Simulation d'un Contrôleur de Petite Machine à Bois

Matthieu Boyer & Sélène Corbineau

15 janvier 2025

Plan

Positionnement du Problème

Implémentation

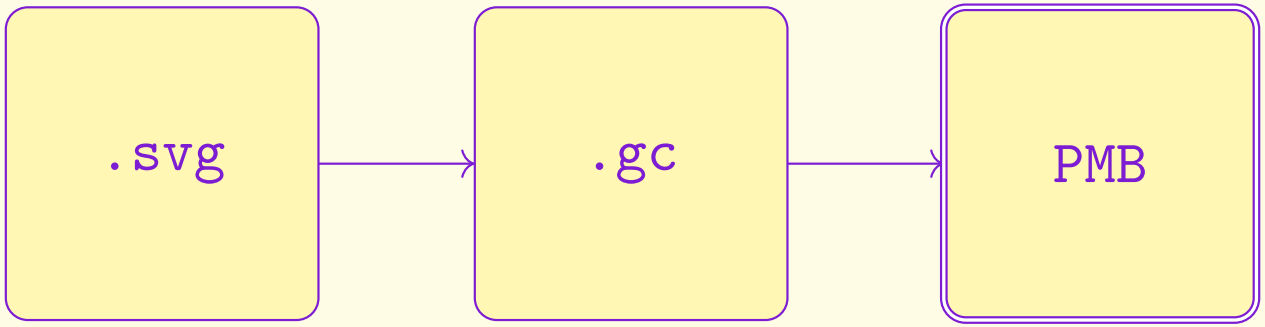
Réalisme

Petite Machine à Bois

Figure – Une Petite Machine à Bois

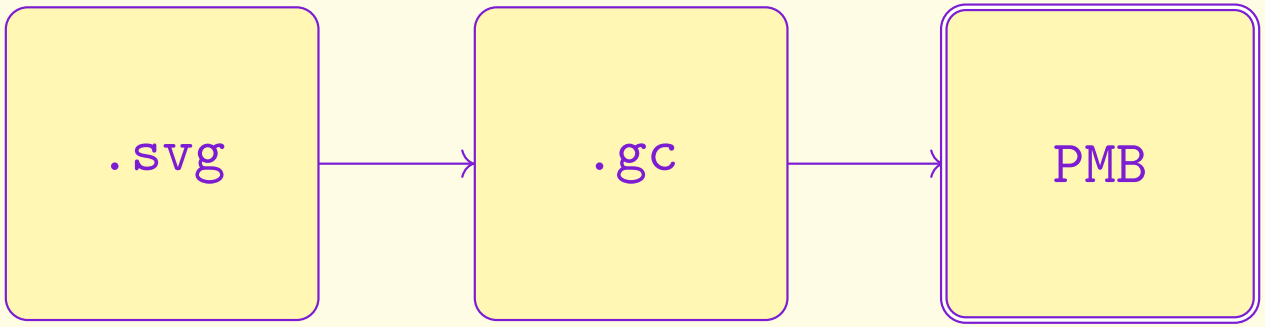
Format d'Entrée

On convertit un fichier `.svg` en fichier `.gc` qui est une abstraction des commandes à exécuter.



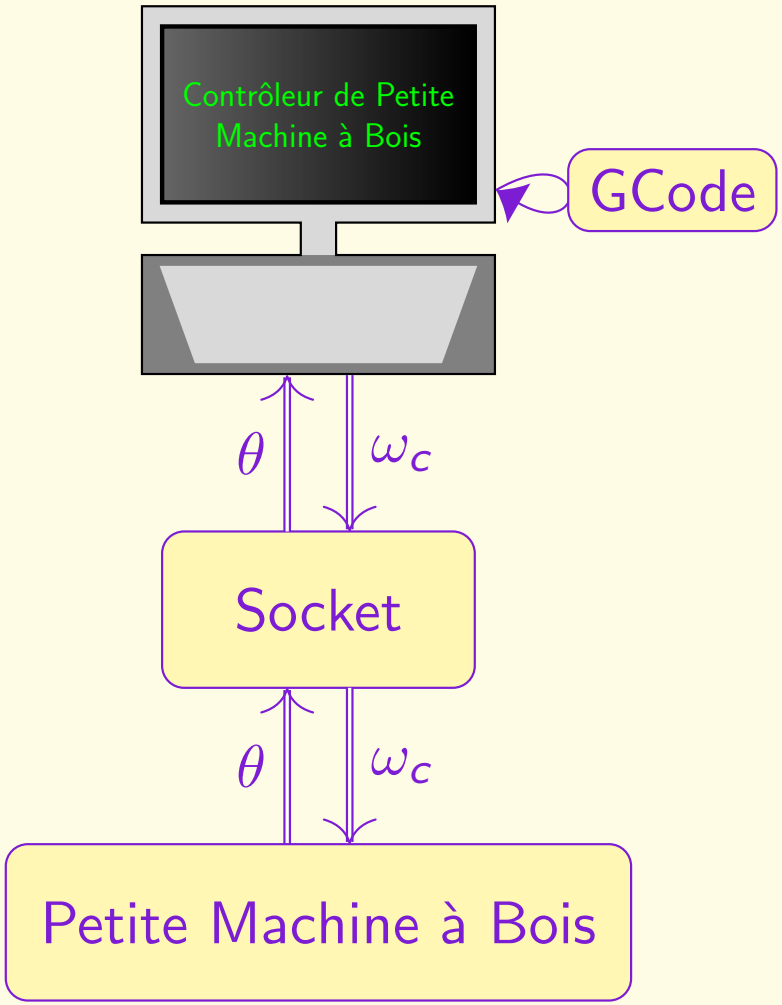
Format d'Entrée

On convertit un fichier `.svg` en fichier `.gc` qui est une abstraction des commandes à exécuter.



Le format GCode est standardisé, et indépendant des capacités de la machine.

Communication



Contrôleur : traduit le GCode et envoie les vitesses de commande ω_c pour les moteurs.

Socket : sert d'intermédiaire pour la connexion (PMB).

Petite Machine à Bois : envoie les positions θ de ses moteurs pas-à-pas.

Plan

Positionnement du Problème

Implémentation

Réalisme

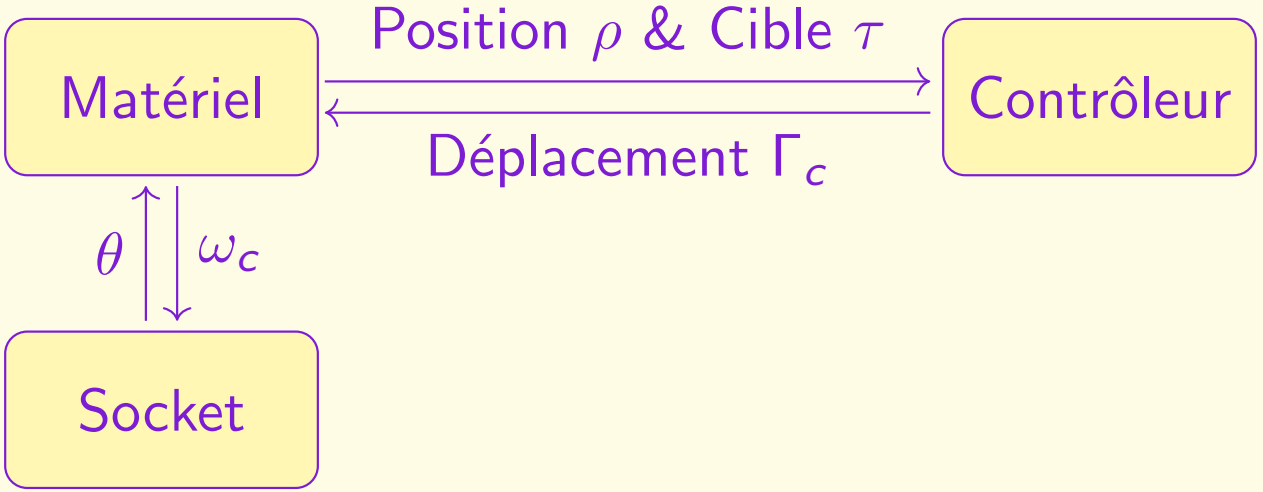
Traducteur

On agit simplement ligne par ligne, voici quelques commandes :

- G0 : Aller au point donné ;
- G1 : Aller au point donné, en coupant au passage ;
- G2,G3 : Faire un arc de cercle, en sens trigonométrique ou non ;
- G4 : Dormir ;
- G5 : Suivre une courbe de Bézier cubique. Permet d'en suivre plusieurs d'affilée de manière dérivable ;
- G20 : WTF IS A KILOMETER ;
- G21 : Changer d'unité en mm ;
- G28 : Retourner à la maison ;
- G30 : Retourner à la maison, en passant par un point donné.

Contrôleur Matériel

Le module matériel du contrôleur maintient une représentation mémoire de la position *théorique* de la tête, transmise au contrôleur. Il demande à la socket (et donc au *vrai* matériel) les positions mesurées des moteurs pas-à-pas.



Il envoie ensuite les vitesses de commandes traduites dans la spécification des moteurs.

Contrôleur Logiciel

Dans une première approximation, on poursuit le point de contrôle à vitesse constante. On calcule $\delta = \rho - \tau$. Selon le signe de δ_x (resp. y, z) on définit ω_c la vitesse de commande de sorte que :

$$\nu\omega_c = 1 \text{ et } (\omega_c)_x \leq \texttt{FAST\_XY\_FEEDRATE} \text{ (resp. } y, z)$$

Un contrôleur plus poussé est obtenu en ajoutant une proportionnalité par rapport à l'écart *théorie - mesure*.

Simulateur

Le simulateur tourne en continu, et ne change de vitesse pour résoudre les équations qu'à chaque changement de direction. On a ensuite par intégration directe du PFD :

$$J\dot{\omega} = \underbrace{-f_0 \times \omega_t}_{\text{Friction}} + \underbrace{\Gamma\left(\omega_c - \omega_t\right)}_{\text{Gain Moteur}}$$

$$\omega_{t+\Delta t} = \omega_t + \underbrace{J^{-1}}_{\text{Inertie}} \times \Delta t \left(-f\omega + \Gamma\left(\omega_c - \omega_t\right)\right)$$

On introduit par ailleurs un terme d'erreur lié aux frottements, et on peut encore raffiner le modèle.

Plan

Positionnement du Problème

Implémentation

Réalisme

Intérêt

L'utilisation d'une socket et d'un modèle théorique restreint comme intermédiaire permet de prendre en compte, durant les phases de test, les différents facteurs d'erreurs pouvant intervenir durant l'exécution :

- Une connexion trop volatile ;
- Des erreurs liées au glissement des moteurs pas-à-pas ;
- Des erreurs liées à la dureté du matériau à couper, et à son anisotropie.

Ceci permet de plus à l'opérateur de modifier la trajectoire en direct pour rectifier/interrompre les opérations.

Réalisme de la Simulation

- La simulation est pour l'instant peu réaliste car nos modèles physiques sont simplifiés.
- Nos moteurs sont modélisés par des moteurs asynchrones plus que pas-à-pas.
- Le contrôleur pour l'instant implémenté ne prend pas en compte la suite des instructions dans sa globalité, et en particulier ne prévoit pas de restreindre l'effort sur les moteurs, en revenant le plus possible à $\omega = 0$.