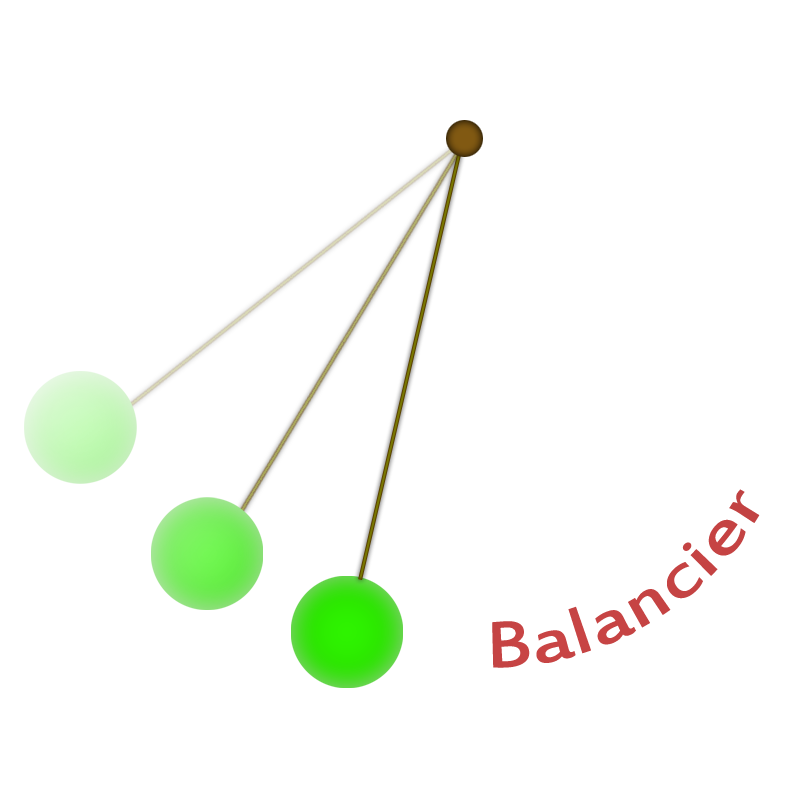
|  |
| --- |
| Détection et calcul d’un mouvement de balancier |
| Projet OpenCV dans le cadre du cours de traitement d’images |
| Christophe Bolinhas et Mathieu Rosser |



# Introduction

Dans le cadre du cours de traitement d’image, il a été décidé de réaliser un projet OpenCV permettant de capturer puis d’analyser un mouvement de balancier afin d’en retirer des données et d’effectuer des comparaisons avec le modèle physique théorique, spécifiant que la période d’un mouvement de balancier est constante et ne dépend que de la longueur du balancier. Pour mener à bien l’objectif du projet, la première phase consistait à acquérir des vidéos de balancier.

Les trois algorithmes de traitement d’images qui ont ensuite été mis en place sont :

* Détection par cercle de couleurs et transformée de Hough. Pour détecter le cercle, on va réaliser une séparation des canaux RVB de l’image pour sélectionner un canal différent de la couleur du cercle. On a alors une image qui sépare bien le cercle (foncé) et le fond (clair), qu’on peut inverser, seuiller et nettoyer avec une ouverture, puis analyser avec Hough ;
* Détection de schéma simple avec algorithme Orb et recherche de correspondances entre points caractéristiques, dans le but de pouvoir effectuer un tracking du schéma qui se balance. Le schéma est défini par une sélection manuelle de l’utilisateur (rectangle à suivre). Cet algorithme est basé sur l’exemple OpenCV feature\_homography, livré avec OpenCV ;
* Détection d’un motif complexe (image) et recherche de ce motif dans chaque image de la vidéo par algorithme SIFT, qui détecte les points caractéristiques, et recherche de correspondances (match). Basé sur <https://gist.github.com/kattern/355d9b27fc29cd195310>.

Les résultats de ces algorithmes sont :

* Centres X, Y des motifs ou cercles détectés ;
* Timestamp de la frame vidéo correspondant à la détection ;
* Nombre de correspondances pour l’algorithme Sift.

Avec ces résultats, il est possible de déterminer :

* Les extremums gauche et droit, en fonction des positions en X ;
* Les maximums locaux de correspondances pour Sift, qui déterminent des extremums de balancement. Le motif est détecté de manière précise aux extrêmes lorsque sa vitesse est minimum, il a donc à ce moment-là un maximum local du nombre de correspondances.

Après avoir effectué l’analyse de la vidéo, on réalise une étape finale de calcul physique pour déterminer les caractéristiques du mouvement de balancier et se comparer au modèle théorique :

* Période de balancement, calculée en connaissant les extremums de balancement (apogée, point mort) et les timestamps des frames de la vidéo. De par sa nature physique, la période réelle est une constante et dépend uniquement de la longueur du bras de balancier,
* Largeur du balancement, soit la distance entre deux extremums successifs (gauche-droite) ;
* Angle au centre au maximum de balancement, déterminé par trigonométrie avec la moitié de la largeur et la longueur du mouvement de balancier ;
* Hauteur maximum du balancement, par rapport au point central ;
* Vitesse maximum au point central.

Les formules physiques liées aux mouvements de balancier sont :

|  |  |
| --- | --- |
| http://www.endmemo.com/physics/pic/spendulum.png  Figure 1. Principe physique d'un mouvement de balancier | **🡪 Période constante** |

Issu de <http://www.endmemo.com/physics/spendulum.php>

La longueur L de notre balancier est de 0.6 mètres.

# Mode d’emploi des programmes réalisés

Les différents programmes réalisés ont été centralisés dans un seul fichier python main.py. Ce fichier reçoit en paramètre un code identifiant le programme à exécuter. **Les programmes fonctionnent avec OpenCV 2.4 et Python 2.7.**

**Pour lancer l’analyse des vidéos de cercles de couleurs par transformée de Hough :**

|  |  |
| --- | --- |
| **Cercle bleu** | python main.py circle\_blue |
| **Cercle rouge** | python main.py circle\_red |
| **Cercle vert** | python main.py circle\_green |
| **Cercle RVB** | python main.py circle\_rgb |

**Pour lancer l’analyse des vidéos avec schémas suivis et détectés par caractéristiques Orb, qui *requiert une sélection du schéma avec la souris et le lancement de la vidéo avec la touche espace* :**

|  |  |
| --- | --- |
| **Premier schéma** | python main.py schema1 |
| **Deuxième schéma** | Python main.py schema2 |

**Pour lancer l’analyse des vidéos avec motif complexe (image), détectée et *matchée* avec Sift :**

|  |  |
| --- | --- |
| **Motif avec analyse par extremum en X** | python main.py motif\_extremum |
| **Motif avec analyse par nombre de correspondances** | Python main.py motif\_match |

A la fin de l’exécution du programme, on trouve une sortie console des résultats calculés pour chaque période de balancement. Voici un exemple de sortie validant le modèle théorique :

|  |
| --- |
| Application de calcul de mouvements de balancier  @author: christophe.bolinhas, mathieu.rosser  Affichage : ON  Exemple avec split couleur bleue, threshold, ouverture et les cercles de Hough  Analyse du mouvement de balancier pour L = 0.600000 [m] (ratio 750.000000 [px]/[m])  Période théorique = 1.553893 [s]  N° Période [ms] Largeur [px] Angle [°] Hauteur [m] Vitesse [m/s]  1 1468.134801 410.000000 27.100685 0.065876 1.136872  2 1434.768101 372.000000 24.414402 0.053652 1.025990  3 1434.768101 342.000000 22.333683 0.045008 0.939712  4 1534.868202 320.000000 20.827497 0.039207 0.877063 |

# Résultats et comparaisons des algorithmes

Les résultats obtenus avec les différents algorithmes sont consignés dans le fichier Excel joint à ce document. La présentation PowerPoint, aussi jointe à ce rapport, reprend et illustre ces résultats avec des graphiques. Les mouvements de balancier en X et Y sont présentés sous forme de graphique radial. Un graphique du nombre de correspondances calculées avec l’algorithme Sift montre les maximas locaux de correspondances, permettant une détection des extremums de balancement.

Figure 2. Graphique des positions X et Y des centres détectés, qui permettent de voir les mouvements de balancement

Figure 3. Graphique du nombre de correspondances par frame, avec des maximas locaux aux extrêmes de balancement

# Conclusion

Au final, les implémentations réalisées avec les trois algorithmes différents donnent des résultats intéressants et correspondants à la réalité. Selon les formules physiques, la période de balancement est constante. L’angle, la hauteur et la vitesse de balancement diminuent avec le temps. Les résultats obtenus valident ces formules, avec une période moyenne proche de la période théorique (erreurs de précisions, de mesure, de correspondance en pixels) et un écart-type peu élevé. Les autres mesures décroissent avec le temps, comme attendu. L’algorithme cherchant les extremums en se basant sur le nombre maximum de correspondances par analyse graphique donne les meilleurs résultats en ce qui concerne la période, avec un écart-type nul.