Apprentissage incrémental de la saillance visuelle pour des applications robotique
Incremental learning of visual saliency for robotics applications

Céline Craye[1,2] David Filliat[1] Jean-François Goudou[2]

[1] ENSTA Paristech -INRIA FLOWERS team,
Unité informatique et Ingénierie des Systèmes, 828 boulevard des Marechaux,
91762 Palaiseau, France [2] Thales - SIX - Theresis / Vision & Sensing
1, avenue Augustin Fresnel
91767 Palaiseau

celine.craye@ensta-paristech.fr

Résumé

Nous proposons une méthode d'apprentissage incrémental de la saillance visuelle par un mécanisme d'exploration de l'environnement. Partant d'une définition géométrique de la saillance des objets, notre système observe de façon attentive et ciblée son environnement, jusqu'à découvrir des éléments saillants. Un classifieur permet alors d'apprendre les caractéristiques visuelles correspondantes afin de pouvoir ensuite prédire rapidement les positions des objets sans analyse géométrique. Notre approche a été testée sur des images RGBD, fonctionne en temps réel et dépasse plusieurs méthodes de l'état de l'art sur le contexte particulier de la détection d'objets en intérieur.

Abstract

We describe a method to learn visual saliency incrementally using an environment exploration mechanism. We define saliency in a geometrical manner and use this definition to discover salient elements given an attentive and selective observation of the environment. Then, a classifier learns the visual features corresponding to salient objects within their environment. Our approach has been tested on RGBD images, is real time, and outperforms several state-of-the-art methods in the case of indoor object detection.

1 Introduction

L'exploration visuelle, la découverte, l'identification et la recherche d'objets dans un environnement quelconque par des robots mobiles restent des problèmes complexes. Dans le cas de scènes simples et restreintes, d'objets proches et suffisamment séparés, le problème est quasiment résolu. Cependant, le cas où les objets sont situés loin du robot, posés ou entassés sur des tables ou des étagères reste ouvert et est largement étudié aujourd'hui [20]. Une stratégie d'attention visuelle efficace est alors indispensable, et l'utilisation du contexte peut être d'une aide précieuse afin de localiser *a priori* des objets dans leur environnement.

Les méthodes d'apprentissage sont désormais incontournables dans le domaine de la détection ou de la reconnaissance d'objets dans des images. Cependant, cet apprentissage est le plus souvent réalisé d'un seul bloc et n'est pas forcément flexible face à de nouveaux environnements ou de nouvelles prises de vues. Au contraire, on se propose ici d'effectuer un apprentissage en continu sur un robot, système actif qui, grâce à des capacités de recherche et d'exploration, peut faire preuve d'adaptation face à de nouvelles situations ou de nouveaux environnements. La robotique développementale, puisant son inspiration dans les mécanismes de développement des enfants, offre des possibilités d'apprentissage incrémental basé sur l'expérience et l'exploration.

Contrairement à des approches classiques de détection d'objets où l'ensemble de l'image est traité avec la même importance dans un ordre quelconque [22], la vision humaine est sélective et séquentielle. Cela signifie que les éléments du champ de vision sont traités tour à tour selon leur importance. Ceci se traduit par un déplacement du regard vers une zone de l'espace jugée intéressante (ou saillante) et l'acquisition d'informations fortement résolues grâce à la vision fovéale de l'œil. Notre approche s'inspire de ce mécanisme par une exploration séquentielle de l'environnement et une alternance entre vision fovéale et vision champ large.

Nos travaux visent à concevoir un mécanisme capable d'apprendre la saillance dans un environnement donné

de manière autonome et incrémentale en partant d'une définition simple du concept de saillance liée aux objets en intérieur. De plus, nous nous appuyons sur la donnée d'un champ fovéal étroit mais riche en information, et celle d'un champ large moins riche. Notre apprentissage va permettre d'apprendre la saillance sur le champ large, et ainsi de diriger et d'utiliser la partie fovéale de plus en plus judicieusement. Dans une première phase, notre agent explore son environnement sans *a priori*, de manière sélective et attentive en utilisant la vision fovéale, et découvre au fur et à mesure des éléments d'intérêt dans son environnement. Par un mécanisme d'apprentissage, il peut alors apprendre les caractéristiques visuelles du champ large correspondant aux éléments saillants découverts. Après cette phase d'apprentissage, l'agent est capable de construire une carte de saillance dédiée à l'environnement dans lequel il se trouve et peut alors l'utiliser pour d'autres tâches.

Cet article est organisé de la manière suivante. Dans la section 2, nous effectuons une revue de l'état de l'art et exposons nos contributions au domaine. La section 3 décrit plus en détail la méthode que nous avons élaborée. Nous présenterons ensuite nos résultats expérimentaux dans la section 4 et terminons l'article par une conclusion et les perspectives.

2 Etat de l'art

Il existe de nombreux modèles d'attention visuelle, pour des applications très diverses. La récente revue de littérature de Borji et Itti [4] en couvre un échantillon important, sans pour autant prétendre être exhaustive. Nous ne nous focaliserons dans cet article que sur l'attention visuelle liée aux applications robotiques. De manière générale, les mécanismes d'attention visuelle passent par la détermination d'une carte de saillance, représentant l'importance d'un stimuli visuel par rapport à son environnement. Par la suite, le maximum global de cette carte est localisé, et l'attention est alors portée vers ce maximum, ce qui se traduit éventuellement par un mouvement mécanique (saccade, zoom ou déplacement) dans le cas de sytèmes robotiques.

Les cartes de saillances sont généralement obtenues à partir de caractéristiques visuelles simples, applicables sur l'ensemble de l'image à traiter. A la fin des années 90, Itti et Koch proposèrent un modèle de saillance avec des caractéristiques basées sur les orientations, l'intensité et les oppositions de couleurs centre-périphérie, par analogie aux traitements rétiniens et corticaux [12]. Harel et al. [10] ont proposé un modèle biologiquement plausible basé sur une normalisation de caractéristiques RGB à partir de convergence de graphes. D'autres caractéristiques peuvent être obtenues par multiple seuillage des images [23], l'utilisation de données statistiques comme la covariance [9], ou l'utilisation de bases de décomposition en cosinus discrets [11]. Il est également possible d'utiliser des caractéristiques temporelles comme le flux optique dans le cas de vidéos [15], ou bien d'exploiter la profondeur dans le cas d'images RGBD [18]. Suivant le cas, les caractéristiques doivent être combinées pour pouvoir produire une carte de saillance. On peut alors se servir de statistiques sur de larges bases de données pour déterminer des poids optimaux [24] ou bien réaliser des post-traitements pour réajuster la prédiction de la saillance [16]. Enfin, certaines méthodes utilisent directement des algorithmes d'apprentissage sur des bases de données disponibles pour apprendre et prédire la saillance dans une image [21].

Les protocoles d'évaluation récents évaluent la qualité d'une carte de saillance par rapport à des statistiques d'observations humaines. Ainsi, la carte de saillance obtenue est comparée à une carte de saillance de référence, représentant la probabilité qu'un être humain fixe tel ou tel point de l'image. Le MIT *saliency benchmark* [6] évalue ainsi des dizaines de méthodes suivant plusieurs métriques.

En robotique, les applications pour lesquelles on utilise des méthodes d'attention visuelle sont en général liées à de la détection, de la reconnaissance ou de la découverte d'objets [3], l'estimation de la pose de certains objets [2] ou de la cartographie d'environnements [7]. Dans tous les cas, la notion de saillance est ici intimement liée à la notion d'objets. Partant de ce constat, il est intéressant d'obtenir des cartes de saillance dédiées aux objets, en fonction de la tâche que l'on souhaite accomplir (saillance descendante) et non aux stimuli visuels de manière générale (saillance ascendante). De nombreux systèmes robotiques ajoutent une approche *top-down* à leur modèle de saillance pour l'adapter à la tâche à accomplir. Par exemple, l'utilisation du contexte dans lequel l'objet recherché doit se trouver [13] peut permettre une exploration plus efficace. Il est aussi possible de moduler des cartes de saillance ascendante soit en excitant ou inhibant les caractéristiques propres à l'objet recherché ou son environnement [24]. Le principe de la vision fovéale a été utilisé à plusieurs reprises pour effectuer de l'identification ou de la découverte d'objets. Elle nécessite des systèmes de prise de vue adaptés, par exemple des têtes anthropomorphiques munies de caméras champ large et champ étroit ainsi que d'un système mécanique d'orientation de la tête [2], ou bien plus simplement des caméras PTZ [7].

Bien que le MIT *saliency benchmark* permette d'apprécier les performances de nombreuses approches, son évaluation reste dans une optique purement ascendante. De plus, la base d'images servant à l'évaluation est biaisée de par leur point de vue, car la plupart des images sont centrées sur les objets d'intérêt. Ce biais représente un problème courant en robotique, car les prises de vue sont liées à la position et aux mouve-

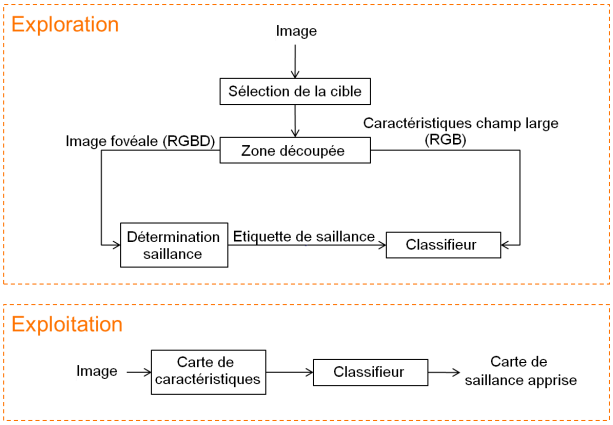


FIGURE 1 – Architecture de notre système.

ments du robot, et elles ne sont donc pas - du moins dans un premier temps - liées à l'intérêt des éléments constituant l'environnement.

Contributions

Jusqu'à présent, l'attention visuelle en robotique n'est utilisée que comme pré-requis à une autre tâche. Comme expliqué plus tôt, on part généralement de cartes de saillance ascendante génériques que l'on module éventuellement au fur et à mesure des observations qui sont faites. À notre connaissance, aucune méthode d'apprentissage incrémental de la saillance pour une tâche dédiée n'a été proposée. Nous proposons dans cet article une méthode d'apprentissage de la saillance directement au sein de l'environnement dans lequel évolue le robot. Pour cela, nous utilisons un mécanisme d'observation fovéale, donnant accès à une information riche et fiable, et d'un champ d'observation plus large pour lequel l'information disponible est moins riche mais plus rapide à traiter. L'idée est d'utiliser l'observation fovéale pour détecter avec certitude des éléments saillants, puis d'apprendre les caractéristiques visuelles de ces éléments sur l'information disponible sur le champ large. Après une phase d'apprentissage, il sera possible de détecter les éléments saillants à partir du champ large.

3 Méthode proposée

Nous décrivons dans cette section notre mécanisme d'apprentissage incrémental de la saillance visuelle dans un environnement donné (intérieur, avec de nombreux objets).

3.1 Architecture générale

L'architecture générale de notre système est présentée à la Figure 1. L'idée est d'entrainer le système dans une phase d'exploration, puis d'utiliser le modèle dans une phase d'exploitation ultérieure afin d'obtenir une carte de saillance spécifique pour une tâche donnée dans l'environnement qui a été préalablement exploré.

Dans la phase d'exploration, une zone de l'image d'entrée est tout d'abord sélectionnée. Nous nous contentons pour l'instant d'une sélection aléatoire, mais celle-ci sera par la suite choisie dans le but d'accélérer l'apprentissage et potentiellement de l'améliorer. Des méthodes de sélection d'actions basées sur le progrès d'apprentissage [17] pourraient permettre un telle sélection.

Nous utilisons le principe de la vision fovéale sur la zone sélectionnée et devons pour cela accéder à une information plus riche dans cette portion de champ. Pour les besoins de l'évaluation, nous utilisons ici des portions d'images RGBD pour l'image fovéale, et la composante RGB de l'image entière pour le champ large. On pourrait toutefois imaginer un système où l'image fovéale serait, par exemple, une image plus résolue associée à une méthode de détection d'objets.

Cette information de profondeur va nous permettre de déterminer si la zone que nous étudions est saillante ou non. Pour le savoir, une méthode de détermination géométrique des éléments saillants a été mise en place et sera plus amplement décrite à la Section 3.2. Si les critères géométriques (accessibles par la donnée de la profondeur) sont compatibles avec la définition de saillance, la zone observée est considérée comme saillante et envoyée au classifieur sous l'étiquette saillante (ou non saillante dans le cas contraire). Notons également que cette opération de détermination de saillance basée sur la profondeur est relativement coûteuse et ne pourrait être appliquée directement sur l'image entière en temps réel.

Enfin, le classifieur prend en entrée des caractéristiques simples et peu coûteuses basées uniquement sur la composante RGB de l'image, ainsi que l'étiquette qui a été attribuée grâce à la donnée de la profondeur. Les détails de l'implémentation seront donnés à la Section 3.3.

3.2 Découverte d'éléments saillants

Dans cet article, les éléments saillants sont des objets de taille restreinte (jusqu'à une trentaine de centimètres) ayant la propriété d'être posés sur une surface plane. Cette définition permet de caractériser une bonne partie des objets dans un environnement statique, sans interaction humaine. Les critères de taille ont été spécifiés de manière arbitraire pour concorder avec les bases de données disponibles, mais ils pourraient tout à fait être réajustés sans dégradation des performances du système. Enfin, suivant le contexte, il est relativement simple de modifier la définition d'objets saillants soit avec d'autres caractéristiques que la profondeur (par exemple une image hautement résolue, une analyse du mouvement, etc.), soit avec d'autres contraintes géométriques.

Notre algorithme a été conçu pour fonctionner avec des images RGBD obtenues par une caméra Kinect,

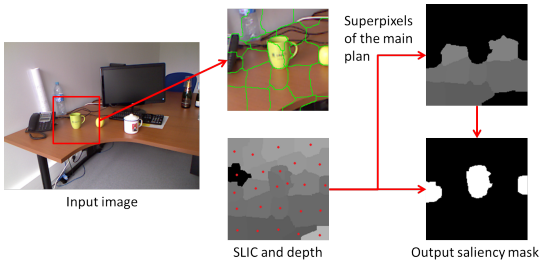


FIGURE 2 – Pricipales étapes de la découverte d'éléments saillants.

mais également sur d'autres types de dispositifs donnant une information de profondeur même très faiblement résolue. Entre autres, il est possible d'utiliser un télémètre laser capable de pointer successivement des points du champ de vue de la caméra. L'action mécanique déplaçant la position du télémètre demandant en soi quelques centièmes de secondes, il n'est alors possible que d'utiliser un nombre limité de points.

En partant d'un point de l'image, nous isolons une zone de l'image centrée sur ce point. Étant donnée la profondeur du pixel central, nous dimensionnons cette zone pour qu'un objet à cette distance puisse être vu dans son intégralité. Puis, nous trouvons des superpixels SLIC [1] (25 dans notre cas) sur cette zone de l'image. La profondeur de chaque centroïde de superpixel est ensuite récupérée, de sorte que nous obtenons un nuage de points, que nous convertissons dans un repère orthonormé. Si la zone sélectionnée est saillante, elle doit comporter une surface plane ainsi qu'au moins un élément posé sur celle-ci. Par conséquent, nous recherchons tout d'abord dans le nuage de points les plans potentiels, puis les éléments posés sur le plan.

Pour chaque triplet de points pris dans ce nuage, nous déterminons l'équation du plan passant par ces points. Nous regardons ensuite si ce plan est physiquement plausible et s'il correspond effectivement à une surface plane dans l'image. Pour ce faire, nous calculons la distance algébrique de chaque point au plan. Pour que le plan soit plausible, il ne doit occulter aucun élément proche (les objets potentiels) et comporter au moins cinq points appartenant au plan. Si au moins un plan répond à ces critères, nous recherchons les éventuels objets saillants posés dessus. Il s'agit alors des superpixels correspondant aux points à moins d'une vingtaine de centimètres du plan. La Figure 2 montre les grandes étapes de l'algorithme.

Cette définition est certes restrictive, mais elle ne produit que très peu de fausses alarmes. Si un élément saillant n'est pas détecté lors d'un passage, il finira par l'être après plusieurs observations successives à des points de vue légèrement différents.

3.3 Apprentissage de la saillance

La découverte d'éléments saillants se fait par un processus d'observation long et coûteux. Nous souhaitons maintenant pouvoir exploiter l'information extraite de cette phase d'observation pour pouvoir reconstuire la saillance à partir de données beaucoup moins lourdes à traiter. Pour ce faire, nous choisissons d'utiliser les mêmes caractéristiques visuelles qu'Itti et Koch [12]. Il s'agit d'un ensemble de filtres comprenant des réponses à des différences de filtres de Gabor d'orientations différentes et d'échelles différentes. Plus précisément, chaque filtre est obtenu par l'équation 1

$$O(\sigma_f, \sigma_c, \theta) = |Gabor(\sigma_c, \theta) - Gabor(\sigma_f, \theta)| \quad (1)$$

où $\theta \in \{0, 45, 90, 135\}$ est l'orientation, $\sigma_f \in \{2, 3, 4\}$ est l'échelle du centre et $\sigma_c \in \{\sigma_f + 1, \sigma_f + 2\}$ celle de la périphérie, soit un total de 24 filtres. Une autre série de filtres est utilisée, représentant les différences d'intensité entre le centre et la périphérie obtenus par l'équation 2

$$I(c, f) = |I(c) - I(f)| \quad (2)$$

où I représente une pyramide Gaussienne créée à partir de l'image en niveaux de gris. $f \in \{2, 3, 4\}$ et $c \in \{f + 1, f + 2\}$ représentent les échelle de cette pyramide, soit un total de 6 filtres. Enfin, on obtient des oppositions de couleur centre-périphérie par les équations 3 et 4

$$RG(c, f) = |(R(f) - G(f)) - (G(c) - R(c))| \quad (3)$$

$$BY(c, f) = |(B(f) - Y(f)) - (Y(c) - B(c))| \quad (4)$$

où R G B Y représentent les pyramides Gaussiennes obtenues à partir des composantes rouges, vertes, bleues et jaunes de l'image, aux même échelles f et c que pour le niveau de gris, soit 12 filtres au total.
Ces 42 caractéristiques ne requièrent que l'image RGB et sont calculables en quelques millisecondes. Ce sont donc elles qui vont nous permettre de calculer notre carte de saillance dans la phase d'exploitation.
Chaque nouvelle observation produit une segmentation de la zone observée, séparant les superpixels détectés comme saillants et non saillants. Grâce à ce masque, nous pouvons générer une série d'exemples qui serviront à notre apprentissage. Dans un premier temps, nous découpons la zone observée en carrés de quelques pixels. Dans notre cas, pour des images 640×480 nous utilisons des carrés de 20 pixels, valeur empirique qui s'est avérée être un bon compromis entre vitesse de calcul et précision. Nous associons chaque carré avec une étiquette saillant ou non saillant suivant que le masque dans la zone du carré est majoritairement saillant ou non. Ensuite, nous associons

à cette étiquette un vecteur de caractéristique représentant la valeur moyenne des caractéristiques Itti et Koch à l'intérieur du carré. L'ensemble des caractéristiques obtenues de cette manière est ajouté aux précédents exemple, et le classifieur est alors ré-entrainé. Nous avons choisi d'utiliser un classifieur à base de forêts aléatoires [5] en raison de leurs bonnes performances, leur rapidité d'apprentissage et leur capacité à gérer des données déséquilibrées (en pratique, nous avons beaucoup moins d'exemples d'éléments saillants que non saillants). Dans notre cas, nous utilisons une forêt de 20 arbres de profondeur 10.

Dans la phase d'exploitation, nous construisons la carte de saillance de la manière suivante : Nous calculons des cartes de caractéristiques Itti et Koch pour l'image entière, puis nous découpons l'image en carrés de même taille que pour l'entraînement, et calculons la valeur moyenne des caractéristiques pour chaque carré. La valeur moyenne de chaque carré est ensuite envoyée au classifieur qui produit un score de sortie dépendant de la saillance de l'objet. Ce score est finalement associé à chaque pixel de l'image pour construire la carte de saillance.

4 Résultats expérimentaux

Dans cette section, nous présentons les résultats obtenus sur des images de Kinect. Nous utilisons ce support car de nombreuses bases de données sont disponibles et il est donc plus simple de se comparer à l'état de l'art. Toutefois, notre algorithme est portable sur d'autres types de dispositifs, dès lors qu'une information de profondeur, même très faiblement résolue, est accessible.

4.1 Comparaison à l'état de l'art

Il existe quelques bases de données dédiées à la saillance ascendante sur des images RGBD [8,18]. Toutefois, toutes les images de ces bases ne sont pas compatibles avec notre définition de la saillance (images prises en extérieur, objets très grands, absence de surface plane). Par conséquent, l'évaluation de notre méthode directement sur ces données n'a pas de sens. D'autres bases existent, dédiées à la reconnaissance d'objet ou à l'estimation de leur pose [14,19]. Dans ce cas, les objets présentés correspondent à notre définition, mais les annotations ne sont pas forcément adaptées à l'évaluation de saillance (utilisation de boîtes, éléments saillants non considérés, etc.). Pour cette raison nous réalisons nos tests sur un sous ensemble de la base GIT (15 images) [8] pour laquelle nous retirons les images incompatibles avec notre définition, puis sur un sous ensemble de *RGBD scenes dataset* [14] (92 images) pour lequel nous ajoutons aux annotations les objets non référencés dans leur liste, mais bien présents sur les images et saillants au sens de notre définition.

Nous comparons notre méthode avec quatre autres méthodes. BMS [23] et GBVS [10] sont parmi les méthodes les plus performantes selon le MIT *saliency benchmark* [6] pour des images RGB. Peng et al. [18] correspond à l'état de l'art sur des images RGBD, et Itti et Koch [12] est utilisée comme référence car nous utilisons les même caractéristiques visuelles, en les combinant différemment.

Pour chaque image, nous réalisons 1000 observations aléatoires de différents points de l'image pour entraîner notre classifieur. Nous entraînons un classifieur dédié pour chaque image, ce qui nous permet de valider les performances de notre architecture et de notre définition géométrique de la saillance.

La Figure 3 montre quelques résultats qualitatifs. On constate que notre approche sélectionne non seulement des zones saillantes tout à fait cohérentes, mais elle permet également d'obtenir des informations bien plus marquées sur la forme et la dimension des objets saillants. Ceci peut s'avérer utile dans la phase d'exploitation.

D'un point de vue quantitatif, la Figure 4 présente les courbes ROC des cinq approches citées plus haut, pour les bases GIT et *RGBD-scenes*. Chaque courbe est obtenue en appliquant une série de seuils à la carte de saillance et en obtenant une matrice de confusions à partir de l'ensemble des pixels de l'image. Nous obtenons les meilleurs résultats, ce qui était prévisible, car la saillance que nous apprenons est optimisée pour la définition que nous en avons, et que les classifieurs se spécialisent pour un environnement donné.

4.2 Exploration et exploitation

L'évaluation précédente ne permet pas d'évaluer la capacité de généralisation de notre méthode face à un point de vue ou bien des éléments légèrement différents de la même scène. En effet, nous nous entrainons sur une image et réalisons la carte de saillance à partir de la même image.

Pour démontrer que notre méthode n'est pas sujette au sur-entrainement et qu'elle fonctionne toujours dans une configuration différente, nous utilisons la séquence *table-small-2* de RGBD-scenes. RGBD-scenes est organisée sous forme de séquences vidéo de quelques centaines de frames dans un environnement fixe mais avec des changements de point de vue. Dans la séquence sélectionnée, nous considérons une quarantaine de frames et sélectionnons aléatoirement la moitié d'entre elles pour l'entrainement, et le reste pour le test.

La Figure 5 montre les courbes ROC des 4 méthodes utilisées à la section 4.1, ainsi que les performances de notre algorithme pour un classifieur dédié à chaque image (*train == test*) et un classifieur global entrainé sur une partie de la séquence et testé sur une autre (*train != test*). On constante que notre méthode est plus performante que les autres, et, de manière sur-

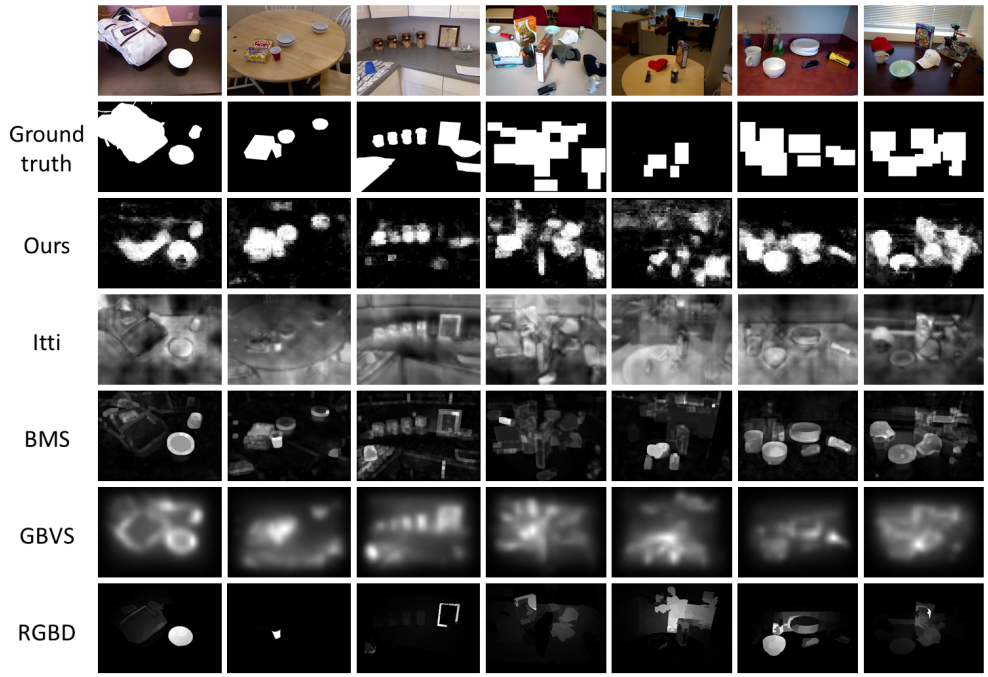


FIGURE 3 – Quelques cartes de saillance pour les cinq méthodes testées.

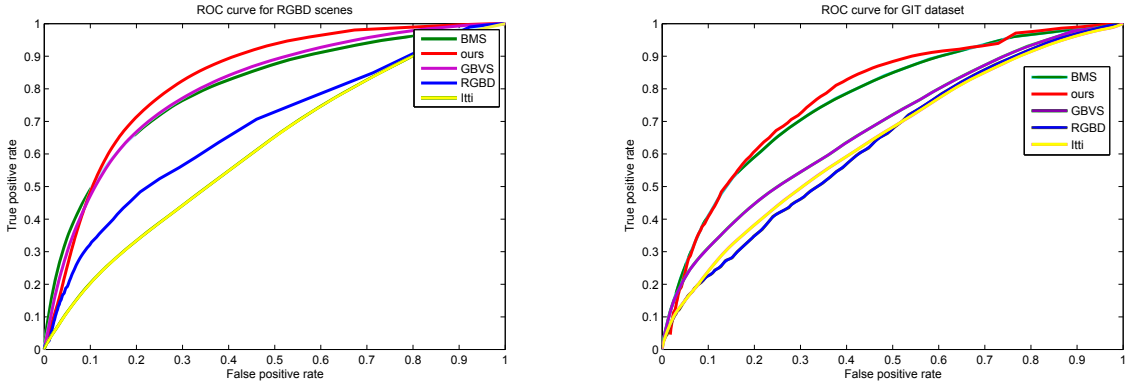


FIGURE 4 – Courbes ROC pour cinq approches différentes, sur les bases GIT et RGBD-scenes.

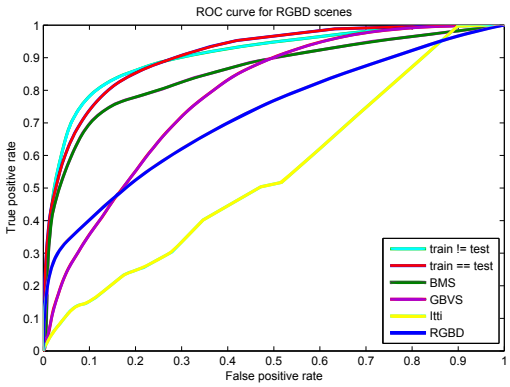
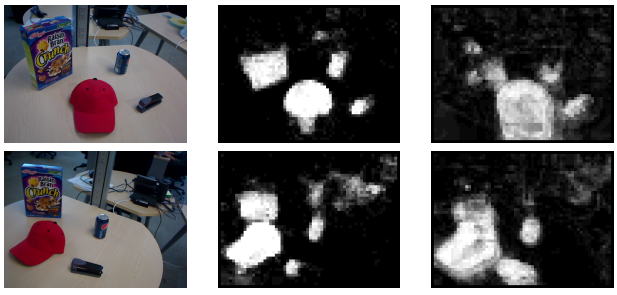


FIGURE 5 – Courbes ROC sur la base RGBD-scenes en entraînant un classifieur dédié pour chaque image (*train == test*), puis en entraînant un classifieur sur une portion de la séquence et testé sur une autre (*train != test*).



Color image (a) (b)

FIGURE 6 – Cartes de saillance en configuration (a) Train = test et (b) Train ≠ test. Bien que la carte (b) soit plus bruitée, l'apprentissage sur plusieurs images permet de trouver des zones saillantes non détectables dans l'image courante.

prenante, l'entrainement sur des images différentes conduit à de meilleures performances en terme de vrais positifs. La Figure 6 permet d'expliquer intuitivement ce résultat. On constate sur ces images que des portions d'objets son manquantes dans la configuration *train = test* (casquette, agrafeuse), probablement parce que ces portions d'objets ne répondaient pas aux critères géométriques de saillance dans cette configuration bien précise. Ces portions sont toutefois trouvées dans la configuration train $\neq$ test, puisque la variété des points de vue a permis de les détecter correctement. En contrepartie, la carte de saillance est plus bruitée, ce qui conduit à un taux de faux positifs plus élevé.

4.3 Implémentation temps réel

Pour finir, nous avons implémenté notre algorithme afin qu'il puisse acquérir en temps réel le flux RGBD d'une Kinect, réaliser les observations image par image, et entraîner un classifieur en ligne dans un *thread* parallèle à mesure que les observations sont réalisées. Nous avons pour cela utilisé les librairies OpenCV et Freenect, ainsi que QT pour la gestion des threads et l'interface graphique. Notre implémentation a été testée sous Ubuntu 10.12 sur un processeur Intel Core i3-3240, CPU cadencé à 3.4GHz quadcore. Nous obtenons les temps d'exécution suivants durant les différentes étapes de l'algorithme :

- Observation fovéale et découverte d'éléments saillants : 50 à 500ms selon la taille de la zone observée.
- Entrainement de la forêt aléatoire (faite sur un thread séparé) : 10 secondes pour 100 000 exemples
- Calcul des cartes de caractéristiques de type Itti et Koch : 90ms
- Construction de la carte de saillance une fois les caractéristiques obtenues : 30ms

On constate donc que le calcul de la carte de saillance est à peine plus long (90ms+30ms) que pour une carte de type Itti et Koch (90ms), et donc capable de traiter les images à 8Hz sans optimisation particulière. De plus, l'observation fovéale peut être relativement coûteuse, mais elle n'est présente que dans la phase d'exploration. Enfin, l'entrainement de la forêt aléatoire dans un *thread* séparé permet de poursuivre l'acquisition de nouveaux exemples sans nuire à la vitesse d'exécution.

La figure 7 montre quelques étapes de l'apprentissage de la carte de saillance après un certain nombre d'itérations. Des vidéos de démonstration montrant l'évolution de l'apprentissage et la robustesse aux mouvements sont disponibles en ligne [1].

1. http ://perso.ensta-paristech.fr/∼ craye/

5 Conclusions et perspectives

Dans cet article, nous avons proposé une approche pour l'apprentissage incrémental de la saillance. Nous utilisons le concept d'attention visuelle et de vision fovéale pour acquérir et traiter des données très riches sur une petite portion du champ de vue, puis apprenons à identifier l'information extraite de ces données à partir de caractéristiques visuelles simples et peu coûteuses, applicable sur l'ensemble du champ de vue. Nos résultats montrent que cette approche est particulièrement appropriée lorsqu'une définition géométrique claire des éléments saillants est possible. Dans le cas d'objets posés sur des surfaces planes, nous avons montré que nous dépassons les performances de l'état de l'art. Notre méthode est également implémentée pour fonctionner en temps réel et est donc relativement bien adaptée à l'utilisation sur des plateformes robotiques. Par la suite, nous nous intéresserons à la stratégie de sélection des points d'observation. En effet, nous les sélectionnons pour le moment de manière aléatoire, mais il est également possible d'inclure des algorithmes de sélection de cibles maximisant les progrès. Cette approche devrait accélérer et améliorer l'apprentissage. D'autre part, nous souhaitons implémenter d'autres méthodes d'observation, comme l'utilisation de zoom permettant d'obtenir une meilleure résolution d'image sur une petite portion du champ visuel. Enfin, nous souhaiterions utiliser cet apprentissage de saillance sur une plateforme mobile, puis exploiter la saillance ainsi obtenue pour de la reconnaissance ou de la découverte d'objets.

Références

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11) :2274–2282, 2012.

[2] M Bjorkman and Danica Kragic. Combination of foveal and peripheral vision for object recognition and pose estimation. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 5, pages 5135–5140. IEEE, 2004.

[3] Mårten Björkman and Danica Kragic. Active 3d scene segmentation and detection of unknown objects. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 3114–3120. IEEE, 2010.

[4] Ali Borji and Laurent Itti. State-of-the-art in visual attention modeling. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(1) :185–207, 2013.

FIGURE 7 – Evolution de la saillance à mesure que de nouvelles observations sont faites. Le nombre d'observations est indiqué sous l'image.

[5] Leo Breiman. Random forests. *Machine learning*, 45(1) :5–32, 2001.

[6] Zoya Bylinskii, Tilke Judd, Frédo Durand, Aude Oliva, and Antonio Torralba. Mit saliency benchmark. http ://saliency.mit.edu/.

[7] José M Cañas, Marta Martínez de la Casa, and Teodoro González. An overt visual attention mechanism based on saliency dynamics. *International Journal of Intelligent Computing in Medical Sciences & Image Processing*, 2(2) :93–100, 2008.

[8] Arridhana Ciptadi, Tucker Hermans, and James M Rehg. An in depth view of saliency. In *Eds : T. Burghardt, D. Damen, W. Mayol-Cuevas, M. Mirmehdi, In Proceedings of the British Machine Vision Conference (BMVC 2013)*, pages 9–13, 2013.

[9] Erkut Erdem and Aykut Erdem. Visual saliency estimation by nonlinearly integrating features using region covariances. *Journal of vision*, 13(4) :11, 2013.

[10] Jonathan Harel, Christof Koch, and Pietro Perona. Graph-based visual saliency. In *Advances in neural information processing systems*, pages 545–552, 2006.

[11] Xiaodi Hou, Jonathan Harel, and Christof Koch. Image signature : Highlighting sparse salient regions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(1) :194–201, 2012.

[12] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11) :1254–1259, 1998.

[13] Thomas Kollar and Nicholas Roy. Utilizing object-object and object-scene context when planning to find things. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 2168–2173. IEEE, 2009.

[14] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011.

[15] Olivier Le Meur, Patrick Le Callet, and Dominique Barba. Predicting visual fixations on video based on low-level visual features. *Vision research*, 47(19) :2483–2498, 2007.

[16] Jia Li, Yonghong Tian, and Tiejun Huang. Visual saliency with statistical priors. *International Journal of Computer Vision*, 107(3) :239–253, 2014.

[17] P-Y Oudeyer, Frédéric Kaplan, and Verena Vanessa Hafner. Intrinsic motivation systems for autonomous mental development. *Evolutionary Computation, IEEE Transactions on*, 11(2) :265–286, 2007.

[18] Houwen Peng, Bing Li, Weihua Xiong, Weiming Hu, and Rongrong Ji. Rgbd salient object detection : A benchmark and algorithms. In *Computer Vision–ECCV 2014*, pages 92–109. Springer, 2014.

[19] Andreas Richtsfeld, Thomas Morwald, Johann Prankl, Michael Zillich, and Markus Vincze. Segmentation of unknown objects in indoor environments. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4791–4796. IEEE, 2012.

[20] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *arXiv preprint arXiv :1409.0575*, 2014.

[21] Eleonora Vig, Michael Dorr, and David Cox. Large-scale optimization of hierarchical features for saliency prediction in natural images. *Computer Vision and Pattern Recognition*, 2014.

[22] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision*, 4 :34–47, 2001.

[23] Jianming Zhang and Stan Sclaroff. Saliency detection : a boolean map approach. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 153–160. IEEE, 2013.

[24] Qi Zhao and Christof Koch. Learning a saliency map using fixated locations in natural scenes. *Journal of vision*, 11(3) :9, 2011.