# **Proposition de projet  - Détection de partie d’échecs –**

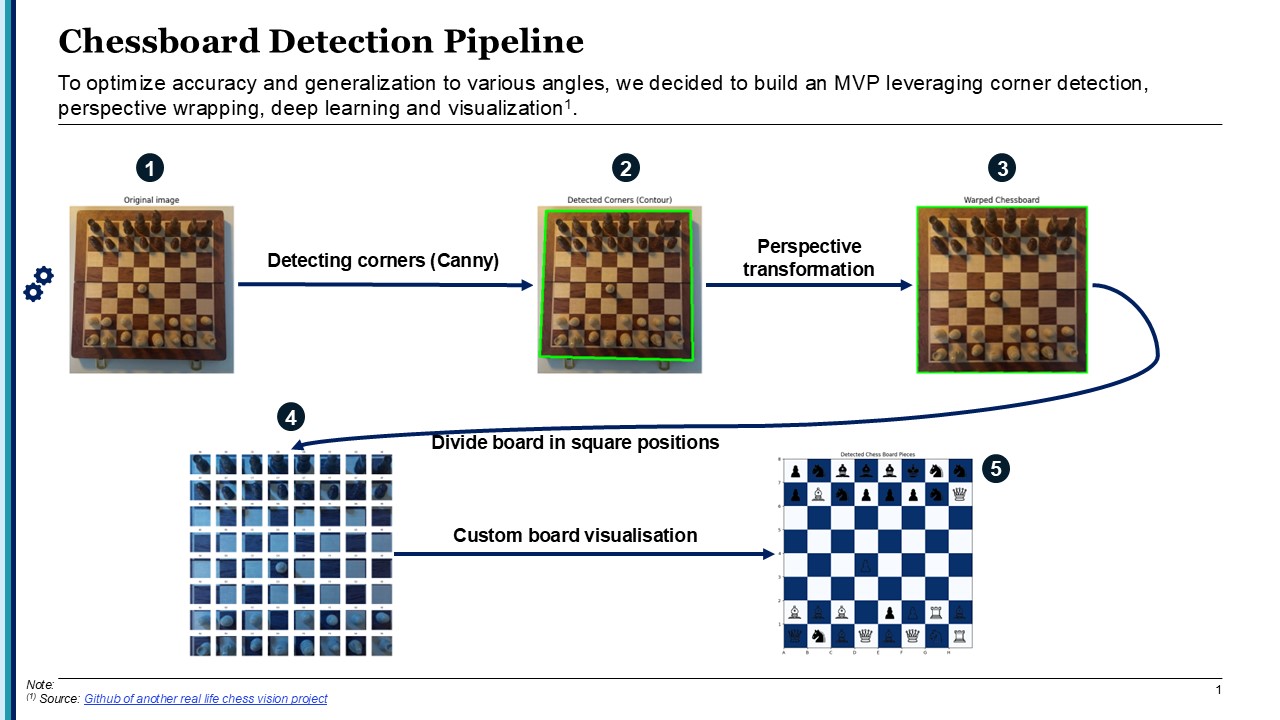
## Problématique & motivations

Le problème que nous cherchons à résoudre est le suivant : « ***Prendre une photo d’un échiquier et ses pièces, et le retranscrire sur une board digital (tracé avec matplotlib ou autre librairie python) »***

Ce problème est intéressant d’un point de vue utilisateur. En effet, il est relativement long et dépourvu d’intérêt de retranscrire une partie d’échecs à la main, et notre pipeline nous permettrait d’automatiser ce processus. Une partie comportant jusqu’à 32 pièces, cela requiert généralement des dizaines de clics au minimum pour retranscrire une partie et analyser les coups !

D’un point de vue technique, cela nous permettrait de mettre en place les techniques vues en cours, tant dans le traitement d’image (saturation, utilisation des différents channels, réalignement, etc…) que dans la détection de contours (détection de coins, contours avec sobel et canny, mais aussi la remise en perspective).

## Méthodologie



En nous inspirant des travaux de [Shainisian](https://github.com/shainisan/real-life-chess-vision)6, nous avons eu l’occasion d’avancer sur le projet et de vérifier sa faisabilité en effectuant un MVP que nous comptons raffiner.

Notre projet se décomposera en cinq étapes :

1. Lecture de l’image et pre-processing pour enlever les artefacts
2. Détections de contours (comparaison des performances de canny et sobel)
3. Transformation en perspective pour avoir un format de ‘board’ uniformisé
4. Division de l’échiquier en positions de cases (Baseline division en cases de longueur uniforme vs. Détection de coins plus affinés via des règles spécifiques et dézoom pour avoir les cases et pièces)
5. Transformation de l’échiquier en board digital via matplotlib et des images SVG trouvées sur wikipedia. Dans cette étape, nous utilisons aussi le modèle dino-v2 et des mesures de similarité pour discriminer les pièces entre elles.

Chacune de ces étapes est relativement compliquée et présente multiples approches possibles. Le but de nos travaux, en plus de les implémenter, sera aussi de comparer leurs performances.

## Métrique

Nous avons trouvé de nombreux datasets3 de pièces pour l’étape n°5, mais des tests empiriques ont montré que la performance était bien meilleure si nous utilisions notre propre dataset (les pièces variant beaucoup d’un échiquier à un autre).

Nous observerons les métriques classiques par rapport aux variables d’évaluation décrites dans la partie suivante.

## Evaluation

Nous effectuerons plusieurs tests d’échiquiers avec différents angles et différentes configurations pour mesurer la précision de nos modèles, avec un niveau de granularité par case.

En plus de la précision de détection de case, nous espérons pouvoir mesurer la précision de notre pipeline sur : La détection de l’échiquier (True/False), Transformation en perspective (True/False), Division de l’échiquier en 64 cases satisfaisantes (True / False) et pour chacune des cases, le F1-Score par rapport à la variable (Détection de pièce / absence de pièce)2.

## Références

[1] : [Deep Learning chessboard detection project](https://colab.research.google.com/drive/1ByRi9d6_Yzu0nrEKArmLMLuMaZjYfygO#scrollTo=WgHANbxqWJPa)

[2] : [Computer vision system for the chess game - M. Piškorec\*, N. Antulov-Fantulin\*\*, J. Ćurić\*, O. Dragoljević\*, V. Ivanac\*, L. Karlović\*](https://matijapiskorec.github.io/static/pdf/Computer%20vision%20system%20for%20the%20chess%20game%20reconstruction.pdf)

[3] : [Kaggle Dataset](https://www.kaggle.com/datasets/anshulmehtakaggl/chess-pieces-detection-images-dataset?utm_source=chatgpt.com)

[4] : [Chess piece location and identification (Bachelor thesis)](https://e-archivo.uc3m.es/rest/api/core/bitstreams/6669354c-a9e7-4883-9090-69b18a627124/content)

[5] : [Chess piece detection senior project](https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1617&context=eesp)

[6] : [Real Life chess vision github project (Big influence on our approach)](https://github.com/shainisan/real-life-chess-vision)