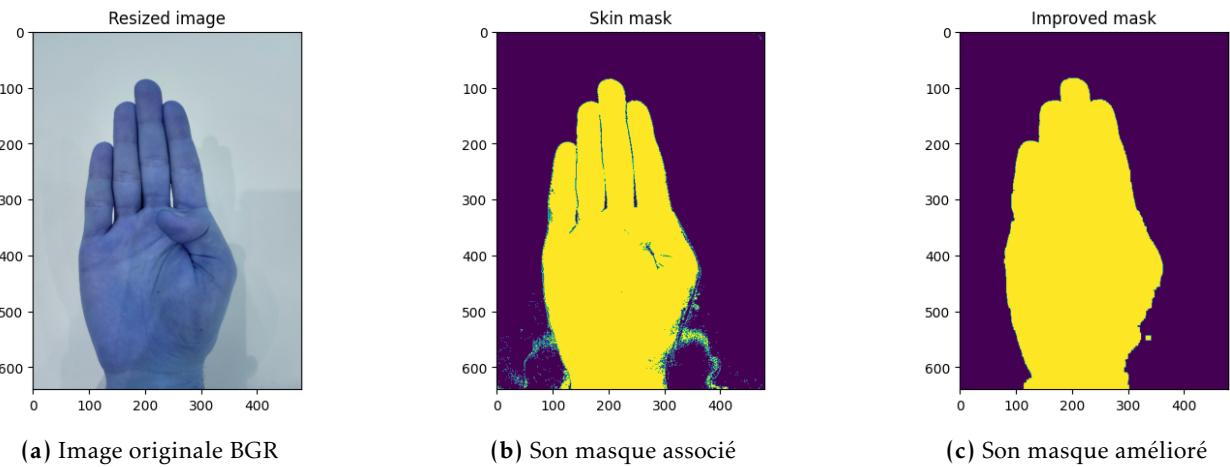


Figure 4: Exemple d'une des images de référence et de son masque de peau associé

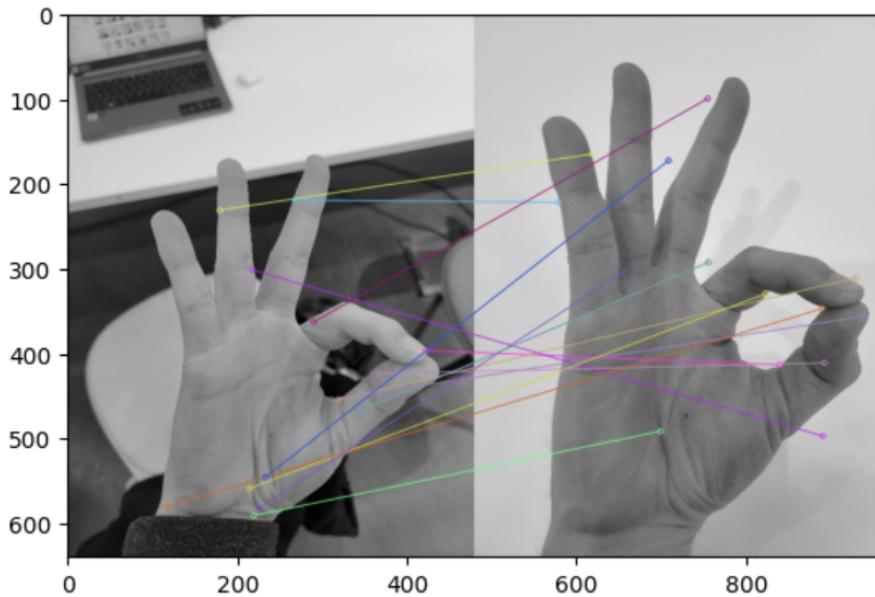


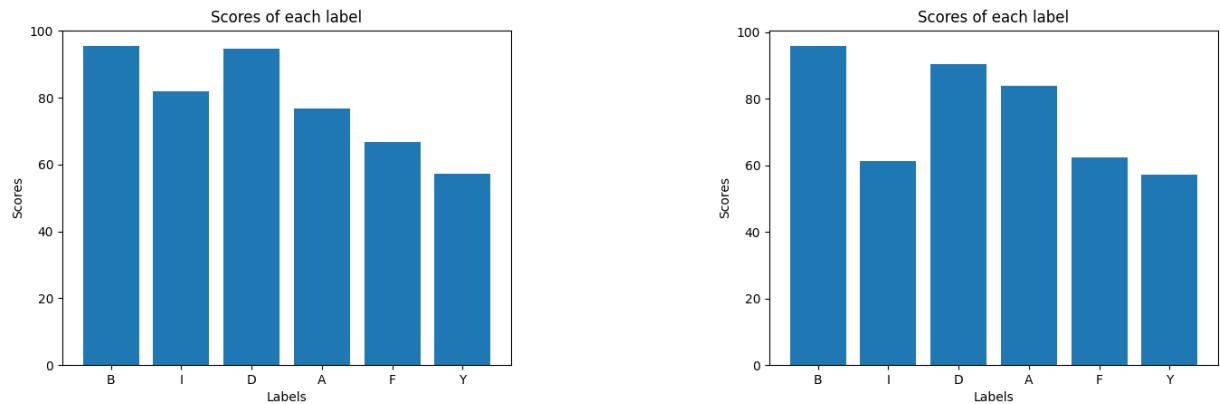
Figure 5: Matching avec SIFT et BFMatcher entre une image "réelle" et une image de référence représentant le signe de la lettre F. On peut voir que le matching ne se fait pas toujours sur les mains, cela est dû à l'imprécision des masques de peau

III Résultats

Lien vers la vidéo démo:

<https://drive.google.com/file/d/1nG91JygNS6HXB6B3Rm2gMBoNi-3RmjDd/view?usp=sharing>

Les résultats obtenus lors de cette étude n'ont pas été très satisfaisants. Initialement, nous avions envisagé d'utiliser le pourcentage de labels correctement identifiés comme métrique de référence. Cependant, cette approche s'est avérée peu efficace, le taux de détection correcte étant quasiment toujours proche de zéro. Pour avoir tout de même une métrique exploitable, nous avons modifié notre approche d'évaluation en nous concentrant davantage sur le nombre de points d'intérêts correctement identifiés. En effet, nous avons observé que les images réussissant à identifier correctement le label étaient celles où il y avait le plus grand nombre de correspondances (matchs) avec ce label. Ainsi, la précision de matching a été recalculée comme le nombre de points ayant correctement matché divisé par le nombre de points maximal ayant matché pour l'image ayant le meilleur score. Par exemple, si une image où le bon label matche 80 points et qu'un autre label obtient un score plus élevé avec 100 points de matchés, alors la précision pour cette image est de 80%. Une précision de 100% signifie que c'est le bon label qui a obtenu le plus de matchs. Cette nouvelle métrique nous a permis de mieux évaluer l'efficacité de notre méthode malgré les faibles performances initiales en termes de détection de labels corrects.



(a) Résultats avec un échantillon de test utilisant un masque simple.

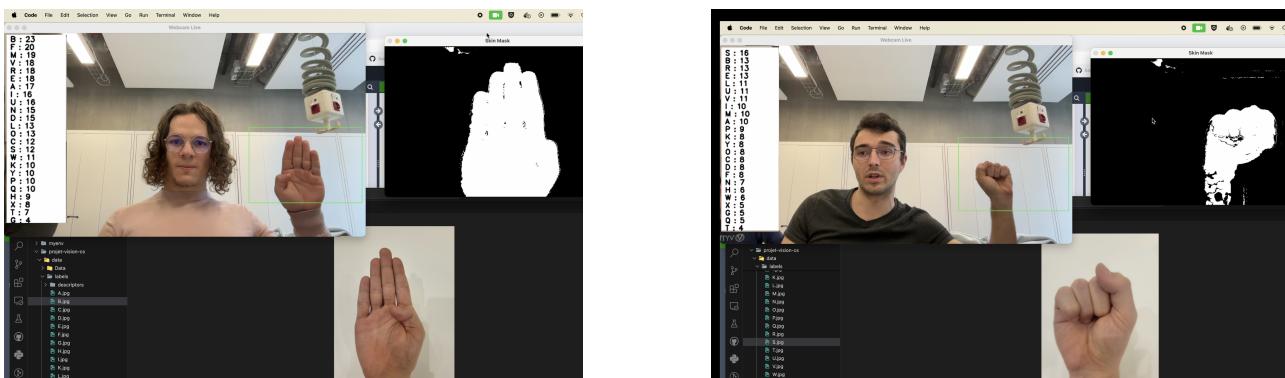
(b) Résultats avec un échantillon de test utilisant un masque amélioré.

Figure 6: Comparaison des résultats obtenus avec différents types de masques.

Finalement, le score de précision moyen dans le cas où nous utilisons le masque de peau sans le retravailler est de 69,1% tandis que dans la version où nous retravaillons le masque pour tenter de l'améliorer en comblant les trous ou en enlevant le bruit, le score moyen est de 65,8%.

Nous avons appliqué cette pipeline à un flux vidéo en temps réel. Cependant, l'opération la plus coûteuse en temps de calcul est le calcul du masque de peau. Cette opération, sur des images avec une résolution de 600 par 400, prend quelques secondes par image. Par conséquent, notre vidéo est extrêmement saccadée.

Notre idée afin d'améliorer la vitesse de nos algorithmes est de calculer à l'avance les descripteurs SIFT de nos images de référence et de les stocker. De cette manière, nous n'avons plus besoin de lire les images de références, calculé leur masque et leurs descripteurs. Nous ne les calculons plus qu'au premier passage et nous les stockons dans une variable globale pour les images suivantes. Nous utilisons la variable globale, car elle nous permet de stocker les données en mémoire. Nous avions aussi essayé en enregistrant dans un dossier les descripteurs, mais l'écriture et lecture est aussi longue et moins performante que de stocké les données en mémoire. De plus, notre variable n'est pas très lourde en termes de taille donc c'est une approche convenable. Au final, nous avons réussi à réduire d'un facteur 5 le temps de calcul pour une image, ce qui améliore énormément la fluidité de notre vidéo.



(a) Lettre B.

(b) Lettre S.

Figure 7: Extrait vidéo temps réel, avec classement par correspondance.

Même si nous n'obtenons que très rarement le bon label, nous remarquons tout de même une tendance qui classe le bon label dans le top 5 des labels les plus probables. Il ne s'agit donc pas de hasard. De plus, nous voyons déjà d'autres axes d'améliorations possibles qui pourraient augmenter considérablement nos performances.

IV Autres tests et axes d'amélioration

Dans cette section, nous allons présenter les autres éléments de code que nous avons essayé d'implémenter ainsi que les potentiels axes d'améliorations.

Pour commencer, nous aimerais noter que nous avons tenté d'égaliser l'histogramme de la composante de luminance de nos images après les avoir convertis en Lab. Le but étant d'uniformiser les contrastes de nos images et de voir si cela pouvait améliorer les performances de notre méthode. Il se trouve qu'appliquer cette égalisation rendait notre détecteur de peau basé sur des histogrammes complètement inefficace. Nous avons donc retiré cette opération de notre pipeline finale.

D'autres tests ont rapidement été tentés. Nous avons voulu voir si ORB fonctionnait bien par rapport à SIFT. Sa vitesse d'exécution supérieure aurait pu être un avantage, mais il ne performait pas mieux. Au contraire sa précision était bien moins bonne.

Un dernier point d'amélioration aurait été de diversifier le contexte d'acquisition des images. En effet, en prenant un plus grand nombre de photos dans des environnements variés, notamment avec différents éclairages, nous aurions pu mieux évaluer la robustesse de nos algorithmes. Cette démarche aurait permis de rendre nos tests plus représentatifs et d'obtenir des scores de précision plus fiables et significatifs. De plus, en regroupant les tests par environnements similaires (par exemple, même type de lumière, même couleur de peau, même distance entre la caméra et le sujet), nous aurions pu extraire des informations plus précises sur les éléments influençant directement les performances de nos méthodes. Cette analyse détaillée aurait été bénéfique pour identifier spécifiquement ce qui fonctionne bien ou moins bien, permettant ainsi de cibler plus efficacement les améliorations futures.

V Conclusion

Nous concluons ce projet avec un sentiment partagé de satisfaction et de regret. D'un côté, nous sommes fiers de ce que nous avons accompli et des connaissances que nous avons acquises en explorant des approches non conventionnelles de la vision par ordinateur sans recourir à l'apprentissage profond. Ce projet nous a non seulement permis de renforcer nos compétences techniques, mais aussi de comprendre en profondeur les défis liés à la reconnaissance des signes de la langue des signes en temps réel. D'un autre côté, nous regrettons de ne pas avoir disposé de plus de temps pour explorer davantage d'améliorations potentielles que nous avons identifiées au cours de nos recherches. Avec plus de temps, nous aurions aimé implémenter ces améliorations pour voir jusqu'à quel point nous pourrions augmenter la précision de notre système. Malgré ces contraintes, nous sommes enthousiastes à l'idée de poursuivre cette recherche, armés de nouvelles idées et d'une meilleure compréhension des problématiques à adresser.

References

- [1] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60:91–110, 2004.
- [2] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.
- [3] Vladimir Vezhnevets, V. Sazonov, and Alla Andreeva. A survey on pixel-based skin color detection techniques. In GRAPHICON. GRAPHICON-2003, 2003.
- [4] Yogam Pratheepan. The pratheepon dataset. http://web.fsktm.um.edu.my/~cschan/downloads_skin_dataset.html. Accessed: date-of-access.
- [5] Opencv documentation: Introduction to sift (scale-invariant feature transform). https://docs.opencv.org/4.x/dc/dc3/tutorial_py_matcher.html.