N° d'ordre : 057 Année: 2006

Université de Technologie de Belfort-Montbéliard
Ecole Doctorale "Sciences Physiques pour l'Ingénieur et Microtechniques"

THESE

Présentée pour obtenir le grade de

Docteur de l'Université de Franche-Comté et de

l'Université de Technologie de Belfort-Montbéliard

par

Sihao DENG

Programmation robotique hors-ligne et contrôle en temps réel des trajectoires : Développement d'une extension logiciel de RobotStudio™ pour la projection thermique

Soutenue le 24 novembre 2006 devant le jury composé de :

Rapporteur :

Monsieur **L. PAWLOWSKI**, Professeur, ENSCL, Lille
Monsieur **J.P. GAUTHIER**, Professeur, Université de Bourgogne, Dijon

Examinateur :

Monsieur **A. BOURJAULT**, Professeur, ENSMM, Besançon
Monsieur **P. CHARLES**, Directeur de R&D, ABB MC, Paris
Monsieur **C. CODDET**, Professeur, UTBM, Belfort
Monsieur **H. LIAO**, Professeur, UTBM, Belfort, Directeur de thèse

Remerciements

Les travaux présentés dans ce mémoire ont été réalisés au Laboratoire d'Etudes et de Recherches sur les Matériaux, les Procédés et les Surfaces (LERMPS) de l'Université de Technologie de Belfort-Montbéliard (UTBM) sous la direction de Monsieur le professeur Hanlin LIAO.

Premièrement, j'adresse mes sincères remerciements à Monsieur le professeur Christian CODDET, le directeur du Laboratoire LERMPS, pour m'avoir accueilli au sein de son équipe, pour m'avoir donné cette occasion de recherche et l'opportunité de m'affirmer.

Ensuite je tiens à remercier Monsieur le professeur Alain BOURJAULT de m'avoir fait l'honneur de présider le jury, ainsi que Monsieur le professeur Lech PAWLOWSKI et Monsieur le professeur Jean-Paul GAUTHIER, pour l'intérêt qu'ils ont manifesté à ce travail en acceptant d'être les rapporteurs de mon travail ainsi que les opinions sur mes travaux et les suggestions pour des futurs travaux.

Je tiens à remercier la société ABB qui fournit la plateforme robotique sur laquelle j'ai fait cette étude. Je remercie spécialement Messieurs Philippe CHARLES, Philippe URIOT chez ABB pour les suggestions de développement et pour le support technique apporté sur cette plateforme.

Je remercie chaleureusement Monsieur le professeur Hanlin LIAO qui m'a soutenu et conseillé beaucoup tout au long de cette étude. Sa sérieuse attitude scientifique m'impressionne fortement et influenceront mes futurs travaux. Il m'a aidé non seulement dans les recherches mais aussi dans la vie quotidienne.

Je tiens à adresser un remerciement à Monsieur Olivier LANDEMARRE pour son aide concernant les corrections exquises de ma thèse et les conseils valables sur mes travaux. J'associe à des remerciements Messieurs Hui LI, Christian ADAM, Jianfei LI pour leurs conseils avisés et discussions techniques. J'exprime aussi ma reconnaissance à tous les membres de l'équipe du LERMPS pour leur sympathies et encouragements.

Enfin, je voudrais adresser un grand merci à ma femme et mes parents qui m'ont soutenus et encouragés tout au long de ces années.

SOMMAIRE

LISTE DES FIGURES

Chapitre 1 Introduction

Chapitre 2 Rappel sur la CFAO-robotisée en projection thermique

Chapitre 3 Extension logicielle pour RobotStudio™

Chapitre 4 Thermal Spray Toolkit (T.S.T.)

LISTE DES TABLEAUX

Chapitre 2 Rappel sur la CFAO-robotisée en projection thermique

Chapitre 3 Extension logicielle pour RobotStudio™

Chapitre 4 Thermal Spray Toolkit (T.S.T.)

Chapitre 1

Introduction

La projection thermique occupe aujourd'hui une place de plus en plus importante parmi les familles de traitements de surfaces [1] qui jouent un rôle incontournable dans de nombreux domaines industriels.

L'usage des robots en projection thermique s'est popularisé parce qu'il fournit une protection dans un environnement de travail dangereux et propose une garantie de qualité de production. Pourtant, aucun système ne propose aujourd'hui une approche complète de la programmation hors-ligne dans le domaine de la projection thermique. Les principaux problèmes rencontrés sont, tout d'abord, l'exploitation des modèles informatiques à assembler, puis la prise en compte des paramètres liés à la projection thermique.

L'approche proposée dans cette thèse vise à intégrer au sein d'un système CFAO (Conception et Fabrication Assistée par Ordinateur) robotique « RobotStudio™[1] » tous les aspects de la programmation d'un robot de projection thermique, de la génération des trajectoires au choix des paramètres opératoires.

Le but de cette étude est de générer la trajectoire robot sur les pièces de forme complexe et de contrôler la trajectoire robot pendant le processus de projection thermique ainsi que l'analyse de caractérisation de la surface du dépôt. Pour ce faire, nous proposons de développer une extension logicielle utilisant les ressources du logiciel RobotStudio™.

Dans le premier chapitre, après avoir donné les paramètres opératoires qui doivent être contrôlés en projection thermique ainsi que la nécessité de cette étude, nous poursuivons par l'étude d'une application réelle de projection thermique, pour ensuite exposer les objectifs de ces travaux.

Le deuxième chapitre présente l'application de la CFAO robotisée pour la programmation hors-ligne et la simulation cinématique du robot. En particulier, nous présentons le logiciel commercial « RobotStudio™ » et ses insuffisances pour une utilisation directe pour ce type d'application. Cette étude propose une extension logicielle dite « Thermal Spray Toolkit (T.S.T.) » qui étend par 3 modules les fonctionnalités du logiciel RobotStudio™ pour l'adapter à l'application de projection thermique.

[1] RobotStudio™ – Un logiciel CFAO robotique développé par Asea Brown Bovrie Ltd (ABB).

Le troisième chapitre exprime les principes et la méthode de développement d'extension logicielle sous RobotStudio™. Ce chapitre contient les indications concernant le développement de l'extension T.S.T., la technologie de la programmation et la présentation des fonctionnalités de l'extension T.S.T.

Le quatrième chapitre spécifie les 3 modules composants de l'extension T.S.T. Le module PathKit permet de créer les trajectoires robot par rapport aux besoins spécifique de la projection thermique. Le module ProfileKit fournit la stratégie de choix des paramètres opératoire de projection et la caractérisation de surface du dépôt (Ra, Rq, Rsk, FD, etc.). Le module MonitorKit est capable de communiquer avec le robot par liaison Ethernet et d'obtenir en temps réel toutes les données de robot durant le procédé de projection. Des essais expérimentaux ont été effectués pour vérifier certains programmes robot et les fonctionnements du logiciel.

Enfin, le dernier chapitre présente les conclusions de cette étude et montre la voie pour des développements futurs.

1.1 Projection thermique

La technique de projection thermique est une méthode de couverture des surfaces permettant d'obtenir des dépôts en mettant en œuvre un large spectre de procédés (projection à la flamme, à la torche à plasma d'arc soufflé, etc.) et de matériaux (alliages métalliques, céramiques, polymères et matériaux composites, etc.). Elle permet de réaliser des revêtements de plusieurs dizaines micromètres à quelques millimètres. Ce procédé consiste à chauffer et fondre le matériau à déposer, présenté sous la forme de poudre ou d'un fil, grâce à une source à haute énergie (électrique ou combustion). Le transfert des particules fondues ou semi-fondues s'effectue à l'aide d'un gaz, le plus souvent chaud. La poudre fondue et accélérée vient ensuite s'écraser puis se solidifier sur la surface du substrat pour former un dépôt [2,3] (Figure 1.1).

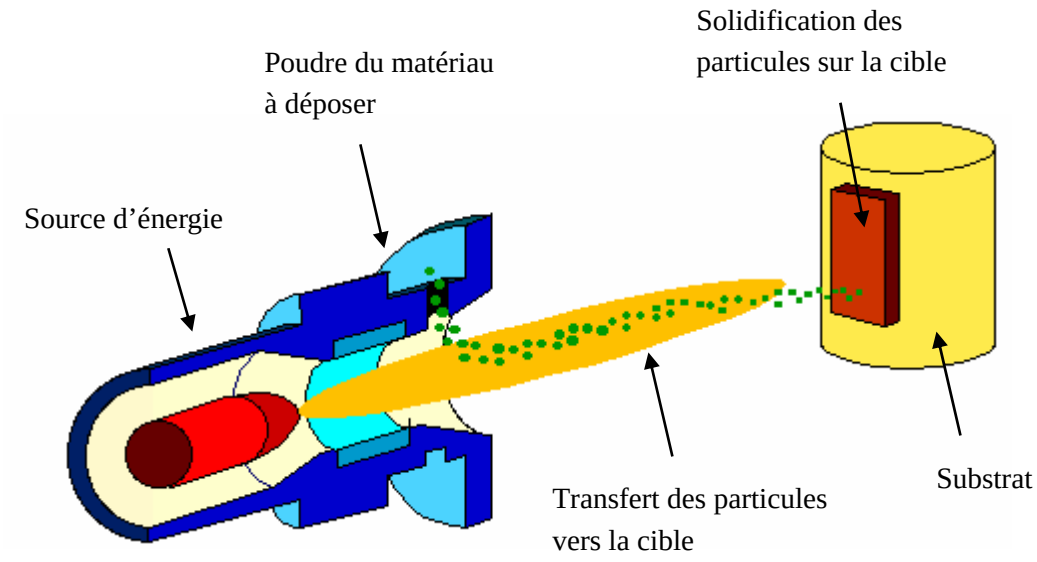


Figure 1.1 Principe général de la projection thermique

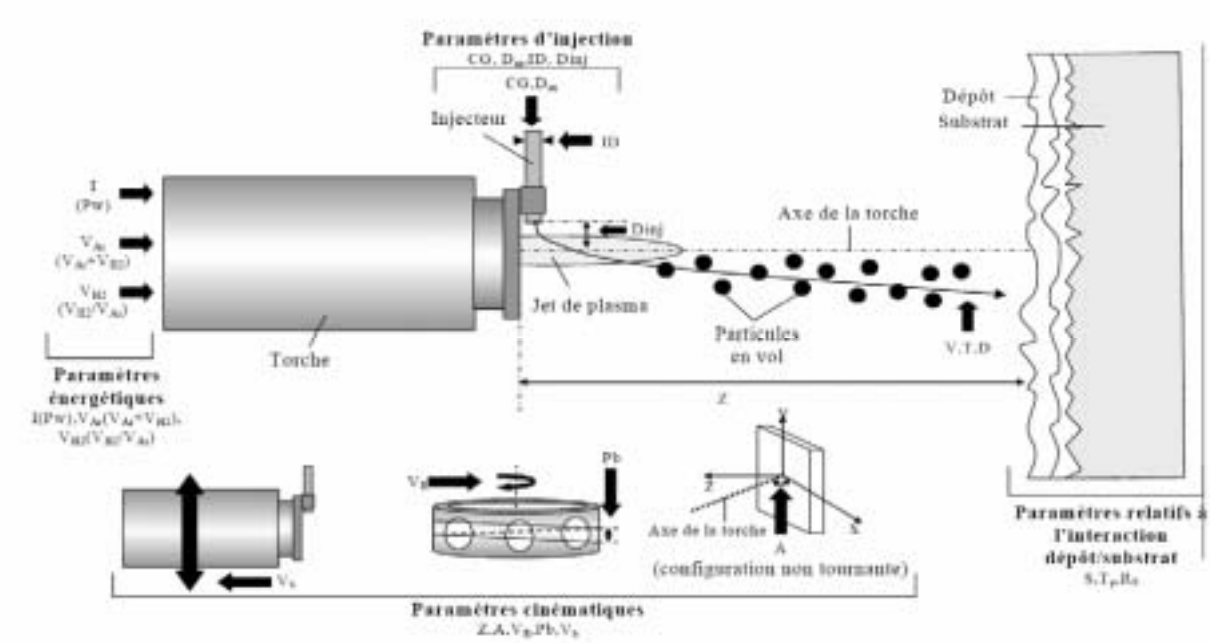
Le procédé de projection thermique à la torche à plasma d'arc soufflé atmosphérique (APS – Atmospheric Plasma Spraying) est l'une des techniques de projection la plus utilisée dans l'industrie (environ 50% des applications) [4]. Les dépôts réalisés avec ce procédé sont

d'adhérence moyenne comparée à d'autres procédés et présentent des taux d'oxydes et de porosités significatifs [5]. Cette technique a été retenue dans cette étude comme support pour valider la faisabilité de la génération et l'optimisation des trajectoires robot de la projection thermique par l'extension logicielle « Thermal Spray Toolkit ».

1.2 Paramètres opératoires en projection thermique

La projection thermique se caractérise par une forte interaction entre les paramètres opératoires, les propriétés et les caractéristiques des dépôts produits. Les paramètres opératoires peuvent se diviser en cinq catégories (Figure 1.2):

- Les paramètres relatifs aux matériaux d'apport.
- Les paramètres d'injection du matériau dans le jet gazeux.
- Les paramètres énergétiques du jet.
- Les paramètres cinématiques.
- Les paramètres relatifs à l'interaction dépôt/substrat.



Paramètres d'injection	
Dm : débit massique de la poudre [g.min^{-1}]	CG : débit du gaz porteur [Nl.min^{-1}]
ID : diamètre de l'injecteur [mm]	Dinj : distance injecteur/axe géométrique de la torche [mm]
AI : angle d'injection	
Paramètres énergétiques	
I : intensité du courant d'arc [A]	Pw : puissance électrique [w]
V_{H2} : débit d'hydrogène [Nl.min^{-1}]	V_{Ar} : débit d'argon [Nl.min^{-1}]
$V_{H2}+V_{Ar}$: débit total [Nl.min^{-1}]	V_{H2}/V_{Ar} : taux d'hydrogène [%]
Paramètres cinématiques	
Z : distance de projection [mm]	A : angle de projection [°]
V_b : vitesse relative torche/substrat [m.s^{-1}]	V_r : vitesse de balayage [mm.s^{-1}]
P_b : pas de balayage [mm]	
Paramètres relatifs à l'interaction dépôt/substrat	
S : nature de substrat [-]	Tp : température de substrat [°C]
R_S : rugosité de la surface de substrat [um]	

Figure 1.2 Catégories des paramètres opératoires [6].

Les paramètres opératoires sont fortement couplés entre eux, ainsi que l'ont montré plusieurs études [6,7,8]. Dans cette étude, nous nous concentrerons sur les paramètres cinématiques qui peuvent être contrôlés directement par le robot.

Les paramètres cinématiques concernent (Figure 1.3) :

- la trajectoire de la torche,
- la vitesse relative de déplacement de la torche par rapport au substrat,
- la distance de projection,
- l'angle de projection,
- le pas de balayage (distance entre deux passages voisins de la torche).

Ces paramètres peuvent avoir une influence sur le rendement de dépôt, sur la température du substrat, sur la morphologie des lamelles et des dépôts et sur la qualité des dépôts [9,10].

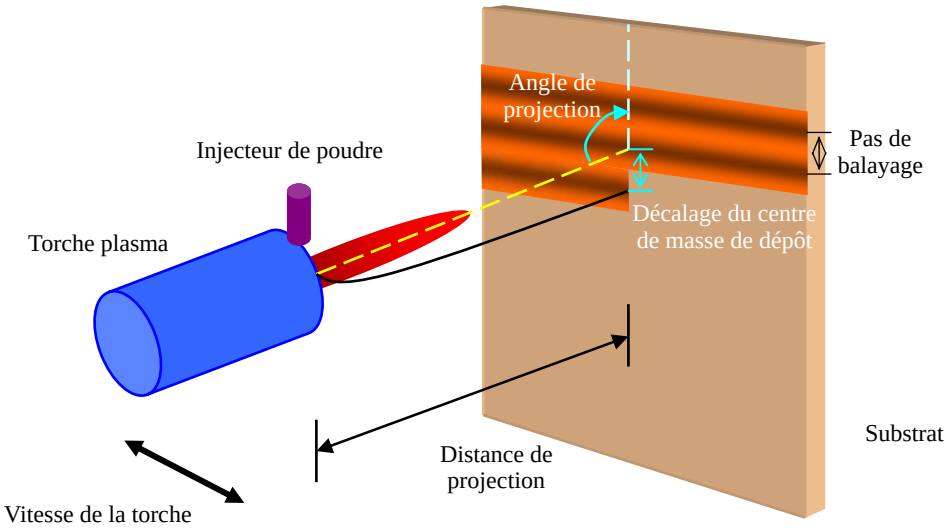


Figure 1.3 Paramètres cinématiques.

1.2.1 Vitesse relative torche – substrat

La vitesse relative torche – substrat représente l'écart de vitesse de déplacement entre la torche et le substrat (Figure 1.3). Avec le débit de poudre, le pas de balayage et le nombre de passes, cette vitesse est l'un des paramètres les plus influents sur la répartition massique de la matière projetée sur le substrat [8].

La qualité d'un dépôt réside généralement non seulement dans son homogénéité structurale (faible porosité, peu d'infondus, peu d'oxydation), mais aussi, dans son homogénéité de répartition [6]. L'obtention d'une épaisseur constante est un gage de gain de matière (respect d'une côte minimale) et d'un post traitement réduit (usinage à la cote). L'épaisseur de matière déposée sur une surface est fonction de la vitesse d'éclairement de la torche par rapport au substrat. En première approximation, l'obtention d'un dépôt homogène en épaisseur doit passer par une vitesse d'impact constante sur tout le substrat si les autres paramètres restent constants. D'où la nécessité de définir une vitesse de déplacement de la torche permettant d'assurer une vitesse d'éclairement constante.

La projection thermique fait partie des procédés dits « continus » au même titre que la peinture et l'encollage. En d'autres termes, la vitesse d'éclairement doit être constante et le mouvement doit être adapté au profil du substrat.

Si le mouvement d'un robot industriel est précis au niveau de la trajectoire géométrique, les caractéristiques cinématiques et dynamiques du déplacement de l'outil sont variables lors de l'exécution de la trajectoire. Rien ne garantit le respect des vitesses spécifiées surtout lors de fortes accélérations. Il est alors primordial de générer des trajectoires garantissant une vitesse d'éclairement constante.

Par ailleurs, la vitesse de balayage doit être modifiée pendant la projection sur de nombreuses pièces en rotation de section variable (configuration souvent rencontrée dans les applications industrielles) (Figure 1.4).

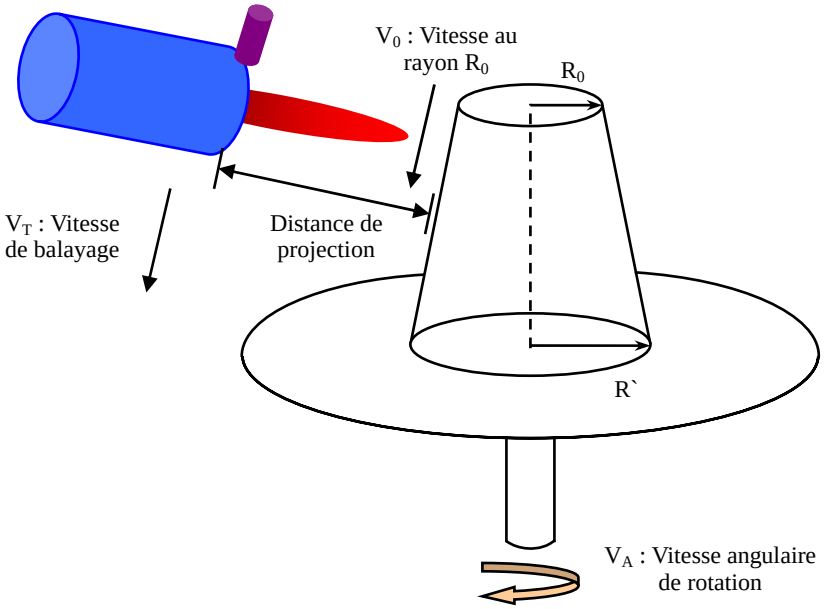


Figure 1.4 Projection sur un échantillon en rotation.

La vitesse de déplacement de la torche, couplée à la rotation de l'échantillon, définit le pas de balayage qui mesure le degré de recouvrement des empreintes de la matière déposée. En plus, la vitesse linéaire de la torche est fonction de la vitesse angulaire de la pièce et du rayon local (Eq.1).

$$V_L \mid 2\phi R \, \mathfrak{f} V_A \qquad \text{(Eq.1)}$$

où V_L est la vitesse linéaire [mm.s^{-1}] et V_A est la vitesse angulaire [tour.s^{-1}]. Dans cette situation, la vitesse de torche doit être adaptée par rapport au rayon R, il en résulte que le pas de balayage doit diminuer quand la vitesse de projection augmente. Donc la vitesse de la torche est, en fonction de la vitesse linéaire en R_0, variable selon le rayon R (Eq.2).

$$V_T \mid \frac{R_0}{R} \, \mathfrak{f} V_0 \qquad \text{(Eq.2)}$$

Il est donc nécessaire de bien contrôler la vitesse de la torche pendant le procédé de projection thermique.

1.2.2 Distance de projection

La distance de projection est l'écart entre le dernier plan géométrique de la tuyère de la torche et le substrat (Figure 1.2, Figure 1.3, Figure 1.4). Idéalement, toutes les particules impactant le substrat sont totalement fondues. Pourtant, à cause de leurs différentes tailles, de leur injection, formes, comportements, des gradients importants de vitesses et températures dans le jet de plasma, les particules arrivent dans différents états de fusion sur le substrat [11]. C'est aussi le paramètre le plus facilement ajustable pour faire varier l'état des particules à l'impact sur le substrat.

Une valeur optimale de la distance de projection doit permettre d'établir une valeur de température relativement égale entre les petites et les grosses particules avant leur impact sur le substrat [12,13]. La distance de projection contrôle les propriétés du dépôt comme le taux de contraintes résiduelles, la porosité, le taux d'infondus et le rendement de projection [14,15,16]. Une distance de projection très courte conduit à un échauffement excessif de substrat à cause des flux thermiques transmis. Par ailleurs, un éloignement trop important peut allonger les trajectoires des particules et donc de les refroidir par transferts convectifs/radiatifs et influence rapidement le rendement du dépôt, le taux de porosité et son adhérence au substrat [17,18] (Figure 1.5).

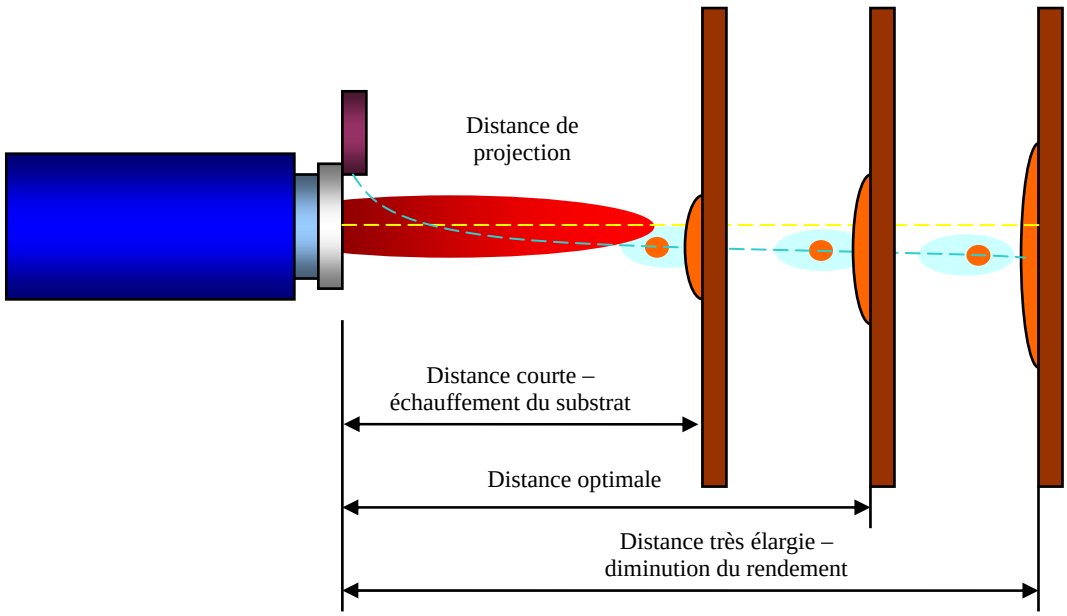
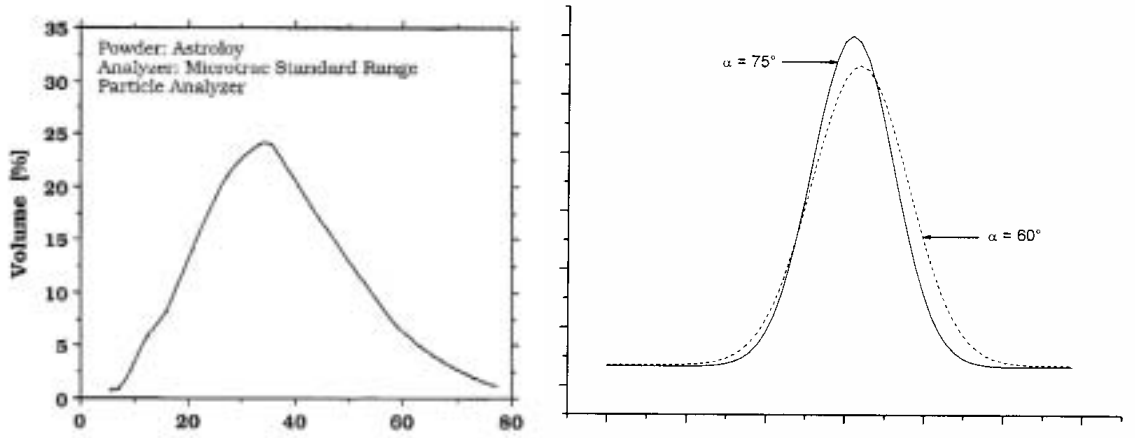


Figure 1.5 Influences de la distance de projection

C'est pourquoi, afin de garantir une homogénéité dans les caractéristiques intrinsèques du dépôt, il convient de conserver une distance de projection constante tout au long de la trajectoire à effectuer.

1.2.3 Angle de projection

L'angle de projection est défini couramment comme l'angle intérieur entre l'axe géométrique de la torche et la tangente à la surface à revêtir (Figure 1.2, Figure 1.3). L'impact des particules sur le substrat peut être classé en deux groupes : impact normal et impact incliné. L'angle de projection influence significativement le profil du dépôt formé. Des études [19,20,21] ont ainsi montré que la hauteur de l'empreinte déposée diminuait avec la diminution de l'angle de projection, diminuant alors également le rendement de projection par suite du rebondissement croissant des particules sur le substrat (Figure 1.6). Cette diminution s'accompagne en général à d'un décalage de l'empreinte [22].



a. Distribution de la forme du dépôt b. Effet de la variation d'angle de projection sur la forme

Figure 1.6 Distribution de la forme du dépôt et effet d'angle de projection [8]

En plus de ces effets macroscopiques, un effet remarquable sur le taux et la morphologie des porosités a été démontré, correspondant à une augmentation du taux de porosités lorsque

l'angle de projection diminue [23,24]. Ceci cause en général une perte d'adhérence et une diminution de la dureté. L'angle de projection « idéal » pour un rendement de projection maximal serait donc l'angle normal. Plusieurs études [19,21] ont aussi souligné que la rugosité de surface augmente pour des angles décroissants de projection.

La nécessité de maîtriser parfaitement l'orientation de la torche est donc évidente et constitue un paramètre important à prendre en compte lors de la projection thermique.

1.2.4 Pas de balayage

Le pas de balayage est la distance entre deux passages consécutifs de la torche de projection. Dans le cas du procédé de projection thermique (APS par exemple), la dimension caractéristique de la zone d'impact des particules projetées à la distance de projection varie entre 8 mm et 20 mm, en fonction du matériau et des conditions opératoires. La valeur du pas de balayage influe naturellement sur l'épaisseur et la forme du dépôt final.

Une valeur optimale du pas de balayage doit permettre d'établir une texture fine de la surface du dépôt et empêcher que la température du substrat n'augmente trop rapidement (Figure 1.7), car un pas fin fournit un dépôt plus homogène mais provoque un échauffement plus local du substrat. En général, une valeur de pas de balayage variant entre 5 mm à 15 mm est souvent utilisée dans le procédé APS.

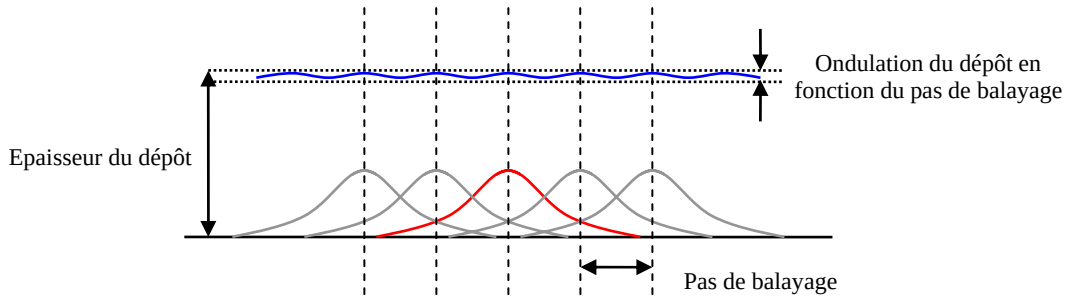


Figure 1.7 Texture de la surface du dépôt

Un choix pertinent de la valeur du pas de balayage est donc important pour obtenir un revêtement de forme adéquate. Ce paramètre opératoire doit de ce fait être bien établi avant la projection thermique et bien contrôlé pendant la projection.

1.3 Nécessité d'utilisation du robot

L'automation flexible et l'automation programmable sont des techniques de plus en plus exigées par l'industrie car elles offrent un temps de préparation plus court et une plus grande productivité. Ces systèmes utilisent invariablement des robots pour manipuler et exécuter la tâche désirée [25].

Ces dernières années, de plus en plus de recherches et développements ont été faits pour améliorer la fiabilité du processus. Cependant, il existe trois problèmes principaux spécifiques à la projection thermique :

- Problème de mouvement (déplacement) : la torche doit suivre précisément la trajectoire conçue surtout pour la projection sur des pièces de formes géométriques complexes. Tous les paramètres opératoires (la vitesse, la distance et l'angle de projection) doivent être constants pendant la phase de projection.
- Problème de modification des paramètres opératoire : les applications variées requierent différents paramètres opératoires incluant la distance et l'angle de projection, la vitesse de la torche, le pas de balayage, etc. Touts les paramètres opératoires doivent donc pouvoir être modifiés facilement.
- Problème d'environnement du processus : la projection thermique génère un environnement de travail potentiellement dangereux, par exemple la haute température des gaz, la basse pression, les radiations, les gaz nocifs et les poussières, ce qui représente des dangers pour les opérateurs.

La projection thermique assistée par robot a résolu tous ces problèmes en raison du mouvement précis de robot et de la capacité du microprocesseur de commander et sauvegarder

les variables du processus. En outre, le robot permet de maintenir l'opérateur à l'abri de l'environnement de poussières et de radiations pendant la projection thermique. Les robots sont donc employés couramment dans l'application de projection thermique [26]. Le laboratoire LERMPS utilise 5 robots (produits par ABB) pour tous types de projection thermique incluant HVOF, Cold spray, APS et VPS (Figure 1.8).

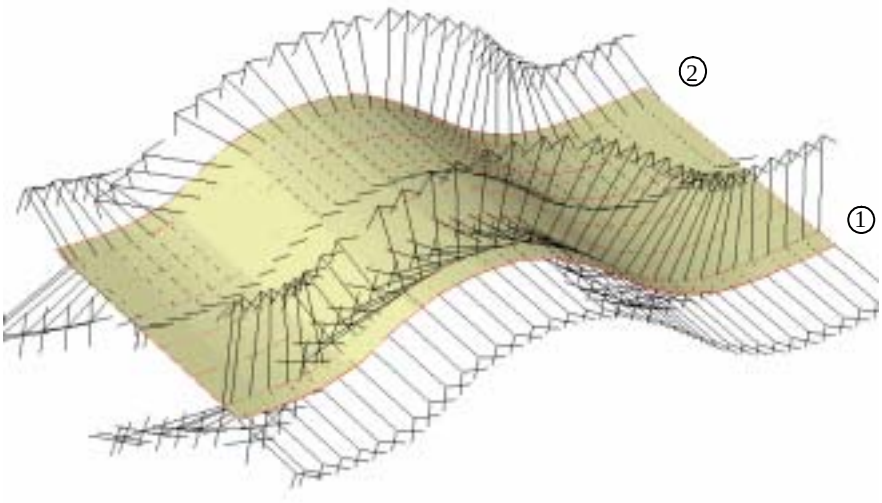
a. Applications APS

b. Applications VPS

Figure 1.8 Exemple de robots utilisés au LERMPS

1.4 Nécessité de la programmation hors-ligne

La programmation manuelle du robot pour la projection thermique reste un point difficile industriellement notamment pour les pièces de forme complexe. En effet, la technique de programmation par apprentissage des points de la trajectoire ne permet pas de générer la position et l'orientation de l'outil de façon précise par rapport la forme de l'échantillon.



Série de trièdres dont les axes z sont normaux à la surface
Série de trièdres dont les axes z sont à 30° de la normale locale à la surface

Figure 1.9 Trajectoires sur l'échantillon complexe

Il faut donc envisager une programmation du robot via un logiciel de CFAO robotique. Cette technique offre des moyens modernes et performants pour définir et contrôler une trajectoire sous contraintes sur des pièces complexes.

L'utilisation d'un logiciel CFAO robotique offre aussi la possibilité de simuler le processus avant l'application réelle, de façon économique et en toute sécurité. Cet outil supplémentaire permet d'optimiser finement la programmation des trajectoires robot, par exemple le réglage de la position, l'orientation de chaque point et la modification des paramètres opératoires (la distance de projection, le pas de balayage, la vitesse de la torche, etc.) selon besoin.

Touts ces avantages de la programmation hors-ligne permettraient aussi de générer les trajectoires robot plus rapidement et plus précisément qu'avec un apprentissage manuel.

PathKit (un module des fonctions du « Thermal Spray Toolkit ») a été développé dans le cadre de cette thèse pour la programmation hors-ligne de trajectoires sur des formes complexes, comme une pièce incurvée, ou une pièce conique en rotation par exemple (cf. Chapitre 4.1).

1.5 Nécessité de prédiction de l'épaisseur du dépôt

Le dépôt est le résultat de la superposition d'une multitude d'événements aléatoires et indépendants : les impacts des particules fondues incidentes sur le substrat (Figure 1.2). Même si les impacts de particules sont des événements aléatoires et indépendants, leur résultat est un dépôt ayant une forme déterministe pouvant être décrite par une fonction de probabilité (courbe de distribution de matière) continue [27,28]. Il est généralement admis que les profils des cordons en projection thermique sont typiquement décrits par une distribution normale (distribution gaussienne) des hauteurs [29] (Figure 1.7, Figure 1.10). Bien entendu, ce profil dépend de la vitesse de torche, du débit de poudre, du matériau, etc.

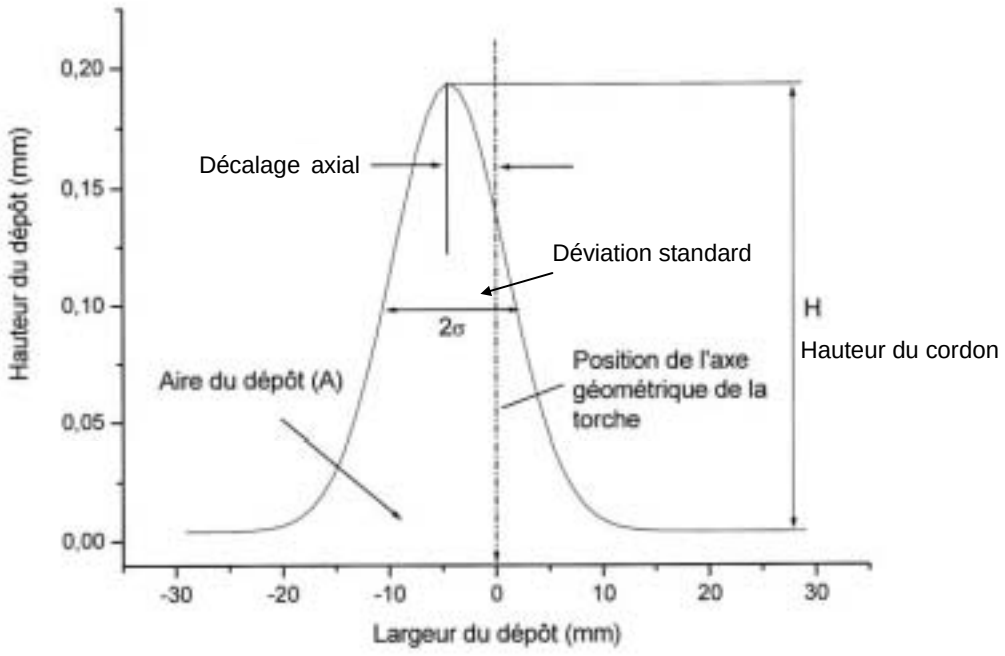


Figure 1.10 Paramètres considérés pour décrire la géométrie d'un dépôt [8].

La programmation hors-ligne doit permettre de générer des trajectoires respectant précisément les paramètres opératoires exposés ci-dessus (section 1.2). L'obtention d'une bonne qualité de dépôt passe par le respect d'une uniformité d'épaisseur sur toute la pièce revêtue. Il faut donc intégrer à la programmation hors-ligne une fonctionnalité permettant de simuler l'épaisseur déposée et le profil final. Des travaux ont déjà été effectués au laboratoire

dans ce domaine (cf. Chapitre 4.2). ProfileKit (un module des fonctions du « Thermal Spray Toolkit ») a été développé pour calculer et simuler l'épaisseur finale du dépôt en projection thermique. Dans le chapitre « ProfileKit », où trouvera une présentation détaillée du développement et de la fonction de ce module.

1.6 Nécessité de contrôle du robot

Lors de l'utilisation d'un robot en projection thermique, la vitesse, l'orientation et la trajectoire de la torche impactent directement la qualité de dépôt. Le logiciel de simulation CFAO permet de programmer une trajectoire robot complexe hors-ligne, mais il ne représente qu'imparfaitement le comportement réel du robot et de la cellule robotisée. En effet, d'une façon générale, les phénomènes sont modélisés de façon approchée pour accélérer les calculs. Des divergences peuvent donc être constatées entre le temps de cycle observé et celui attendu. Les points suivants peuvent être cités comme phénomènes généralement non pris en compte (Figure 1.11):

- ü Les modèles CAO des robots sont théoriques, des écarts entre le robot réel et le virtuel sont inévitables (axes, positionnement, initialisation des moteurs, etc.).
- ü Le poids de câbles électriques, de la torche et des accessoires peut causer des divergences dynamiques entre la simulation et la réalité.
- ü Les frottements, les flexibilités des bras de robots, les non linéarités des transmissions, etc. Ces erreurs non géométriques dépendent de la charge, des vitesses et des accélérations mises en jeu lors des déplacements. Elles peuvent, de plus, évoluer avec la température de l'atelier ou à l'échelle de la vie du robot selon son usure.
- ü Au niveau de l'outil, les erreurs géométriques sont d'autant plus importantes que le centre d'outil est éloigné du centre du poignet.
- ü Au niveau de la cellule, les erreurs géométriques et les incertitudes de positionnement des repères « pièce » peuvent influer sur la précision des trajectoires.

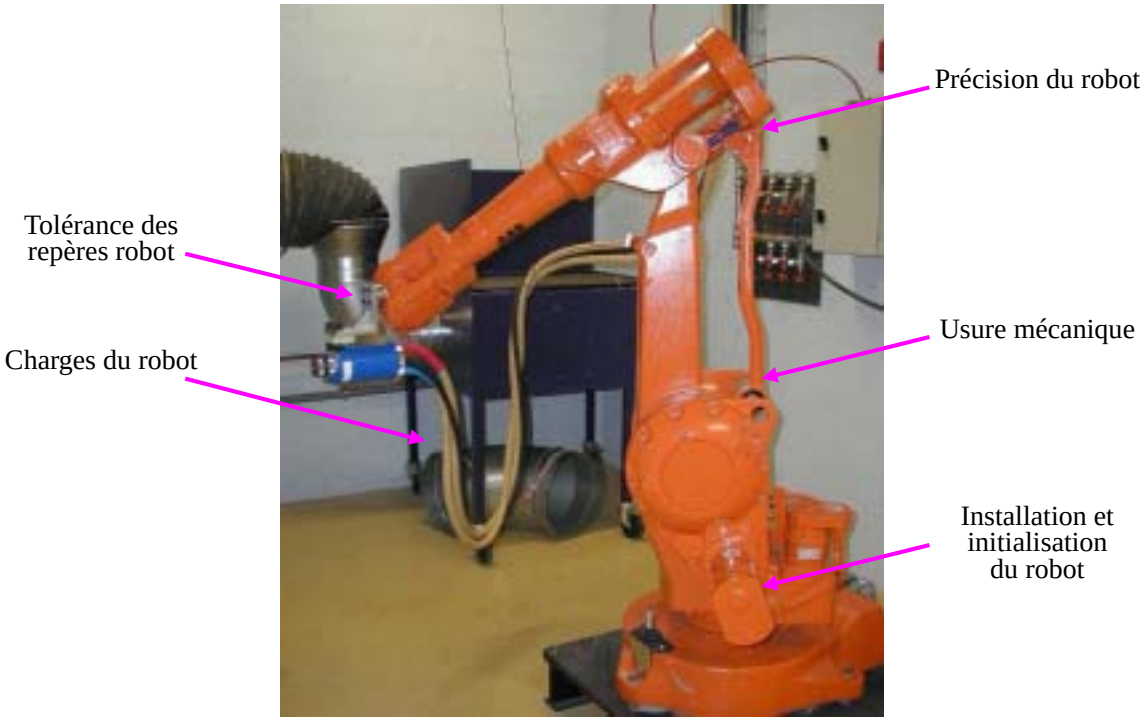


Figure 1.11 Facteurs de divergence entre le robot réel et le robot virtuel.

Fort heureusement, la part des erreurs « inévitables » (usure, frottement, etc.) est généralement très minime face aux autres. Cependant, la programmation hors-ligne ne peut être efficace que dans la mesure où une phase d'étalonnage est pratiquée. Cette phase peut être plus ou moins longue selon la précision exigée par le procédé mis en jeu.

Il est donc recommandé de contrôler les trajectoires de robot en temps réel pour garantir la qualité d'élaboration du dépôt. MonitorKit (un module des fonctions du « Thermal Spray Toolkit ») a été conçu en vue de contrôler le robot en temps réel par liaison Ethernet et d'obtenir toutes les données opératoires du robot lors du processus de projection (cf. Chapitre 4.3). Une présentation détaillée du module est proposée dans le chapitre « MonitorKit ».

1.7 Application actuelle en projection thermique robotisée

1.7.1 Le concept CIM

Le concept CIM (Computer Integrated Manufacturing ou Fabrication Intégrée par

Ordinateur) regroupe toutes les phases de réalisation d'un produit, depuis la conception jusqu'à la fabrication [30,31]. En général, l'étape de conception ne concerne que le produit. Toutefois, la modélisation de l'atelier dans lequel le produit va être fabriqué fait également partie intégrante de cette étape. Jusqu'à ces dernières années, le processus de conception était orienté sur la fabrication du produit lui-même, reflet du processus traditionnel.

Le manque de communication, donc l'absence de cohérence et de réactivité par rapport aux évolutions de la pièce, entre les unités de recherche - développement et les unités de production, représentaient un coût élevé pour l'entreprise.

La prise en compte de la conception des outils nécessaires à la production de l'objet s'effectue désormais très en amont de sa fabrication, réalisant ainsi une économie majeure. Grâce aux nouvelles générations de logiciels, l'expertise technique de la réalisation s'intègre dès la conception de la cellule. Cette intégration correspond au concept d'ingénierie simultanée.

L'étape de conception d'atelier peut également avoir lieu avant que la conception du produit ne soit totalement terminée. Puisque le produit est modélisé dans le système de CFAO, il devient possible de concevoir les cellules de production et de simuler le travail du robot et des axes externes nécessaires à la réalisation du produit, dans le même temps [32]. La mise en œuvre du CIM peut être illustrée par la Figure 1.12.

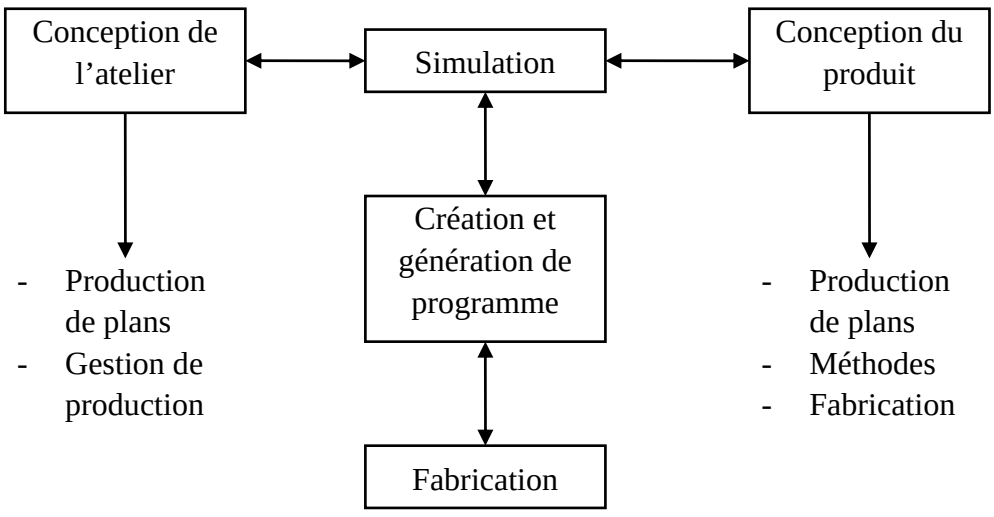


Figure 1.12 Mise en œuvre du concept CIM.

La simulation permet de valider à la fois la modélisation de la pièce et celle de l'atelier, afin de passer à l'étape suivante : la génération automatique des programmes pour le robot de l'atelier [33].

Ces programmes sont ensuite transférés vers les robots en vue de démarrer le processus de production sur site. Le concept CIM, dans l'objectif d'une application ayant recours à la robotique, se divise en trois étapes principales :

- la phase de conception et de modélisation du produit,
- la phase de conception et de modélisation de la cellule,
- l'étape de programmation et simulation.

Puis:

- la création et la modification des mouvements du robot,
- la génération des programmes sur site,
- les calibrations,
- la phase de correction et de reprises éventuelles.

La programmation hors-ligne peut donc se définir comme la génération d'un programme robot de façon déportée, c'est à dire sans mobilisation de la ligne de production. Cela est réalisable dès lors que l'on dispose d'un système numérique de représentation de la cellule. Son plus grand atout réside dans l'économie réalisée par l'absence d'utilisation des outils de production.

La simulation, quant à elle, se présente comme une visualisation animée du comportement de la cellule, avec tous les avantages (confort, puissance de calculs, antériorités des études) qu'engendre l'utilisation de l'ordinateur. Le gain majeur de la simulation réside dans la possibilité de prévoir et donc de corriger, avant la descente sur site, les défauts éventuels qui pourraient conduire à une mise en service longue et coûteuse des outils de production.

1.7.2 L'état actuel

L'arrivée massive de l'électronique et de l'informatique au travers des ordinateurs, des automates et des robots a permis des progrès notables dans les domaines de la conception et de la fabrication, avec l'apparition de la CAO / FAO. On parle aujourd'hui de concept CIM, de conception intégrée et d'ingénierie simultanée dont les objectifs sont d'intégrer les différents aspects liés à l'étude et à la réalisation de produits finis.

Dans cet environnement industriel en constante évolution, la projection thermique est parvenue aujourd'hui à un niveau d'automatisation où les robots et les machines dédiées remplacent souvent l'opérateur humain pour accroître la sécurité, la qualité, la répétitivité des dépôts et pour réduire les coûts en augmentant la productivité [34].

La programmation par apprentissage est le mode actuellement utilisé sur la quasi-totalité des robots de projection thermique. Elle permet un contrôle immédiat des actions effectuées, mais impose l'immobilisation de l'outil de production. De plus, ce mode de programmation s'avère être une tâche longue et fastidieuse avec une précision de trajectoire médiocre pour des pièces de géométries complexes.

La programmation hors-ligne peut quant à elle, être effectuée en dehors du site de projection permettant ainsi de réduire la durée d'immobilisation. De plus, l'utilisation de l'outil CFAO permet la génération de trajectoires beaucoup plus précises que l'apprentissage manuel.

Le concept de programmation hors-ligne peut se synthétiser sous la forme du schéma présenté dans la Figure 1.13.

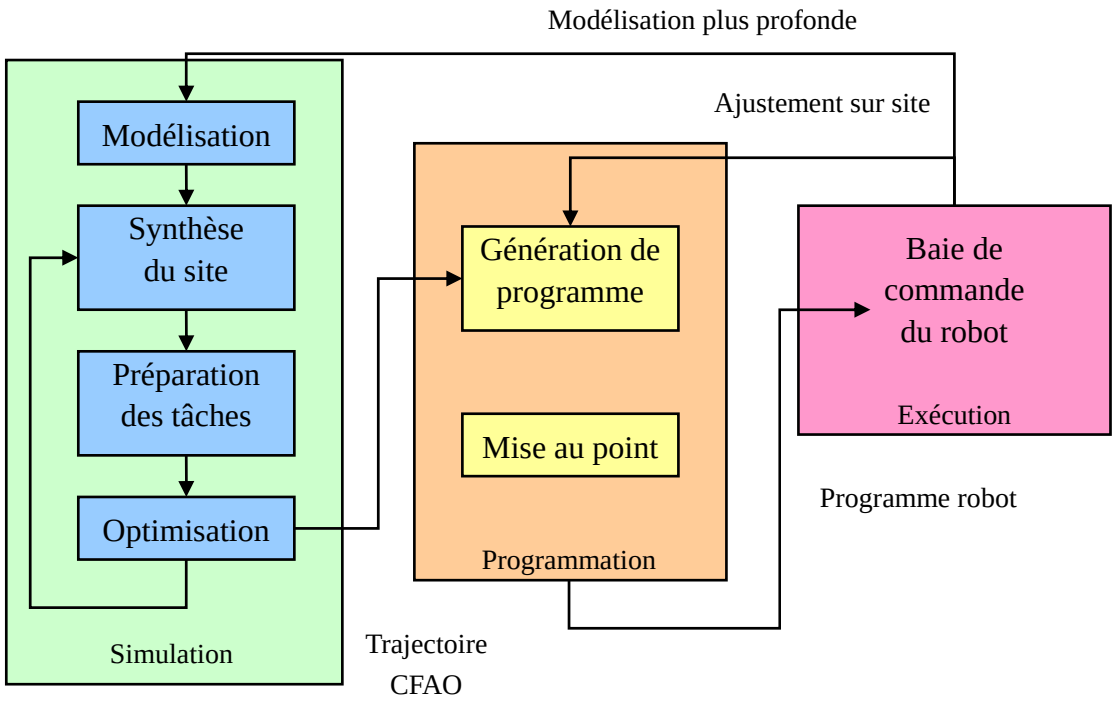


Figure 1.13 Démarche de conception et de programmation d'un site robotisé [35]

Le développement de la programmation hors-ligne a été le sujet de recherche de beaucoup d'entreprises et de laboratoires ces dernières années. Des logiciels commerciaux existent (Igrip, Robcad, Cimstation). Avec l'utilisation de tels logiciels, les problèmes de portée de robot, d'accessibilité, de collisions peuvent être éliminés. Les données CAO des parties à revêtir, la géométrie, la cinématique et la dynamique des robots, ainsi que les données relatives à la projection sont utilisées pour la simulation.

Ces logiciels de robotique fournissent de nombreuses fonctions de simulations dédiées aux applications industrielles dans plusieurs domaines, par exemple le soudage, l'emballage, la peinture, etc. Cependant, il n'existe pas de logiciel commercial CFAO dédié à la projection thermique. Dans le cadre de cette thèse, nos travaux ont consisté à combler cette lacune dans le domaine de la projection thermique.

1.8 Conclusion sur les objectifs des travaux

La projection thermique connaît un développement soutenu et continu (10% par an depuis

1990 [36]) grâce à des applications des plus en plus diversifiées. Pourtant, pour assurer la pérennité et le développement du procédé, un accent important doit être mis sur l'amélioration de la qualité et de la reproductibilité des propriétés des dépôts. Ces propriétés découlent de l'ajustement de 50 à 60 paramètres qui requièrent une excellente compréhension de leurs effets et un rigoureux contrôle des procédés [37,38,39,40].

Une étude récente [41] a estimé que des gains importants de productivité sont possibles par deux voies principales de progrès :

- Une meilleure connaissance et maîtrise du procédé (permettant des gains de temps estimés à 40% par rapport à la situation actuelle).
- Une méthode rapide et précise de programmation des trajectoires des robots employés dans la projection thermique (permettant des gains de temps estimés à 60%).

Certaines études [34,42,43] ont montré des voies de progrès et de développements en projection thermique robotisée. L'aspect robotique fait donc partie intégrante des procédés modernes de projection thermique.

La contribution de cette thèse se situe au niveau de l'optimisation des trajectoires robot en projection thermique. Les objectifs sont alors multiples :

- Améliorer la productivité en réduisant les temps de cycle
- Améliorer les rendements en réduisant la quantité de matière gaspillée
- Etablir des trajectoires robot pour des pièces complexes sans immobilisation de l'outil de production
- Modifier les paramètres opératoires rapidement
- Contrôler le robot en temps réel lors du procédé de projection thermique
- Améliorer la qualité des dépôts par une analyse du profil de la surface

Les conséquences de ces améliorations seront les suivantes :

- Réduction du coût des revêtements

- ü Augmentation de la flexibilité de l'outil de production
- ü Qualité accrue des dépôts

Références

[1] Handbook of Thermal Spray Technology, (Ed.) J.R. Davis, ASM International, Materials Park, Ohio, USA, 2004.

[2] L. Pawlowski. The Science and Engineering of Thermal Spray Coatings. John Wiley & Sons ltd, New York, USA, 1995.

[3] P. Fauchais, M. Vardelle. Plasma spraying: present and future. Pure & Applied Chemistry, Vol. 66, No. 6, pp 1247-1258, 1994

[4] P. Fauchais, A. Vardelle. Heat, Mass and Momentum Transfer in Coating Formation by Plasma Spraying, Int. J. Therm. Sci. (2000) 39, 852-870.

[5] Manuel des traitements de surface à l'usage des B.E, Ed. CETIM, Senlis, France, p.139, 1987.

[6] R. Bonnet. La projection thermique sur des formes complexes: simulation et étalonnage du procédé robotise. Thèse de doctorat. Université de Technologie de Belfort-Montbéliard, France, 2000.

[7] S. Guessasma. Optimisation et contrôle de procédés de projection thermique par coopération de méthode d'intelligence artificielle. Thèse de doctorat. Université de Technologie de Belfort-Montbéliard, France, 2003.

[8] F.I. Trifa. Modèle de dépôt pour la simulation, la conception et la réalisation de revêtements élaborés par projection thermique. Thèse de doctorat. Université de Technologie de Belfort-Montbéliard, France, 2004.

[9] C.J. Li, B. Sun. Effects of spray parameters on the microstructure and property of Al2O3 coatings sprayed by a low power plasma torch with a novel hollow cathode. Thin Solid Films 450 (2004) 282-289.

[10] F.I. Trifa. G. Montavon, C. Coddet. On the relationships between the geometric processing parameters of APS and the Al2O3-TiO2 deposit shapes. Surface and Coatings Technology; Volume 195, Issue 1, 23 May 2005, 54-69.

[11] S. Guessasma, G. Montavon, C. Coddet. Modeling of the APS plasma spray process using artificial neural networks: basis, requirements and an example. Computational Materials Science, Volume 29, Issue 3, (2004) 315-333.

[12] JE Döring, R. Vassen, D. Stöver, The influence of spray parameters on particle properties, Proc. 2002 International Thermal Spray Conference and Exposition, E. Lugscheider, PA. Kammer (eds), DVS-Verlag GmbH, Düsseldorf, Germany, 2002, p. 440, 2002.

[13] A. Vardelle, Etude numérique des transferts de chaleur, de quantité de mouvement et de masse entre un plasma d'arc à la pression atmosphérique et des particules solides, Thèse de Doctorat, Université de Limoges, France, N° d'ordre : 29-87, 1987.

[14] J-F. Li, Modélisation de la formation des contraintes résiduelles dans les dépôts élaborés par projection thermique, Thèse de doctorat. Université de Technologie de Belfort-Montbéliard, France, 2005.

[15] C.-J. Li, A. Ohmori, Relationship between the microstructure and properties of thermally sprayed deposits, Journal of Thermal Spray Technology, 11 (3), p.365, 2002.

[16] S-H. Leigh, Stereological investigation on structure/property relationships of plasma spray deposits, Thèse de Doctorat, Department of Materials Science and Engineering, State University of New York, Stony Brook, USA, 1996.

[17] M. Prystay, P. Gougeon, C. Moreau, Structure of plasma sprayed zircona coatings tailored by controlling the temperature and velocity of the sprayed particles, Journal of Thermal Spray Technology, 10 [1], p. 67, 2001.

[18] K.A. Khor, Y.W. Gu ; D. Pan, P. Cheang. Microstructure and mechanical properties of plasma sprayed HA/YSZ/Ti-6Al-4V composite coatings. Biomaterials 25 (2004) 4009-4017.

[19] R. Bonnet, O. Landemarre, R. Bolot, C. Coddet, Thermal spray coating simulation – step 2, rapport N° 01026-020830, LERMPS, France.

[20] J. Ilavsky, A. Allen, G.G. Long, S. Krueger, Influence of spray angle on the pore and crack microstructure of plasma-sprayed deposits, Journal of the American Ceramic Society, 80, p. 733, 1997.

[21] M.F. Smith, R.A. Neister, R.C. Dykhuizen, An investigation of the effects of droplet impact angle in thermal spray deposition, Thermal Spray Industrial Applications, C.C. Berndt, S. Sampath (eds.), ASM International, Materials Park, OH., USA, p. 603, 1994.

[22] G. Montavon, S. Sampath, C.C. Berndt, H. Herman, C. Coddet, Effects of the spray angle on splat morphology during thermal spraying, Surface and Coatings Technology, 91, p. 107, 1997.

[23] J. Ilavsky, A. Allen, G.G. Long, S. Krueger, Influence of spray angle on the pore and crack microstructure of plasma-sprayed deposits, Journal of the American Ceramic Society, 80, p. 733, 1997.

[24] V.V. Sobolev, J.M. Guilemany, Flattening of droplets and formation of splats in thermal spraying: a review of recent work- Part 2, Journal of Thermal Spray Technology,8(2), p. 301, 1999.

[25] K. Sivayoganathan, D. Al-Dabass, Simulation and robot calibration, The Nottingham Trent University, Nottingham, NG1 4BU, UK.

[26] K.R. Abram, J. Bustamante, R.D. Etzenhouser, Automated metal spray applications. Thermal Spray Research and Applications, Proceedings of the Third

National Thermal Spray Conference, Long Beach, CA, USA, 20-25 May 1990.

[27] B. Bhushan. Modern tribology handbook, Vol. 1 – Principles of Tribology, CRC press LLC, 2001, Boca Raton, FL, USA, p.1760

[28] NF EN ISO 13565-3

[29] Y. Dodge. Premiers pas en statistique. Springer-Verlag, Paris, France, 2001, p.427

[30] I. ESTEBE, Simulation et programmation hors ligne en robotique. Technique de l'Ingénieur. Tome R8 Mesures et Contrôle – Automatique et Robotique. p. R7735-1 – R7735-9

[31] Actes de MICAD 1995. Editions Hermès.

[32] A. Liégeois, Modélisation et commande des robots manipulateurs. Techniques de l'ingénieur, R 7 730 (4-1988), vol R 8, traité Mesures et Contrôle.

[33] R. Bernhardt, G. Schreck, C. Willnow, The realistic robot simulation (RRS) interface. Fraunhofer – Institute for production systems and design technology (IPK), Berlin.

[34] I. Gary, G. Louis, Thermal Spray Robots. METCO Inc, Westbury New York.

[35] P. Chedmail, E. Dombre, P. Wenger, La CAO en robotique. Edition Hermès, Paris, 1998. ISBN 2-86601-695-5. p 255.

[36] F. Kassabji, G. Jacq, J. P. Durand, Thermal spray applications for the next millennium: a business development perspective. (Proc.) Thermal Spray: Meeting the challenges of the 21st century, (Ed.) C. Coddet, (Pub.) ASM International, Materials Park, OH, USA 1998, p1677-1680

[37] P. Fauchais, A. Vardelle, B. Dussoubs. Quo vadis thermal spray? Thermal Spray

2001: New Surfaces for a New Millennium, (Ed.) C. C. Berndt, K. A. Khor, E. F. Lugscheider, (Pub.) ASM International, Materials Pack, OH, USA 2001, p1-32.

[38] P. Nylén, J. Zigren, J. Idetjarn, L. Pejryd, M. Friis, P. Moretto, On-line microstructure and property control of a thermal sprayed abrasive coating. Thermal Spray 2001: New Surfaces for a New Millennium, (Ed.) C. C. Berndt, K. A. Khor, E. F. Lugscheider, (Pub.) ASM International, Materials Pack, OH, USA 2001, p1213-1219.

[39] P. Fauchais, A. Vardelle, A. Grimaud. Les dépôts par plasma thermique. Journal of high temperature chemical processes, Colloque, supplément au no 3, vol 1, September 1992, p255-271

[40] G. Montavon, C. Coddet, Quantification of partice morphologies in view of quality control of the thermal spray processes, Materials Characterization 36 (1996) p257-269

[41] L. Pejryd, L. Wigren, N. Hanner, The ultimate spray booth. Thermal Spray: A United Forum for Scientific and Technological Advances, (Ed.) C. C. Berndt, (Pub.) ASM International, Materials Pack, OH, USA 1997, p445-450.

[42] S. Ahmaniemi, P. Vuoristo, T. Mäntylä, J. Latokartano, Optimisation of the robot controlled plasma spraying of thermal barrier coating for gas turbine transition duct. International Thermal Spray Conference, Essen 4.-6.3.2002. p. 208-212.

[43] I. Hammad, Robotic arc spray. Thermal Spray Research and Applications, Proceedings of the Third National Thermal Spray Conference, Long Beach, CA, USA 1990, p 651-653

Chapitre 2

Rappel sur la CFAO-robotisée en projection thermique

2.1 Introduction

Dans le processus continu, le robot est manipulé par une série de commandes et de fonctions stockées dans un fichier dit « programme robot ». La trajectoire du robot, et par conséquence de la torche, doit être programmée avant son exécution. Deux modes de programmation des trajectoires robot sont couramment utilisés [1]:

- ü La programmation en ligne : PEL
- ü La programmation hors-ligne : PHL

Ce chapitre présente ces deux modes de programmation robot ainsi que le logiciel puissant de PHL « RobotStudio™ ». Il est ensuite explicité les améliorations nécessaires à apporter à ce logiciel pour son utilisation en projection thermique et le but de ce développement dans le cadre de cette thèse.

2.2 Programmation en ligne

La programmation en ligne est aujourd'hui le mode de programmation le plus utilisé industriellement pour la projection thermique principalement la projection sur des pièces de forme simple (Figure 2.1). Il existe deux méthodes de programmation en ligne : l'apprentissage direct et l'apprentissage indirect point à point.

Figure 2.1 Programmation en ligne [2]

2.2.1 L'apprentissage direct

Le principe de cette méthode de programmation consiste à mémoriser, lors de l'apprentissage, l'ensemble des configurations articulaires du robot pendant l'exécution de la tâche ou pendant une simulation de celle-ci. L'opérateur manipule directement le robot (ou un pantin de même morphologie) à l'aide d'un pupitre d'apprentissage pour lui « apprendre » la tâche à accomplir. L'acquisition de la trajectoire est continue. Ce type de programmation est simple et demande peu de formation pour le programmeur. Toutefois, en projection thermique, ce mode de programmation ne paraît guère viable. Tout d'abord, les trajectoires créées suivant cette méthode ne peuvent pas être modifiées par la suite par un réapprentissage. De plus, l'utilisation de ce mode d'apprentissage n'est pas compatible avec la précision requise par les procédés de projection (au niveau de la position, de l'orientation et de la vitesse d'éclairement par rapport au substrat).

Cette méthode est plutôt utilisée pour des applications qui exigent des mouvements de précision limitée, dans la projection sur les pièces simples ou le domaine de la peinture.

2.2.2 L'apprentissage indirect point par point

Le principe utilisé par cette méthode est également basé sur une démonstration matérielle de la tâche à réaliser. L'opérateur déplace le robot en mode manuel, utilisant pour cela le pupitre de commande à touches. Cependant, la configuration du robot n'est enregistrée qu'en certains points caractéristiques de la trajectoire. La liaison entre ces différents points est spécifiée par les caractéristiques de la trajectoire entre ces points (linéaire, articulaire voire circulaire) et par la définition d'une vitesse de déplacement spécifiée par l'opérateur [3].

Ce mode de programmation est actuellement le plus répandu pour les robots de projection thermique. Pour des pièces de géométrie complexe, l'élaboration d'une trajectoire précise au niveau position et orientation de la torche devient très vite extrêmement longue et fastidieuse par la multiplication des points à mémoriser.

2.3 Programmation hors-ligne

La programmation hors-ligne permet la génération de programmes avec une précision beaucoup plus importante au niveau des points de passage de l'outil (distance de projection, contrôle de l'orientation de l'outil). Le temps consacré à la programmation sur site est fortement réduit. Le positionnement des trajectoires ainsi que la conception des outils peuvent être optimisés.

Un des principaux intérêts de ce type de méthode est qu'il ne nécessite aucune immobilisation de l'outil de production durant la phase de programmation du robot. On distingue 2 approches principales pour la programmation hors-ligne des robots : la programmation par langage et la programmation graphique à partir d'un système de CFAO (Conception et Fabrication Assistée par Ordinateur).

2.3.1 Programmation par langage robotique

Les langages de programmation des robots sont proches des langages informatiques

classiques. Ils proposent des instructions particulières pour commander les mouvements des robots. Ils sont très performants pour définir des positions par calculs ou pour implanter des instructions conditionnelles ou des boucles.

Le principal inconvénient de ces langages est qu'ils restent généralement spécifiques à chaque marque de robot; il n'existe pas de standard reconnu. Cela oblige l'opérateur à connaître un langage pour communiquer avec le robot. Les langages suivants peuvent être cités: RAPID (ABB); VAL II (UNIMATION), RAIL (AUTOMATIX), PLAW (KOMATSU). La Figure 2.2 montre un outil d'éditeur du programme robot chez ABB [4].

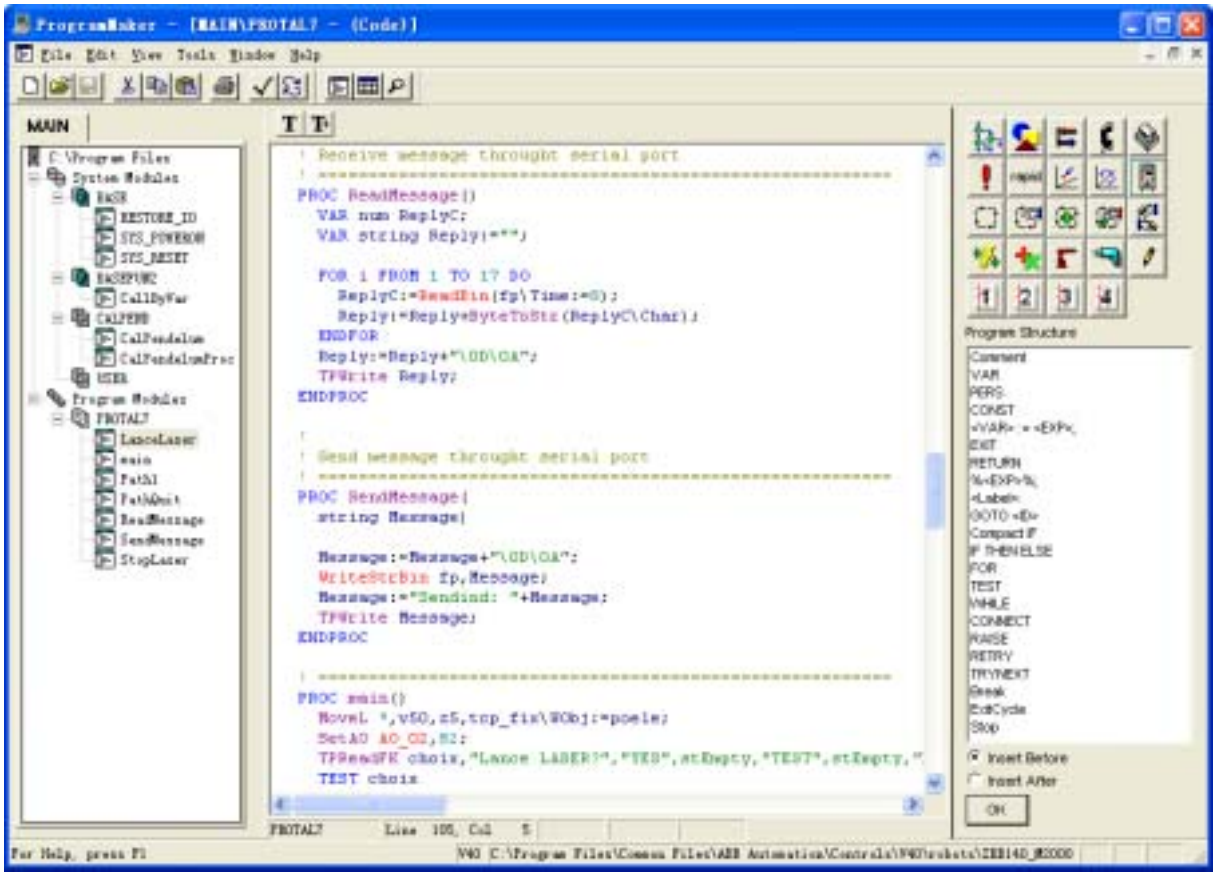


Figure 2.2 Programmation par langage robotique – RAPID (Robots ABB)

Ce type de programmation nécessite de connaître à l'avance les points de passage, l'orientation de l'outil ainsi que la vitesse locale de déplacement pour chaque point. Il est envisageable de programmer les robots de projection thermique de cette manière sur des géométries très simples de substrat, ne nécessitant que quelques points de passage reliés par un

mouvement linéaire ou circulaire à vitesse constante. En revanche, cette solution ne peut être retenue pour des géométries complexes de par le nombre de points, les orientations et les vitesses à définir.

2.3.2 Programmation graphique

Cette méthode de programmation est une méthode plus pratique pour programmer des trajectoires robot sur les pièces de forme complexe. Les logiciels de CFAO actuels permettent de représenter et de modéliser les robots, les pièces à revêtir et l'environnement de projection [5]. La Figure 2.3 présente la programmation hors-ligne à l'aide d'un logiciel CFAO robotique.

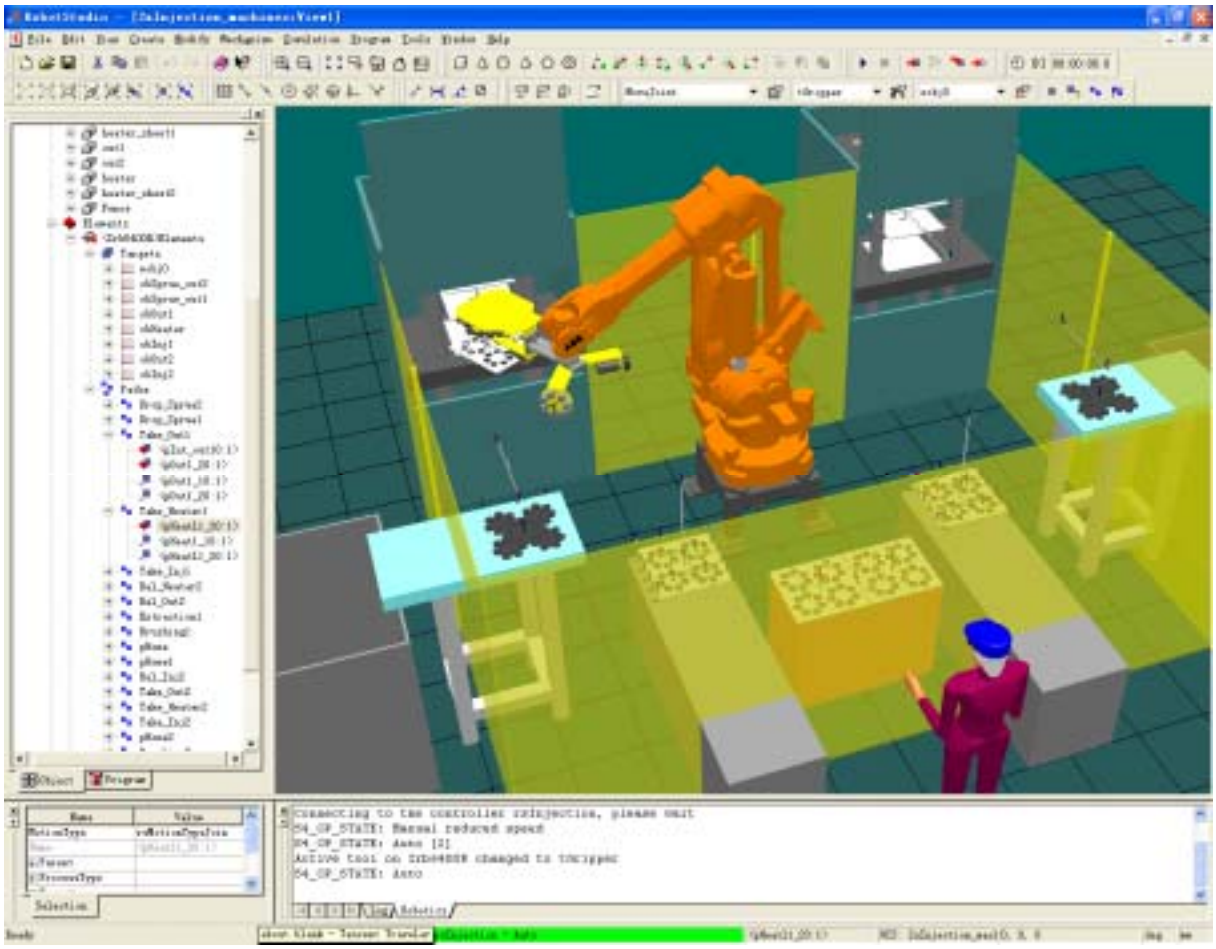


Figure 2.3 Programmation graphique

Cette approche de la programmation hors-ligne présente des avantages indéniables par rapport à l'approche par langage :

- ü les modes de visualisation en 3D utilisés par les systèmes CFAO permettent à l'opérateur d'apprécier le travail en cours,
- ü l'interactivité et la convivialité de ces systèmes permettent des modifications rapides des paramètres géométriques,
- ü il est possible de créer des bases de données géométriques pour les cellules robotisées,
- ü la puissance offerte par les systèmes informatiques actuels permet de résoudre certains problèmes de création de trajectoire et de détection de collision.

Les logiciels de CFAO mécanique proposent de nombreuses fonctions pour la création d'entités géométriques complexes. Ils possèdent souvent des modules dédiés à certaines applications (usinage, calculs de structure, éléments finis, etc.) et parfois même des modules spécialisés en robotique. Par ailleurs, les logiciels spécialisés en robotique possèdent toutes les fonctionnalités requises en terme de programmation hors-ligne. En revanche, les modeleurs géométriques dont ils disposent sont souvent trop simplifiés par rapport à ceux proposés par les logiciels de CFAO.

L'absence de standard au niveau des langages de programmation ne permet pas aux différents modules robotiques de piloter directement les robots. Les modules robotiques permettent la génération de trajectoires, mais la liaison avec le contrôleur robot est confiée à des post-processeurs qui créent des programmes dans les langages des robots par discrétisation de la trajectoire.

Dans le cadre de cette étude, le logiciel de CFAO robotique dit « RobotStudio™ » a été utilisé en raison de sa bonne compatibilité entre les robots virtuels et les robots réels. RobotStudio™ est un logiciel puissant d'ABB qui permet de simuler et de programmer hors-ligne des systèmes robots à l'aide d'un PC standard fonctionnant sous Windows [6].

2.4 RobotStudio™ [6]

RobotStudio™ est fondé sur le « Virtual Controller » d'ABB qui est une copie exacte du logiciel réel qui commande le robot en production. Il est ainsi possible d'effectuer des simulations très réalistes à l'aide de programmes de robots réels et de fichiers de configuration identiques à ceux utilisés dans l'atelier. Le Virtual Controller contient aussi un « Pupitre mobile d'apprentissage » virtuel qui permet d'utiliser le robot simulé exactement comme un vrai robot.

RobotStudio™ augmente la rentabilité du système robot en permettant d'effectuer des tâches comme la formation, la programmation et l'optimisation sans gêner la production. RobotStudio™ offre ainsi plusieurs avantages comme :

- ü Réduction des risques - vérification de nouvelles installations en créant rapidement sa propre station, en menant des études de faisabilité précises, et en vérifiant l'accessibilité, les collisions et les problèmes de singularité.
- ü Mise en service plus rapide - le temps consacré à la programmation peut être réduit et, plus important encore, cette tâche peut être effectuée avant l'installation du système
- ü Changements plus rapides - il est possible de tester les modifications apportées à des installations existantes ou de mettre à jour les programmes de production de nouvelles pièces
- ü Optimisation des programmes - disposition de plusieurs outils qui facilitent l'optimisation des programmes de robot
- ü Environnement de simulation 3D - représentation 3D complète de systèmes robotiques
- ü Système de commande virtuel - système de commande de robot intégré à un PC standard
- ü Bibliothèque de robots - modèles exacts de la gamme de produits complète ABB
- ü Pupitre mobile virtuel d'apprentissage - parfait pour la formation hors-ligne des opérateurs
- ü Éditeur de programmes RAPID - vérification automatique des erreurs de programme

- ü Simulateur des E/S - interaction avec des entrées / sorties simulées
- ü VBA (Visual Basic for Applications) - permet de développer des fonctionnalités particuliers

Pour mieux expliquer cette étude, il est nécessaire de présenter certains principes de base du RobotStudio™. Dans les paragraphes suivants, plusieurs éléments et l'usage du logiciel RobotStudio™ seront présentés.

2.4.1 Stations de RobotStudio™

D'une certaine façon, une station de RobotStudio™ correspond à une cellule de robot dans une usine. La station contient le robot, l'outil, les pièces de travail, les programmes et tout ce qui peut être utile lors de la simulation. Une seule station peut être ouverte lors du l'utilisation du logiciel.

Cette station peut être enregistrée dans un fichier. Les fichiers de station portent l'extension « .stn ». Un fichier de station contient :

- ü Un système de coordonnées dans lequel sont positionnés les objets de la station.
- ü Des références aux objets de la station.
- ü Des informations sur le système de commande virtuel utilisé dans la station.

Outre les informations stockées dans le fichier de la station, toutes les stations possèdent un dossier contenant des fichiers de support. Ce dossier contient les éléments suivants (Tableau 2.1):

Tableau 2.1 Éléments contenues dans une station

Élément	Dossier	Description
fichier *.jt	dossier *nom de la station* __supporting_files	Fichiers graphiques 3D
fichier *.sat	dossier *nom de la station*	Fichiers CAO

	__supporting_files	
fichier *.vba	dossier *nom de la station* __supporting_files	Fichiers de code Visual Basic
fichier *.stationBackup.stn	dossier *nom de la station* __supporting_files	Fichier de sauvegarde de la station. Ce fichier est une sauvegarde du dernier enregistrement du fichier de la station.
fichier *.AddinData	dossier AddinData	Fichiers enregistrés par les modules add-in utilisés dans la station.

2.4.2 Systèmes de coordonnées

Pour qu'un robot puisse manipuler un objet physique, il doit pouvoir le localiser. La détermination cohérente des positions dans l'espace représente une tâche fondamentale pour tous les systèmes robots. C'est pourquoi des systèmes de coordonnées sont utilisés.

Dans RobotStudio™, plusieurs systèmes de coordonnées peuvent être utilisés pour effectuer des positionnements, ou pour organiser les positions et d'autres objets (Figure 2.4). Les systèmes de coordonnées peuvent être imbriqués les uns dans les autres ; par exemple, une entité est située dans une pièce qui se trouve dans le système de l'utilisateur de l'atelier.

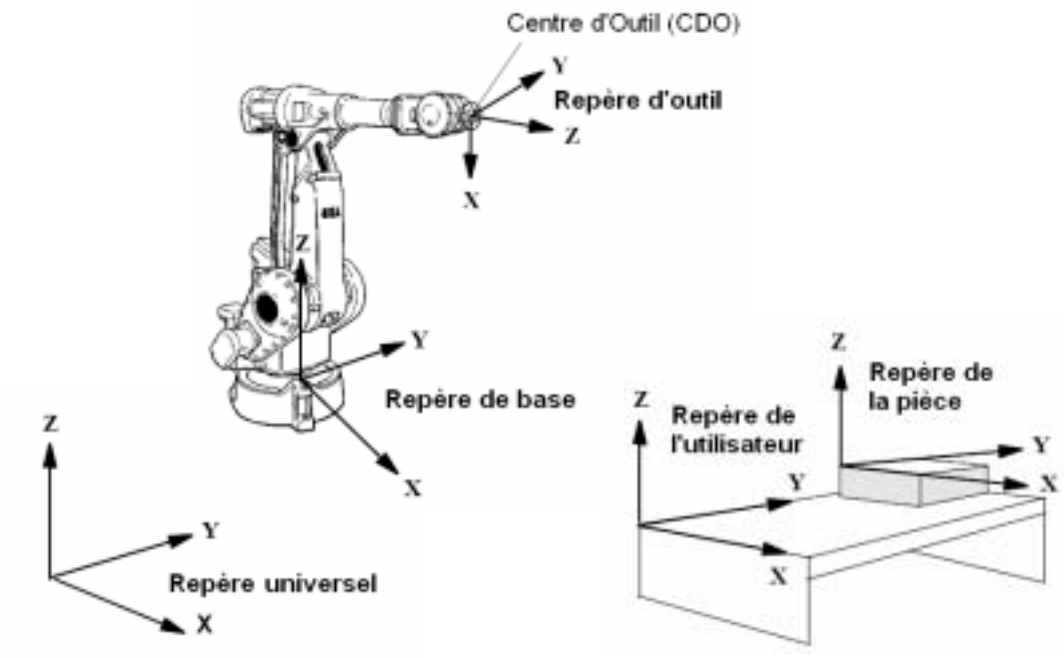


Figure 2.4 Systèmes de coordonnées utilisés dans RobotStudio™ [7]

Les coordonnées du centre d'outil CDO (appelé également TCP, Tool Center Point) sont toujours exprimées par rapport au repère associé à la plaque du robot. La Figure 2.5 décrit les repères plaque et outil d'un robot. Il faut pouvoir définir précisément les coordonnées $/\div x_{CDO}, \div y_{CDO}, \div z_{CDO}0$ du centre outil dans le repère plaque. Cet étalonnage relève plus des procédés de métrologie classique que de robotique. Pour les applications de projection thermique, l'orientation de l'outil est généralement identique à celle de plaque et le décalage peut être déterminé par mesure manuelle (précision < 1mm).

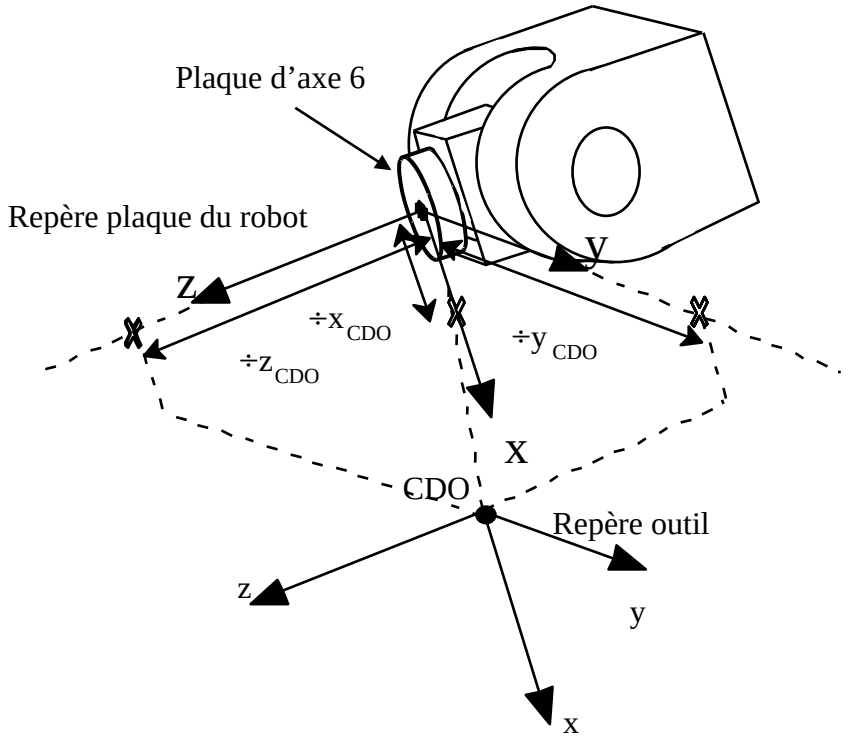


Figure 2.5 Position des repères plaque et outil

Les rotations $/\chi x, \chi y, \chi z0$ par rapport au repère plaque conditionnent l'orientation de l'outil ; pour les robots ABB, cette orientation est transformée en quaternions (cf. Annexe 1).

En projection thermique, le point qui effectue la trajectoire (point qui passe sur chacun des trièdres) n'est pas la sortie de la torche, mais un point fictif situé à une distance de l'ordre de 120 mm en APS ou 270 mm en VPS par rapport à la sortie de la buse. Il faut donc positionner

le centre d'outil CDO à la distance de projection en prenant en compte la traversée de la poudre (décollage du jet de matière par rapport à l'axe de la torche).

2.4.3 Création de la trajectoire robot

La création de trajectoires robot adaptées aux applications de projection thermique comporte 3 phases distinctes :

- ü Positionnement des pièces
- ü Création des positions
- ü Création des trajectoires

Dans les paragraphes suivants, les étapes de création d'une trajectoire seront exposées.

2.4.3.1 Positionnement des pièces

Le positionnement précis des éléments extérieurs est un gage de sécurité lors de l'élaboration de la trajectoire. La méthode la plus classique consiste à utiliser le robot comme appareil de mesure. Une pointe calibrée est alors montée sur la plaque du robot (Figure 2.6). Elle sert à définir des coordonnées caractéristiques de la pièce (angle, axe de rotation…) exprimées dans le repère du robot. Pour une pièce de révolution, deux points sur l'axe de rotation sont suffisants ; pour une pièce quelconque, 3 points sont nécessaires et suffisants. Les données recueillies sur site permettent d'ajuster précisément le positionnement de la pièce à revêtir dans l'enceinte virtuelle.

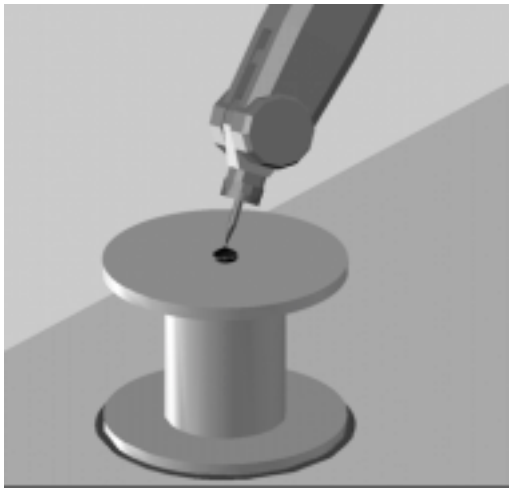


Figure 2.6 Définition d'une position

Un bon positionnement des pièces en simulation et l'absence de collision (plus d'éventuelles zones de sécurité) permet de garantir une programmation hors-ligne sans risque.

2.4.3.2 Création des points de passage

RobotStudio™ fournit 2 méthodes pour créer une position de point : l'apprentissage d'une position ou la création directe d'une position. Il est possible de créer une position en pilotant manuellement le robot vers l'emplacement de cette position (Figure 2.7).

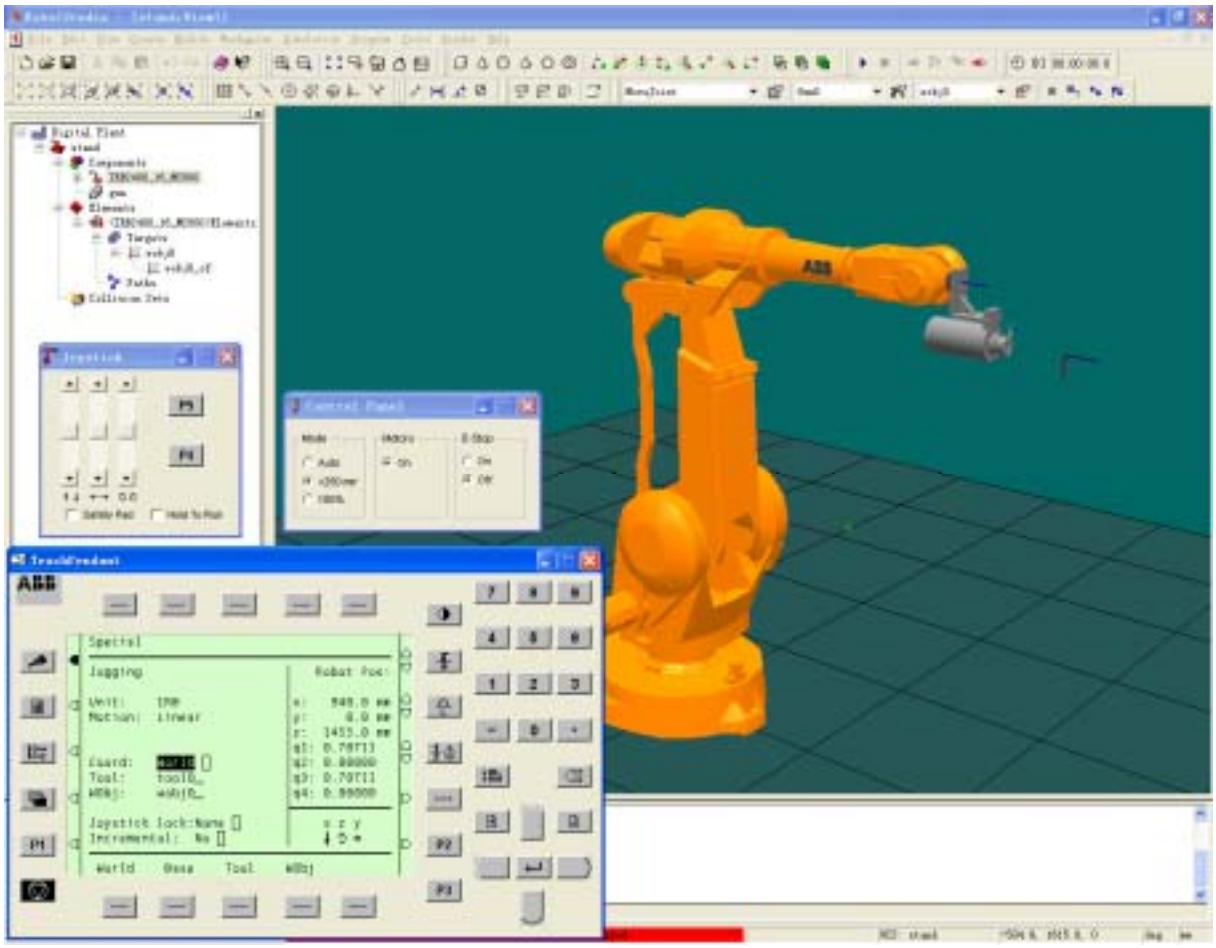


Figure 2.7 Création des positions par pupitre mobile virtuel d'apprentissage [8]

Une autre méthode de création consiste à entrer les coordonnées et l'orientation de la position directement dans la boîte de dialogue « Créer une position ». La Figure 2.8 montre les paramètres à remplir dans la boîte de dialogue.

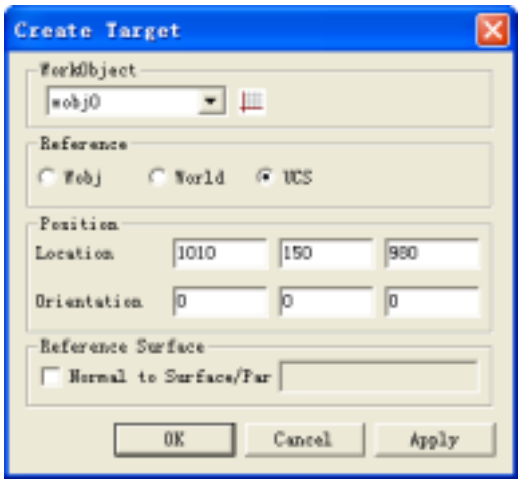


Figure 2.8 La boîte de dialogue « Créer une position »

Chaque point de passage est un trièdre direct dont l'axe z est toujours normal à la surface. Cette propriété est très intéressante pour la projection thermique puisque, dans la plupart des cas, la projection s'effectue en gardant l'outil proche de la normale à la surface locale. Toutefois, pour des problèmes de flux gazeux par exemple ou d'attaque de surface avec un angle particulier, chaque point de passage peut être ré-orienté en fonction de l'expérience du métier.

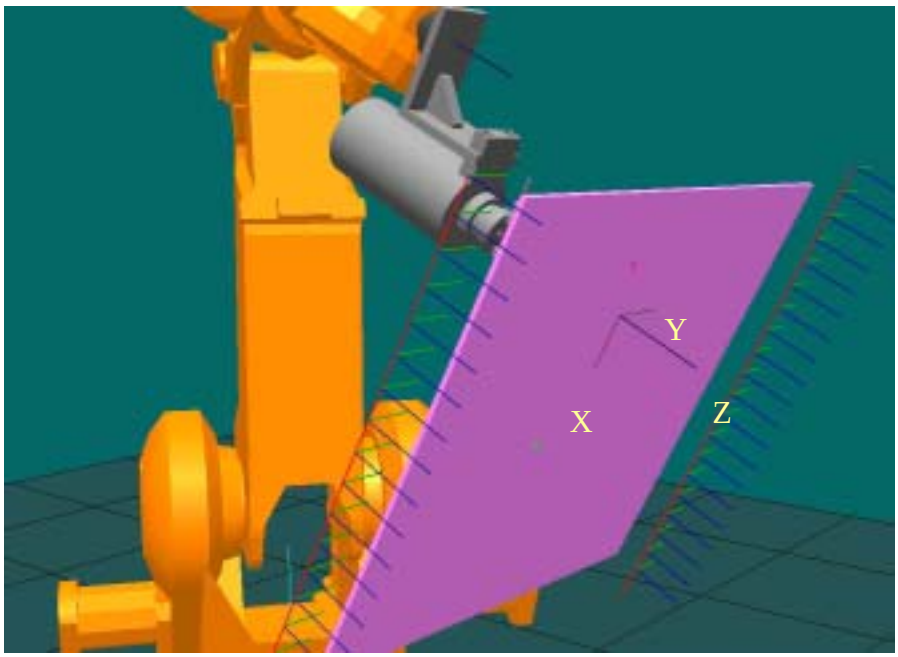


Figure 2.9 L'axe Z est normal à la surface de la pièce

Lors de l'exécution de la trajectoire, l'axe z de l'outil vient se positionner colinéairement à l'axe z du trièdre concerné à chaque point de passage. L'outil s'oriente ensuite en positionnant son axe x colinéaire à l'axe x du trièdre. Les points de passage sous forme de trièdre gouvernent donc complètement la position de l'outil dans l'espace au gré de son déplacement. Le déplacement de l'outil est contrôlé par un « path » qui regroupe les points de passage dans l'ordre de leur enchaînement.

2.4.3.3 Création des trajectoires

Une fois les points de passage créés, la trajectoire robot peut être généré simplement en joignant touts les points de passage dans un ordre conçu. Lors de la création d'une trajectoire (path), le type de mouvement entre chaque point, la zone de précision affectée à chaque point, la vitesse de l'outil aux différents points, peuvent être paramétrés comme lors de la création d'une trajectoire réelle de robot (Figure 2.10) [9].

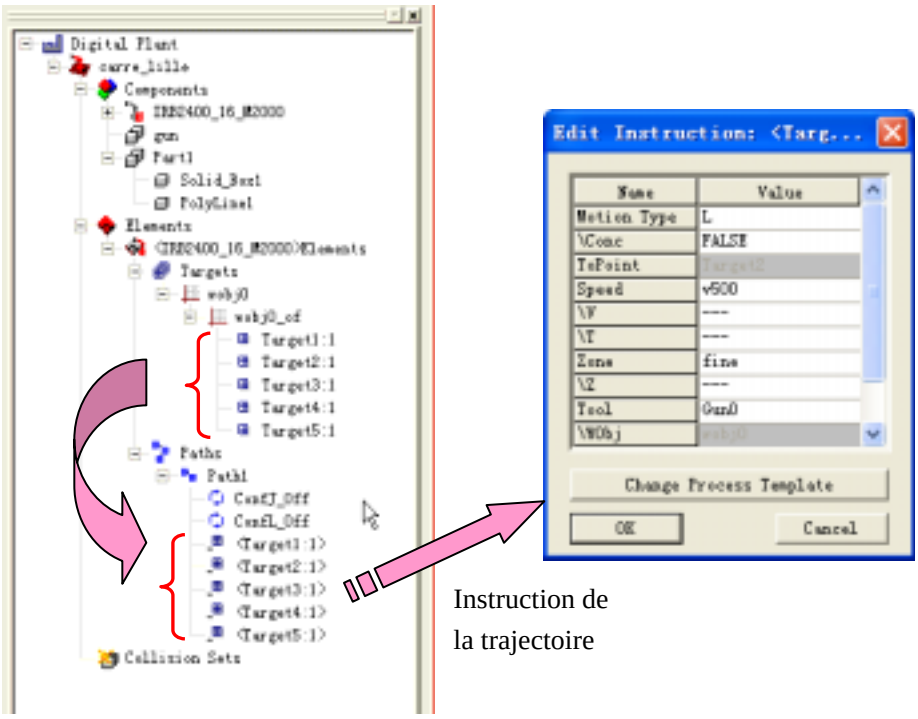


Figure 2.10 Création des trajectoires

RobotStudio™ fournit une autre manière dite « AutoPath » pour créer une trajectoire à partir d'une courbe. Cette fonction permet de générer une trajectoire sur des pièces de forme complexes efficacement et rapidement surtout pour des applications de soudage (Figure 2.11). Mais ce type de création n'est pas adaptable pour les applications de projection thermique. Les inconvénients et les limites sont exposés dans les paragraphes suivants.

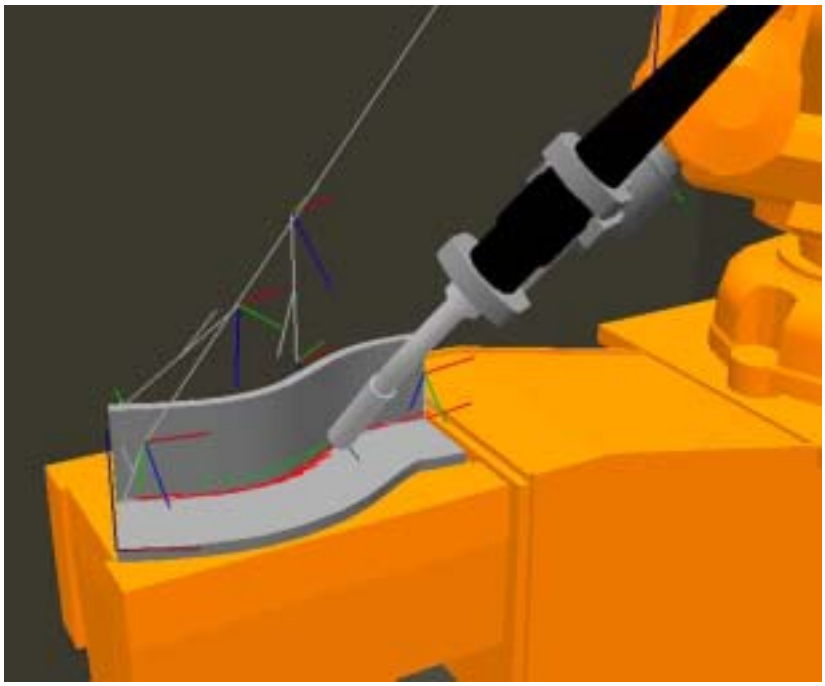


Figure 2.11 Création d'une trajectoire à l'aide d'« AutoPath » dans une application de soudage

2.4.4 Simulation de la trajectoire

La trajectoire de la torche en projection thermique doit satisfaire plusieurs critères [10]:

- ü Aucune collision avec aucun élément de la cellule
- ü Aucune reconfiguration[1] de la position de la torche de projection en vis-à-vis de la

[1] Reconfiguration = rotation non justifiée d'axes du robot qui cause une rotation d'outil du robot sur lui-même.
Elle est causée par la configuration ambiguë d'axes du robot qui dispose de plusieurs configurations d'axes pour obtenir la

pièce à revêtir [1]

- ü La distance de projection doit être maintenue constante lorsque la torche est en vis-à-vis de la pièce à revêtir
- ü L'orientation de la torche par rapport à la surface à revêtir doit être spécifiée (en principe normal) en touts les points de la surface de la pièce
- ü Le temps d'exécution de la trajectoire doit être minimal pour maximiser la productivité
- ü La vitesse de projection doit être en principe constante lorsque la torche est en vis-à-vis de la pièce à revêtir afin de garantir une épaisseur de dépôt homogène
- ü Aucun arrêt de la torche de projection en vis-à-vis de la surface à revêtir n'est permis

Une fois déterminés les points de passage, la configuration de l'outil, et éventuellement la plume de matière, l'exécution de la trajectoire doit être simulée avant la vraie projection. Un panneau permet de contrôler les variations angulaires de chaque axe du robot tout au long de la simulation (Figure 2.12).

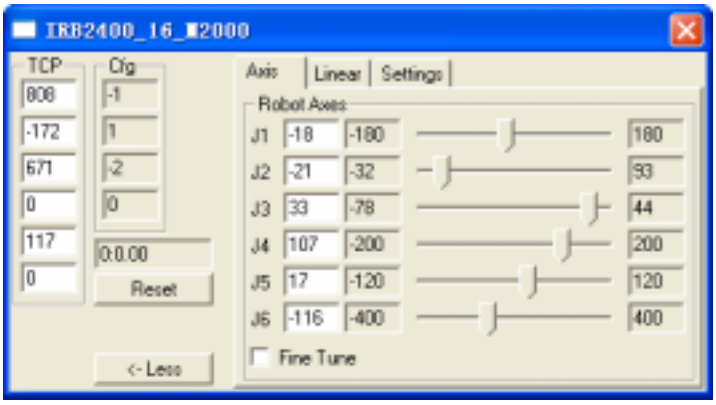
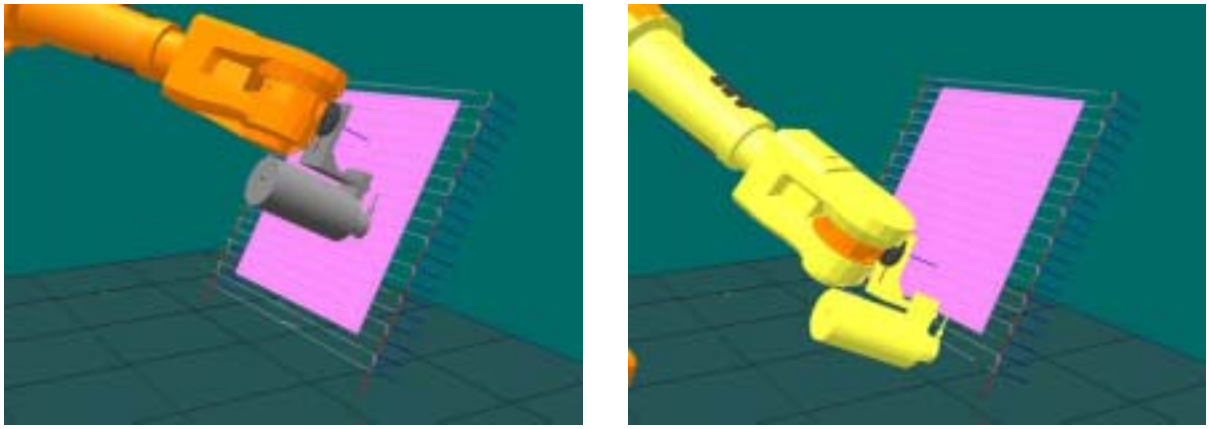


Figure 2.12 Panneau de contrôle des axes d'un robot

Si un point dans la trajectoire n'est pas atteignable, la simulation s'arrête. RobotStudio™ fournit aussi une fonction pour détecter une collision possible en affectant une couleur sur les objets dans la zone de collision. Le jaune par exemple signifie qu'une proximité de collision

même position et orientation d'outil. Il faut donc éviter l'alignement d'axes 4 et 6 du robot.

est détectée (Figure 2.13).



a. position sans risque b. proximité de collision

Figure 2.13 Détection de collision dans RobotStudio™

Une trajectoire sans collision est bien sûr une garantie très importante de sécurité et de qualité du procédé en projection thermique. Toutes les trajectoires robot effectuées dans cette étude ont évidement passé la détection de collision.

2.5 Nécessités d'amélioration pour RobotStudio™

2.5.1 Caractéristiques de la trajectoire robot pour la projection thermique

La trajectoire pour les applications de projection thermique est relativement particulière par rapport à celle des autres applications industrielles, comme le soudage et le coupage par exemple. Pour élaborer une couverture complète de la pièce, il faut générer une trajectoire en composant une série de plusieurs courbes parallèles sur la surface à revêtir. La distance entre deux courbes constitue le pas de balayage, un des paramètres opératoires du procédé (cf. Chapitre 1.2.4). La Figure 2.14 présente une forme quelconque sur laquelle un certain nombre de sections parallèles ont été effectuées.

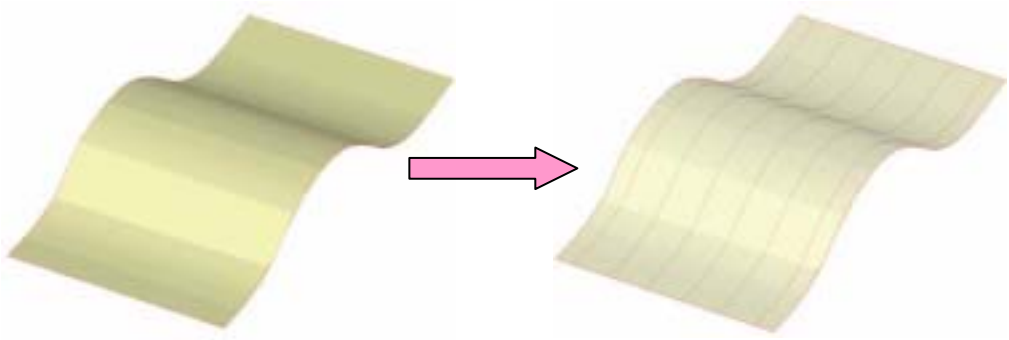
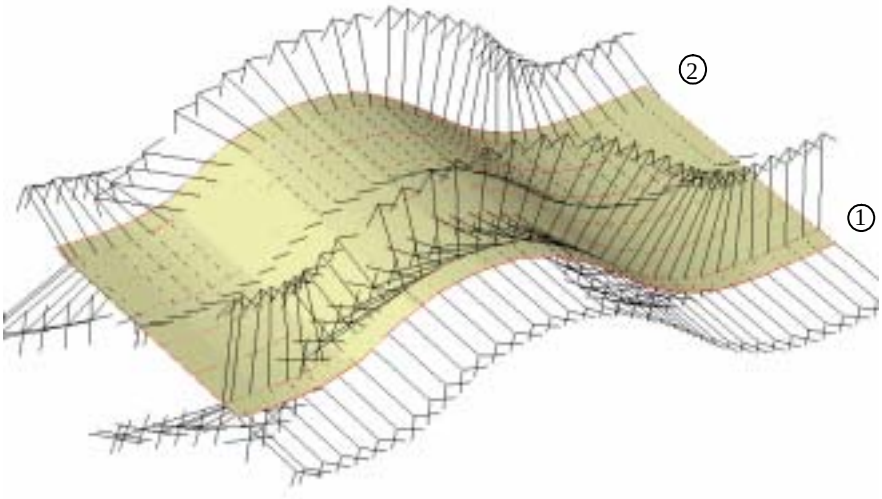


Figure 2.14 Les courbes parallèles sur la surface de la pièce définissent le pas de balayage

En plus, l'orientation de chaque point sur la courbe doit être normal ou faire un angle constant par rapport à la surface à revêtir afin d'obtenir un rendement identique lors de la projection.



Série de trièdres dont les axes z sont normaux à la surface
Série de trièdres dont les axes z sont à 30° de la normale locale à la surface

Figure 2.15 Création et orientation de points de passage [1]

Dans le pratique de la projection, bien que seuls les points sur la pièce soient à atteindre (Figure 2.16 a), il s'avère nécessaire, afin d'offrir au robot la possibilité d'accélérer et

décélérer et donc garantir le respect de la vitesse définie en regard de la pièce, de prévoir un certain débordement latéral (Figure 2.16 b). C'est à dire que certains points de passage ne sont pas sur la courbe de forme de la pièce mais plutôt sur une prolongation de cette courbe.

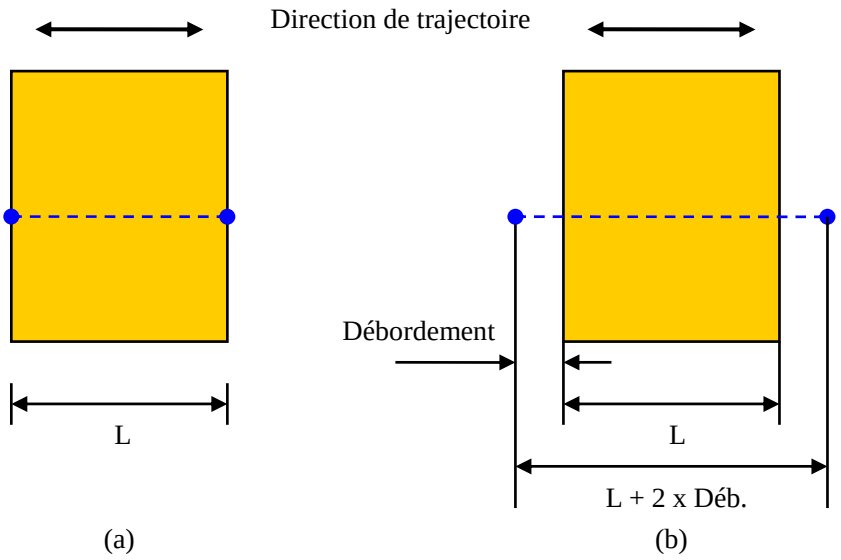


Figure 2.16 Trajectoire de la torche en dessus d'une pièce

(a) sans débordement (b) avec un débordement égal de chaque coté de la pièce

La valeur de débordement dépend de la charge supportée par le robot (le poids de la torche, de l'outil) et de la vitesse demandée relative torche – substrat. La Figure 2.17 présente des diagrammes de la distance de transitoire en fonction de la vitesse. Pour garantir une vitesse constante de projection sur la surface de la pièce, il faut donc prévoir une distance de débordement assez longue pour permettre d'éviter les zones d'accélération et de décélération ; par exemple 30 mm pour une torche de 5 kg (montage et câbles inclus) et une vitesse de 300 $mm.s^{-1}$.

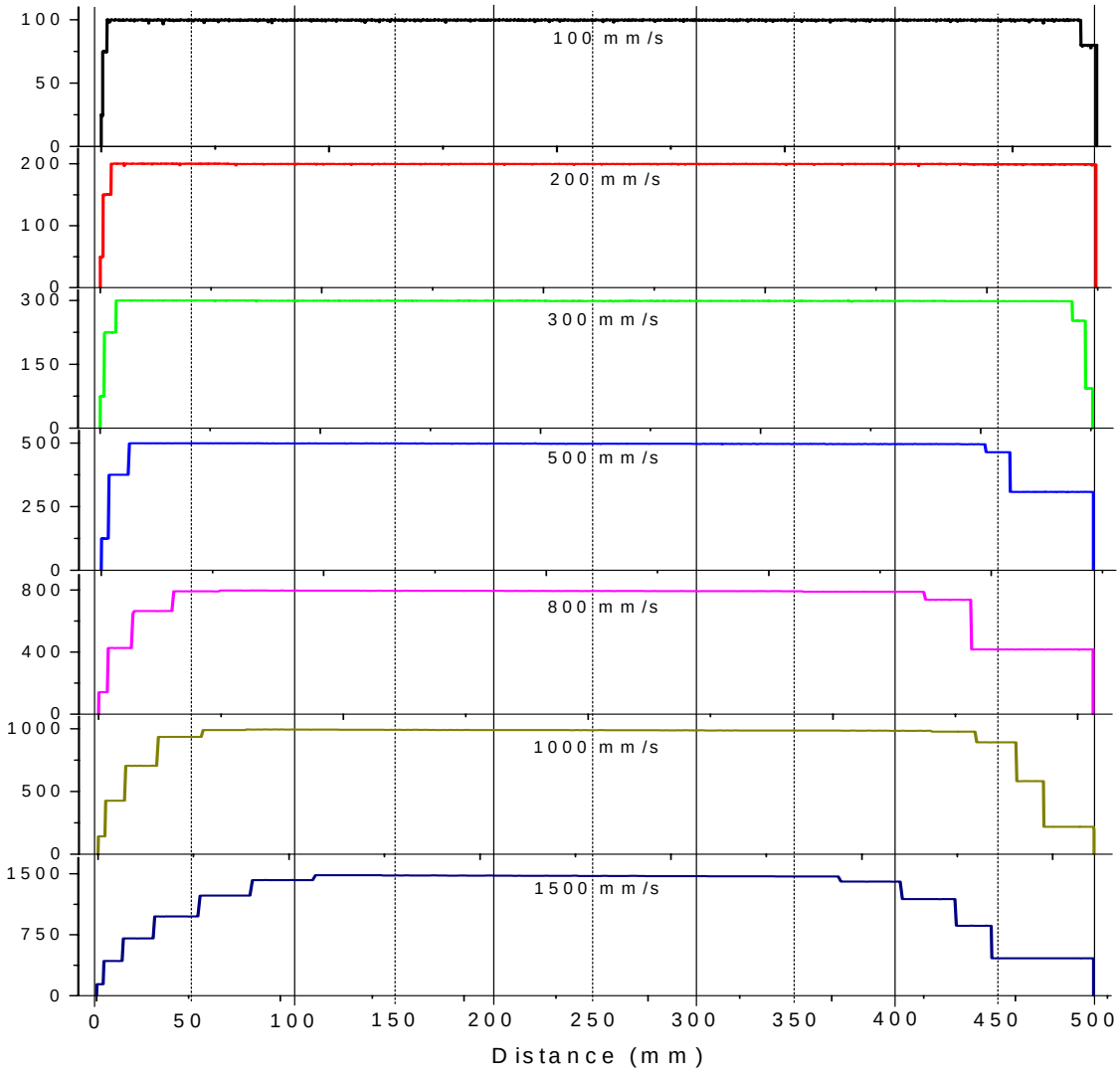


Figure 2.17 Vitesses réelles relevées sur une trajectoire linéaire de 500 mm en fonction de la vitesse différente demandée (robot ABB IRB 4400 équipé d'une torche F4)

D'après la Figure 2.17, il est constaté que plus la vitesse de déplacement programmée est grande, plus la zone transitoire correspondant aux périodes d'accélération / décélération est grande. Ainsi, la distance de débordement doit être déterminée en fonction de la vitesse demandée.

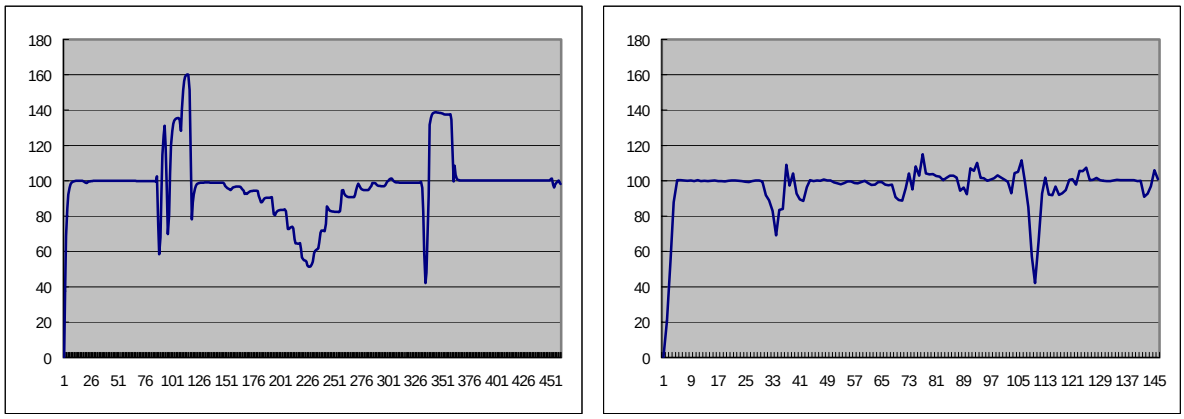
2.5.2 Inconvénients de la génération des trajectoires par RobotStudio™

Un logiciel de simulation robotique est un outil informatique qui peut permettre des gains de temps et de qualité dans le cas d'un système de conception et / ou de fabrication assistée par ordinateur (CAO et FAO). RobotStudio™ est un logiciel de CFAO robotique basé sur l'usage d'un modèle mathématique d'un robot réel. Il offre ainsi à l'utilisateur un cadre de travail virtuel pour concevoir et simuler l'espace de travail et programmer hors-ligne les trajectoires robot.

RobotStudio™ fournit de nombreuses méthodes pour générer des trajectoires spéciales dans différents cas industriels, surtout pour les applications de soudage ou d'encollage qui occupent une grande part de marché des applications robotiques. Par exemple, RobotStudio™ permet de générer une trajectoire de soudage par rapport à la courbe d'intersection de deux surfaces (Figure 2.11). Mais, ce type de trajectoire n'est pas convenable pour la projection thermique puisque la projection thermique a besoin de plusieurs courbes parallèles par rapport au profil de surface (Figure 2.14, Figure 2.15) ainsi que d'autres spécificités déjà décrites pour générer une trajectoire adaptée (orientation constante, vitesse constante, …).

RobotStudio™ fournit même des commandes spéciales pour le soudage à points et le soudage à l'arc ; par exemple la commande « SpotL » bouge le robot par rapport une trajectoire linéaire et effectue un soudage par points au début du mouvement [11]. Cependant, il n'existe pas de commandes pour faire varier la vitesse du robot pendant le mouvement entre deux ou plusieurs points, cela sera utile dans l'application de projection thermique sur pièces en rotation.

RobotStudio™ permet aussi de simuler la trajectoire et la vitesse du mouvement du robot. La Figure 2.18 présente la vitesse simulée dans RobotStudio™ et la vitesse acquise par MonitorKit (cf. Chapitre 4.3). On observe ainsi des différences entre la simulation et la réalité [12]. S'agissant d'un paramètre opératoire très important, la seule simulation de la vitesse n'apparaît pas suffisante pour l'application de projection thermique.



a. Vitesse simulée dans RobotStudio™ b. Vitesse réelle acquise par MonitorKit

Figure 2.18 Tolérances entre la simulation et la réalité

Il est donc apparu nécessaire d'améliorer les fonctions de ce logiciel pour le rendre plus adapté aux applications de projection thermique.

2.5.3 Améliorations pour RobotStudio™

Compte tenu de ce qui précède, un certain nombre d'améliorations du logiciel RobotStudio™ ont été envisagées [3,13]:

- ü Projets de génération automatique de trajectoire adaptée à l'application de projection thermique (répétition parallèles et débordement)
- ü Traitement et conception des courbes sur pièces de forme géométrique complexe
- ü Facilité de modification et/ou régénération des paramètres opératoire de projection thermique
- ü Contrôle en temps réel du robot lors du processus de projection thermique
- ü Exportation des paramètres cinématiques de la trajectoire robot
- ü Analyse et caractérisation de la surface / interface du dépôt pour évaluer la qualité de projection thermique
- ü Stratégie de choix des meilleurs paramètres opératoires par rapport au profil du

cordon et aux conditions de projection thermique

Le chapitre suivant présentera le développement d'une extension logicielle nommée « Thermal Spray Toolkit » qui améliore les fonctionnalités de RobotStudio™ pour les applications de projection thermique.

Références

[1] R. Bonnet. La projection thermique sur des formes complexes: simulation et étalonnage du procédé robotise. Thèse de doctorat. Université de Technologie de Belfort-Montbéliard, France, 2000.

[2] ABB, RobotStudio™ Trainning CD, Sweden, 2001

[3] S. Deng, H. Liao, C. Zeng, C. Coddet, New functions of Thermal Spray Toolkit, a software developed for off-line and rapid robot programming. Thermal Spray 2006: Building on 100 years of success, ASM International, Materials Park, Ohio, USA, 2006

[4] ABB, ProgramMaker 4.4, Guide de l'utilisateur, 3HAC 021 414-001 Rev. A, Sweden, 2001.

[5] S. Deng, H. Liao, C. Zeng, C. Coddet, Robotic trajectory autogeneration in thermal spraying. Thermal Spray 2005: Explore its potential!, (Ed.) E. Lugscheider, ASM. International, Materials Park, Ohio, USA, 2005

[6] ABB, RobotStudio™ Guide de l'utilisateur, 3HAC 7793-1 for BaseWare OS 4.0. Sweden; 2001.

[7] ABB, Product Specification – Industral controller 3HAC. 13335-1/M2000 BaseWare OS 4.0/Rev.0, Sweden, 2001.

[8] ABB, QuickTeach 5.4, Guide de l'utilisateur, 3HAC 021409-001 Rev. A. Sweden, 2001.

[9] ABB, Product Specification – RobotWare Options 3HAC 9218-1/Rev.2 BaseWare OS 4.0. Sweden, 2001.

[10] F.I. Trifa. Modèle de dépôt pour la simulation, la conception et la réalisation de revêtements élaborés par projection thermique. Thèse de doctorat. Université de Technologie de Belfort-Montbéliard, France, 2004.

[11] ABB, RAPID Reference Manual, Overview / System Data Types and Routines, 3HAC 7783-1 / 7774-1, For BaseWare OS 4.0, Sweden, 2001.

[12] S. Deng, H. Liao, R. Bonnet, C. Zeng, C. Coddet, Real time monitoring of robot trajectory in thermal spraying. Thermal Spray 2004: Advances in technology and application, ASM International, Materials Park, Ohio, USA, 2004

[13] S. Deng, H. Liao, C. Zeng, P. Charles, C. Coddet, Development of robotic trajectories autogeneration in thermal spraying - A new extended program of ABB RobotStudio™, Rencontres Internationales sur la projection thermique 2005, Dec 2005, Lille France

Chapitre 3

Extension logicielle pour RobotStudio™

Ce chapitre présente la méthode et les étapes de développement du logiciel « Thermal Spray Toolkit » (T.S.T.), ainsi que la méthode de génie logiciel employée pendant toute la période de développement du T.S.T.

3.1 Introduction

Compte tenu des éléments décrits dans le chapitre précédant, RobotStudio™ nécessite une personnalisation d'une extension de ses possibilités dédiées à la projection thermique. Il existe deux types d'extension logicielle dans RobotStudio™ : l'Add-in et la macro [1].

- ü Add-in – un programme qui ajoute des fonctions et des commandes optionnelles à RobotStudio™. Pour pouvoir utiliser un add-in, une installation et un chargement sont nécessaires. Une fois que l'add-in est chargé, il est disponible dans RobotStudio™ et il ajoute les commandes associées aux menus appropriés.
- ü Macro – une série de commandes et de fonctions stockées dans un programme Visual Basic, qui peuvent être exécutées qu'une fois pour effectuer une certaine tâche. Il est utile d'automatiser une tâche répétitive dans RobotStudio™ à l'aide d'une macro.

Comparé à la macro, l'add-in peut manipuler les éléments de RobotStudio™ (le robot, la position, la trajectoire, etc.) directement à l'aide de RobotStudio™ API (Application Programming Interface). Comme il s'est révélé nécessaire de développer une extension logicielle aux fonctionnalités flexibles pour l'application de projection thermique, l'extension « Thermal Spray Toolkit » a donc été développée à partir de RobotStudio™ API (cf. Annexe 2) comme un programme d'add-in qui doit être chargé avant exécution [2,3] (cf. Annexe 3).

Les paragraphes suivants présentent les détails et les étapes de développement du T.S.T.

3.2 Prérequis au développement

Cette étude nécessite la mise en œuvre d'une installation robotisée de projection APS, des

logiciels de CFAO et des outils de développement.

3.2.1 Installation de projection thermique

L'installation de projection est nécessaire pour la réalisation et l'exécution des trajectoires robot créées par T.S.T.

La torche de projection thermique utilisée pour les essais de validation est une torche à plasma d'arc soufflé atmosphérique de type F4 (fabriquée par Sulzer Metco). Cette torche est très couramment employée en projection APS. Elle est généralement équipée d'une buse « standard » de 6 millimètres et munie d'un porte – injecteur de poudre de 1.5 - 2 mm de diamètre interne placé à 6 millimètres en aval de la sortie de la torche (Figure 3.1).

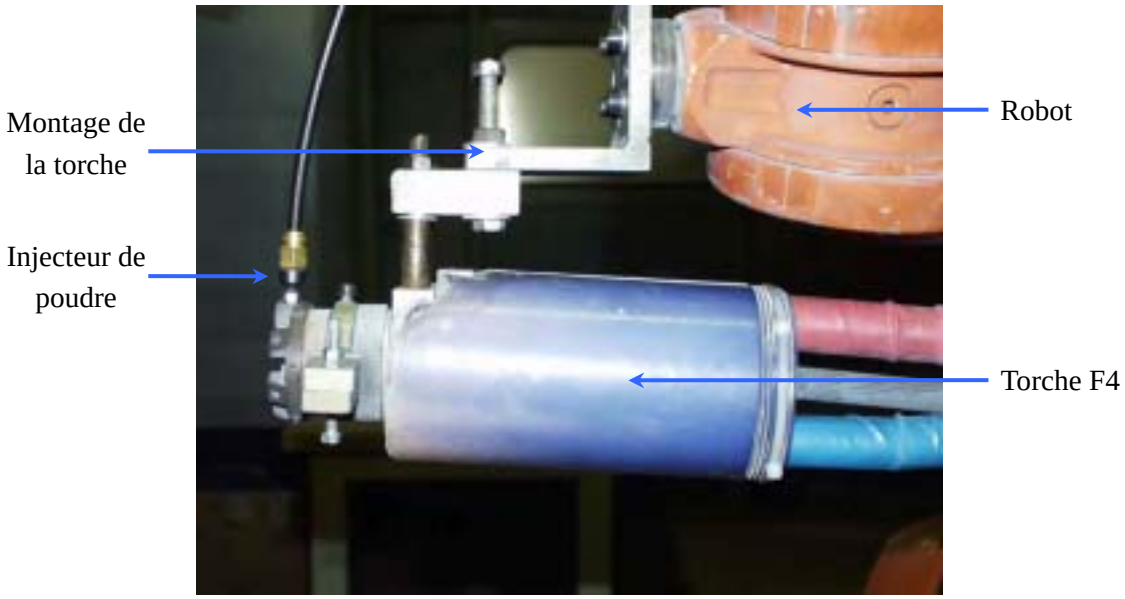


Figure 3.1 Torche utilisée: F4 Sulzer Metco

Le manipulateur de la torche considéré dans nos essais est un robot industriel d'ABB à 6 degrés de liberté (6 axes). Un robot IRB 4400 M98 par exemple, est composé de 6 axes de mobilité. Il a un rayon maximal d'action approximatif de 1.955 m [4] (Figure 3.2). Le Tableau 3.1 présente les différents mouvements du robot IRB 4400.

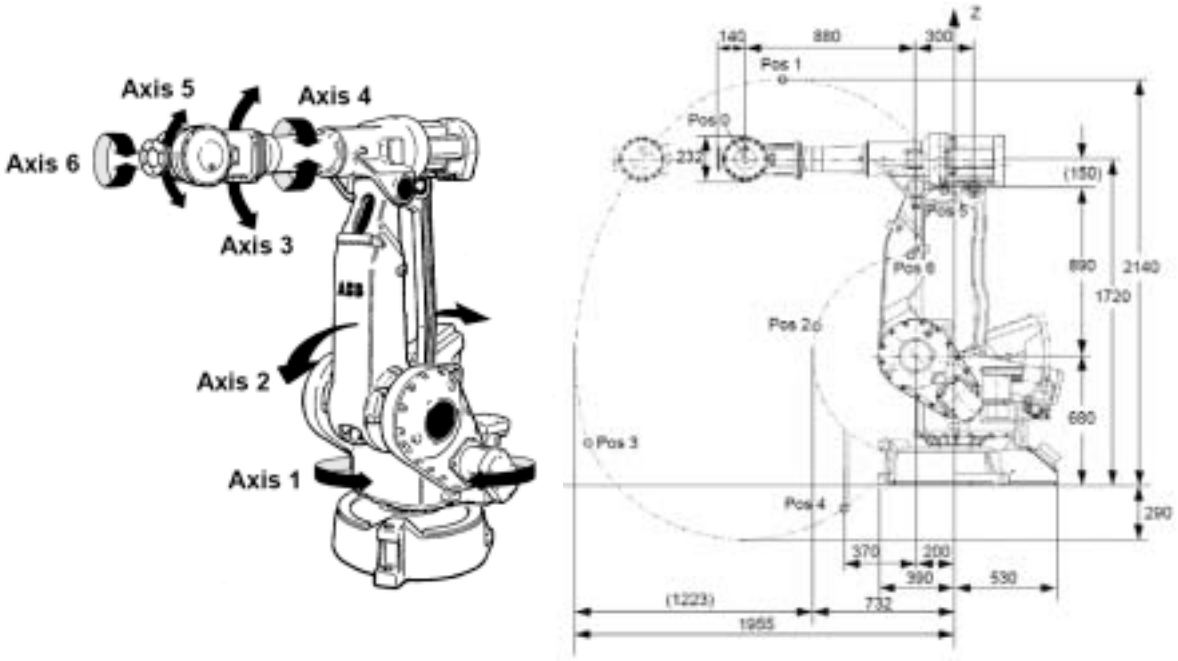


Figure 3.2 Robot IRB 4400 (ABB) et description de la zone d'action

Tableau 3.1 Mouvements des axes

Axe	Type de mouvement	Rayon d'action
Axe 1	Mouvement de rotation	-165° à +165°
Axe 2	Mouvement du bras	-80° à +95°
Axe 3	Mouvement du bras	-60° à +65°
Axe 4	Mouvement du poignet	-200° à +200°
Axe 5	Mouvement de flexion	-120° à +120°
Axe 6	Mouvement de pivot	-400° à +400°

3.2.2 Logiciels de CAO / FAO

Dans le cadre de cette étude, les logiciels commerciaux utilisés sont Pro/ENGINEER[1] (version 2001i), CATIA[2] (version 5.9) et RobotStudio™ (version 3.2).

Pro/ENGINEER et CATIA sont dédiés à la modélisation solide paramétrique des pièces à revêtir ainsi que des éléments présents dans l'environnement de travail du robot. Les logiciels de CAO mécanique proposent de nombreuses fonctions pour la création d'entités géométriques complexes. Ils possèdent souvent des modules dédiés à certaines applications (usinage, calculs de structure, éléments finis...) et parfois même des modules spécialisés en robotique.

Le logiciel RobotStudio™ a été employé dans cette étude pour toutes les fonctionnalités requises en terme de programmation hors-ligne et de simulation cinématique du robot. En revanche, le modeleur géométrique dont il dispose est souvent trop limité par rapport à ceux proposés par les logiciels de CAO mécanique. RobotStudio™ est l'environnement logiciel où l'extension logicielle (T.S.T.) a été développée.

3.2.3 Outils de développement

Dans les travaux de cette étude, le développement du T.S.T. a été réalisé à l'aide de plusieurs outils de développement incluant :

- Microsoft Visual Basic 6.0[3] – outil de développement principal du T.S.T.
- Borland C++ Builder 5.0[4] – outil de développement du module MonitorKit
- Microsoft Visual C++ 6.0[5] – outil de développement de l'interface du protocole RAP

[1] Pro/ENGINEER – un logiciel CAO développé par Parametric Technology Corporation (PTC) [http://www.ptc.com/]

[2] CATIA – un logiciel CAO développé par Dassault Systèmes [http://www.dassault.fr/]

[3] Microsoft Visual Basic 6.0 – un logiciel de développement en langage BASIC développé par Microsoft Corporation [http://www.microsoft.com/]

[4] Borland C++ Builder 5.0 –un logiciel de développement en langage C/C++ développé par Borland Software Corporation [http://www.borland.com/]

- ü Distinct ONC RPC/XDR Toolkit[6] – compilateur du protocol de RPC (Remote Procedure Call) pour la génération de l'interface du protocole RAP (Robot Application Protocol)
- ü RobotStudio™ API – interface de programmation d'application de RobotStudio™ qui permet de lecture/écriture des variables ou des paramètres dans l'environnement de RobotStudio™
- ü SGI OpenGL[7] – une spécification employée dans MonitorKit qui définit une API multi plate-forme pour la conception d'applications générant des images 3D/2D.

3.3 Technologies de programmation utilisées

Certaines technologies de programmation ont été employées lors du développement du T.S.T. incluant le compteur de temps (timing) à haute précision sous l'environnement Windows 32 et la programmation multitâche (multi-thread). Les paragraphes suivant présentent les détails de ces techniques, puis la structure des données utilisées dans T.S.T.

3.3.1 Compteur de haute précision

Il existe deux méthodes dans le module MonitorKit pour obtenir la vitesse du robot : une méthode de calcul et une méthode d'acquisition. Ces deux méthodes ont besoin d'un compteur de haute précision qui permette de compter l'intervalle entre les points et d'enregistrer le temps sur chaque point. Il est donc nécessaire de disposer un compteur de haute précision dans l'environnement Windows 32 (32 bit).

Plusieurs minuteries de différentes précisions sont offertes par le système d'exploitation

---

[5] Microsoft Visual C++ 6.0 – un logiciel de développement en langage C/C++ développé par Microsoft Corporation [http://www.microsoft.com/]

[6] Distinct ONC RPC/XDR Toolkit – une trousse à outil logiciel de compilateur du protocole RPC développé par Distinct Corporation [http://www.distinct.com/]

[7] OpenGL – Open Graphics Library, une interface d'application graphique développé par Silicon Graphics, Inc. [http://www.sgi.com/]

Windows (Tableau 3.2).

Tableau 3.2 Minuteries de différente précision

Fonction	Unité	Résolution
Now, Time, Timer	secondes	1 seconde
GetTickCount	millisecondes	approx. 10 ms
TimeGetTime	millisecondes	approx. 10 ms
QueryPerformanceCounter	QueryPerformanceFrequency	niveau de µs

Windows est un système d'exploitation à temps partagé qui est capable d'exécuter plusieurs processus de façon « quasi-simultanée ». Le sens de processus doit être pris comme quelque chose qui prend du temps, donc qui a un début et (parfois) une fin. Un processus peut-être démarré par un utilisateur par l'intermédiaire d'un périphérique ou bien par un autre processus : les applications des utilisateurs sont des (ensembles de) processus. Les fonctions comme Timer, GetTickCount, TimeGetTime peuvent être interrompues par un autre processus ayant une priorité plus élevée. C'est pour cette raison que ces fonctions ne proposent que quelque milliseconde de résolution. Dans notre cas il est requis un compteur de haute résolution, pour cela il faut donc utiliser « QueryPerformanceCounter » et « QueryPerformanceFrequency » pour effectuer des minutages de haute résolution (niveau de la microseconde) [5].

« QueryPerformanceCounter » et « QueryPerformanceFrequency » sont des fonctions de minutage implémentées dans kernel32.dll (Dynamic Link Library, Bibliothèque de Liens Dynamiques) qui permettent de lire le nombre de cycles processeurs depuis l'allumage de l'ordinateur [6,7]. Ces fonctions et les types de données qu'elles utilisent sont définis dans le fichier « windows.h ». Les prototypes et les types de données sont définis comme suit :

```
BOOL QueryPerformanceFrequency (LARGE_INTEGER *lpFrequency);
BOOL QueryPerformanceCounter (LARGE_INTEGER *lpCount);
```

Le nombre de cycles processeurs est stocké dans le type de données LARGE_INTEGER qui se définit par une déclaration de la forme :

```
typedef union _LARGE_INTEGER
{
    struct
    {
        DWORD LowPart ;          // le type entier de 4 octets
        LONG HighPart ;          // le type entier de 4 octets
    };
    LONGLONG QuadPart ;          // le type entier de 8 octets
} LARGE_INTEGER ;
```

La fonction « QueryPerformanceCounter » enregistre une position de départ et de fin du processus du compteur de haute précision. La fonction « QueryPerformanceFrequency » retourne la valeur de résolution du compteur système pour déterminer le temps écoulé. Le code de réalisation du compteur de haute précision est le suivant :

```
LARGE_INTEGER litmp;                    // définir les variables
LONGLONG QPart1,QPart2;                 // définir les variables
double dfMinus, dfFreq, dfTim;
QueryPerformanceFrequency(   litmp);    // obtenir la résolution du système
dfFreq = (double)litmp.QuadPart;        // extraire la valeur de la résolution
QueryPerformanceCounter(   litmp);      // noter la position de départ
……
……                                      // le module à compter
……
QueryPerformanceCounter(   litmp);      // noter la position de fin
QPart1 = litmp.QuadPart;                // extraire la valeur de la position
QPart2 = litmp.QuadPart;                // extraire la valeur de la position
```

```
dfMinus = (double)(QPart2   QPart1);        // obtenir la différence
dfTim = dfMinus / dfFreq;                   // calculer le temps écoulé
```

3.3.2 Multithread

La terme « multithread » désigne la possibilité qu'à une application de répartir ses tâches de façon indépendante les unes des autres pour exploiter au mieux le temps du processeur [8].

Par « thread », on entend une unité de traitement, et par « fonctionnement multitâche » l'exécution simultanée de plusieurs threads. Les systèmes d'exploitation Windows NT, 2000, XP sont des systèmes « multitâche préemptif » : le processeur décide d'affecter à chaque thread un certain créneau de temps par rapport à leurs priorités (cf. Annexe 4), puis bascule d'un thread à l'autre après épuisement de cette durée. Dans ce cas, le programmeur n'a pas besoin de se préoccuper de rendre la main au processeur.

Quand le créneau de temps affecté au thread en cours d'exécution est épuisé, le processeur mémorise dans la pile le « contexte » (état des registres, etc.) du thread en cours, puis restaure dans ses registres le contexte du thread auquel il va redonner la main. Ce processus porte le nom de « basculement de contexte ».

Un autre avantage du multithread est la simplification de la conception. Le multithread permet de simplifier l'architecture des systèmes complexes en employant des files d'attente et des traitements asynchrones. Cette philosophie permet de créer des systèmes fiable, robustes et extensibles. Dans le MonitorKit, les modules de fonctionnalité individuels sont donc mis dans différents threads (files d'attente) en vue d'une gestion des tâches plus pratique et plus fiable.

Quatre threads individuels sont créés dans le processus principal du MonitorKit ; ce sont :

- ü Thread de communication – communication avec le robot connecté par le protocole RAP pour obtenir les paramètres cinématiques du robot en temps réel.
- ü Thread d'acquisition – acquisition en temps réel de la vitesse en chaque point de la trajectoire par la carte d'acquisition externe.
- ü Thread de monitor (contrôle) – contrôle en temps réel du mouvement du robot et

lecture d'état du contrôleur robot et des signaux entré/sortie.

- ü Thread de playback – démonstration d'une trajectoire robot obtenue par le thread d'acquisition ou une trajectoire enregistrée dans un fichier.

C'est le processus principal du MonitorKit qui manipule et gère les différents threads pour appeler et arrêter l'exécution de la fonctionnalité correspondante par rapport à l'opération d'utilisateur.

3.3.3 Structure de données

En informatique, une structure de données est une structure logique destinée à contenir des données, afin de leur donner une organisation permettant de simplifier leur traitement [9]. Une structure de données implémente concrètement un type abstrait qui est une spécification mathématique d'un ensemble de données et de l'ensemble des opérations qui peuvent y être effectuées.

Une bonne conception de la structure des données permet de simplifier la structure du logiciel et donc une exécution du logiciel plus efficace. Elle occupe donc une position importante dans le cycle du développement du logiciel. Après l'analyse des données opératoires du module MonitorKit, la « liste doublement chaînée » a été employée pour la structure de données du logiciel [10].

Le MonitorKit collectionne continûment les données (la position, l'orientation, etc.) de chaque point à intervalles régulières. Toutes ces données construisent une série de points qui n'ont pas d'autre relation logique que la liaison en temps linéaire. Par ailleurs, à cause de la précarité d'acquisition (on ne sait pas combien de points seront relevés), les « tableaux statiques » ne sont pas convenables pour ce type de système d'acquisition. Une structure dynamique dite « la liste doublement chaînée » a été donc utilisée dans le module MonitorKit pour stocker les données de chaque point de la trajectoire.

Une liste est un ensemble d'objets de même type constituant les éléments de la liste [11,12]. Les éléments sont chaînés entre eux et on peut facilement ajouter ou extraire un ou

plusieurs éléments. Une liste simplement chaînée est une structure de données telle que chaque élément contient (Figure 3.3) :

- ü des informations caractéristiques de l'application (les coordonnées d'un point de la trajectoire par exemple)
- ü un pointeur vers un autre élément ou une marque de fin s'il n'y a pas d'élément successeur.

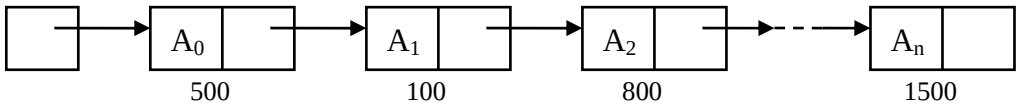


Figure 3.3 Structure d'une liste simplement chaînée

Chaque élément est alloué dynamiquement à la demande, les différents éléments sont donc dispersés en mémoire centrale. L'espace est réservé au fur et à mesure des créations d'éléments. La seule limite est la taille de la mémoire centrale de l'ordinateur ou l'espace mémoire alloué au processus. Sur l'exemple de la Figure 3.3, la structure de A_0 se trouve à l'adresse 500, celle de A_1 à l'adresse 100. Ainsi, le suivant de A_0 se trouve à l'adresse 100 (A_1) qui est indiquée par le pointeur sur l'élément suivant de A_0.

Il est évidemment observé qu'il y a un seul pointeur qui indique l'élément suivant. Ceci signifie qu'une liste simple ne peut être parcourue que dans une direction, et ne peut donc pas être adressée aléatoirement. Pour éviter cet inconvénient la liste simplement chaînée a été remplacée par une liste doublement chaînée dans le module MonitorKit afin que les informations de chaque point puissent être parcourus dans les deux sens.

La liste doublement chaînée est une liste telle que chaque élément pointe sur l'élément suivant et sur l'élément précédent. La Figure 3.4 présente une structure de liste doublement chaînée.

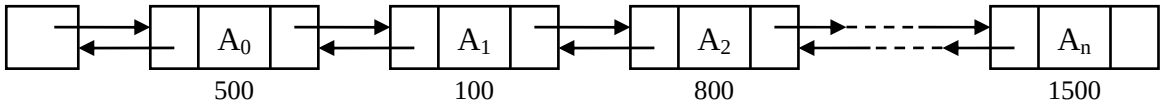


Figure 3.4 Structure d'une liste doublement chaînée

Dans la structure de données du module MonitorKit l'élément de la liste concerne non seulement les pointeurs sur l'élément suivant et l'élément précédent, mais aussi les informations de chaque point (les coordonnées du point par exemple). La déclaration pour la liste doublement chaînée dynamique à la demande est :

```
struct DATA_NODE{
    int no;                          // nombre d'élément dans la liste
    RAPVAR_POS_TYPE pos;             // les coordonnées du CDO
    RAPVAR_ORIENT_TYPE ori;          // l'orientation du CDO
    LONGLONG timecount;              // compteur du temps intervalle (64bit)
    float timecount_float;           // compteur du temps intervalle (32bit)
    float delta_t;                   // intervalle entre deux points
    float speed;                     // vitesse du CDO calculée
    float speed2;                    // vitesse du CDO par acquisition
    DATA_NODE* prior;                // pointeur sur l'élément précédent
    DATA_NODE* next;                 // pointeur sur l'élément suivant
};
```

3.4 Fonctionnalités du T.S.T.

Comme explicité précédemment, un logiciel CAO robotique permet la création de trajectoires robot. Cependant, il n'existe pas de logiciel CAO robotique dédié à la projection thermique car les caractéristiques de la trajectoire robot pour cette application sont très spécifiques. Une trousse à outil « Thermal Spray Toolkit » orientée projection thermique a

donc été développé au LERMPS dans le cadre de cette étude.

Cette trousse à outil comporte 3 modules : PathKit, ProfileKit et MonitorKit. La Figure 3.5 présente la structure du logiciel T.S.T.

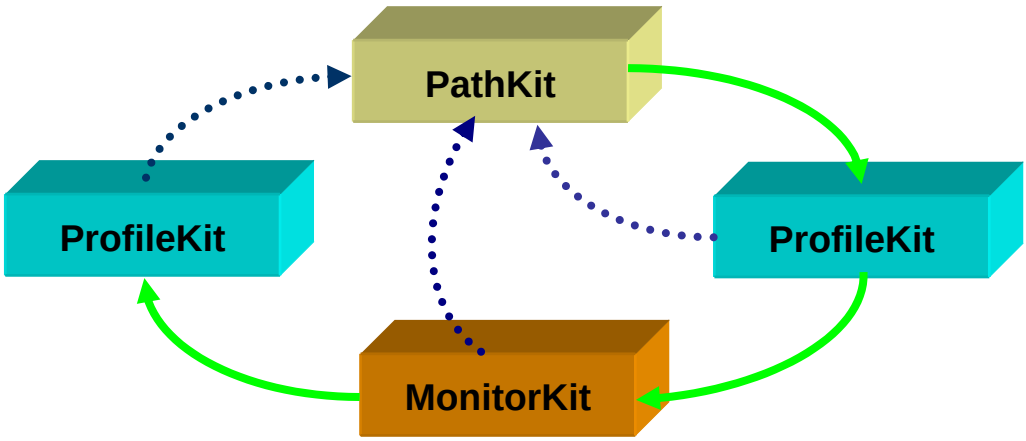


Figure 3.5 Composants du T.S.T.

- ü PathKit – un module intégré dans RobotStudio™ qui permet la génération des trajectoires de robots dédiés à la projection thermique
- ü ProfileKit – un module intégré dans RobotStudio™ qui fournit la stratégie de choix des paramètres opératoires de projection et la caractérisation de surface du dépôt
- ü MonitorKit – un module extérieur qui est capable de communiquer avec le robot et d'obtenir en temps réel toutes les données robot durant le procédé de projection

Pour pouvoir utiliser cette trousse à outil, on doit la charger dans RobotStudio™ (Figure 3.6). Après le chargement, le panneau du T.S.T. apparait dans RobotStudio™ (Figure 3.7). L'interface du module extérieur MonitorKit est présentée dans la Figure 3.8.

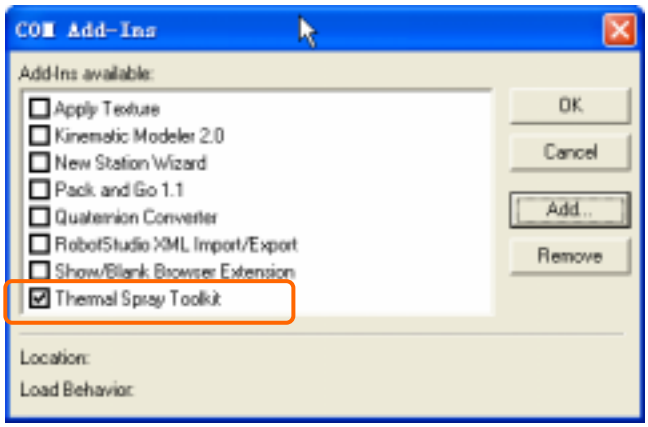
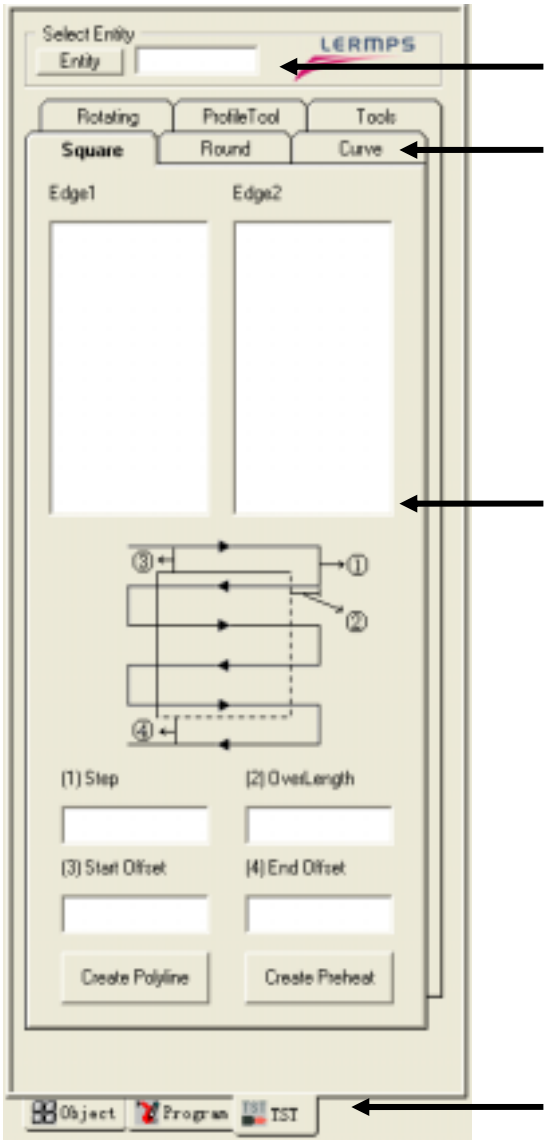


Figure 3.6 Chargement du T.S.T.



Sélection de l'entité à procéder, ce champ doit contenir un nom d'entité dans RobotStudio™

Sélection d'une fonctionnalité du T.S.T.

Fenêtre d'opération du T.S.T.

Sélection de panneau actuel

Figure 3.7 Représentation de panneau intégré dans RobotStudio™

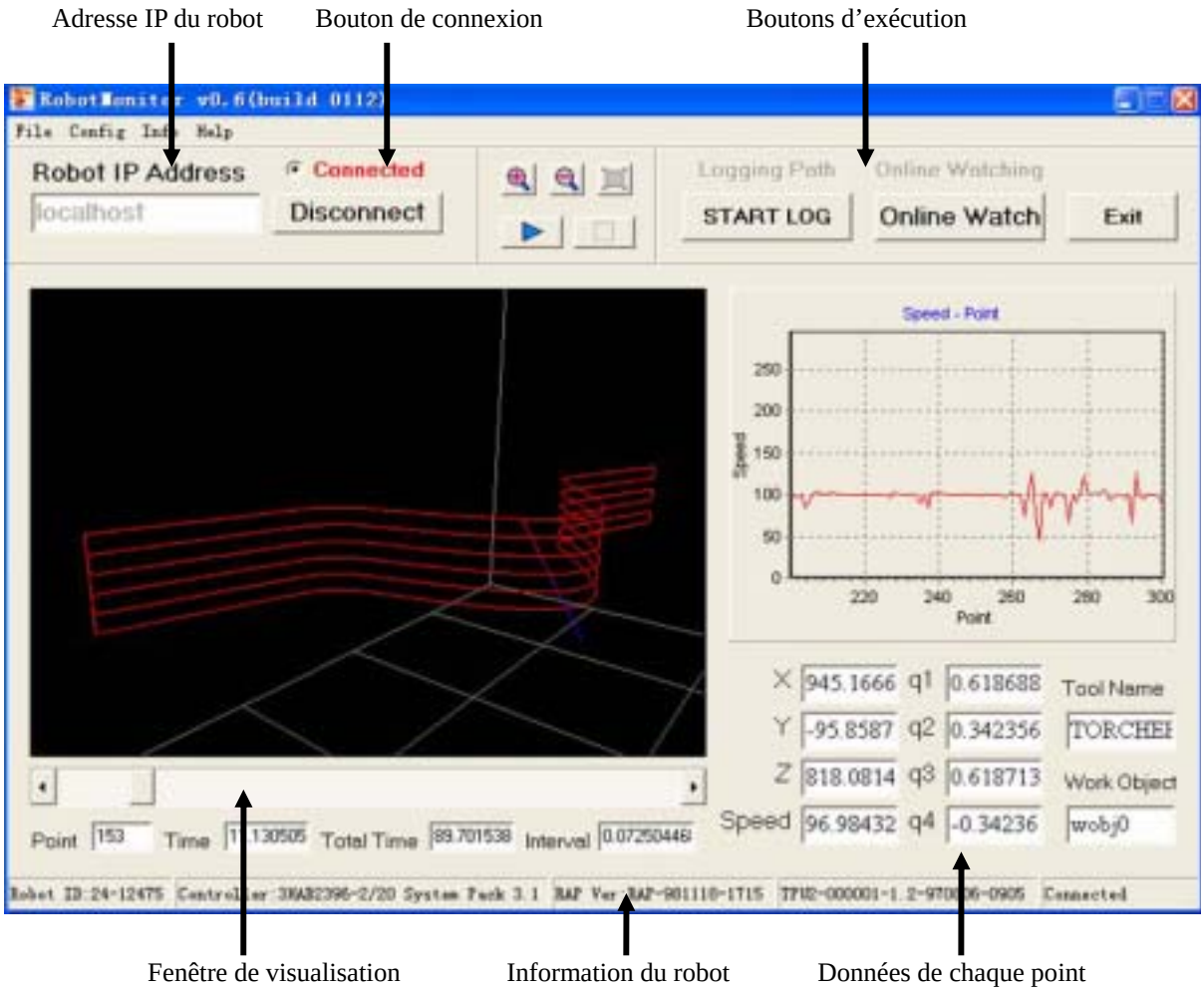


Figure 3.8 Représentation de l'interface du MonitorKit

Références

[1] ABB, RobotStudio™ Guide de l'utilisateur, 3HAC 7793-1 for BaseWare OS 4.0. Sweden; 2001.

[2] S. Deng, H. Liao, C. Zeng, C. Coddet, Robotic trajectory autogeneration in thermal spraying. Thermal Spray 2005: Explore its potential!, (Ed.) E. Lugscheider, ASM. International, Materials Park, Ohio, USA, 2005.

[3] S. Deng, H. Liao, C. Zeng, C. Coddet, New functions of Thermal Spray Toolkit, a software developed for off-line and rapid robot programming. Thermal Spray 2006: Building on 100 years of success, ASM International, Materials Park, Ohio, USA, 2006.

[4] ABB, Product Specification IRB 4400 3HAC 9117-1 / Rev 2. ABB Automation Technology Products AB Robotics, SE-721 68 Västerås, Sweden, 2001.

[5] Microsoft, Comment utiliser QueryPerformanceCounter pour chronométrer le code. http://support.microsoft.com/default.aspx?scid=kb%3Bfr%3B172338

[6] developpez.com, Lire la fréquence du processeur, http://haypo.developpez.com/article/frequence_cpu/

[7] codeproject.com, High-Performance Timer in C#, http://www.codeproject.com/csharp/highperformancetimercshar.asp

[8] T. Archer, Formation à Microsoft C#. Microsoft Press, 2001.

[9] http://fr.wikipedia.org/wiki/Structure_de_donn%C3%A9es, Structure de données, Wikipédia

[10] S. Deng, H. Liao, R. Bonnet, C. Zeng, C. Coddet, Real time monitoring of robot

trajectory in thermal spraying. Thermal Spray 2004: Advances in technology and application, ASM International, Materials Park, Ohio, USA, 2004

[11] M. Divay, Algorithmes et structures de données génériques. 2e édition, Dunod, Paris, 2004.

[12] Y. Noyelle, Belle programmation et langage C. Ellispses edition marketing S.A., Paris, 2001.

Chapitre 4

Thermal Spray Toolkit (T.S.T.)

Ce chapitre présente les modules fonctionnels de l'extension logicielle « Thermal Spray Toolkit » (T.S.T.), incluant « PathKit », « ProfileKit » et « MonitorKit ». Leurs principes et développements ainsi que les fonctionnalités seront présentés. Les résultats obtenus par les essais expérimentaux sont présentés et comparés afin de vérifier la validation du T.S.T. dans les applications réelles de projection thermique.

4.1 Module PathKit

4.1.1 Introduction

Pour une application de projection thermique la conception de la trajectoire robot représente une étape très importante. La reconnaissance récente par le monde de l'industrie de la projection thermique est la conséquence principale de la quantité relativement faible d'articles traitant des problèmes de la génération de trajectoire hors-ligne et de la détermination de l'épaisseur du revêtement en touts points d'une pièce revêtue [1]. En effet, la très grande majorité des articles relatifs à la projection traite de sujets tels que la mise au point ou l'étude de la projection de nouveaux matériaux sur des formes simples (plaque, cylindre, disque). Pour ces études, la génération d'une trajectoire est relativement triviale et peut s'effectuer aisément par apprentissage. Quelques auteurs se sont cependant déjà intéressés à la programmation hors-ligne et à la simulation de l'épaisseur déposée :

- Chettibi et al. [2], Ismael et al. [3] et Suk-Hwan et al. [4] proposent des méthodes et des théories de programmation des trajectoires robot sur des formes simples (plaque, cylindre).
- Antonio et al. [5], Ramabhadram et al. [6] et Hyotyniemi et al. [7] optimisent les trajectoires robot pour l'application de projection automatique. Ses descriptions correspondantes sont utilisées pour créer des trajectoires pour le procédé de peinture.
- P. Herling et al. [8], W. Persoons et al. [9] et Y. Itoh et al. [10] développent des méthodes et des systèmes pour générer la trajectoire robot sur des formes complexes à

partir d'un fichier numérique CAO.

L'ensemble de ces études est basé sur l'application des méthodes développées pour la peinture effectuée sur une surface plane ou courbe. Devant la faiblesse du niveau de modélisation/simulation des trajectoires et des épaisseurs de revêtement au niveau de la projection thermique, il est certainement intéressant de se tourner vers le monde de la peinture industrielle par pulvérisation. Ce métier revendique une expérience beaucoup plus grande au niveau des trajectoires et de la simulation de l'épaisseur des dépôts [9] et [4]. En effet, une expérience existe pour programmer les trajectoires du robot hors-ligne en utilisant une description CAO de la surface à revêtir [11]. La recherche de la correspondance projection/peinture constitue donc un axe rapide pour le développement de modules informatiques dédiés à la génération de trajectoires robot hors-ligne et à la simulation des épaisseurs déposées. Cette stratégie a été utilisée par Nylén et al. [12] et semble être en mesure de répondre très rapidement aux nouvelles exigences en terme de précision de trajectoire et d'homogénéité d'épaisseur.

Néanmoins, il existe des différences notables entre le procédé de peinture et le procédé de projection thermique. La projection thermique est un procédé de sédimentation très rapide pour lequel le revêtement se construite à l'état quasi solide. À l'inverse, un revêtement de peinture peut s'homogénéiser automatiquement par le biais de la tension superficielle du liquide déposé. Il est donc nécessaire de contrôler plus précisément la trajectoire (vitesse, orientation, etc.) pour la projection thermique.

Le module développé PathKit est un outil de génération de trajectoires dédié à la projection thermique qui permet de créer des courbes et des trajectoires sur les pièces de forme complexe et de modifier les paramètres de la trajectoire commodément et précisément.

4.1.2 Définitions géométriques

La programmation hors-ligne nécessite un certain nombre d'opérations telles que :

- ü la définition géométrique de l'outil, de la pièce à revêtir, ainsi que de l'ensemble des pièces situées dans l'environnement proche du robot et de la pièce,
- ü le positionnement et surtout l'étalonnage de la position de la pièce à revêtir. Toute pièce dont le positionnement est important en terme de sécurité (absence de collision) doit également être étalonnée,
- ü la création d'une trajectoire respectant des critères tels que la distance de projection, l'orientation des points de passage, les vitesses aux points de passage, l'accessibilité de chacun des points par l'outil, l'évitement de tout point singulier, l'absence de toute collision avec la pièce à revêtir et tout élément de l'environnement,
- ü le transfert de la trajectoire optimisée vers le robot concerné pour la tâche à effectuer.

La définition géométrique est une opération qui constitue la base de la programmation hors-ligne. L'ensemble des éléments tells que l'outil, la pièce à revêtir, les objets environnants doivent être modélisés en 3D. Pour ce faire, RobotStudio possède un modeleur volumique qui permet de développer des modèles 3D simples (génération de solides par des primitives simples, génération de courbes 2D ou 3D, création de solides 3D par extrusion d'une section (courbe 2D) suivant une courbe 2D ou 3D, opérations booléennes classiques sur les solides).

Cette panoplie d'outils est très utile car elle permet de générer directement dans RobotStudio de nombreuses formes de pièces volumiques. Toutefois, la puissance de ce modeleur est limitée par rapport aux possibilités affectés par la plupart des systèmes de CAO classiques. En outre, les pièces à intégrer dans la simulation peuvent avoir été modélisées préalablement à partir d'un système de CAO. La modélisation des pièces intervenant dans la programmation hors-ligne peut donc être générée soit dans RobotStudio, soit importée dans RobotStudio à partir de logiciels de CAO conventionnels.

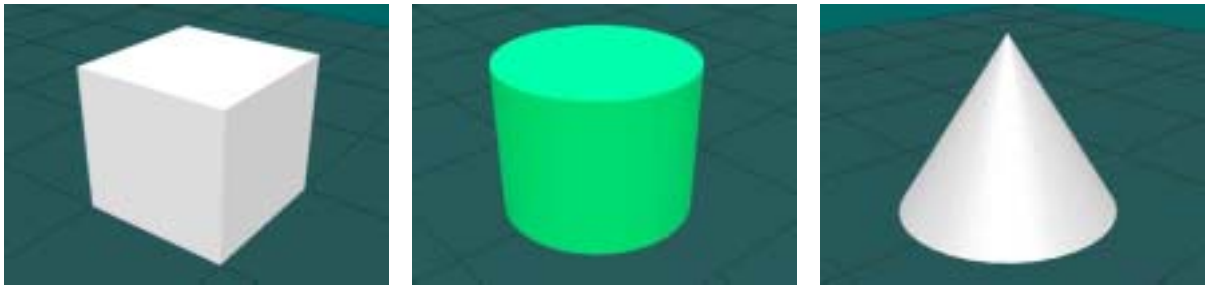


Figure 4.1 Exemples de pièces de géométrie simple mode liées dans RobotStudio

Au-delà des possibilités de modélisation géométrique, l'étape primordiale de la programmation hors-ligne consiste à réaliser le positionnement et surtout l'étalonnage des modèles géométriques.

4.1.3 Positionnement et étalonnage

Afin de mettre en correspondance l'enceinte réelle de projection avec l'enceinte virtuelle, une phase d'étalonnage est nécessaire.

Les méthodes de programmation hors-ligne nécessitent que soient maîtrisées les incertitudes dues aux imperfections de structure des mécanismes et des objets manipulés par rapport aux modèles implantés en simulation. En admettant que ces modèles soient bien caractérisés, le problème consiste en l'identification des valeurs géométriques (position et orientation) qui les lient. C'est ce que l'on appelle l'étalonnage.

L'étalonnage géométrique d'une cellule robotisée doit permettre de définir les transformations qui lient :

- la base du robot au repère atelier (peu usitée en projection thermique),
- le repère terminal (flange du robot) à la base du robot,
- le repère outil au repère terminal,
- le ou les repères pièce au repère atelier ou robot.

Il s'agit ainsi de réduire l'écart entre la position programmée et la position réellement

atteinte, donc d'améliorer la précision absolue du robot. Cette notion, sans intérêt pour la programmation par apprentissage, est absolument essentielle en programmation hors-ligne.

Sur une cellule robot non étalonnée, du type de celles utilisées pour le soudage par points par exemple, la précision absolue peut descendre à plusieurs centimètres même si la précision relative ou répétitivité est de l'ordre du dixième de millimètre pour les robots industriels actuels [13]. La Figure 4.2 présente une cellule de projection thermique robotisée virtuelle.

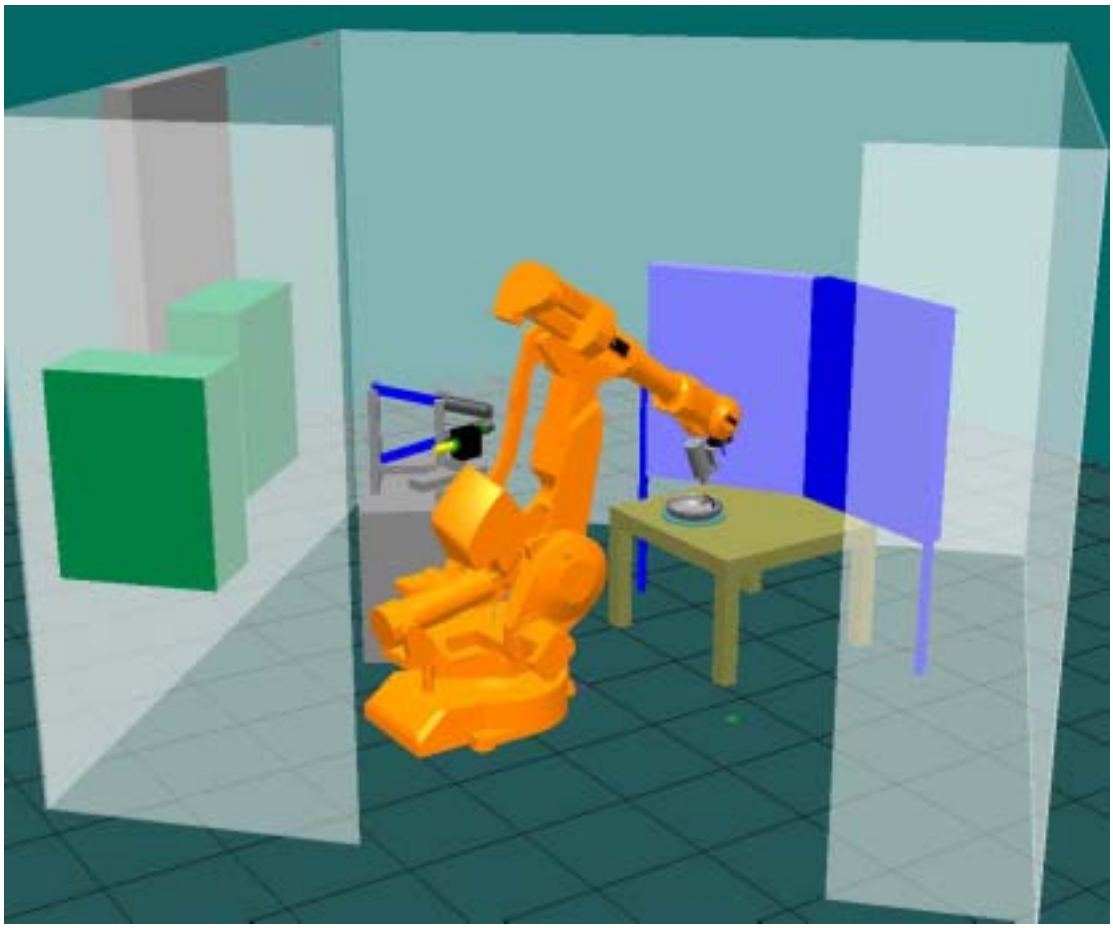
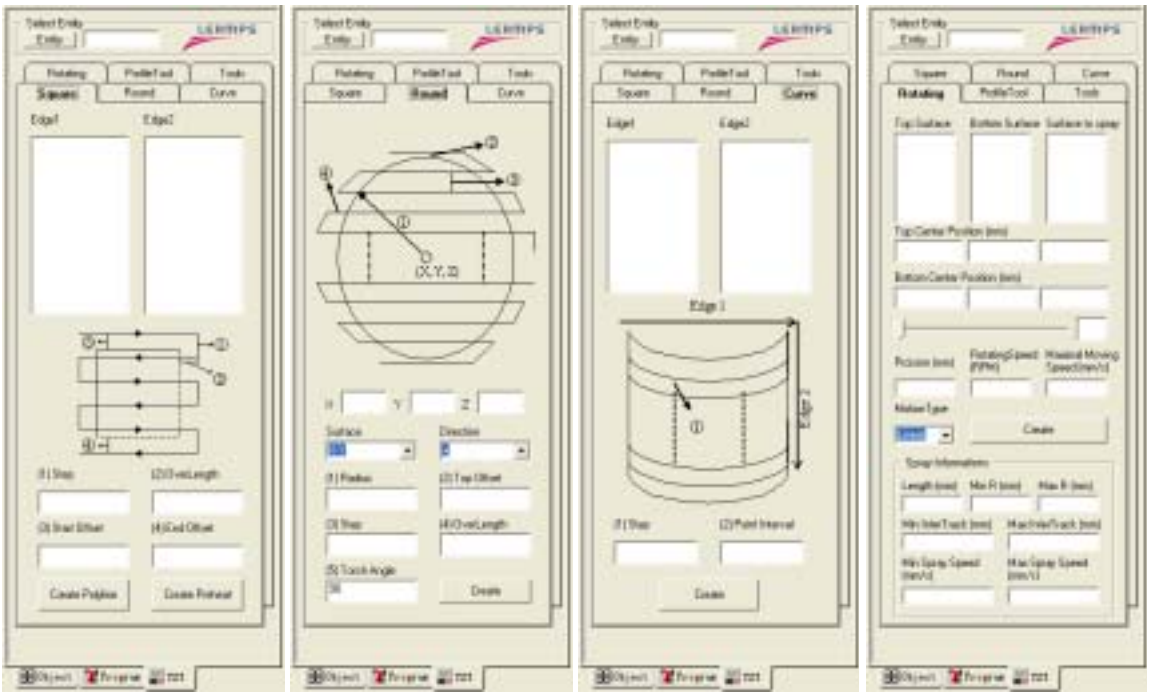


Figure 4.2 Cellule robotisée virtuelle

4.1.4 Création de la trajectoire

PathKit est destiné à fournir plusieurs projets de génération de la trajectoire sur des pièces de formes différentes ainsi que les outils de création de courbes sur les pièces complexes. Les

paragraphes suivants présentent les fonctionnalités de cette trousse à outil.



(a) Pièces carrées (b) Pièces circulaires (c) Pièces incurvées (d) Pièces en rotation

Figure 4.3 Interfaces du PathKit pour différentes pièces

4.1.4.1 Trajectoire pour une pièce carrée

La projection sur une surface carrée est couramment utilisée. La trajectoire est composée par plusieurs points réguliers comme indiquée sur la Figure 4.4.

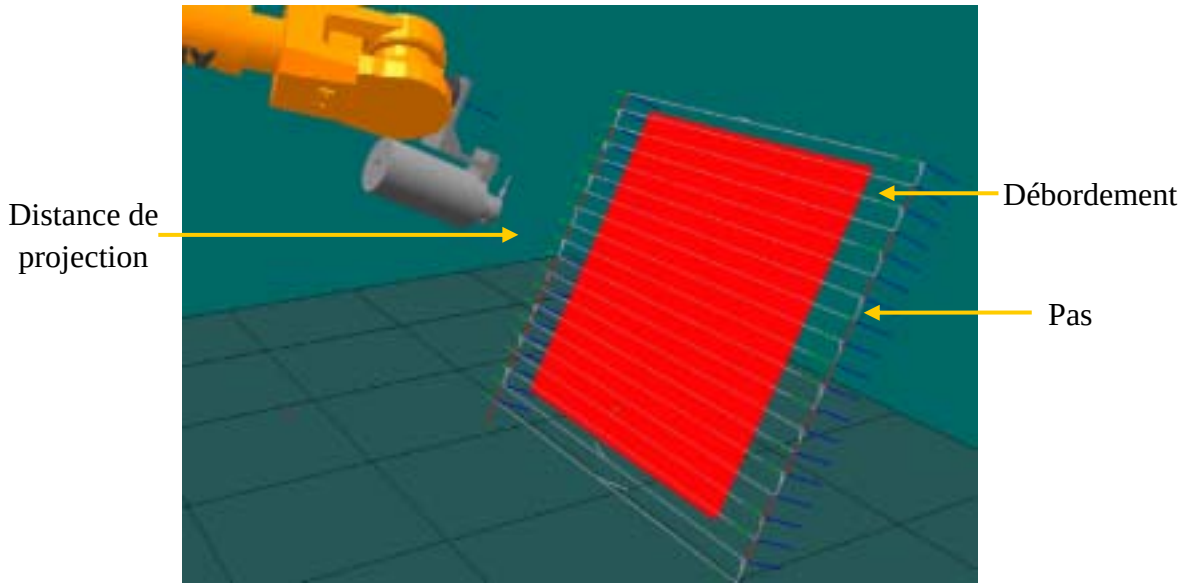


Figure 4.4 Projection sur une pièce carrée

Il n'est bien entendu ni pratique ni efficace de chercher tous les points manuellement. À l'aide de PathKit, les choses deviennent faciles et efficaces. Une seule définition des paramètres de projection (le débordement, le pas, l'offset, etc.) est nécessaire pour la génération automatique de la trajectoire. PathKit permet aussi la création d'une trajectoire pour le procédé de préchauffage (Figure 4.5). Dès que les paramètres sont entrés, la trajectoire peut être générée en quelques secondes.

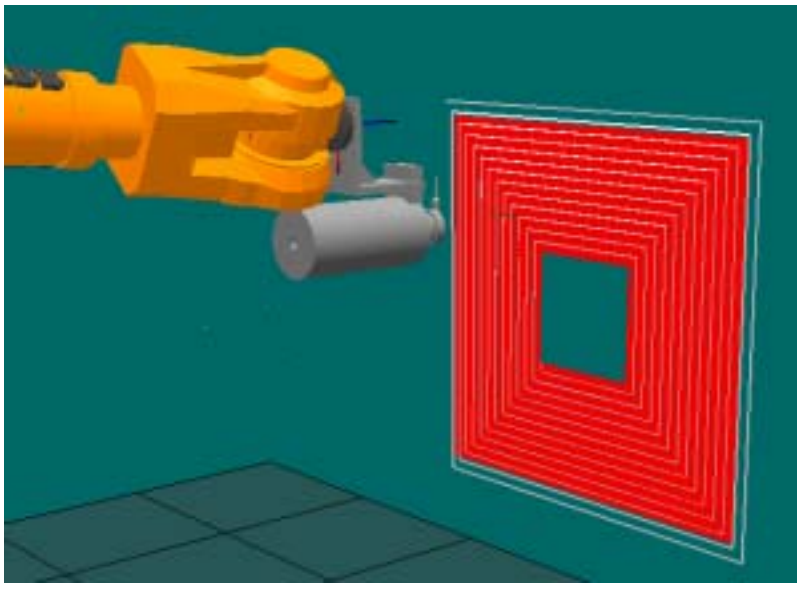


Figure 4.5 Exemple de trajectoire pour le préchauffage

4.1.4.2 Trajectoire pour une pièce circulaire

La trajectoire adaptée à la pièce carrée (Figure 4.4) peut être utilisée sur une pièce circulaire. Mais il existe alors des pertes de rendement sur les quatre coins de la trajectoire carrée (Figure 4.6).

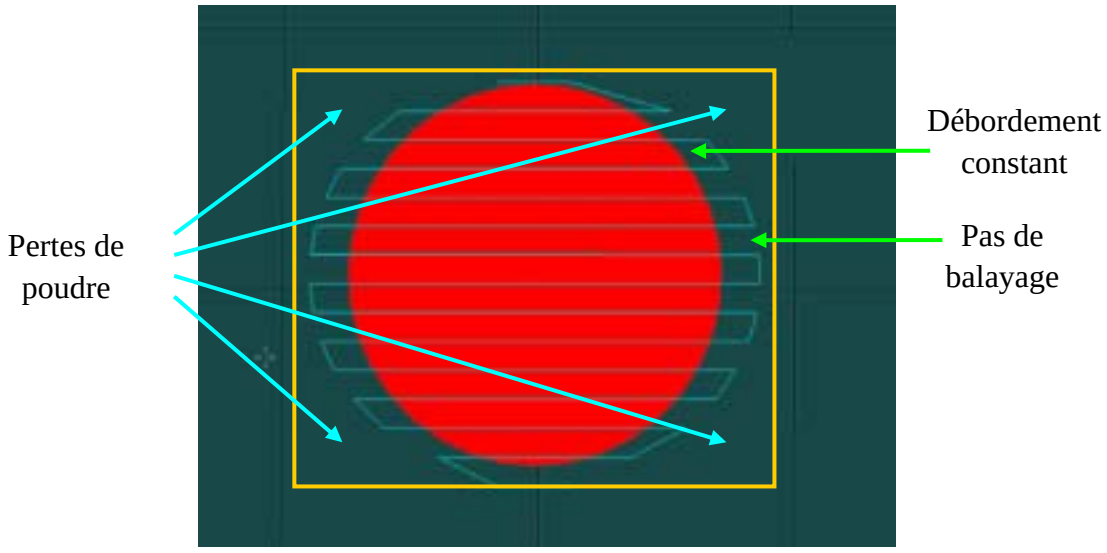


Figure 4.6 Exemple de trajectoire pour la pièce circulaire

La meilleure solution consiste à garder la même distance de débordement pour chaque point du bord du cercle comme montrée Figure 4.6. Cependant, il est très coûteux en temps de déterminer tous les points manuellement. Dans PathKit, la génération de ce type de trajectoire devient pratique et précise. Elle nécessite seulement quelque paramètres à remplir pour une création automatique sous RobotStudio™.

4.1.4.3 Trajectoire pour une pièce incurvée

En principe, il est très difficile de faire de la projection thermique sur les pièces incurvées, parce que la trajectoire sur une surface incurvée n'est plus une ligne, mais une courbe. Par

ailleurs, l'orientation de la torche sur chaque point est modifiée (Figure 4.7a). Il est donc presque impossible de créer une trajectoire manuellement sur ce type de pièce.

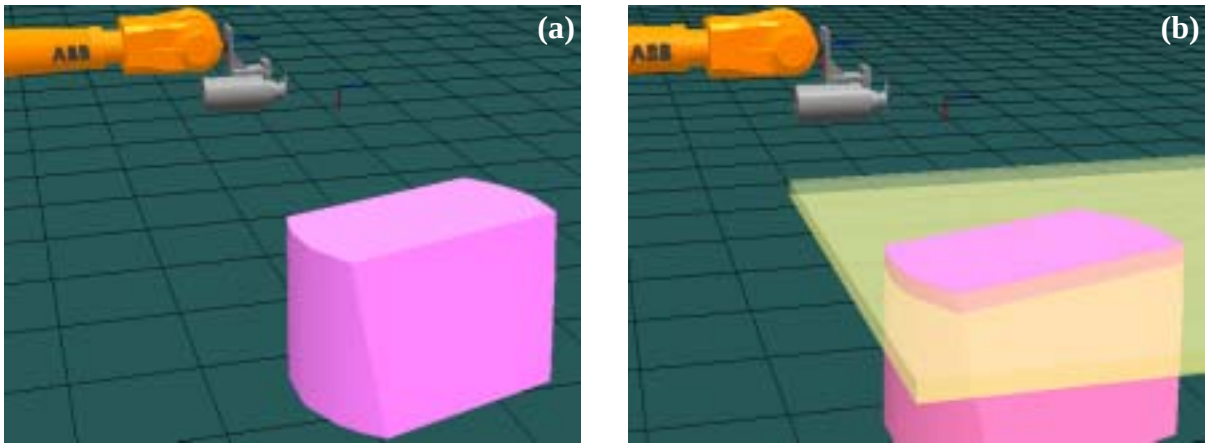


Figure 4.7 Projection sur une pièce incurvée

(a) Pièce incurvée (b) Coupage sur la pièce

PathKit coupe la pièce par un parallélépipède d'une certaine épaisseur (Figure 4.7b). L'épaisseur du parallélépipède sera le pas entre deux passes et les intersections entre la pièce et le parallélépipède seront les trajectoires du robot. Dans PathKit, seules la définition de la surface à revêtir et la configuration des paramètres de projection sont nécessaires. La trajectoire sera créée en quelques secondes (Figure 4.8).

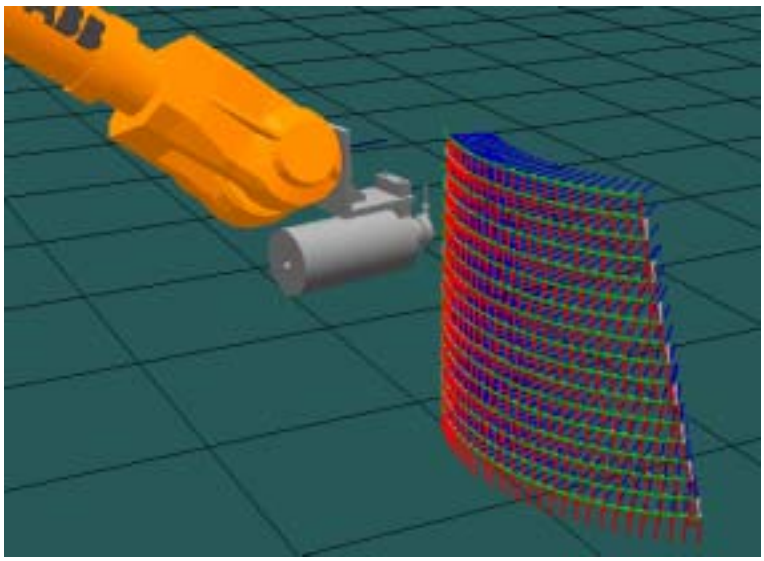


Figure 4.8 Trajectoire pour une pièce incurvée

4.1.4.4 Trajectoire pour une pièce rotationnelle

La projection sur des pièces en rotation est couramment utilisée, surtout pour des grosses pièces axisymétriques. La Figure 4.9 présente une pièce trapéziforme fixée sur un plateau rotationnel. Pour obtenir un dépôt d'épaisseur constante, la vitesse de la torche doit être variée par rapport au rayon R ; il en résulte que le pas de balayage doit diminuer quand la vitesse de projection augmente.

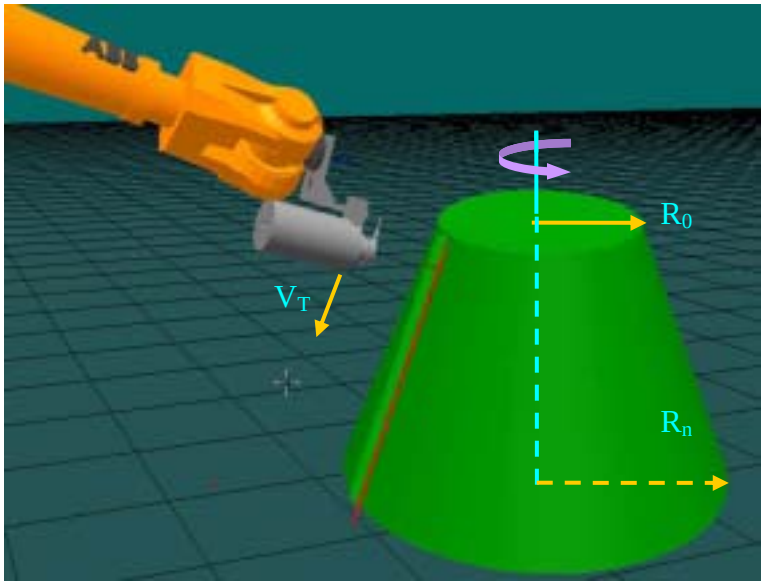


Figure 4.9 Projection sur la pièce en rotation

PathKit permet de changer la vitesse de chaque point sur la trajectoire selon le profil du substrat. Cette fonction a été employée pour le revêtement d'une poêle. Une trajectoire optimisée par PathKit garantit une très bonne qualité du dépôt sur la surface interne de la poêle. La Figure 4.10 montre les configurations physiques pour cette expérience.

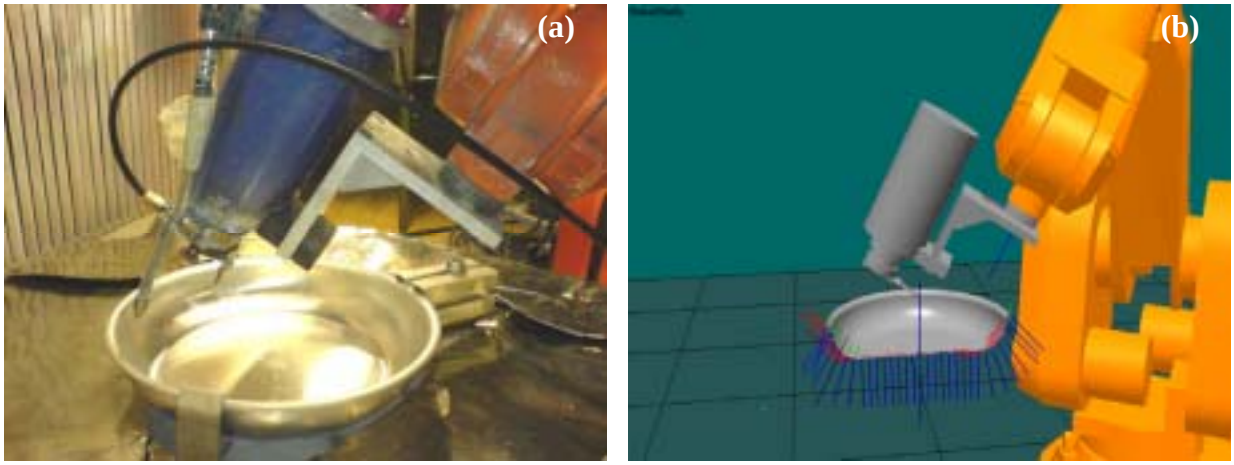


Figure 4.10 Projection sur la poêle rotationnel

(a) Cellule réelle sur site (b) Cellule virtuelle sous RobotStudio™

La Figure 4.11 présente la variation de la vitesse du CDO en fonction du temps par rapport au profil de la poêle.

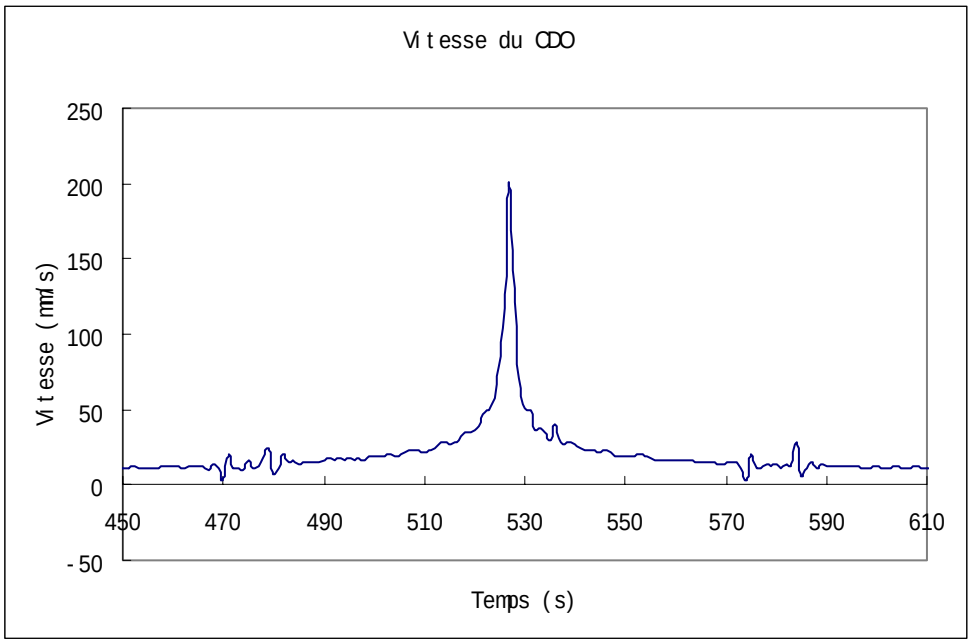


Figure 4.11 Courbe de vitesse du CDO sur le poêle

En raison de la confidentialité de projet, les détails du revêtement de cette expérience ne peuvent pas être présentés dans ce mémoire. Les résultats ont montré que le fonctionnement du PathKit est parfaitement adapté et que les gains en terme de productivité sont évidents.

4.1.4.5 Trajectoire pour une pièce complexe

Dans plusieurs cas de projection thermique industrielle, les pièces ne sont pas de formes régulières. Les solutions de génération préconçue de trajectoire ne sont pas convenables dans ces cas. PathKit fournit alors un outil universel de création de courbes à la surface d'une pièce permettant ainsi de générer une trajectoire qui respecte le profil du substrat. Figure 4.12 présente l'interface de cette fonction et la création des courbes parallèles sur une pièce incurvée.

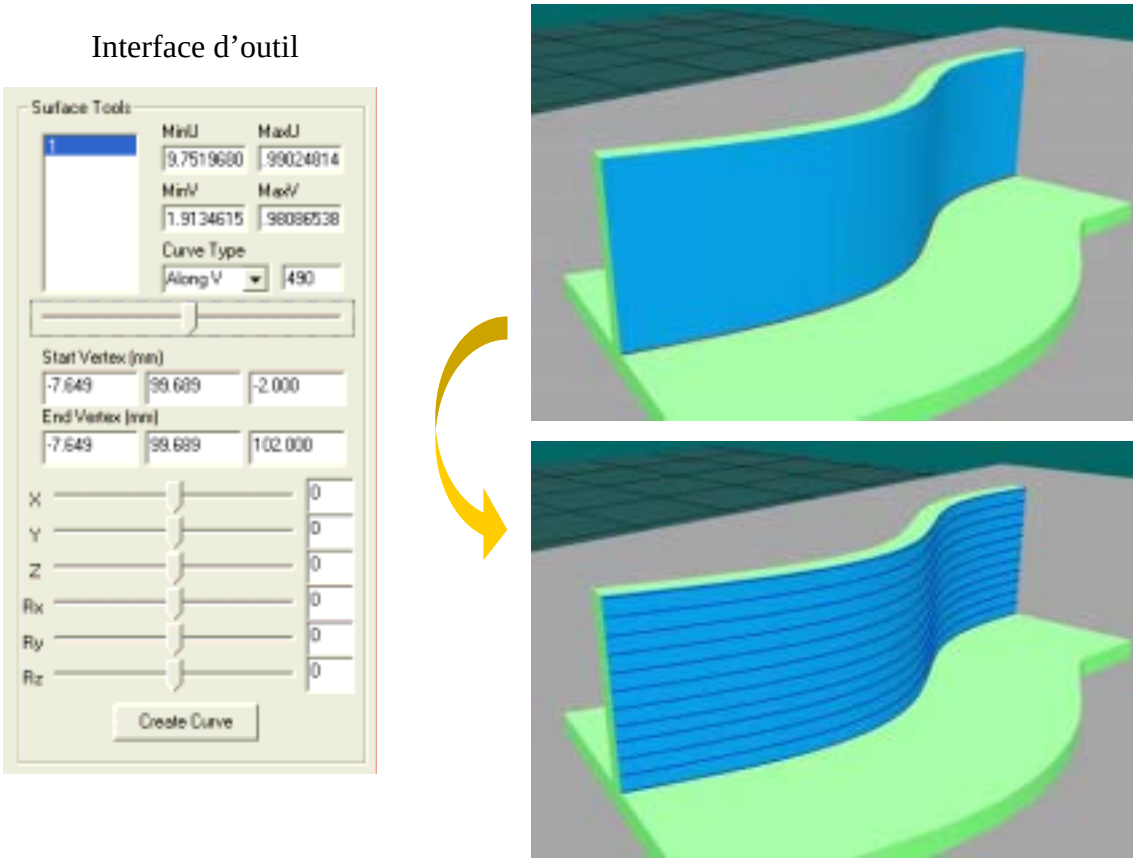


Figure 4.12 Création des courbes parallèles sur une pièce incurvée

PathKit fournit aussi la fonctionnalité de joindre les courbes parallèles dans une seule courbe afin de créer une trajectoire complète sur une pièce.

4.1.5 Post-processeur

Comme explicité précédemment, PathKit permet la création de trajectoires d'outil sur toutes sortes de pièces. Chaque point de passage est défini par une position spatiale et une orientation du CDO dans un repère qui peut être la base du robot ou de la pièce dont le positionnement est défini par rapport au robot. La définition complète du mouvement doit inclure un certain nombre de paramètres tels que la précision au point de passage, la vitesse de l'outil à ce point, de même que le type de mouvement (linéaire ou articulaire).

Cependant, dans certain cas de projection, il faut accepter de modifier les paramètres les moins cruciaux (l'angle de projection par exemple) afin que la trajectoire robot dans le

processus de projection soit exécutable, ou afin d'évider une collision mécanique entre le robot et le montage torche.

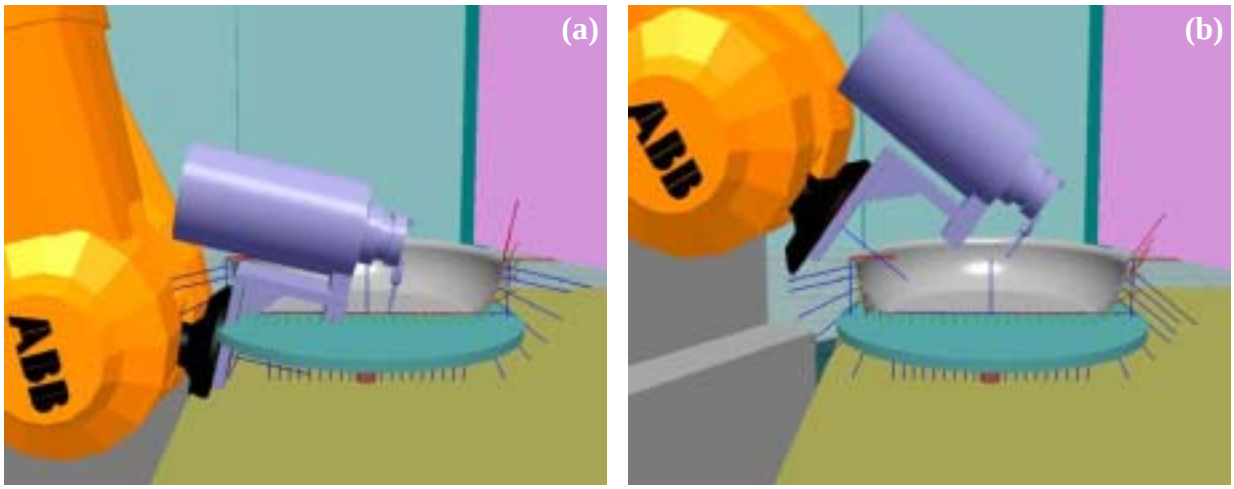


Figure 4.13 Optimisation des positions et orientation relative torche / pièce

(a) avant optimisation (b) après optimisation

La Figure 4.13 montre une situation de projection sur une poêle. Dans la Figure 4.13a, tout les points respectent la direction normale (90 dégrées) de la torche par rapport à la surface de la poêle. Il est évidemment observé que le montage de la torche touche la poêle pendant le processus. Il faut donc sacrifier l'angle de projection pour assurer la faisabilité de la projection. La Figure 4.13b montre une optimisation (20 dégrées de déviation) de l'orientation de la torche à l'aide du post-processeur du PathKit.

Ce post-processeur est entièrement intégré dans PathKit. Deux menus ont été développés afin de permettre de modifier l'orientation de la position du point, et de déplacer le point par rapport au repère local ou global. Ces deux types d'optimisation du CDO sont présentés dans la Figure 4.14.

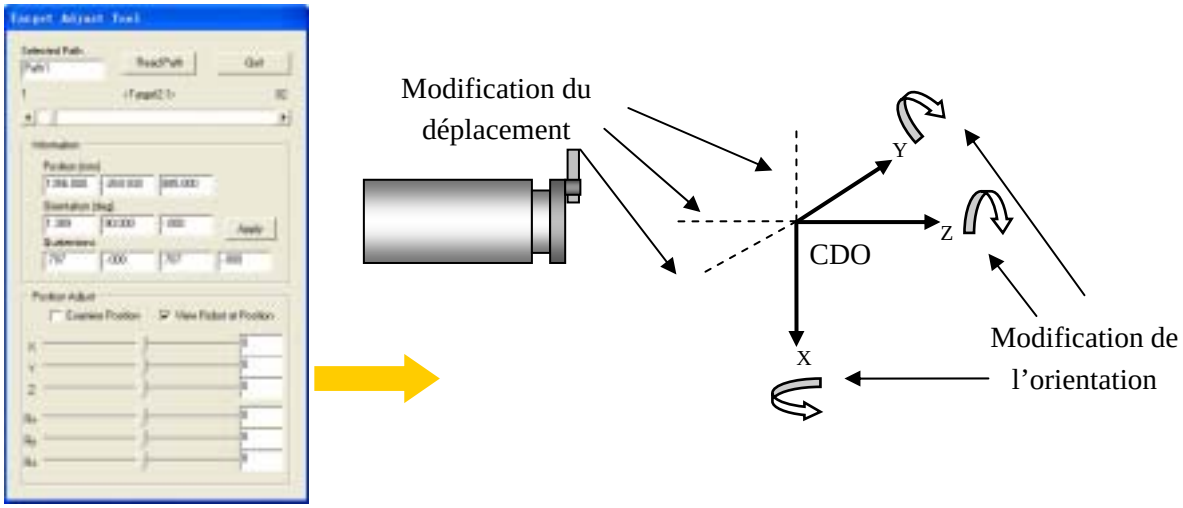


Figure 4.14 Optimisation du CDO

4.2 Module ProfileKit

4.2.1 Introduction

La simulation de l'épaisseur déposée en tous points de la pièce revêt une importance particulière. En effet, la prédiction de l'épaisseur permet de valider ou non la trajectoire (position, orientation, vitesse de déplacement) définie par la programmation hors-ligne. Si une bonne correspondance est établie entre les résultats de la simulation et l'épaisseur réelle mesurée sur le substrat, alors la simulation permettra de générer des programmes robot en garantissant l'épaisseur déposée. Cet atout est un gage d'économie au niveau de :

- ü la réduction de la phase de mise au point du programme robot,
- ü la réduction du temps de projection,
- ü la réduction de matière utilisée,
- ü la réduction des post-traitements.

Tous ces gains représentent une économie substantielle (coût et temps) dans la mise au point d'un revêtement sur une pièce et permettent d'envisager le procédé de projection thermique pour des séries de pièces plus petites.

L'état de la surface du substrat joue également un rôle important dans l'ancrage mécanique des dépôts, surtout dans le cas des céramiques [14]. Plusieurs études ont mis en évidence la dépendance entre l'état des particules à leur impact (vitesse et température notamment) et la rugosité résultante de la surface des dépôts.

ProfileKit fournit une fonctionnalité pour simuler l'épaisseur finale du dépôt à partir d'un profil de cordon. Il fournit ainsi l'évaluation d'état de l'interface substrat/dépôt ou de la surface du dépôt par rapport aux caractéristiques de surface (rugosité, dimension fractale, etc.).

4.2.2 Principe de la simulation de l'épaisseur du dépôt

Cette simulation est basée sur la caractérisation du cordon (une passe de projection) afin d'aboutir une modélisation de l'épaisseur finale. De nombreux auteurs se sont intéressés à l'étude des caractéristiques et de la morphologie du dépôt :

- ü Fasching et al. [15] et [16], Figueroa et al. [17], Cirolini et al. [18] proposent des modèles de simulation d'épaisseur sur des formes simples (plaque, cylindre),
- ü Vivekanandhan et al. [19] digitalisent la surface devant être revêtue et créent une modélisation de la surface de type B-Spline. Cette description est utilisée pour créer une trajectoire robot qui maintient une distance constante et une orientation constante. Cependant, la simulation de l'épaisseur n'est pas prise en compte.
- ü Goedjen et al. [20] ont utilisé une approche par différences finies pour prédire les épaisseurs de revêtement sur des géométries complexes. L'approche débute par une description empirique du taux de dépôt et calcule l'épaisseur en chaque point par incrémentation du pas spatial. Cependant, le modèle est limité à 2 dimensions et suppose que la distribution de matière est symétrique et de type gaussien.
- ü Nylén et al. [21] ont développé une méthode de simulation de l'épaisseur déposée en utilisant un modèle expérimental de dépôt. Ce modèle est intégré à un logiciel de simulation robotique pour prédire l'épaisseur du dépôt en fonction de la trajectoire du robot.

A partir de la stratégie du PathKit, ProfileKit utilise une approche dérivée du procédé de peinture pour simuler l'épaisseur du dépôt en projection thermique. ProfileKit fournit la fonctionnalité de simuler la superposition de différentes passes parallèles sur une surface plane. La Figure 4.15 présente deux exemples de superposition du dépôt.

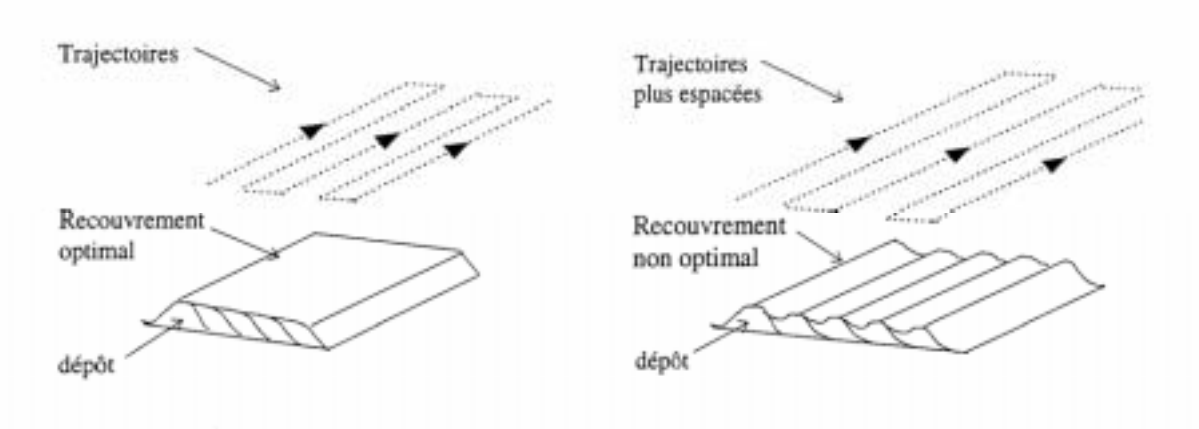


Figure 4.15 Superposition des cordons pour la formation du dépôt

4.2.3 Obtention de la fonction de dépôt

Pour simuler l'épaisseur du dépôt, une expérimentation sur une plaque plane est nécessaire. Un balayage à vitesse constante est réalisé en vue d'obtenir un profil de cordon pour un matériau et des paramètres de projection donnés. Ce profil peut être mesuré de différentes manières : palpeur mécanique, profilomètre optique, photographie optique ou échographie acoustique (Figure 4.16).

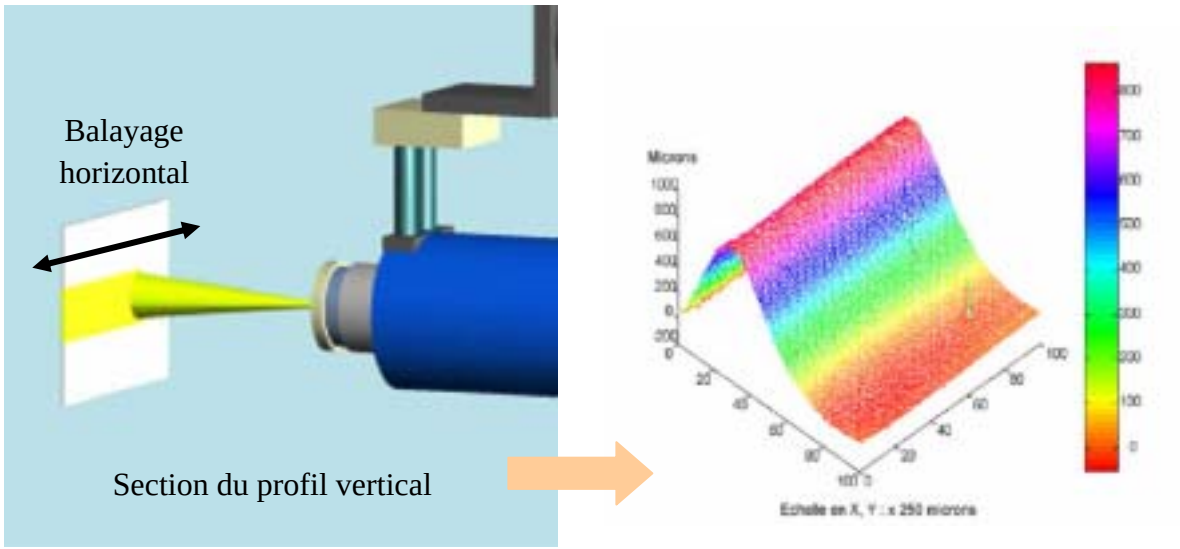


Figure 4.16 Obtention d'un profil de cordon par échographie acoustique

L'obtention du profil de matière en 3D par voie acoustique constitue un moyen efficace, précis et rapide. Avec la technique acoustique, un profil moyen de matière est obtenu. Il est associé à une plume de matière sous la forme d'un profil mesuré. Tous les paramètres opératoires de projection peuvent être associés à cette plume de matière par une interface réalisée par R. Bonnet [1].

Pour chaque plume, il est possible de sauvegarder l'ensemble des paramètres opératoires d'obtention du profil de matière. La Figure 4.17 présente l'interface réalisée pour le procédé APS.

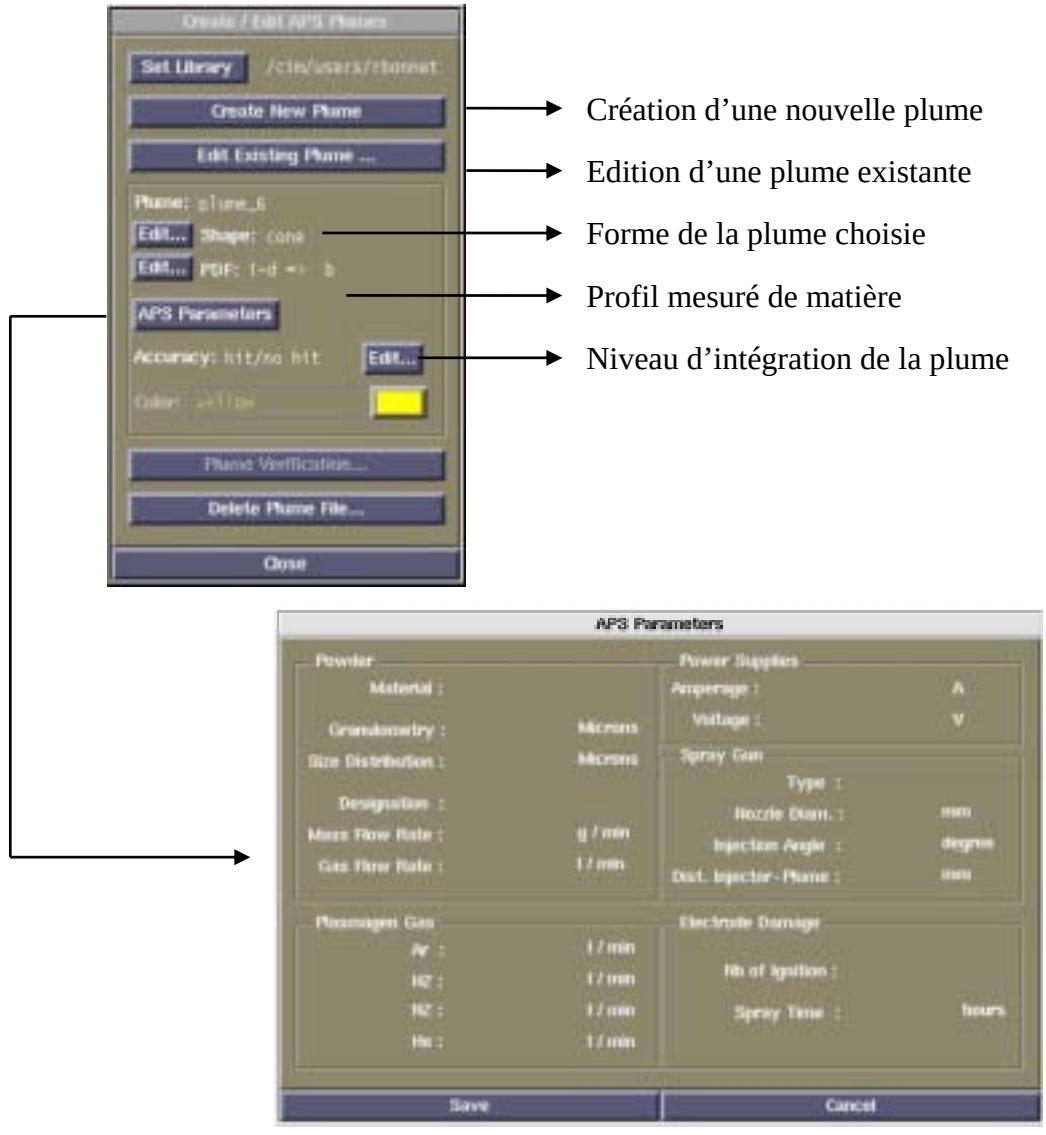


Figure 4.17 Panneaux relatifs à l'introduction des paramètres pour une plume APS [1]

4.2.4 Superposition des cordons pour la simulation d'un dépôt

Les courbes obtenues après palpage sont formées d'un ensemble de points donnant pour une abscisse y l'épaisseur e en micromètres. Chaque fichier est composé d'environ 500 à 1000 points. Un lissage de la courbe est donc nécessaire pour éviter les aléas de la courbe et les écarts d'acquisition. La Figure 4.18 présente une forme expérimentale de cordon et le lissage correspondant du profil.

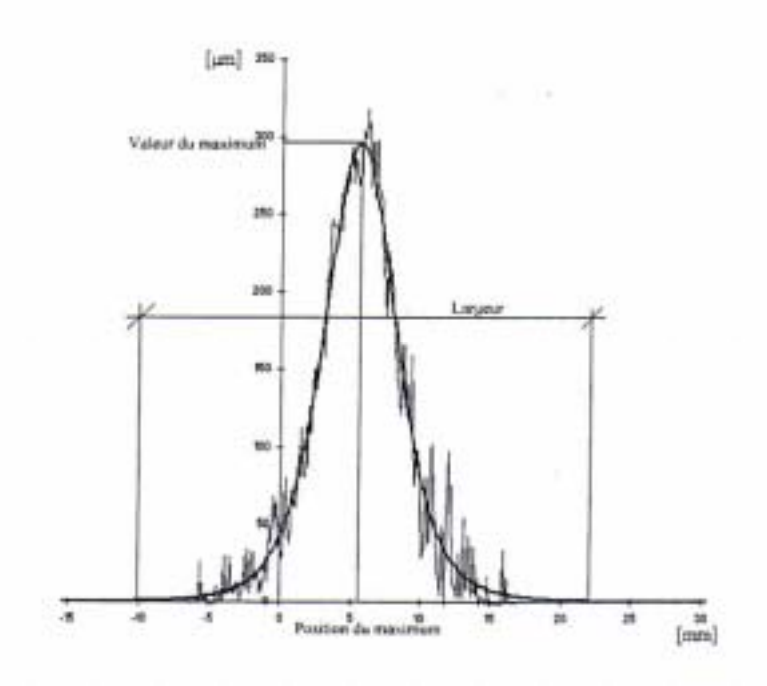


Figure 4.18 Définition des caractéristiques du cordon

Après le traitement de la courbe d'acquisition, le profil de forme peut être importé dans ProfileKit. La Figure 4.19 montre un exemple de profil dans ProfileKit.

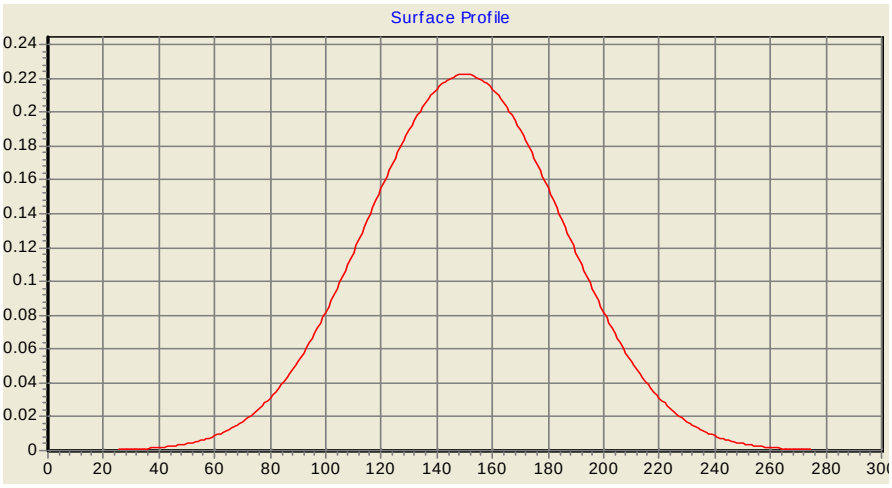


Figure 4.19 Exemple de profil de cordon importé dans ProfileKit

Ensuite, par la mise en œuvre d'un système d'expert [22] et l'optimisation des paramètres opératoire [23], ProfileKit détermine les paramètres optimaux (pas et nombre de passe) en fonction de l'épaisseur finale du dépôt demandée.

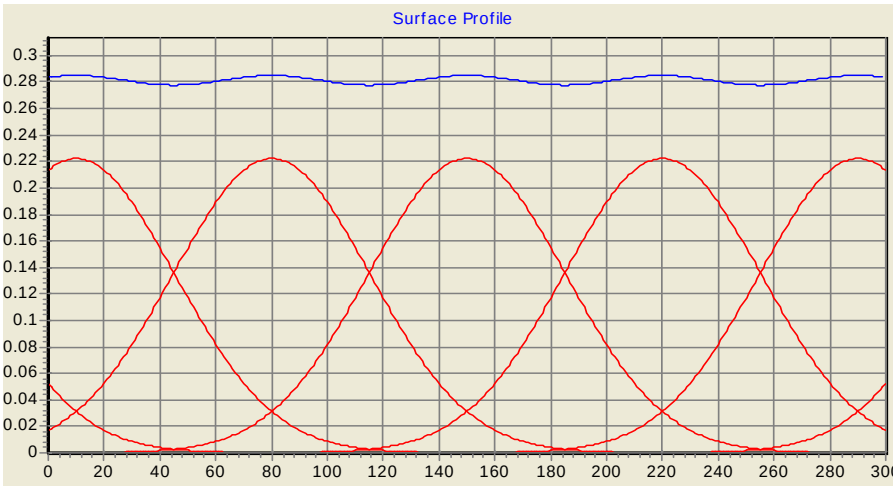


Figure 4.20 Exemple de la superposition des cordons et simulation de l'épaisseur finale du dépôt

La Figure 4.20 donne un exemple de simulation, de la formation d'un dépôt par superposition de cordons.

4.3 Module MonitorKit

4.3.1 Introduction

La programmation hors-ligne assistée par des logiciels de simulation robotique permet aujourd'hui la définition de trajectoires complexes et précises d'outil. Elle permet d'accroître la productivité des cellules robotisées par le transfert d'un programme directement exécutable par le robot. Ce transfert nécessite :

- ü l'utilisation de post-processeurs pour générer un programme lisible par la baie de commande du robot,
- ü la capacité d'étalonner la cellule robotisée (définir précisément le positionnement relatif pièces / robot).

Les problèmes principaux rencontrés aujourd'hui se situent à plusieurs niveaux.

Tout d'abord, le respect de la vitesse spécifiée au niveau de l'outil présente une obligation incontournable pour l'ensemble des procédés dits continus (encollage, soudage, projection thermique...). Or, d'une manière générale, pour des raisons de sécurité, un robot privilégie toujours la trajectoire à la vitesse. Aussi, lorsque les contraintes du procédé concerné imposent un respect des vitesses spécifiées, il est important de pouvoir vérifier que le robot peut respecter les consignes de vitesse.

Par ailleurs, la programmation hors-ligne nécessite une phase d'étalonnage. Cet étalonnage permet aux cellules robotisées réelles et virtuelles d'être tout à fait conformes au niveau des positionnements relatifs des pièces et de l'environnement par rapport au robot. Cependant, cette phase nécessite aujourd'hui une procédure manuelle de palpage contraignante. Cette procédure d'étalonnage est suffisamment astreignante pour que la programmation des robots s'effectue encore manuellement, même si la faisabilité de la programmation hors-ligne a été montrée.

A partir de ces constatations, un système de contrôle en temps réel, dit « MonitorKit »,

intégré dans T.S.T., a été développé. Les objectifs sont :

- ü de définir rapidement la position de la pièce sur laquelle le robot doit travailler sans démontage de l'outil,
- ü d'effectuer un contrôle de la trajectoire en termes de position spatiale et de vitesse de l'outil tout au long de la trajectoire.

Cette section présente les matériels d'étalonnage existants, les paramètres d'influence sur la précision des mesures d'acquisition ainsi que les développements et fonctionnalités du MonitorKit pour le contrôle des vitesses d'outils et l'étalonnage géométrique des cellules robotisées.

4.3.2 Rappels

4.3.2.1 Divergence entre la simulation et les cellules robotisées

D'une façon générale, la chaîne de simulation CAO ne représente qu'imparfaitement le comportement réel du robot et de la cellule robotisée car les phénomènes sont modélisés de façon approchée pour accélérer les calculs. Ceci conduit à des différences entre le temps de cycle observé et celui attendu. Les points suivants peuvent être cités comme phénomènes non pris en compte :

- ü les modèles CAO des robots sont théoriques ; des écarts entre le robot réel et le virtuel sont inévitables (jeux, positionnement),
- ü les frottements, les flexibilités des bras de robots, les non linéarités des transmissions sont des paramètres difficiles à intégrer. Les erreurs non géométriques correspondantes dépendent de la charge, des vitesses, des accélérations mises en jeu lors des déplacements. Elles peuvent, de plus, évoluer avec la température de l'atelier, à l'échelle de la vie du robot selon son usure,

- ü au niveau de l'outil, les erreurs géométriques sont d'autant plus importantes que le centre d'outil est éloigné du centre du poignet,
- ü au niveau de la cellule, les erreurs géométriques et les incertitudes de positionnement des repères "pièce" peuvent interférer avec la précision des trajectoires,
- ü les poids de la torche et des câbles, les pressions du gaz et de l'air refroidissement sont des facteurs d'influence sur les déplacements du robot réel.

Fort heureusement, la part des erreurs "inévitables" (usure, frottement...) est très largement minime face aux autres. Par contre, la phase d'étalonnage est importante pour assurer la précision exigée par le procédé mis en jeu.

4.3.2.2 Etalonnage ou calibrage de cellules robotisées

L'étalonnage géométrique d'une cellule robotisée doit permettre de placer en conformité les cellules robotisées réelles et virtuelles et de calculer les transformations exactes qui lient :

- ü la base du robot au repère atelier,
- ü le repère terminal (poignet du robot) à la base,
- ü le repère outil au repère terminal,
- ü les repères pièces au repère atelier.

L'étalonnage ne concerne pas tous les éléments d'une cellule robotisée de la même façon puisque les caractéristiques à identifier n'ont pas la même évolution dans le temps :

- ü l'étalonnage de la structure mécanique pratiqué lors de la mise en service est valide pour toute la durée de vie du robot (sauf en cas de choc ou de déplacement),
- ü la localisation du robot et des pièces par rapport au repère de la cellule,
- ü l'étalonnage des offsets codeurs : après chaque opération de maintenance ou après un choc du robot avec l'environnement,

- l'étalonnage du centre d'outil qui relève des procédés de métrologie classique.

Il convient alors d'adapter les moyens utilisés pour l'étalonnage à la précision souhaitée. Bien des systèmes de calibrage existent sur le marché. La projection thermique, de par la profondeur de champ tolérée (quelques millimètres) et l'incertitude angulaire (quelques degrés) ne nécessite pas une procédure draconienne de calibrage de la cellule. Des systèmes de pointes calibrées et des procédures de palpage donnent des résultats tout à fait satisfaisants pour des investissements financier et temporel très réduits. Au niveau du robot, il convient tout de même d'étalonner les offsets des codeurs après chaque opération de maintenance ou en cas de choc du robot avec son environnement. D'après Caenen [24], 50 à 60% de l'erreur en positionnement liée au robot est due aux offsets articulaires.

4.3.2.3 Mise en œuvre de l'étalonnage

L'étalonnage fait intervenir l'utilisateur directement sur le site robotisé. Un système de mesures doit satisfaire plusieurs contraintes opérationnelles parfois contradictoires :

- facilité de mise en œuvre, d'autant plus cruciale que l'étalonnage se déroule sur un site robotisé très encombré et s'effectue par un opérateur non spécialisé,
- minimisation des temps d'immobilisation du robot, si possible sans démontage de l'outil,
- faible coût.

Il doit en outre répondre le mieux possible au cahier des charges fonctionnelles suivant :

- mesures sans contact (pour éviter toute déformation du robot faussant alors les mesures),
- mesures effectuées en automatique,
- précision supérieure ou du même ordre de grandeur que la répétabilité du robot,

- ü mesures de situation effectuées dans tout le volume de travail du robot,
- ü stockage des mesures réalisées dans des fichiers.

4.3.3 Méthodes de contrôle en temps réel

Les procédures de contrôle d'un robot en trois dimensions peuvent être classés en deux familles selon qu'elles procèdent par mesure directe des coordonnées des points ou par acquisition indirecte des données via le contrôleur robot.

4.3.3.1 Mesure directe des coordonnées des points

Plusieurs techniques peuvent être mises en œuvre, mais elles présentent toutes des limitations importantes. Il est par exemple possible d'utiliser des capteurs de proximité.

Un système à base de capteurs de distance doit être composé d'au moins trois paires de capteurs pour mesurer la distance de chaque axes dans l'espace de travail (Figure 4.21).

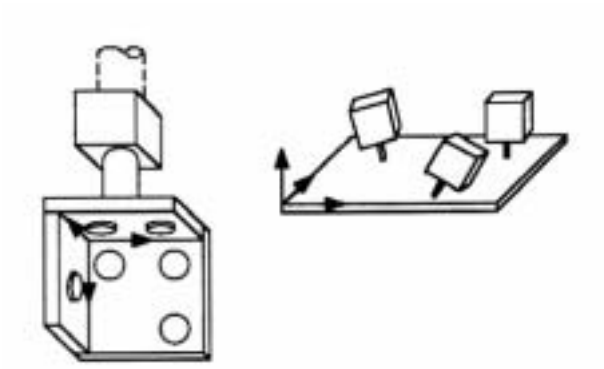


Figure 4.21 Principe de l'étalonnage par proximètres sans contact

Chaque proximètre mesure la distance le séparant d'une des faces du cube. Par un traitement approprié, les équations des trois plans du trièdre sont retrouvées. L'intersection de ces trois plans donne l'orientation du trièdre. La dimension, la position et l'orientation des

cubes doivent être parfaitement connues. Les proximètre utilisés doivent avoir une très grande résolution (quelques micromètres) ce qui correspond à une faible plage de mesure (de l'ordre du centimètre).

Il est aussi possible d'utiliser un système à base de théodolites. Un théodolite est un télescope instrumenté donnant l'azimut et la hauteur d'une cible par rapport à une ligne de visée. L'étalonnage des robots se fait par triangulation grâce à deux théodolites Thl et Th2 visant une ou plusieurs cibles M placées sur l'organe terminal du robot (Figure 4.22).

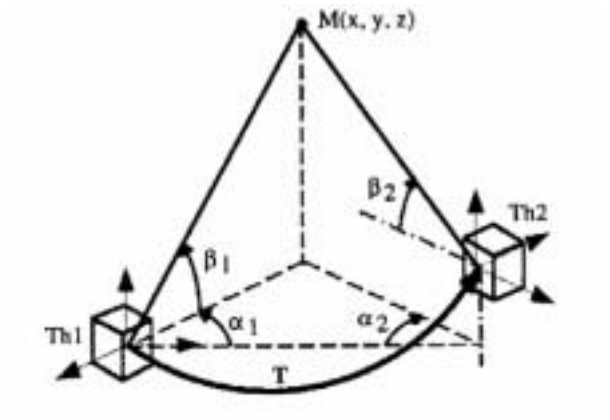


Figure 4.22 Principe de l'étalonnage par théodolites

En connaissant la transformation T qui lie les deux théodolites, les coordonnées du centre de la cible visée sont déterminées. Avec trois cibles, la position et l'orientation du repère outil peuvent être déterminées dans un repère de référence lié à un des théodolites.

La précision des mesures peut être très bonne (de l'ordre de 5/100 mm sur une distance de 1 mètre) mais la mise en œuvre est complexe (étalonnage du système pour déterminer T, acquisition longue, opérateur expérimenté). Le coût des équipements est de surcroît élevé. Par ailleurs, le montage des capteurs peut influencer le mouvement du robot et l'installation de la torche.

A partir de ces constatations, une autre solution a été étudiée. Elle utilise la base du système de commande du robot. Les contraintes imposées ont été d'obtenir un système précis, peu onéreux et ne nécessitant pas une mise en œuvre complexe.

4.3.3.2 Acquisition indirecte des coordonnées des points

Un robot se compose de deux parties principales : le système de commande et le manipulateur [25] (Figure 4.23).

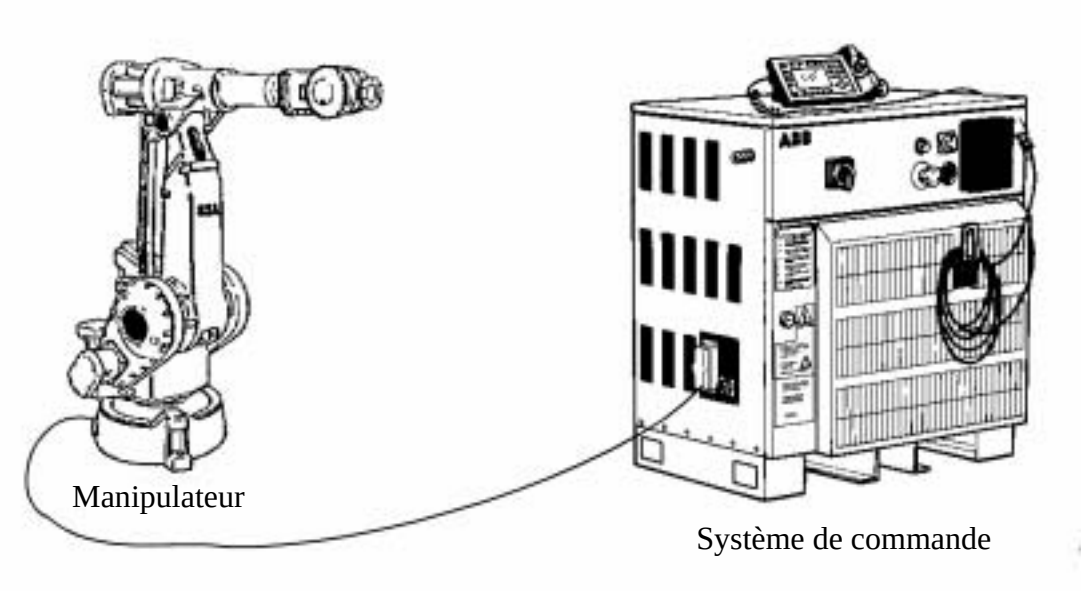


Figure 4.23 L'unité de commande et le manipulateur sont reliés par deux câbles

Le système de commande contient l'électronique nécessaire pour commander un éventail d'unités mécaniques, des axes externes et les équipements périphériques. Il commande le robot pour exécuter le programme et bouge le CDO à la position demandée. Toutes les données du robot (la position et l'orientation du CDO, l'angle de chaque axe, l'information de l'outil, etc.) se trouvent sur le pupitre mobile d'apprentissage. La Figure 4.24 présente les données affichées sur le pupitre mobile d'apprentissage.

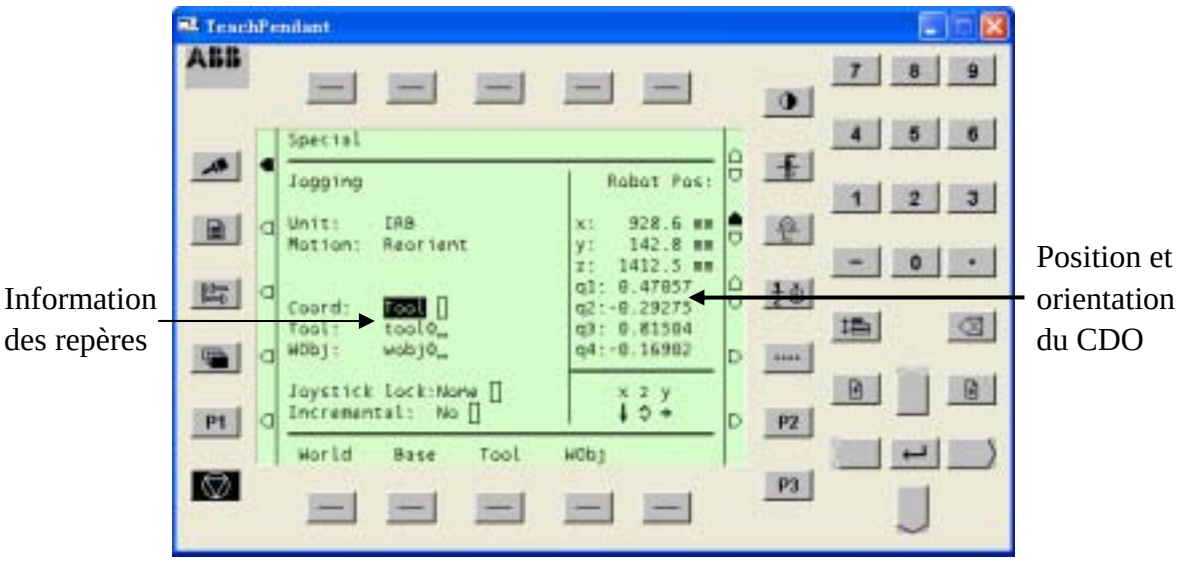


Figure 4.24 Pupitre mobile d'apprentissage (virtuel)

Evidement, le système de commande reçoit toutes les données du robot qui sont obtenues par les capteurs intégrés au robot. Si les données du robot peuvent être sorties de façon continue, il est alors possible d'obtenir la trajectoire complète réalisée. De plus, le contrôleur robot peut parvenir à la précision de 0.01 mm.

Le logiciel MonitorKit développé obtient les données cinématiques du robot par acquisition indirecte à partir du système de commande du robot. Cette solution très peu onéreuse offre une excellente résolution et une excellente fiabilité.

4.3.4 Communication avec les robots

Comment peut on sortir les données cinématiques à partir du contrôleur robot ? Le système de commande ne fournit pas les signaux analogiques correspondant aux valeurs des capteurs intérieurs, mais il permet de sortir les données numériques à l'aide d'un ordinateur extérieur.

Le système de commande des armoires ABB S4C dispose de trois sorties séries pour usage permanent : deux RS232 et un RS422 duplex intégral. Les signaux recueillis de forme point à point, peuvent être dirigés vers une imprimante, un terminal, un ordinateur ou tout autre équipement [26]. Les sorties périodiques peuvent être utilisés au débit de 19.200 bit/s

(maximum 1 canal avec vitesse 19.200 bit/s). Le système de commande a deux canaux Ethernet qui peuvent être employés à 10 Mbit/s ou à 100 Mbit/s par canal. Le débit de communication est adapté automatiquement. La Figure 4.25 présente le schéma des connexions disponibles.

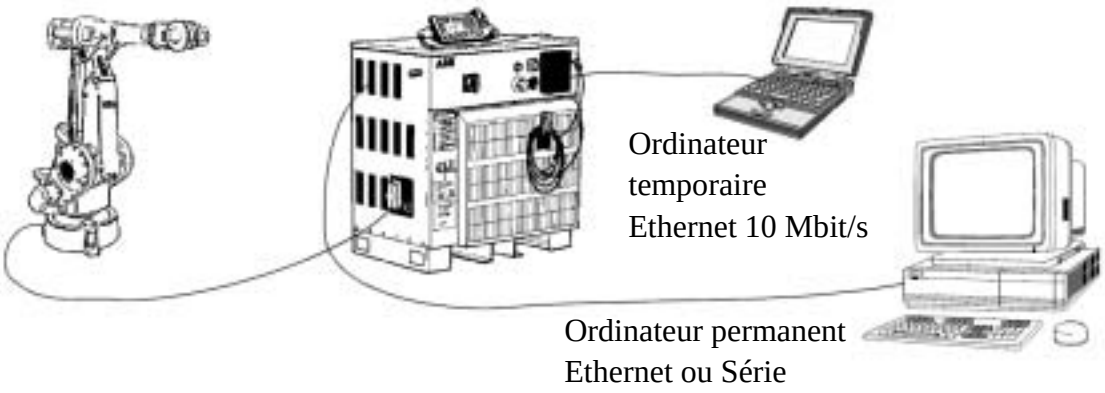


Figure 4.25 Communication point à point

La communication permet de plusieurs possibilités comme :

- ü DNS, DHCP etc. (incluant passerelle multiple)
- ü FTP/NFS client et FTP serveur
- ü Contrôle de système avec le protocole RAP (Robot Application Protocol)
- ü Démarrage/Mise à niveau du logiciel du système de contrôle via le réseau ou un PC

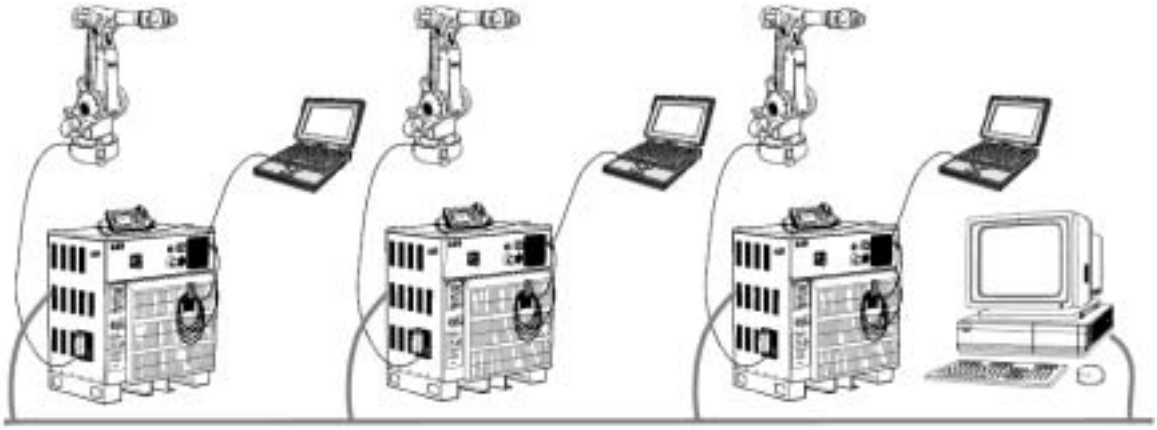


Figure 4.26 Communication LAN

Dans le cas de notre étude, la stabilité et le débit de la communication sont des paramètres importants. L'Ethernet 10Mbit/s a été employé comme liaison physique de communication. Le protocole dit RAP (Robot Application Protocol) a été utilisé pour demander / recevoir les données du robot. Ce protocole sera présenté dans le paragraphe suivant. La Figure 4.27 présente la configuration de la communication utilisée par le module MonitorKit.

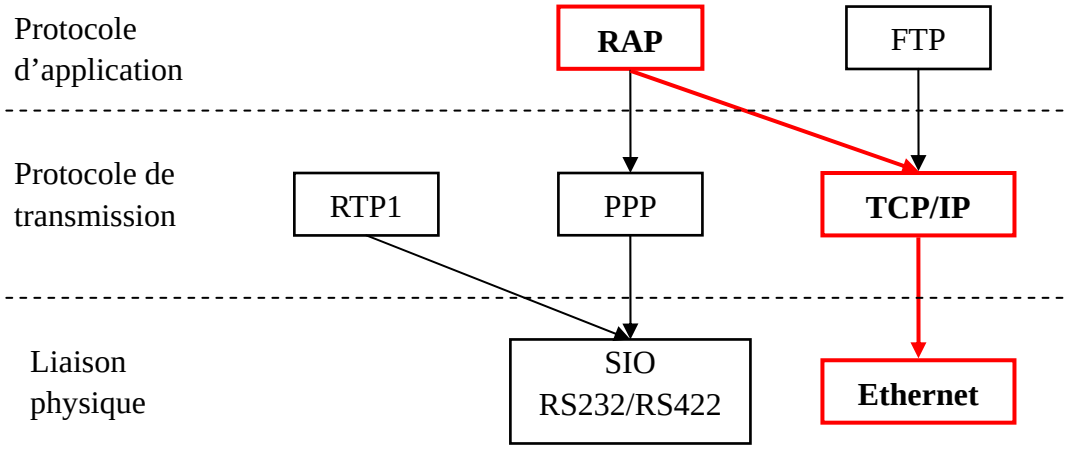


Figure 4.27 Structure de la configuration de communication [27]

Les configurations de la communication sont :

- ü Liaison physique – Ethernet 10Mbit/s
- ü Protocole de transmission – TCP/IP
- ü Protocole d'application – RAP (Robot Application Protocol)

4.3.4.1 Le protocole RAP

Afin d'obtenir les données du robot, il faut communiquer avec le robot en suivant une

certaine méthode. ABB fournit un protocole dit « RAP » pour effectuer l'échange des données avec le robot. Ce protocole est un protocole à base du standard « RPC » (Remote Procedure Call) qui fournit une interface au contrôleur robot ainsi qu'une série de services.

Les services sont groupés dans les catalogues suivants [28] :

- ü General Management Services (Services de la gestion générale)
- ü Variable Access Services (Services d'accès aux variables)
- ü File Management Services (Services de gestion des fichiers)
- ü Program Control Services (Services de commande de programme)

Dans notre étude, les service d'accès aux variables ont été employés pour obtenir les données (variables) du robot, la position du CDO par exemple.

4.3.4.2 Le standard RPC

Le protocole RAP fonctionne sur la base du standard RPC qui est un protocole permettant d'appeler des procédures sur un ordinateur distant à l'aide d'un serveur d'application.

Le standard RPC est une technique développée par SUN Microsystems Inc. qui est couramment utilisée dans le modèle « client–serveur ». Les appels de procédure à distance sont semblables aux appels de procédure locaux. Dans un appel de procédure local, le programme place les paramètres dans un endroit bien connu (habituellement la pile), et entre la procédure. Quand la procédure est accomplie, le programme principal reprend l'exécution. Dans des appels de procédure à distance, le programme principal fait un appel normal à une procédure locale dite « CLIENT STUB » avec les paramètres nécessaires. Le « CLIENT STUB », après toutes les opérations nécessaires au niveau local, localise le serveur à distance et envoie le message au « SERVER STUB » qui va exécuter la procédure avec les paramètres donnés. Une fois le service fait, le « SERVER STUB » contacte le « CLIENT STUB » et renvoie les résultats de la procédure. Le « CLIENT STUB » renvoie ensuite les résultats au programme principal qui peut reprendre l'exécution.

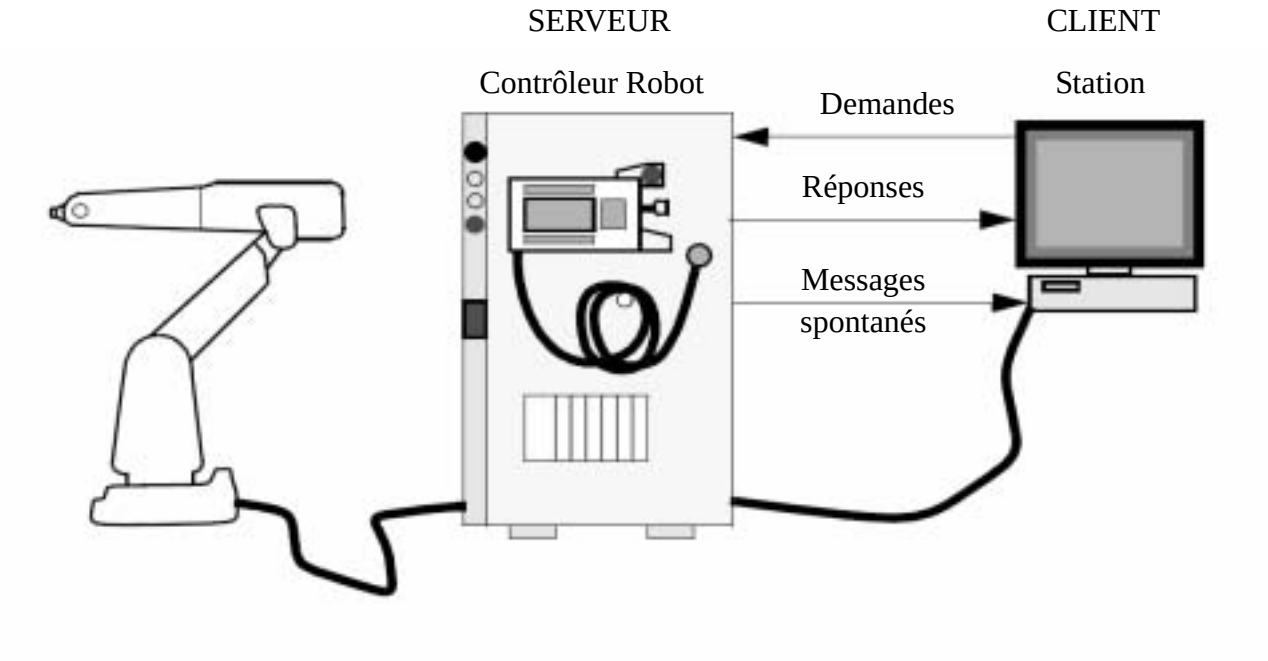


Figure 4.28 Procédure RAP

Dans le cas de cette étude, le système de commande du robot joue le rôle du « SERVEUR », l'ordinateur extérieur est la console du « CLIENT ». L'ordinateur extérieur appelle une procédure (obtenir la position actuelle par exemple), le message est transféré au contrôleur robot rapidement. Le contrôleur robot exécute cette procédure localement et prépare les données des résultats (les coordonnées d'une position). Ensuite, le contrôleur robot renvoie les données à l'ordinateur extérieur. Dans le cas de cette étude, le protocole RAP clientèle est compilé sous l'environnement windows32 à l'aide d'une trousse à outil de Distinct ONC RPC/XDR (cf. Annexe 5). La Figure 4.28 présente le processus de l'appel RAP.

MonitorKit appelle les services d'accès des variables continuellement pour obtenir les données du robot incluant la position et l'orientation actuelles du robot, l'information de l'outil, etc. Un appel complet est présenté dans la Figure 4.29.

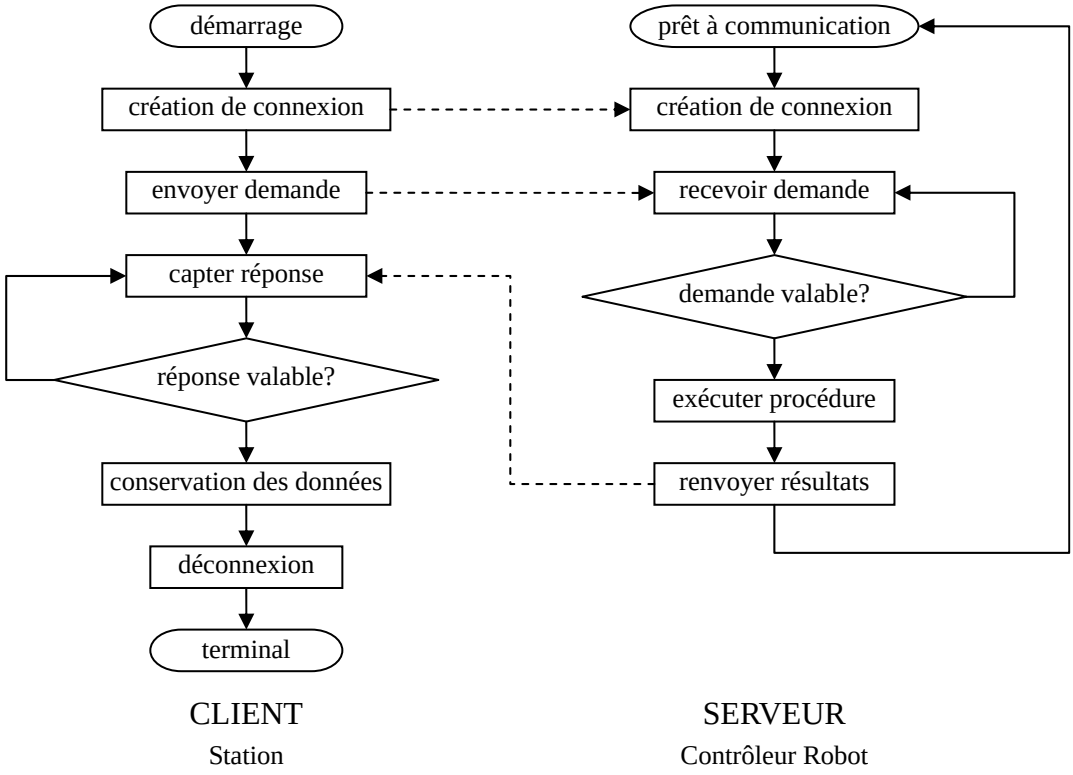


Figure 4.29 Une cycle d'appel RAP

4.3.5 Obtention de la vitesse du robot

La vitesse du robot est un paramètre très important qui influence la qualité du dépôt. Dans notre cas d'application, le module MonitorKit fournit deux manières pour obtenir la vitesse du robot.

La première manière est de calculer la vitesse par rapport à la position de chaque point en fonction du temps. Les coordonnées de chaque point peuvent être obtenus à l'aide du protocole RAP (Figure 4.30).

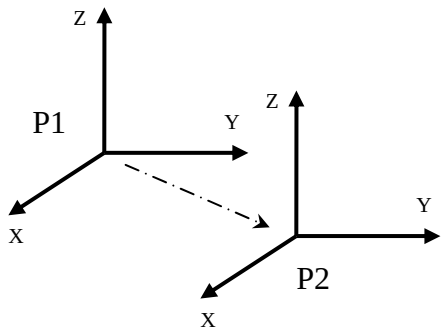


Figure 4.30 Deux points sur la trajectoire

La distance entre les deux point peut être calculé par :

$$P1 = (x1, y1, z1) \qquad P2 = (x2, y2, z2)$$
$$D = \sqrt{(x2 - x1)^2 + (y2 - y1)^2 + (z2 - z1)^2}$$

Le temps intervalle entre deux points peut être compté précisément par un compteur de haute précision sous windows (cf. 3.3.1). La vitesse moyenne entre deux points est alors donnée par :

$$\overline{V} = \frac{D}{t}$$

Une autre manière d'obtenir la vitesse est de capter le signal analogique du contrôleur robot qui correspond à la vitesse du CDO actuelle. ABB fournit un module combi d'Entrée / Sortie à relais DSQC 327 qui peut configurer une sortie analogique comme un indicateur de la vitesse du robot.

Pour sortir la valeur de la vitesse sur la sortie analogique, il faut définir le signal de sortie et la gamme de la valeur. L'élément le plus important est la configuration du ratio de la valeur incluant « Max./Min. Logique » et « Max./Min. Physique ». La Figure 4.31 présente le schéma d'échelle des valeurs [27].

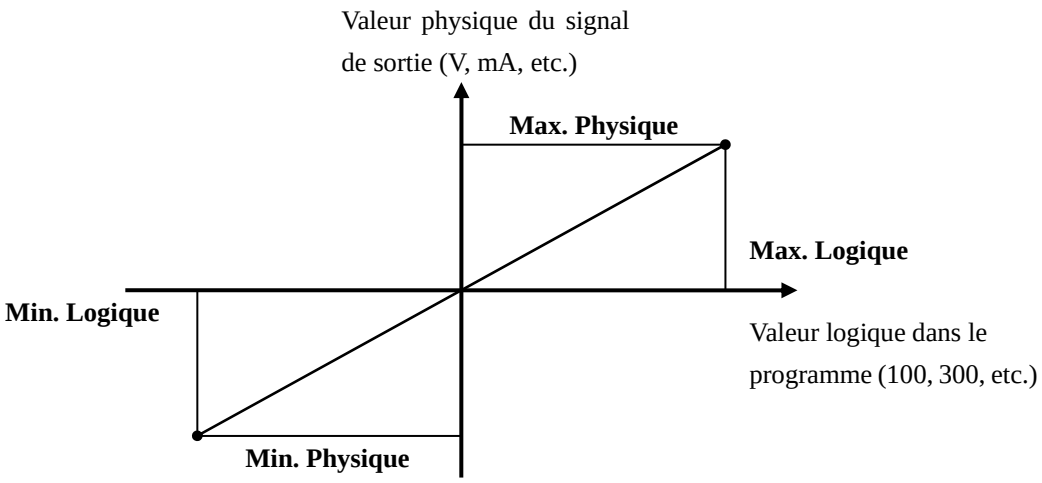


Figure 4.31 Changement d'échelle des valeurs d'un signal analogique

« Max./Min. Logique » représente la valeur max/min sur laquelle peut être amenée une entrée ou une sortie analogique, à partir d'un programme RAPID ou à partir du pupitre mobile d'apprentissage. Les unités sont définies par l'utilisateur (longueur, temps par exemple). « Max./Min. Physique » représente la valeur physique max/min qui peut être appliquée sur la sortie ou sur l'entrée. Si les valeurs physiques et logiques sont mises à la même valeur, il est résulte un facteur d'amplification de 1.

Dans le cas du module MonitorKit, les paramètres suivant ont été utilisés :

Max. Physique = 10 V

Min. Physique = 0 V

Max. Logique = 2000 mm/s

Min. Logique = 0 mm/s

Avec ces configurations des valeurs, l mV de tension sur la sortie analogique correspond à 0.2 mm/s de la vitesse du robot. Par exemple, la tension de « 2.38 V » correspond la vitesse du CDO de « 476 mm/s ».

Pour numériser le signal de tension, une carte d'acquisition NI DAQPad-6020E (produit de National Instruments Corporation) a été employée. Cette carte fournit une multifonction

d'acquisition à 16 entrées analogiques 12 bit jusqu'à 100 Kéch./s et 2 sorties analogiques 12 bit [29]. Dans notre cas d'application, la tension d'entrée maximal est 10 V, la résolution de numérisation correspondante est 2.4414 mV. Donc la résolution de vitesse obtenue est 0.488 mm/s.

Un PC extérieur a été utilisé pour conserver les données obtenues par la carte d'acquisition. Le PC et la carte d'acquisition sont reliés par un câble USB, et le démarrage / arrêt de l'acquisition est commandé par MonitorKit afin de synchroniser les données. Le schéma du système est présenté dans la Figure 4.32.

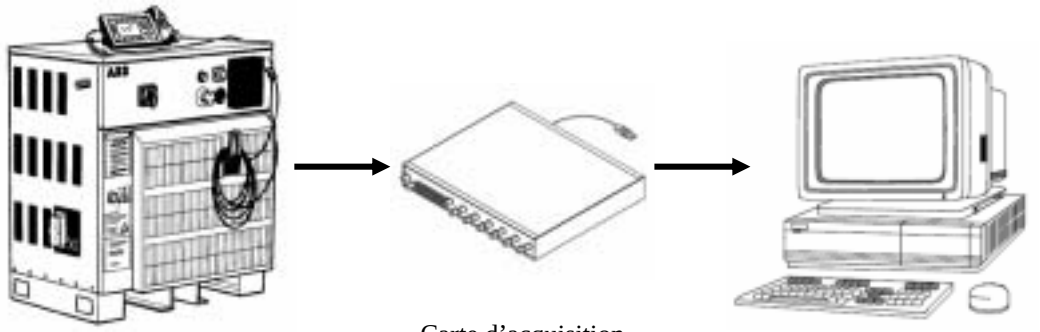


Figure 4.32 Système du MonitorKit

4.3.6 Visualisation des trajectoires robot

Pour contrôler visuellement la trajectoire robot, le logiciel doit dessiner la trajectoire dans un espace en 3-Dimensions sur l'écran. Il existe deux standards principaux d'application générant des images 3D (mais également 2D) :

- ü SGI OpenGL
- ü Microsoft Direct3D

OpenGL (Open Graphics Library) est une spécification qui définit une API multi

plate-forme pour la conception d'applications générant des images 3D. L'interface regroupe environ 250 fonctions différentes qui peuvent être utilisées pour afficher des scènes tridimensionnelles complexes à partir de simples primitives. Du fait de son ouverture, de sa souplesse d'utilisation et de sa disponibilité sur toutes les plate-formes, elle est utilisée par la majorité des applications scientifiques, industrielles ou artistiques 3D et certaines applications 2D vectorielles. Cette bibliothèque est également très populaire dans l'industrie du jeu vidéo où elle est en rivalité avec Direct3D (sous Microsoft Windows). Plusieurs implémentations d'OpenGL (exploitant l'accélération 3D fournie par le matériel) existent pour Windows, de nombreuses stations Unix et Mac OS. Ces implémentations sont généralement fournies par les constructeurs de matériels graphiques et y sont étroitement liées. Il existe une implémentation libre de cette bibliothèque, nommée Mesa, créée en 1993 par Brian Paul.

Direct3D est un composant de l'API Microsoft DirectX. Direct3D est utilisé uniquement dans les multiples systèmes d'exploitations Windows de Microsoft (Windows 95 et au-delà). Direct3D sert à générer des graphismes en trois dimensions pour les applications où la performance est importante, comme les jeux vidéo. Direct3D permet également à des applications de fonctionner en plein écran, plutôt qu'intégrées dans une fenêtre, bien qu'elles puissent toujours tourner dans une fenêtre si elles sont programmées pour cette utilisation. Direct3D utilise l'accélération matérielle si elle est disponible à travers une carte graphique. Microsoft effectue des mises à jour continuelles de Direct3D pour permettre l'exploitation des dernières technologies disponibles sur les cartes graphiques 3D.

Comme explicité précédemment, la localisation des deux APIs est extrêmement claire. OpenGL se voue aux applications industrielles et scientifiques. Par contre, Direct3D est une des fonctions de DirectX (qui est essentiellement utilisé pour le jeu) la plus en vogue actuellement. Afin d'avoir une meilleur stabilité et compatibilité du logiciel, nous avons utilisé l'OpenGL comme l'interface de programmation graphique. La forme de programmation de l'OpenGL est présentée en détail dans l'Annexe 6.

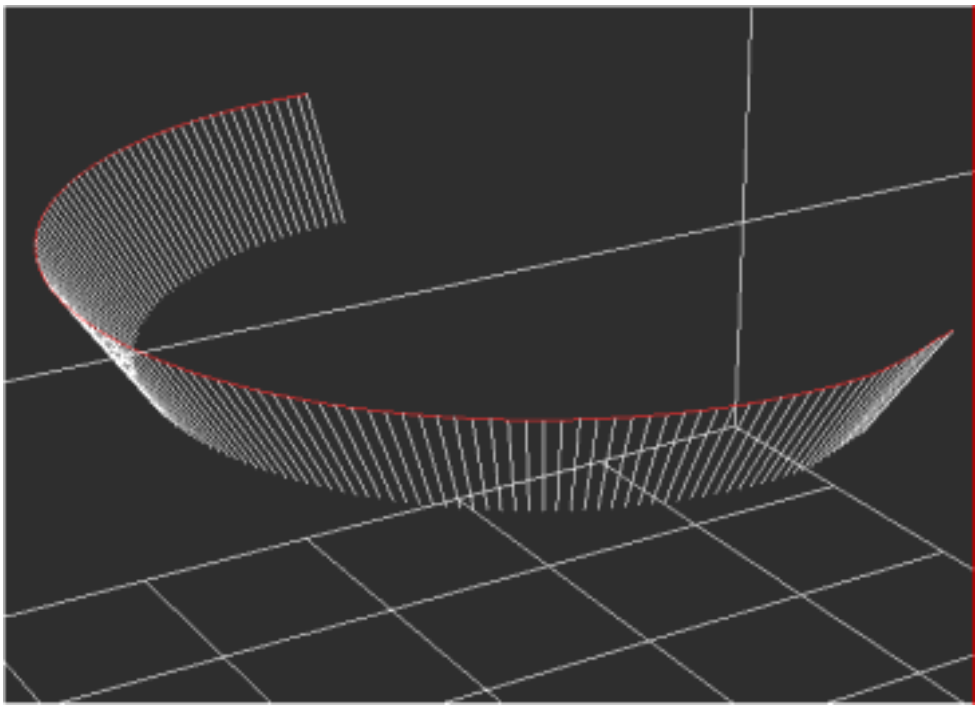


Figure 4.33 La fenêtre graphique du MonitorKit

La Figure 4.33 montre la fenêtre de visualisation du MonitorKit. La courbe rouge est la trajectoire robot qui est couplée par les points obtenus via le contrôleur robot. Les indicateurs en jaunes représentent l'orientation sur chaque point de la trajectoire.

4.3.7 Comparaison entre la trajectoire conçue et la trajectoire réelle

D'après un système de contrôle, une comparaison graphique et statistique de la trajectoire doivent être présentées afin de savoir la précision cinématique du processus de la projection thermique. MonitorKit fournit une fonctionnalité de comparaison entre la trajectoire programmée et la trajectoire réelle incluant la trace et la vitesse sur la trajectoire.

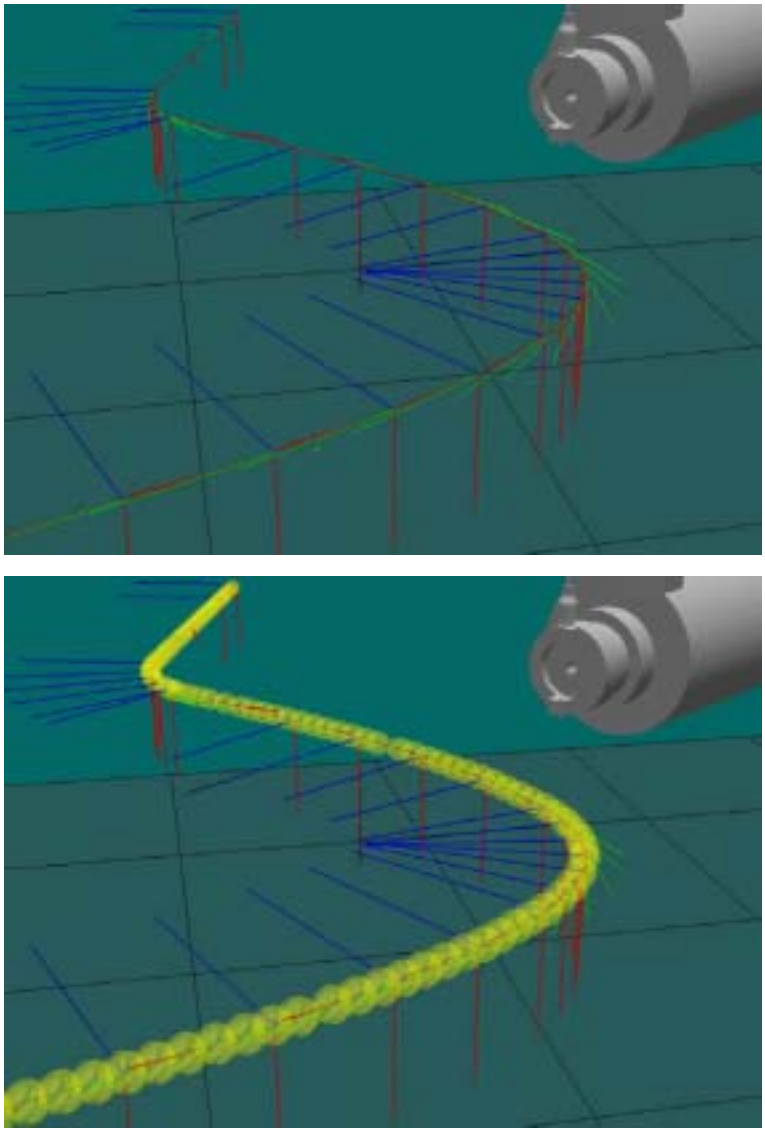


Figure 4.34 Comparaison entre la trajectoire conçue et la trajectoire réelle

La Figure 4.34 montre une comparaison entre une trajectoire conçue et une trajectoire réelle. La courbe rouge est la trajectoire programmée dans RobotStudio qui est composée par plusieurs positions clés. Le tunnel composé par les sphères jaunes présente la trajectoire réelle effectuée par robot dans processus de projection. Ce tunnel rubané est dessiné à partir des points de la trajectoire obtenus par MonitorKit. Il est envisageable de vérifier si la trajectoire programmée est bien respectée grâce à la trace de la trajectoire réelle (si la trajectoire programmée est située dans le milieu du tunnel) à partir de la comparaison visualisée dans RobotStudio.

Dans le cas de la projection thermique, c'est le contrôle de la vitesse du CDO qui nous intéresse car il correspond à la vitesse relative torche – substrat qui influe directement sur l'épaisseur du dépôt. Une comparaison entre la vitesse réelle et conçue est aussi fournie par MonitorKit. La Figure 4.35 montre l'interface de la statistique de la vitesse.

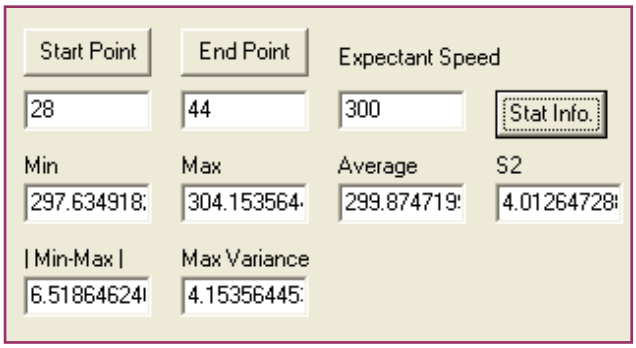


Figure 4.35 Interface de statistique sur la vitesse sur la trajectoire

Cette fenêtre dialogue fournit les paramètres statistiques sur la vitesse d'une session donnée :

- la vitesse minimale et maximale,
- la vitesse moyenne,
- la variance de la vitesse,
- la valeur absolue entre la vitesse maximale et minimale,
- la valeur de tolérance maximale sur la vitesse conçue.

La Figure 4.36 présente le schéma de ces paramètres statistiques.

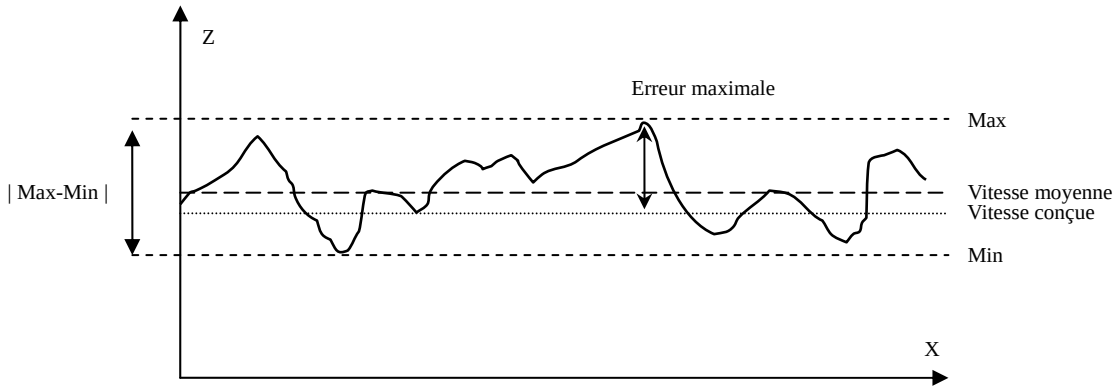


Figure 4.36 Les paramètres statistiques de la vitesse sur une trajectoire donnée

4.4 Vérification du T.S.T. par expérimentations

Pour vérifier les fonctionnalités du T.S.T., certaines expériences ont été effectuées. En premier lieu, le protocole d'expérimentation sera présenté incluant les matériaux de projection, les paramètres opératoires, les substrats, les trajectoires générés par PathKit, etc. Après avoir donné les simulations d'épaisseur du dépôt effectuées par le module ProfileKit avant les essais de projection thermique, nous poursuivrons le contrôle en temps réel pendant le processus réel des essais à l'aide de MonitorKit. Finalement, les résultats d'expérience seront discutés et les conclusions sur cette extension logicielle de projection thermique seront présentées.

4.4.1 Protocole de l'expérimentation

4.4.1.1 Matériaux projetés

Pour exprimer les différentes fonctions du PathKit, trois formes des pièces (pièce carrée, pièce circulaire et pièce incurvée) sont nécessaires afin de vérifier les trois type de trajectoire correspondante.

Pour conduire ces expériences, la poudre Al_2O_3/TiO_2 60/40 (Starck Amperit 746.074 +45-15 µm) a été choisie. Cette poudre est largement utilisée en industrie.

4.4.1.2 Substrats à revêtir

L'acier d'usage général (XC-10) a été choisi comme matériau de base pour élaborer les substrats des échantillons de forme simple. Par ailleurs, l'inox 316 a été choisi comme matériau de base pour l'échantillon de forme complexe. Ces matériaux sont utilisés usuellement dans les constructions mécaniques et métalliques générales, assemblées ou soudées.

4.4.1.3 Paramètres opératoires

Les paramètres opératoires choisis correspondent à des conditions classique de projection thermique. Les valeurs des paramètres opératoires de cette préparation sont regroupées dans le Tableau 4.1.

Tableau 4.1 Paramètre opératoires utilisés pour la pièce circulaire

Paramètres	**Valeur**
Angle d'injection de poudre	90 degrés
Diamètre interne d'injecteur	1.8 mm
Distance entre l'injecteur et le plasma	6 mm
Débit de gaz Ar	30 l/min
Débit de gaz H_2	8 l/min
Débit de gaz porteur (argon)	3.4 l/min
Débit de poudre	30 g/min
Courant électrique	550 A
Distance de projection	110 mm
Angle de projection	90 degrés
Pas de balayage	5 mm
Vitesse de la torche	500 mm/s

4.4.1.4 Configuration de projection et génération de trajectoire

Pour la pièce carrée, les substrats ont été découpés aux dimensions nominales 350×100×5 mm^3. Afin de vérifier la trajectoire réelle du robot, la pièce virtuelle dans RobotStudio™ est définie aux dimensions nominales 270×100×5 mm^3. PathKit crée la trajectoire par rapport à la forme géométrique de la pièce virtuelle et réserve une distance de débordement de 20 mm. Après avoir bien étalonné la cellule virtuelle par rapport à la cellule réelle, une plaque carrée a

été mise en position sur un support. La Figure 4.37 montre les cellules réelles et virtuelles de la même tâche.

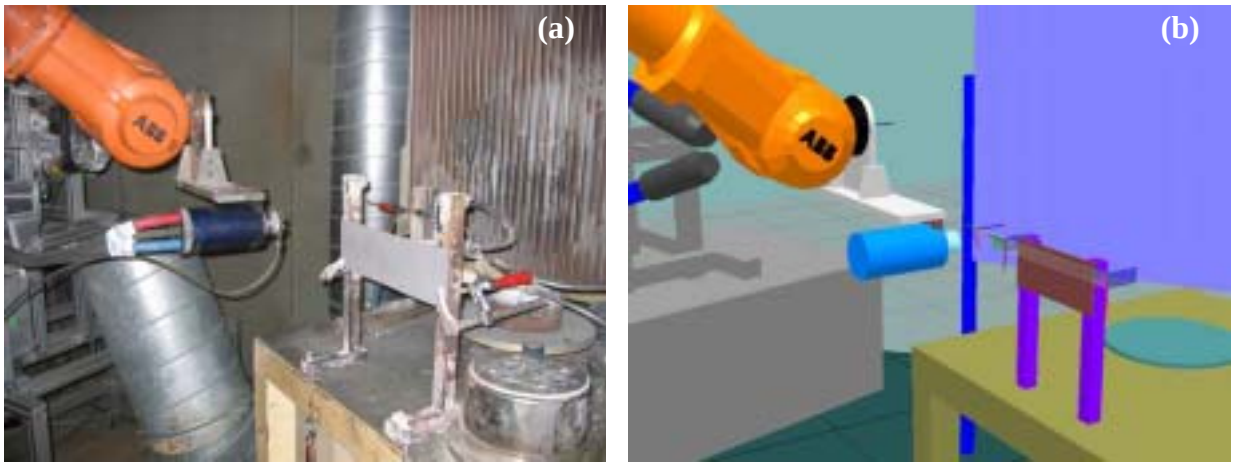


Figure 4.37 Projection sur une pièce carrée

(a) Cellule réelle sur site (b) Cellule virtuelle sous RobotStudio™

Pour étudier le cas de la forme circulaire sur plaque, un diamètre de 280 mm a été choisi. Afin de vérifier la trajectoire, les substrats ont été découpés aux dimensions nominales 350×350×5 mm^3. Les quatre coins de pièce ne sont pas sciés afin que la trajectoire exécutée par le robot puisse être clairement observée sur la pièce. La Figure 4.38a montre le montage de la pièce sur site, la Figure 4.38b s'affiche la trajectoire dessinée lors de la projection sous RobotStudio™.

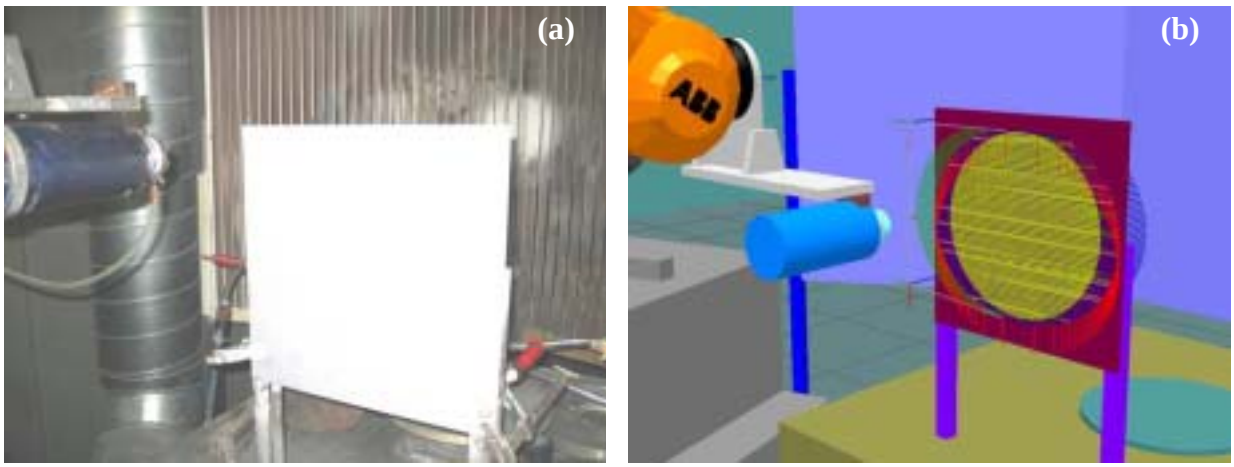


Figure 4.38 Projection sur la pièce circulaire

(a) Cellule réelle sur site (b) Cellule virtuelle sous RobotStudio™

Pour étudier le cas d'une pièce incurvée, un substrat en inox a été découpé aux dimensions nominales 700×100×1 mm^3. Ensuite, l'échantillon a été incurvé pour définir un profil (Figure 4.39).

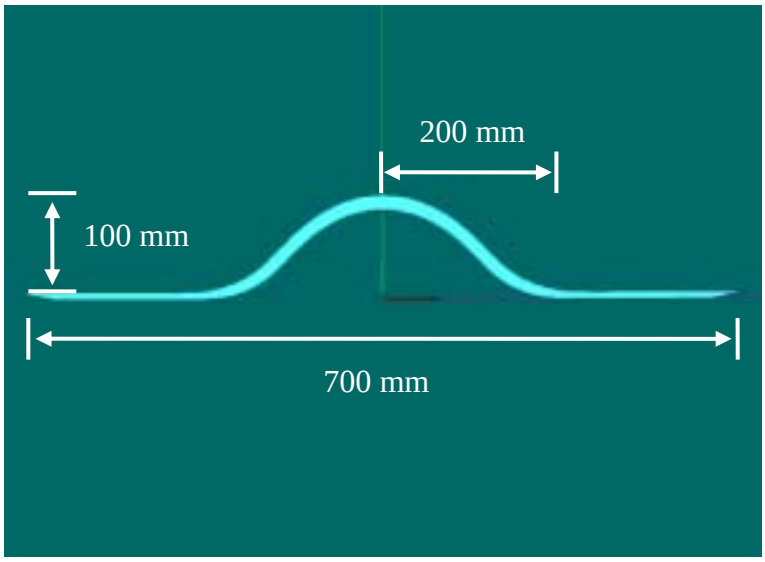


Figure 4.39 Profil conçu de la pièce incurvé

La pièce a été montée sur un support qui empêche sa déformation pendant la projection thermique. La Figure 4.40 présente la préparation sur site et la trajectoire conçue pour cette pièce sous RobotStudio™.

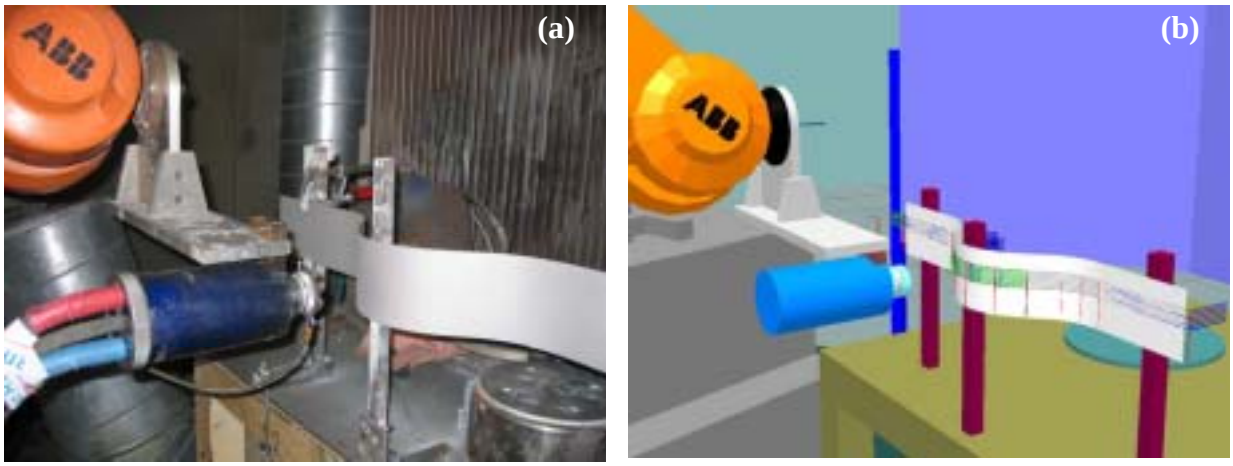


Figure 4.40 Programmation pour la pièce incurvée

(a) Cellule réelle sur site (b) Cellule virtuelle sous RobotStudio™

Avant projection thermique, les substrats ont été nettoyés a l'alcool et sablés manuellement avec une pression à 3 bar avec des particules de corindon (moyen 500 µm) selon la procédure classique.

Toutes les trajectoires robot adaptées les pièces correspondantes ont été générées par PathKit sous RobotStudio™ avec peu effort. Certains paramètres cinématiques (le pas de balayage, la distance de débordement par exemple) se remarquent dans la création de la trajectoire robot. La Figure 4.41 présente les trajectoires crées par PathKit sous RobotStudio™.

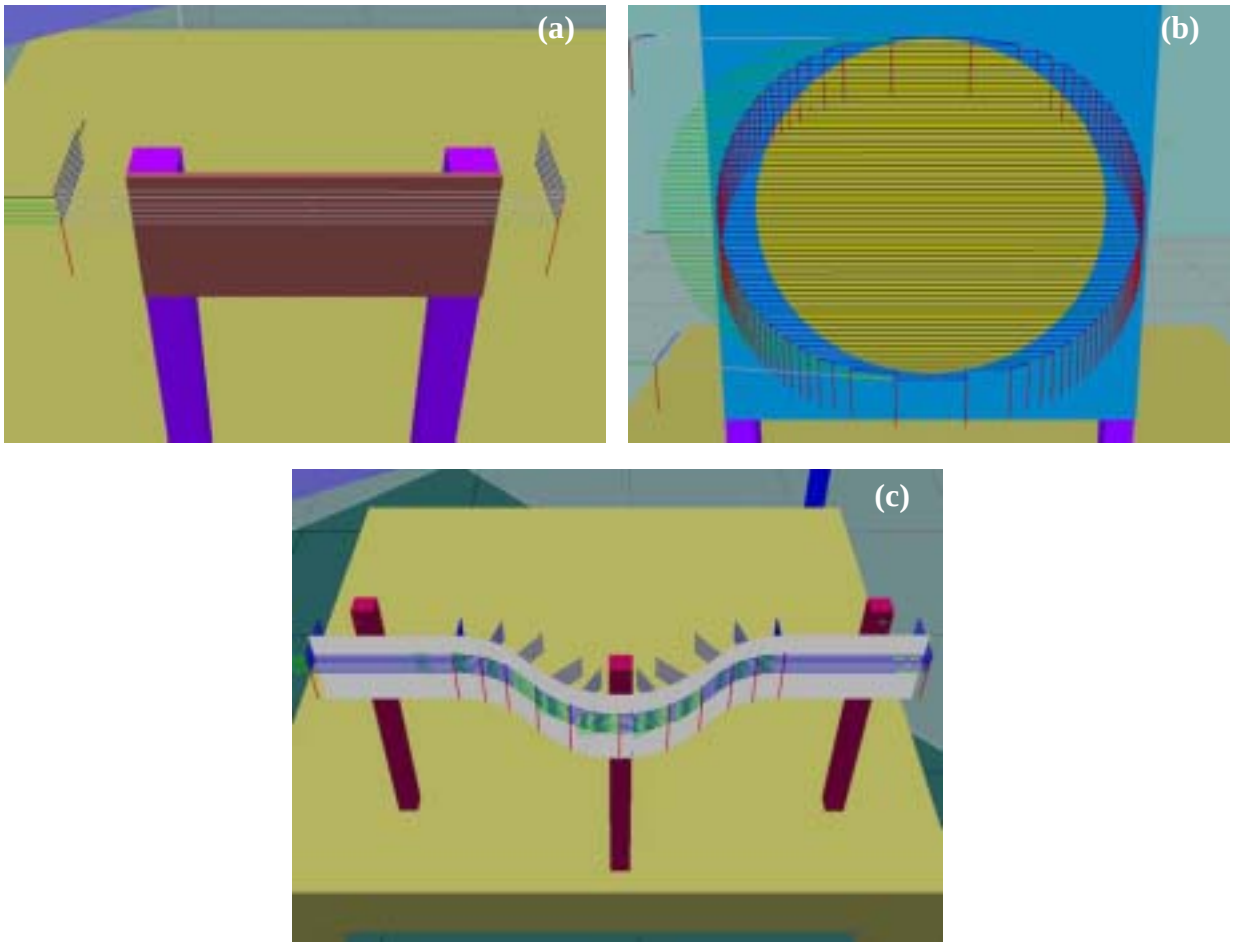


Figure 4.41 Exemple de trajectoires générées par PathKit sous RobotStudio™

(a) pièce carrée (b) pièce circulaire (c) pièce incurvée

4.4.1.5 Torche de projection et robot manipulateur

La torche de projection utilisée dans ces expériences est une torche à plasma d'arc soufflé atmosphérique de type F4 (Sulzer Metco AG). Cette torche est très couramment employée. Il s'agit d'une torche de 55 KW de puissance électrique maximale équipée d'une buse standard de 6 mm et d'un injecteur de poudre de 1.8 mm de diamètre interne qui est placé à 6 mm en bas de la sortie de la torche.

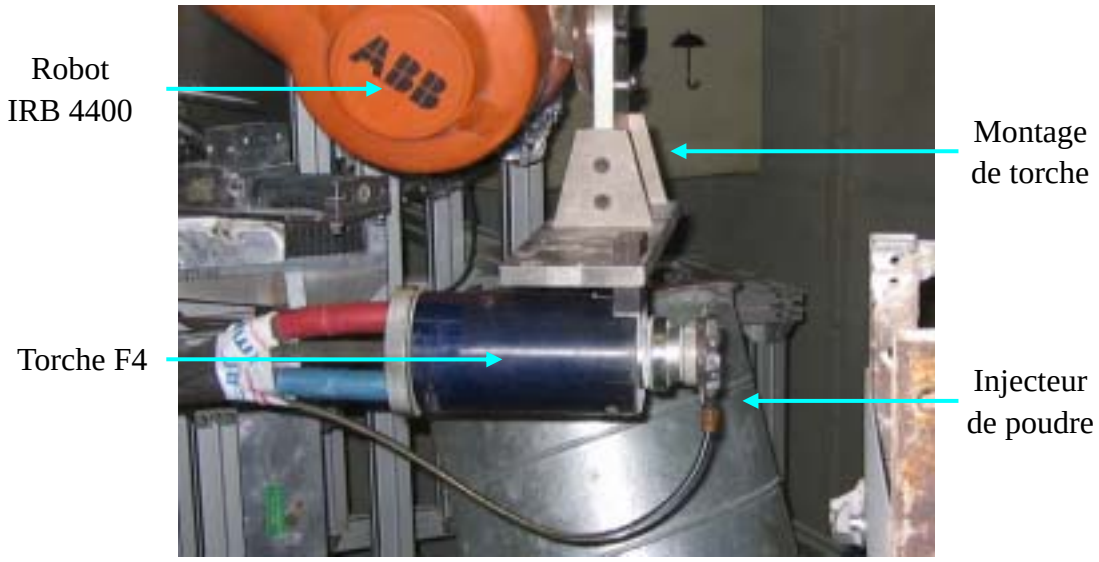


Figure 4.42 Torche utilisée

Un robot IRB 4400 M98 à six degrés de liberté (axes) a été employé pour manipuler la torche de projection dans ces expériences. Le robot a un rayon maximal d'action approximé de 1500 mm avec une précision linéaire de 0.8 mm à 1.3 mm [30].

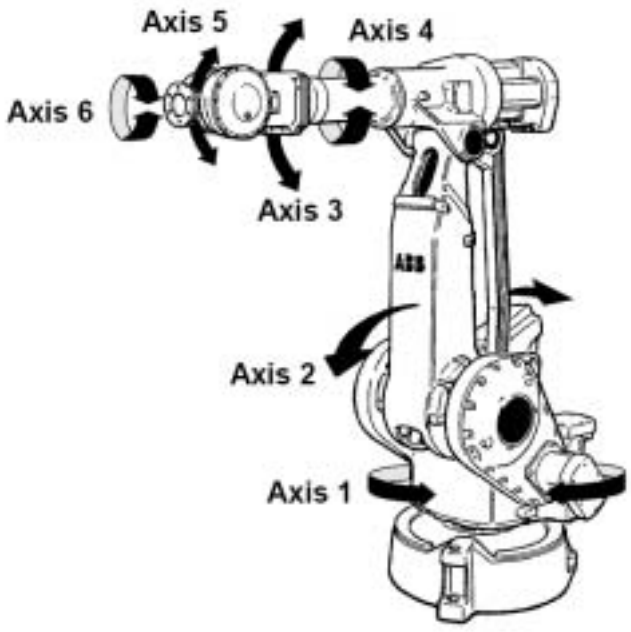


Figure 4.43 Robot IRB 4400 (ABB)

Le Tableau 4.2 liste la performance du robot utilisé.

Tableau 4.2 Performance du robot ABB IRB 4400

Répétablité unidirectionnelle	Précision de trajectoire linéaire	Répétablité de trajectoire linéaire	Résolution des axes	Vélocité CDO max.
0.07 mm	0.8 – 1.3 mm	0.25 – 0.4 mm	0.01 °	2200 mm/s

4.4.2 Simulation de construction du dépôt

4.4.2.1 Expérience de quantification de géométrie

Un dépôt est formé par la superposition de plusieurs couches résultant de plusieurs passes de la torche. Pour vérifier la fonction de simulation de la superposition du ProfileKit, il est nécessaire de disposer d'un modèle de profil du dépôt. Une expérience de quantification de géométrie d'un cordon doit donc être réalisée en premier.

Puisque la simulation d'épaisseur du dépôt est basée sur un modèle de distribution du

cordon pré déterminé, les conditions de projection thermique doivent être maintenues constantes, incluant le matériau projeté, le substrat à revêtir, les paramètres de projection, etc. Donc, cet essai de projection thermique est réalisé une pièce identique à la pièce carrée déjà préparé (expliquée précédente) avec les même conditions (cf. chapitre 4.4.1.1-4.4.1.5).

Pour bien mesurer le profil du dépôt, un cordon d'une certaine épaisseur est nécessaire. Dans cet essai, 10 passes de balayages ont été réalisées sur la plaque de test. Le profil du cordon est ensuite mesuré par un profilomètre mécanique.

4.4.2.2 Quantification de la géométrie du dépôt

Il existe différentes manières de quantifier le profil d'un dépôt : le palpeur mécanique et le profilomètre optique par exemple. Les mesures de la géométrie de pièces par contact à l'aide de machines de mesure en coordonnées sont couramment effectuées en raison de la bonne précision et l'applicabilité (grande variété possible des formes et des dimensions des pièces à mesurer) offertes par cette méthode.

Dans le cadre de cette étude, un profilomètre ETALON Derby® (produit par ETALON Switzerland) a été employé pour mesurer et quantifier le profil du cordon de dépôt.

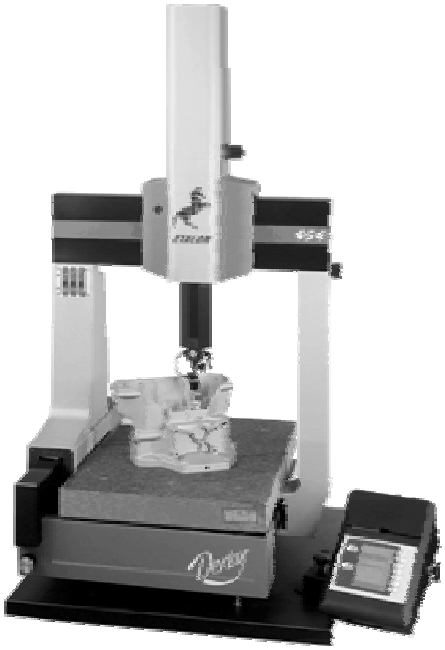


Figure 4.44 Machine à mesurer tridimensionnelle (Derby® ETALON)

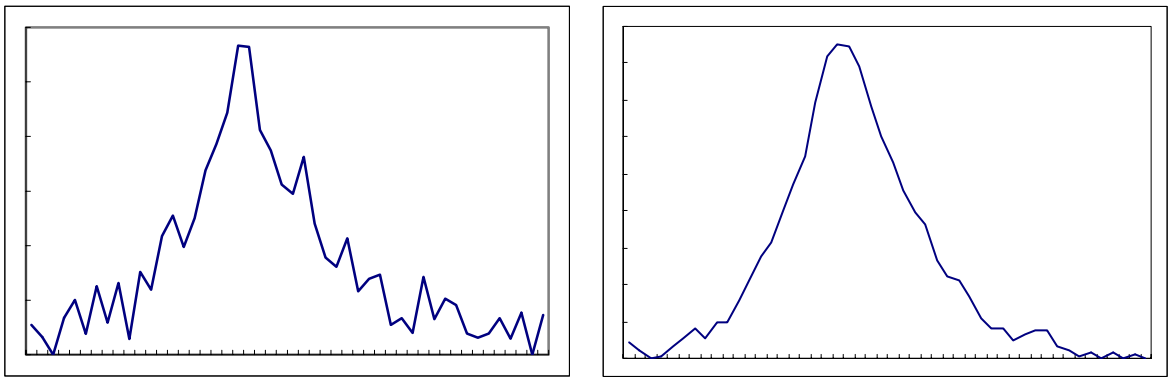
Le Tableau 4.3 liste les spécifications techniques de ce profilomètre tridimensionnel.

Tableau 4.3 Spécifications techniques

Répétablité	Précision volumétrique	Précision linéaire	Résolution	Plage d'affichage	Vitesse de mesure (max.)
0.004 mm	0.010 mm	0.005 mm	0.0625 µm	±9999.999mm	760 mm/sec

4.4.2.3 Simulation d'épaisseur du dépôt

Le profil du cordon obtenu après palpage est formé d'un ensemble de points donnant pour une abscisse y l'épaisseur e en millimètre. Le fichier de données est composé d'environ une cinquantaines de points. Pour simplifier l'utilisation des données, nous avons eu recours à un lissage par des fonctions mathématique. La Figure 4.45 présente un traitement de lissage des données du profil de cordon.



Données originales Données après lissage

Figure 4.45 Traitement de lissage des données

Après avoir traité les données originales, le profil du cordon est ensuite importé dans ProfileKit (Figure 4.46a). Dans cet essai, le pas de balayage (distance entre deux passes parallèles) est défini comme 5 mm. L'épaisseur simulée du dépôt final est 228 µm pour 10 cycles de balayages, 457 µm pour 20 balayages et 684 µm pour 30 balayages. La Figure 4.46b montre le profil importé dans ProfileKit et la simulation de superposition du dépôt final pour un pas de 5mm. Les valeurs d'épaisseur calculées seront vérifiées sur les pièces réelles.

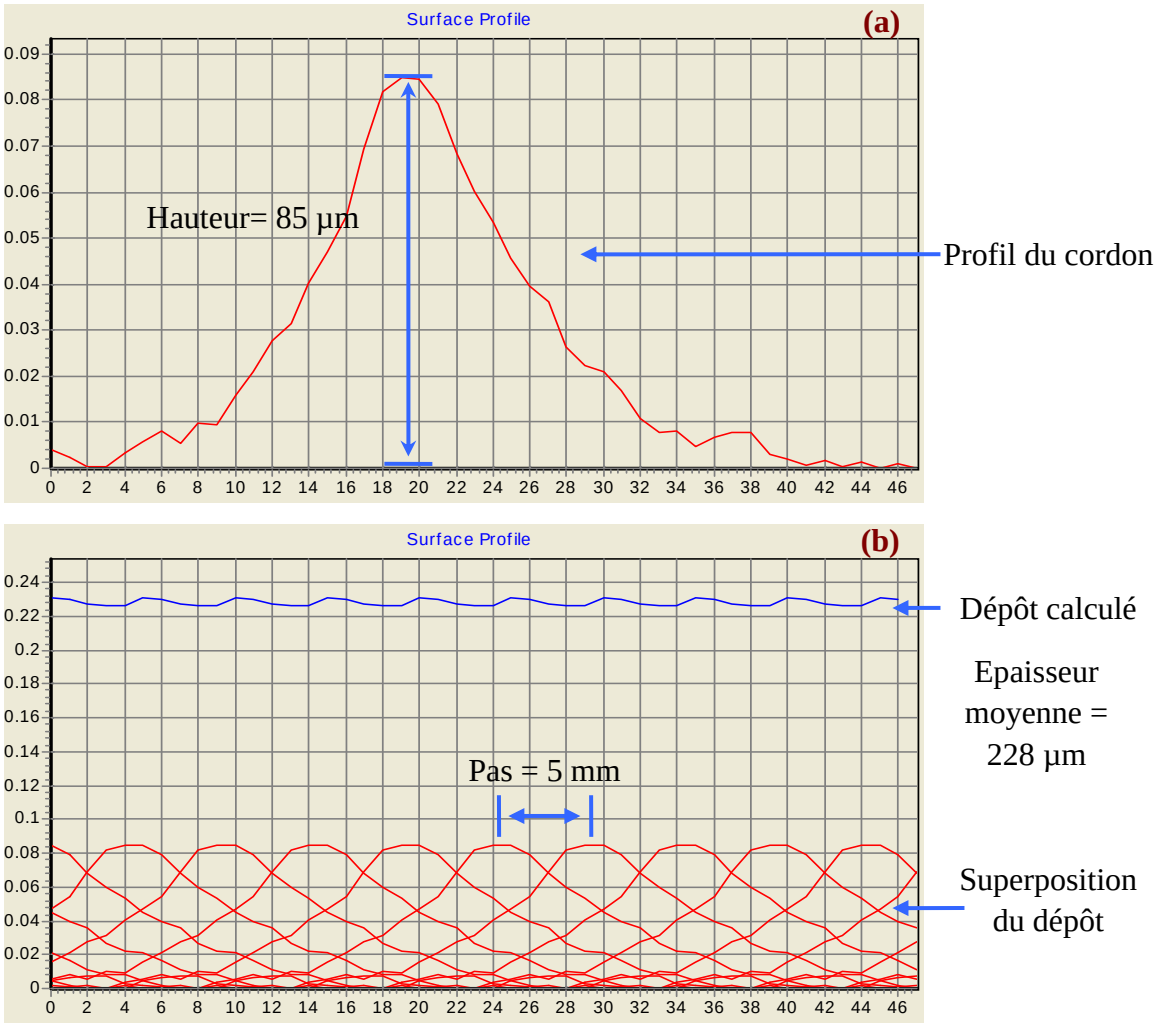


Figure 4.46 Simulation de construction du dépôt final

Pour vérifier l'épaisseur du dépôt, différents nombres de passes sont effectués sur différentes pièces. Le Tableau 4.4 liste les épaisseurs attendues correspondantes en fonction du nombre de passe.

Tableau 4.4 Epaisseurs calculées en fonction du nombre de passes

Nombre de passe	10	20	30
Epaisseur calculée (µm)	228	456	684

Dans les expériences de vérification, 30 passes de balayage ont été réalisés sur la pièce carrée. Pour la pièce circulaire, le nombre de passes a été défini à 10. En raison d'un

changement de la nature du matériau du substrat, l'épaisseur sur la pièce incurvée ne peut pas être anticipée de façon fiable.

4.4.3 Résultats expérimentaux

Après avoir élaboré les trajectoires sous RobotStudio™, les programmes robot sont ensuite exportés et chargés sur le vrai robot. Les projections thermiques sur différentes pièces sont accomplies sous le contrôle du MonitorKit. Les mouvements du robot (la trajectoire, la vitesse, etc.) concernant les essais évoqués précédemment sont enregistrés totalement par MonitorKit. Les résultats d'expériences seront présentés dans les paragraphes suivants.

4.4.3.1 Projection sur la pièce carrée

La Figure 4.47 montre la pièce après projection et la trajectoire réelle acquise par MonitorKit. Dans cette expérience, on peut observer que le robot accomplit parfaitement la trajectoire conçue et respecte bien la vitesse constante (500 mm/s) sur la pièce à revêtir.

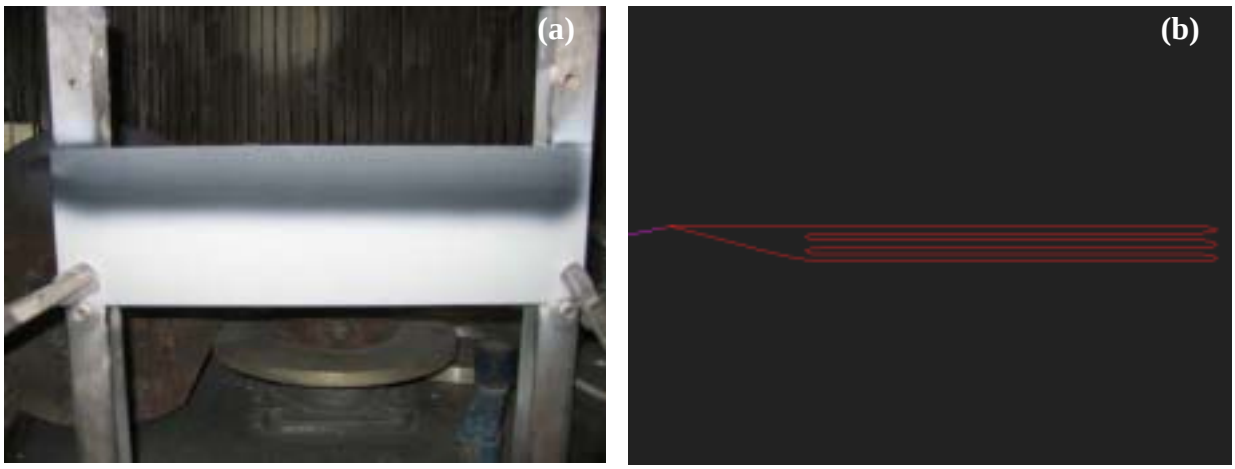


Figure 4.47 Résultats d'essai sur la pièce carrée

(a) Dépôt final sur la plaque (30 passes) (b) Trajectoire réelle sous MonitorKit

La Figure 4.48 présente la courbe de vitesse de CDO du robot qui représente la vitesse relative torche – substrat lors d'une seule passe sur le substrat.

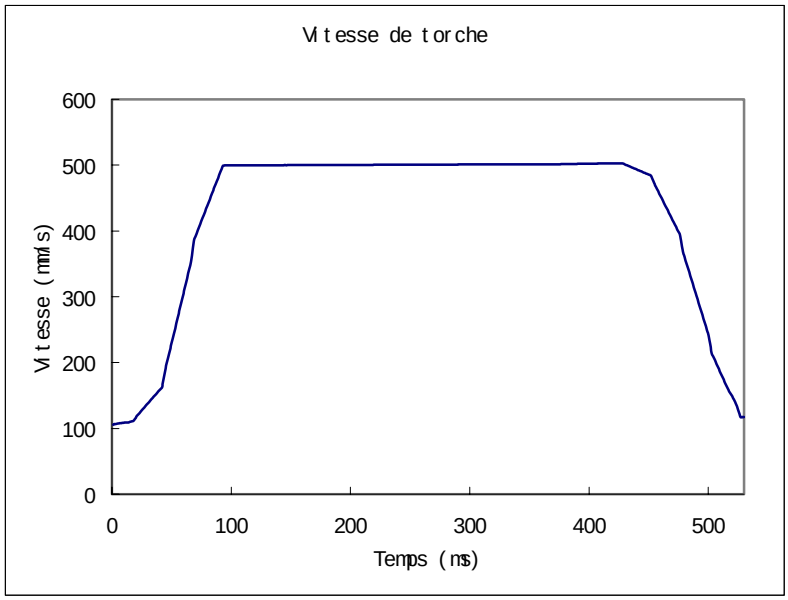


Figure 4.48 Vitesse de torche sur la pièce carrée lors d'une seule passe

En dehors des phases d'accélération et de décélération, lors d'une passe de balayage, le robot respecte bien la vitesse choisie sur le substrat à revêtir.

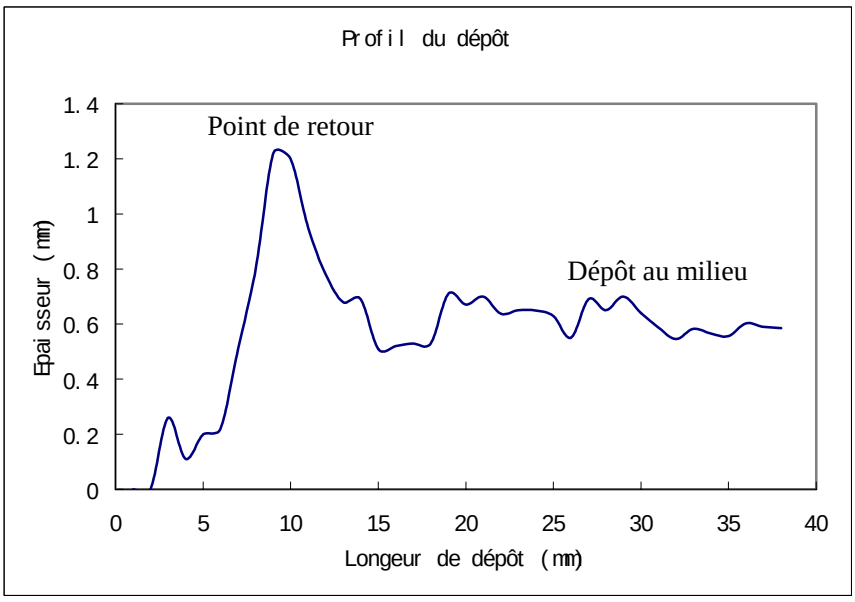


Figure 4.49 le profil du dépôt sur la pièce carrée (en partie)

Le profil du dépôt sur le substrat est présenté dans la Figure 4.49. L'épaisseur sur le point de retour lors de l'accélération / décélération du robot est plus élevée que celle au milieu du dépôt (la vitesse stable sur le substrat). Dans une opération pratique de projection thermique, un débordement de distance convenable est donc nécessaire afin d'obtenir un revêtement d'épaisseur constante.

L'épaisseur moyenne du dépôt mesurée en 10 points sur l'échantillon (zone centrale) est 574±18 µm (Tableau 4.5). L'épaisseur du dépôt est donc homogène sur toute la surface à revêtir du substrat.

Tableau 4.5 Epaisseurs mesurées par jauge micrométrique

Point	1	2	3	4	5	6	7	8	9	10	Moyenne
Epaisseur (µm)	588	546	583	556	603	590	585	582	530	581	574±18

En fort, l'épaisseur simulée par ProfileKit pour 30 passes est de 684 µm. La différence entre l'épaisseur réelle et l'épaisseur calculée est donc de 110 µm (environ 16.1 %). Les raisons qui peuvent expliquer cette déviation peuvent être :

- ü la tolérance de mesure,
- ü la variation du rendement de dépôt

La tolérance de mesure est bien entendu difficile à éviter. En ce qui concerne l'évaluation du rendement, le modèle de simulation du ProfileKit pourrait être optimisée en introduisant une compensation de taux de rendement de dépôt par rapport à l'augmentation de nombre de passe.

Dans cet essai simple, il est observé que le robot respecte la trajectoire que PathKit a créée et que l'épaisseur simulée par ProfileKit fournit une première approche raisonnable.

4.4.3.2 Projection sur la pièce circulaire

La Figure 4.50 présente le dépôt final (10 passes de projection) sur le substrat et la trajectoire réelle obtenue par MonitorKit. Il est observé que la couverture d'une forme circulaire est applicable et qu'une économie de poudre est aussi envisageable (Figure 4.50a). Le robot manipule la torche strictement par rapport à la trajectoire projetée par PathKit (Figure 4.50b).

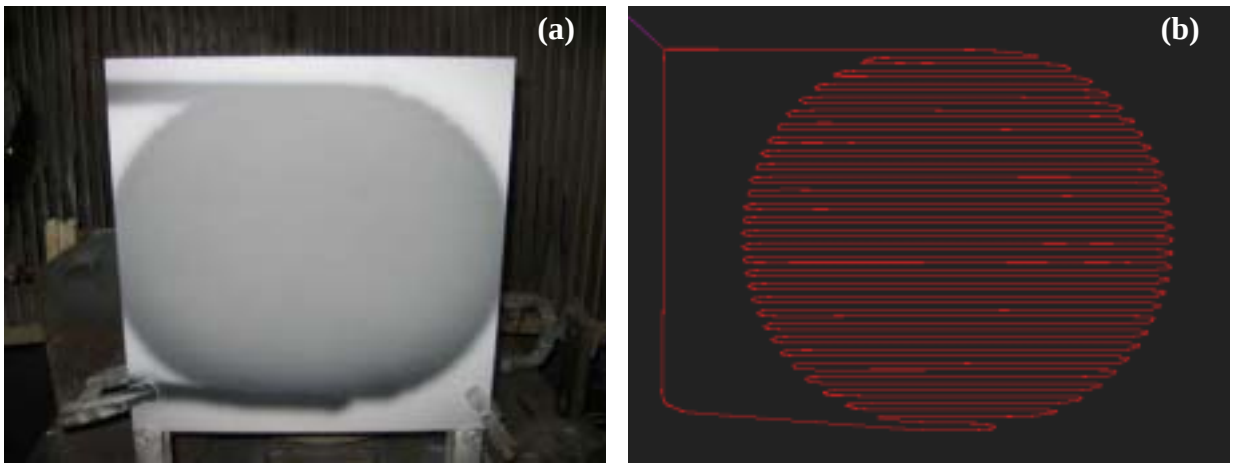


Figure 4.50 Résultats d'essai sur la pièce circulaire

(a) Dépôt final sur la plaque (10 passes) (b) Trajectoire réelle sous MonitorKit

La courbe de vitesse CDO du robot qui représente la vitesse relative torche – substrat lors d'une passe horizontale sur la pièce circulaire est présenté dans la Figure 4.51.

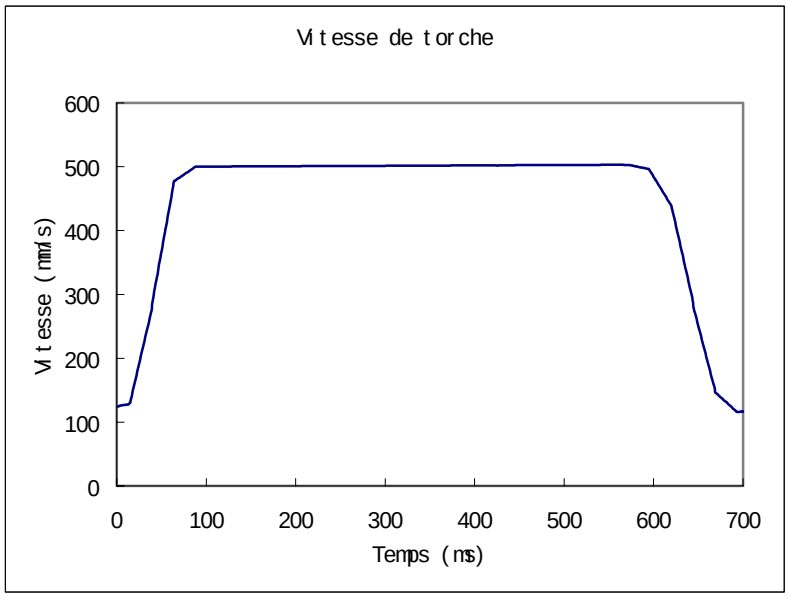


Figure 4.51 Vitesse de torche sur la pièce circulaire (une seule passe)

L'épaisseur du dépôt est mesurée de même façon que sur la pièce carrée. Les épaisseurs sur 10 points d'échantillon ont été mesurées par jauge micrométrique. Le Tableau 4.6 liste les valeurs d'épaisseur mesurées. L'épaisseur moyenne du dépôt est 206±8 µm. Il signifie que l'épaisseur du dépôt est homogène sur toute la surface du substrat.

Tableau 4.6 Epaisseurs mesurées par jauge micrométrique

Point	1	2	3	4	5	6	7	8	9	10	Moyenne
Epaisseur (µm)	205	198	199	214	194	207	221	215	211	197	206±8

L'épaisseur calculée par ProfileKit est 228 µm (10 passes). La déviation entre l'épaisseur réelle et l'épaisseur simulée est 22 µm (environ 9.6 %).

Cet essai prouve qu'on peut créer facilement à partir de PathKit les trajectoires de couverture d'une forme circulaire qui ne pouvaient pas être générées aisément auparavant.

4.4.3.3 Projection sur la pièce incurvée

Pour examiner plus clairement l'influence de la vitesse sur l'épaisseur du dépôt, le nombre de passes a été porté à 30. La Figure 4.52 montre les résultats de cet essai. La Figure 4.52a présente le dépôt final (30 passes de projection) sur la pièce incurvée ; la Figure 4.52b montre la trajectoire réelle obtenue par MonitorKit. Il est observé que le robot a du mal à respecter la trajectoire et la vitesse programmées, surtout dans les zones qui nécessitent une grande variation de plusieurs axes du robot (Figure 4.52c).

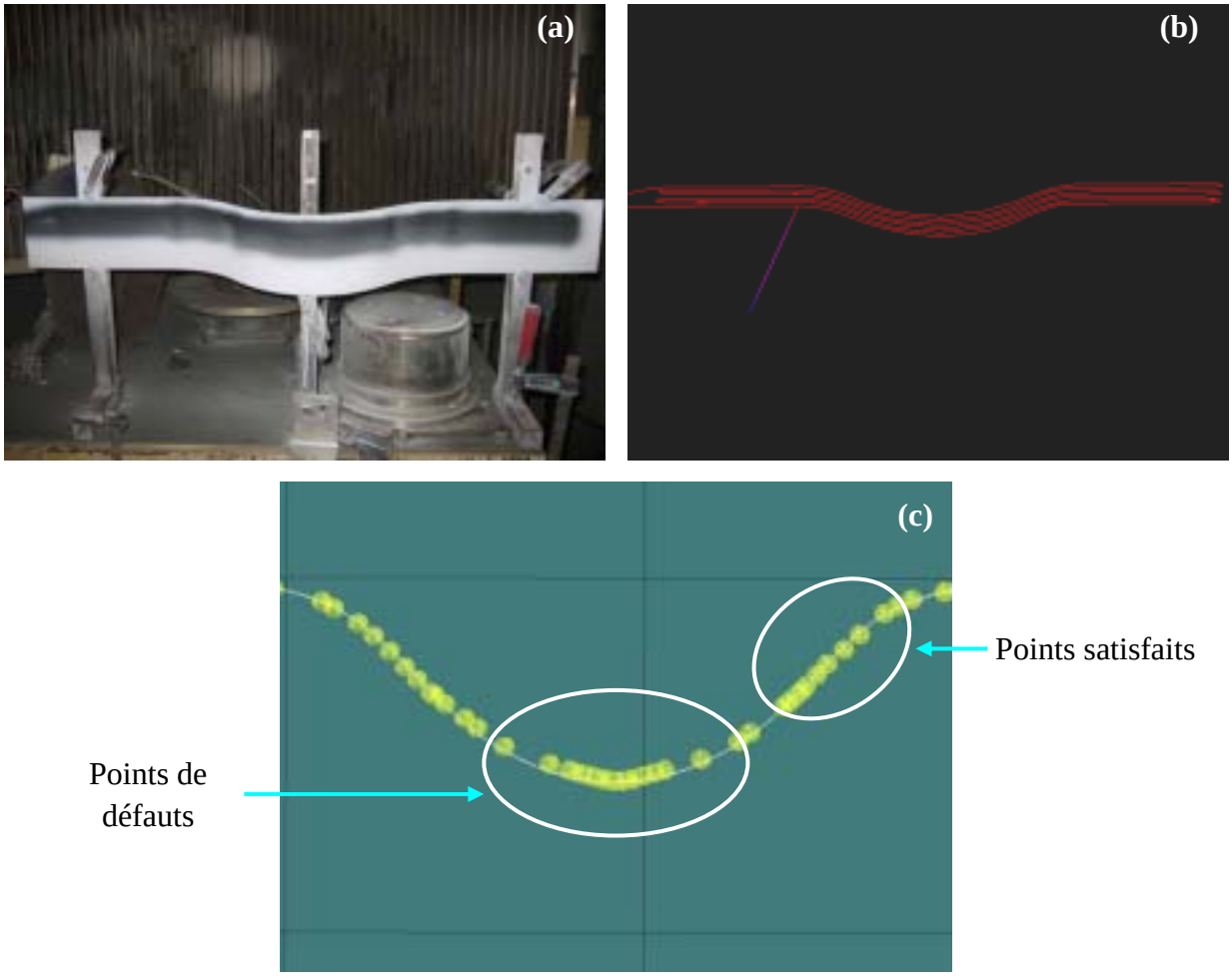
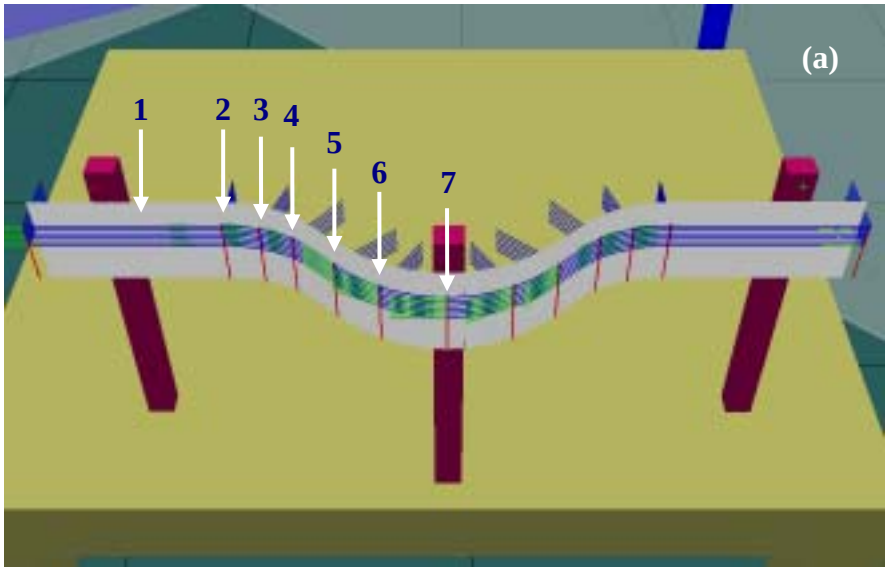
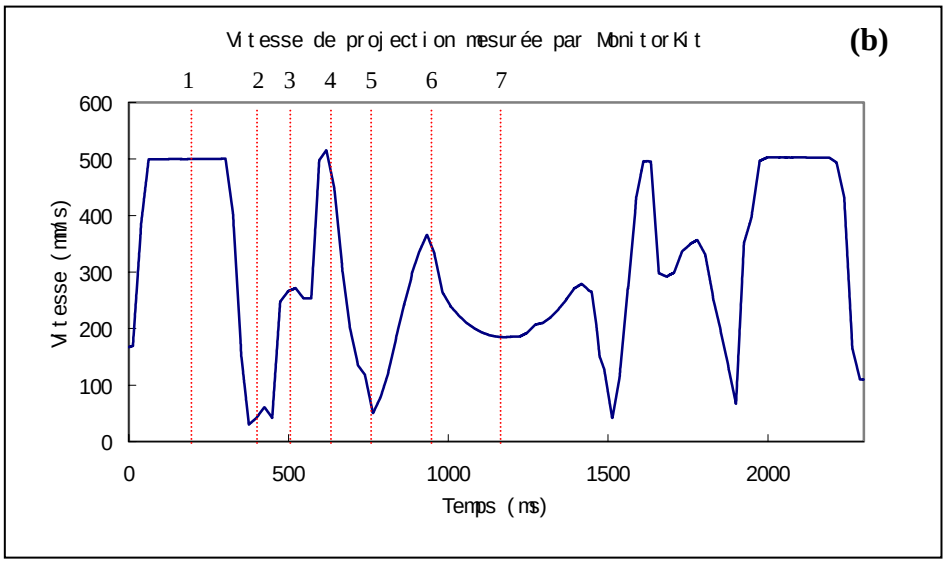


Figure 4.52 Résultats d'essai sur la pièce incurvée

(a) Dépôt final sur la plaque (b) Trajectoire réelle sous MonitorKit

(c) Comparaison des trajectoires





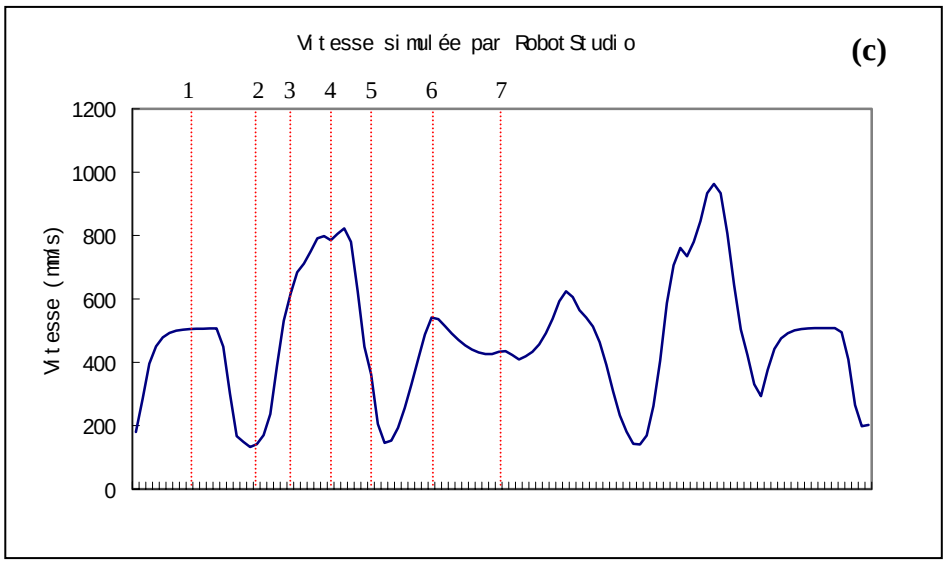


Figure 4.53 Positions des points de mesure et vitesse correspondante

Pour une vitesse choisie de 500 mm/s, la vitesse réelle mesurée varie entre 65 à 513 mm/s. Les Figure 4.53a et Figure 4.53b présentent les points de mesure d'épaisseur et la courbe de vitesse réelle du robot obtenue par MonitorKit.

Pour vérifier qualitativement le résultat obtenu par MonitorKit, une simulation de trajectoire a été fait sus RobotStudio™. Sur la Figure 4.53c, il est observé que la forme de la courbe de la vitesse est proche de celle de la vitesse réelle, mais que les valeurs de vitesse ne sont pas correctes : par exemple il existe un pic de vitesse de 1000 mm/s peu crédible et en autre, les valeurs sont en moyenne plus élevées d'environ 200 mm/s. Par conséquence, la simulation de vitesse proposée par RobotStudio™ ne peut pas être validée.

Le Tableau 4.7 montre évidement que l'épaisseur du dépôt et la vitesse de balayage sont fortement couplées, ainsi que l'ont montré plusieurs études [22, 23]. Les points de mesure sont indiqués dans la Figure 4.53.

Tableau 4.7 L'épaisseur mesurée sur les points correspondants les vitesses différentes

Point	1	2	3	4	5	6	7
Vitesse simulée (mm/s)	502	143	633	785	291	518	524
Vitesse réelle (mm/s)	504	65	268	494	68	378	188
Epaisseur (mm)	0.813	2.298	1.413	0.996	2.143	1.106	1.503

D'après Trifa [23], l'épaisseur du dépôt et la vitesse de projection ont une relation en fonction exponentielle. Les résultats de cet essai confirment cette conclusion. La Figure 4.54 décrit la relation en fonction exponentielle entre l'épaisseur du dépôt et la vitesse de torche par rapport aux points mesurés sur la pièce (Figure 4.53).

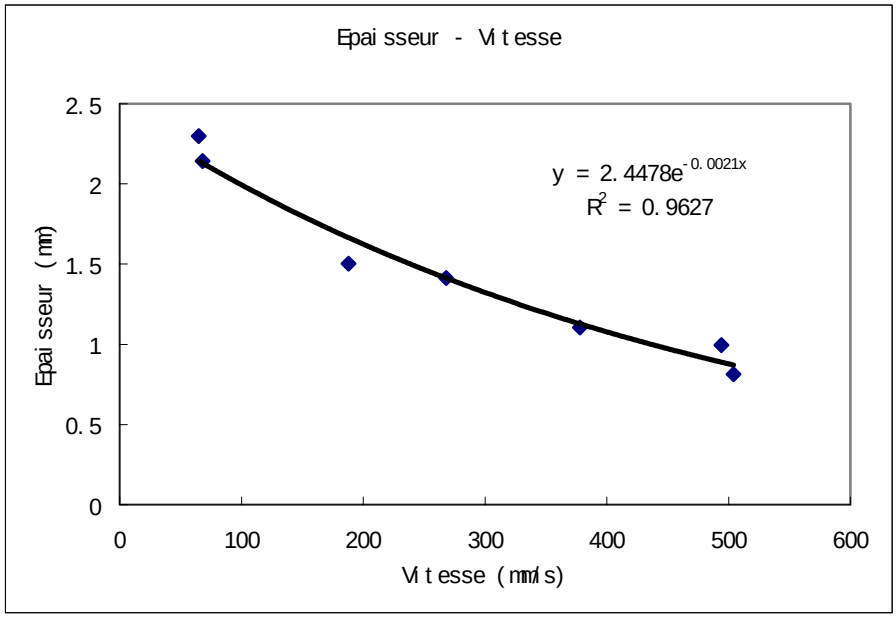


Figure 4.54 Relation épaisseur – vitesse

Selon les résultats de cet essai, le robot ne garantit pas la précision d'exécution de la vitesse sur une trajectoire compliquée avec combinaison de mouvements articulaires. Il est donc nécessaire de réaliser une optimisation de la trajectoire robot sur les pièces de forme complexe. Par exemple, un pré changement d'orientation de la torche peut éviter une variation très forte des axes du robot. Cet essai montre que le robot a atteint la performance maximale de certains moteurs d'axes. Donc la performance du robot doit être aussi considérée comme un facteur très important pour la conception d'une trajectoire en projection thermique.

4.4.3.4 Expérience sur la cinématique du robot

Comme nous l'avons vu précédemment, dans la plupart des essais, le robot peut parfaitement respecter la cinétique programmée. Cependant, dans certain cas de mouvements difficiles (les pièces irrégulières), on peut observer des différences importantes entre la réalité et la conception, surtout pour la vitesse du CDO.

En autre, les distances d'accélération / décélération doivent être pris en compte pendant la conception de la trajectoire pour la projection thermique. En raison des limités de performance

du robot, les temps d'accélération / décélération sont d'autant plus importants que la vitesse est grande et que la charge est lourde.

Une expérience sur la cinématique du robot a été réalisée afin de mesurer la distance d'accélération / décélération du robot en fonction des contraintes imposées. La trajectoire robot est définie comme une seule passe de 500 mm selon l'axe Y (Figure 4.55). Différentes vitesses ont été imposées sur la même trajectoire pour connaître le comportement du robot dans des cas différents.

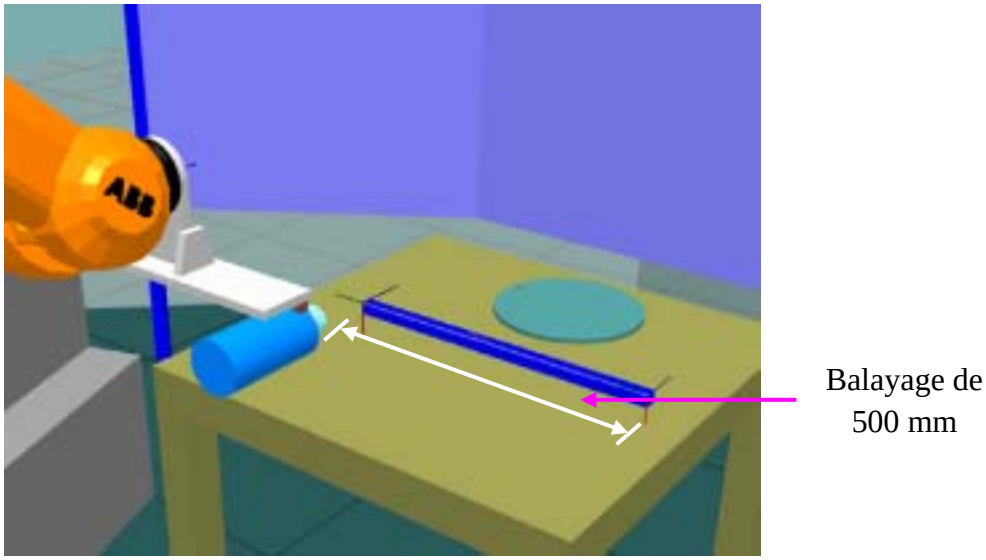


Figure 4.55 Configuration de l'expérience

La Figure 4.56 présente les résultats des expériences d'accélération pour les vitesses de 100 mm/s, 500 mm/s, 1000 mm/s et 1500 mm/s. L'axe X (distance) a été gradué en fonction logarithmique.

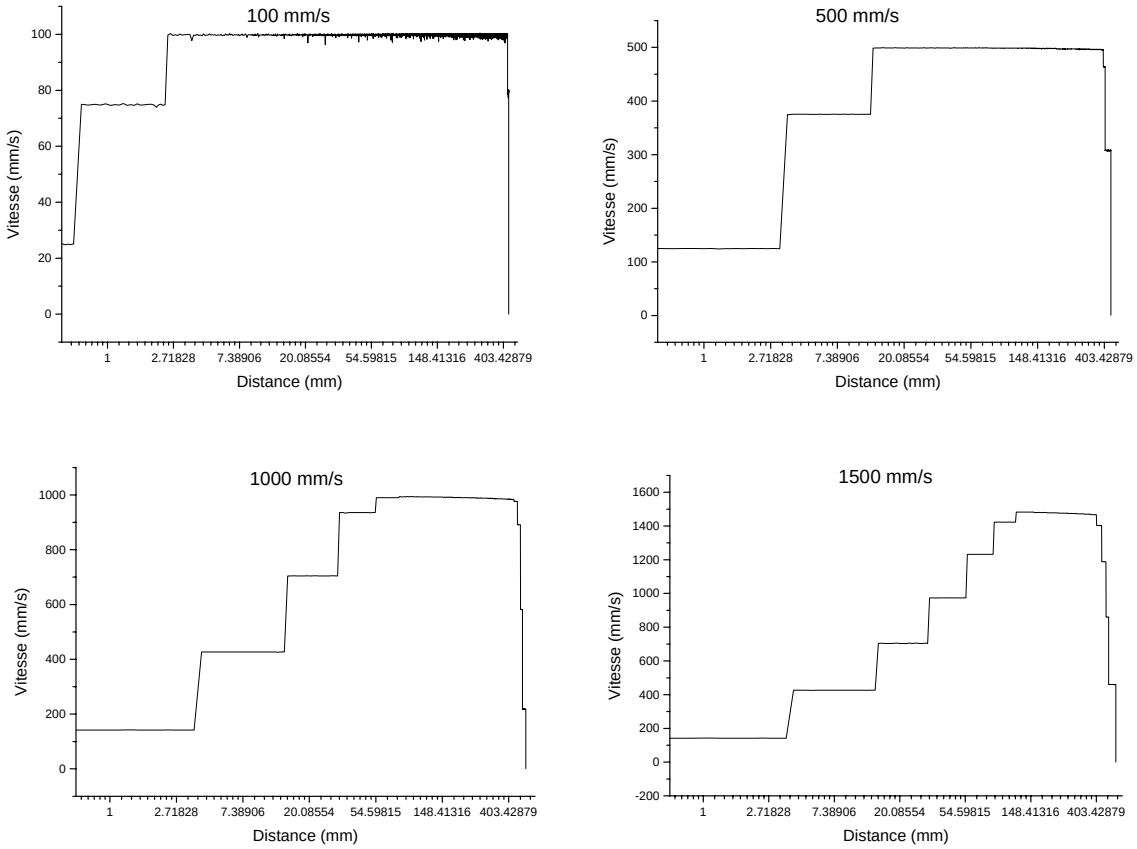


Figure 4.56 La distance d'accélération pour des vitesses différentes

A cause de la limitation de performance du robot et de la vitesse maximale des différents axes, le robot a besoin d'un certain temps et d'une certaine distance afin d'accélérer le CDO à la vitesse demandée. Le Tableau 4.8 présente les distances d'accélération mesurées nécessaires.

Tableau 4.8 Distance d'accélération correspondant aux différentes vitesses attendues

Vitesse (mm/s)	100	200	300	500	800	1000	1500
Distance d'accélération (mm)	2.34	4.58	7.49	12.46	34.65	40.12	68.59

Une vitesse plus grande nécessite une distance d'accélération plus grande. La Figure 4.57 illustre ce phénomène et donne la courbe de distance d'accélération – vitesse pour le robot ABB IRB 4400.

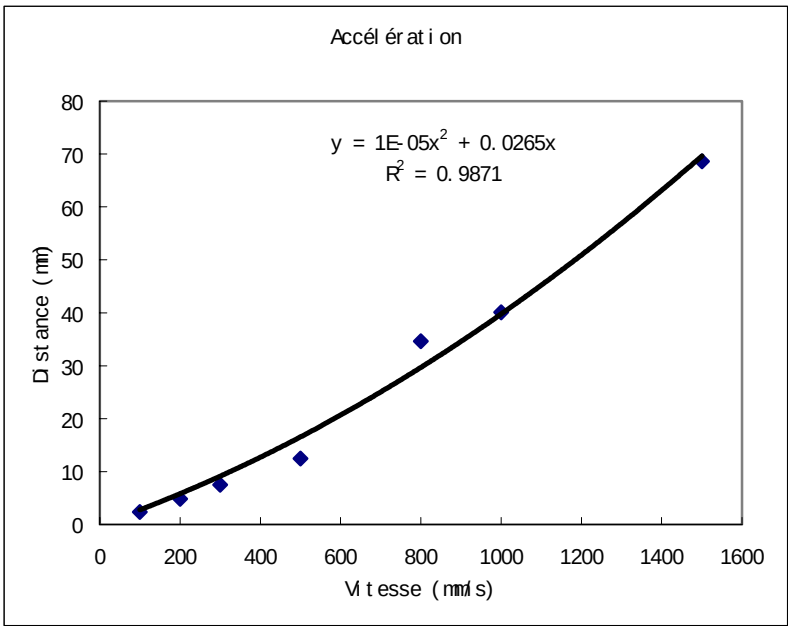


Figure 4.57 Distance d'accélération nécessaire pour atteindre une vitesse donnée

Il est donc nécessaire de prévoir une certaine distance de débordement dans l'opération de projection thermique afin d'éviter que la période d'accélération / décélération du robot se situe dans la zone à revêtir.

4.4.4 Conclusions d'expérimentation

Les conclusions à tirer des résultats d'expérimentation présentés sont les suivant :

- ü PathKit est un module fonctionnel du T.S.T. (Thermal Spray Toolkit) qui fournit des solutions de génération des trajectoires robot pour des pièces courantes (carrées, circulaires, incurvées, rotationnelles…) ainsi qu'un outil de création de courbes pour des pièces irrégulières.

  Bien que la trajectoire robot soit créée en stricte conformité à la forme du substrat, l'optimisation des paramètres cinématiques de la trajectoire est nécessaire. Car rn raison des limites de performance du robot, la vitesse ne peut pas rester constante pour tous les mouvements requis par le procédé. Les changements dans l'orientation de la

torche doivent être considérées de façon particulière.

- ProfileKit fournit la stratégie de choix des meilleurs paramètres opératoire de projection, la distance de pas entre deux passes par exemple. Pour bien utiliser cette fonction de l'outil, une base de données concernant les conditions et caractéristiques associées à chaque type de projection est nécessaire.

  ProfileKit fournit aussi un outil d'évaluation des caractéristiques de la surface (ou interface) qui ont une forte incidence sur la qualité du dépôt (homogénéité, adhésion …). Pour cela, une numérisation du profil de la surface est obligatoire. Ce profil de surface peut être déterminé à l'aide d'un palpeur mécanique ou d'un profilomètre optique par exemple. A partir de ces données numériques, les paramètres de surface peuvent alors être calculés.

- MonitorKit est un module qui contrôle la trajectoire et la vitesse du robot en temps réel. La trajectoire réelle peut être affichée graphiquement sur l'écran et les vitesses en chaque point peuvent être exportées dans un fichier pour une analyse ultérieure. En effet, une fluctuation de vitesse peut fortement influencer la qualité du dépôt (l'épaisseur du dépôt par exemple). Par conséquence, d'une manière générale, une optimisation est nécessaire après la génération de la trajectoire robot sur les bases géométriques de la pièce.

Références

[1] R. Bonnet. La projection thermique sur des formes complexes: simulation et étalonnage du procédé robotise. Thèse de doctorat. Université de Technologie de Belfort-Montbéliard, France, 2000.

[2] T. Chettibi, H.E. Lehtihet, M. Haddad, S. Hanchi, Minimum cost trajectory planning for industrial robots. European Journal of Mechanics A/Solids 23 (2004), p 703-715.

[3] A. Ismeal F. Vaz, Edite M.G.P. Fernandes, M. Paula S.F. Gomes, Robot trajectory planning with semi-infinite programming. European Journal of Operational Research 153 (2004), p 607-617.

[4] Suk-Hwan Suh, In-Kee Woo, Sung-Kee Noh, Development of an automatic trajectory planning system (ATPS) for spray painting robots. Proceedings of the 1991 IEEE International Conference on Robotics and Automation. Sacramento, California, avril 1991. (Pub) Piscataway, NJ : IEEE Computer Society Press, 1991

[5] J.K. Antonio, Optimal trajectory planning for spray coating. Proceedings of the International Conference on Robotics and Automation, pages 2570--2577, 1994. (Pub) Piscataway, NJ : IEEE Computer Society Press, 1994

[6] R. Ramabhadran, J.K. Antonio, Fast solution techniques for a class of optimal trajectory planning problems with applications to automated spray coating. IEEE Transactions on Robotics and Automation, Vol. 13, No. 4, August 1997.

[7] H. Hyotyniemi, Locally controlled optimization of spray painting robot trajectories. Proceeding of the IEEE International Workshop on Intelligent Motion Control (1990). (Pub) Piscataway, NJ : IEEE Service Center, [1990]

[8] P. Hertling, L. Hog, R. Larsen, J. Perram, H. Petersen, Task curve planning for painting robots – Part I: Process Modeling and Calibration. IEEE Transactions on Robotics and

Automation. Vol. 12, No. 2, April 1996.

[9] W. Persoon, H.V. Brussel, CAD-based robotic coating of highly curved surfaces. Proc. ISIR'93, Vol. 14, p. 611-618.

[10] Y. Itoh, M. Idesawa, T. Soma, A study on robot path planning from a solid model. Journal of Robotic Systems, 3(2), 191-203, 1986.

[11] A. Klein, CAD Based off-line programming of painting robots. Robotica (1987) Volume 5. P. 267 – 271

[12] P. Nylén, I. Frasson, A. Wretland. N. Martensson, Coating thickness prediction and robot trajectory generation of thermal sprayed coatings. Thermal Spray: Practical Solutions for Engineering Problems, Proceedings of the 9th National Thermal Spray Conference. 7-11 Octobre 1996, Cincinatti-Ohio. P 693 - 698. Edité par CC. Berndt, ASM International, Materials Park, OH., USA

[13] P. Chedmail. E. Dombre, P. Wenger, La CAO en robotique. Edition Hernies. Paris. 1998. p257. ISBN 2-86601-695-5

[14] P. Fauchais, J.F. Coudert, A. Vardelle, M. Vardelle, A. Grimaud, P. Roumilhac, State of the art for the understanding of the physical phenomena involved in plasma spraying at atmospheric pressure, Proc. 1st National Thermal Spray Conference, Thermal spray: advances in coatings technology, D.L. Houck (ed.), ASM International, Materials Park, OH., USA, p. 11, 1987.

[15] M.M. Fasching, L.E. Weiss, F.B. Prinz, Optimization of robotic trajectories for thermal spray shape deposition. Thermal Spray: International Advances in Coatings Technology, Proceedings of the International Thermal Spray Conference. 28 Mai - 5 Juin 1992; Orlando - USA. P. 221 - 226. Edité par C.C. Berndt, ASM International, Materials Park, OH., USA

[16] M.M. Fasching, L.E. Weiss. F.B. Prinz, Planning robotic trajectories for thermal spray shape deposition. Journal of Thermal Spray Technology, P 45 - 50. Volume 2 (N°l) Mars 1993.

[17] H. Figueroa, O. Diaz, Thermal spray modeling of flat surfaces and cylinders. Proceedings of the Fourth Thermal Spray Conference, Pittsburgh - USA. 4 - 10 Mai 1991. P. 549 - 556. Edité par C.C. Berndt, ASM International, Materials Park, OH., USA

[18] S. Cirolini, J.H. Harding, G. Jacucci, Computer simulation of plasma-sprayed coatings, I. Coating Deposition Model. Surface and Coatings Technology, 48 (1991). P. 137 – 145

[19] T. Vivekanandhan, R.A. Kashani, W.A. Johnson, Intelligente plasma torch trajectory generation for thermal spraying of parts with Complex Geometry. Thermal Spray: International Advances in Coatings Technology, Proceedings of the International Thermal Spray Conference. 28 Mai - 5 Juin 1992: Orlando - USA. P. 217 - 220. Edité par C.C. Berndt, ASM International, Materials Park, OH., USA

[20] J. G. Goedjen, R. A. Miller, W. J. Brindley, G. W. Leissler, A simulation technique for predicting thickness of thermal sprayed coatings. NASA Technical Memorandum 106939, Army Research Laboratory Technical Report ARL-TR-762. Juin 1993. P. 1 - 15.

[21] P. Nylén, I. Fransson, A. Wretland, N. Martensson. Coating thickness prediction and robot trajectory generation of thermal spray coatings. Proceedings of the Thermal Spray: Practical Solutions for Engineering Problems. (Ed.) C.C. Berndt. (Pub) ASM International, Materials Park, OH, USA, 1996, p 693-698.

[22] S. Guessasma. Optimisation et contrôle de procédés de projection thermique par coopération de méthode d'intelligence artificielle. Thèse de doctorat. Université de Technologie de Belfort-Montbéliard, France, 2003.

[23] F.I. Trifa. Modèle de dépôt pour la simulation, la conception et la réalisation de revêtements élaborés par projection thermique. Thèse de doctorat. Université de Technologie de Belfort-Montbéliard, France, 2004.

[24] J.L. Caenen, Contribution à l'identification de paramètres géométriques et non géométriques d'un modèle de robot. Application à l'amélioration de la précision en statique.

Thèse de Doctorat, Université de Valenciennes et du Hainault-Cambrésis, Janvier 1993.

[25] ABB, Fonctionnement de base, 3HAB 5825-1 / RobotWare 2.0, ABB Automation Technology Products AB Robotics, SE-721 68 Västerås, Sweden, 2001.

[26] ABB, Product Specification, 3HAC 13335-1/M2000/BaseWare OS 4.0/Rev. 0, Industrial Controller IRC S4Cplus, ABB Automation Technology Products AB Robotics, SE-721 68 Västerås, Sweden, 2001.

[27] ABB, User's Guide, 3HAC 7793-1, For BaseWare OS 4.0. ABB Automation Technology Products AB Robotics, SE-721 68 Västerås, Sweden, 2001.

[28] ABB, RAP Service Specification, 3HAC 7697-1. ABB Automation Technology Products AB Robotics, SE-721 68 Västerås, Sweden, 2001.

[29] NI, Spécification de NI DAQPad-6020E.
http://sine.ni.com/nips/cds/view/p/lang/fr/nid/11922

[30] ABB, Product Specification IRB 4400 3HAC 9117-1 / Rev 2 M2000, ABB Automation Technology Products AB Robotics, SE-721 68 Västerås, Sweden, 2001.

Chapitre 5

Conclusions générales et perspectives

L'objectif de cette étude était d'étudier les techniques à mettre en œuvre pour permettre la réalisation de revêtement sur les échantillons de forme complexe dans de bonnes conditions. Si la projection sur des formes géométriques simples est bien maîtrisée depuis de nombreuses années, il n'en est pas de même pour les pièces plus complexes. De longues phases de mise au point de la trajectoire robot sont généralement nécessaires, ce qui induit un coût d'élaboration plus élevé. L'intérêt du sujet se situe donc au niveau de l'introduction de nouveaux concepts pour résoudre la difficulté représentée par la forme complexe de la pièce à revêtir. Depuis quelques années, un logiciel CAO de programmation robotique « RobotStudio™ » a été mis sur le marché par la société ABB. Ce logiciel permet une programmation hors-ligne et la simulation des mouvements du robot. Toutefois, sa fonctionnalité de génération de trajectoires n'est pas adaptée pour la projection thermique, car ce procédé nécessite en particulier un contrôle parfait de la vitesse de l'outil.

Dans ce contexte, un travail de réflexion a été d'abord nécessaire pour définir la meilleure orientation de travail possible. A partir de cette réflexion, une extension logicielle dédiée à la projection thermique a été développée, testée et validée. Cette extension logicielle, dénommée « PathKit » permet :

- ü la génération de trajectoires sur des formes de géométrie complexe (pièce carré, pièce circulaire, pièce incurvée, pièce en rotation, etc.),
- ü de modifier et optimiser les paramètres cinématiques des trajectoires robot (la vitesse, l'orientation, la distance de pas) ainsi que l'orientation de l'outil en chaque point,
- ü de créer des courbes convenables sur des formes complexes pour une optimisation de la trajectoire.

Les expérimentations ont montré que les trajectoires générées par PathKit étaient parfaitement réalisées au niveau géométrique sur site réel. En revanche, une divergence importante a été constatée entre les trajectoires programmées et réelles du robot, principalement au niveau de la vitesse d'execution. Cette divergence est logique dans la mesure où le poids de la torche et des câbles, le fardeau dynamique du robot, joue un rôle dans son comportement. En outre, un robot ne peut pas privilégier la vitesse par rapport à la

précision de la trajectoire. Dans certains cas, les vitesses et accélérations demandées au robot dépassent les capacités maximales de celui-ci. Sans la possibilité d'obtenir un comportement dynamique réaliste du robot réel, il est difficile d'assurer la qualité de production ainsi que le contrôle des épaisseurs de dépôt et du comportement thermique de la pièce à revêtir.

Par conséquence, il apparaît primordial de pouvoir contrôler le mouvement du robot en temps réel ainsi que de vérifier la vitesse de l'outils reste dans la capacité du robot car la maîtrise de la vitesse de l'outil constitue le point clef pour l'obtention d'un dépôt homogène en épaisseur. C'est pourquoi un logiciel spécifique « MonitorKit », a été développé pour :

- ü communiquer avec le robot afin d'obtenir et de pouvoir vérifier les coordonnées de l'outil en temps réel,
- ü joindre tous les points obtenus pour constituer la trajectoire réelle du robot pendant le processus de projection thermique,
- ü obtenir les vitesses en chaque point de la trajectoire,
- ü comparer la trajectoire réelle avec la trajectoire programmée, ainsi que les vitesses d'exécution.

En projection thermique, la capacité de simulation de l'épaisseur déposée en tout point de la pièce revêt une importance particulière. En effet, la prédiction de l'épaisseur permet de valider ou non la trajectoire (position, orientation, vitesse de déplacement) définie par la programmation hors-ligne. Si une bonne correspondance est établie entre les résultats de la simulation et l'épaisseur réelle mesurée sur le substrat, alors la simulation permettra de générer des programmes robot efficaces.

Par ailleurs, les relations entre les caractéristiques de la surface et des interfaces et la qualité du dépôt ont été constatées par de nombreux chercheurs. Les résultats montrent que les paramètres de rugosité et la dimension fractale des profils représentent un indicateur effectif de qualité (l'homogénéité, l'adhésion du dépôt par exemple). C'est pourquoi, un autre module logiciel, dénommé « ProfileKit », a été développé afin de fournir :

- ü la simulation de la superposition des cordons ainsi que l'épaisseur de dépôt final,

- ü la stratégie de choix des paramètres opératoires, le pas entre deux passes par exemple,
- ü l'évaluation des paramètres de la surface / interface du dépôt par rapport au profil correspondant.

Une fois ces trois directions de travail défrichées, une combinaison entre les trois modules logiciels a été établie et une trousse à outil logicielle dédié à la projection thermique dénommé « Thermal Spray Toolkit » a été développée. Cette trousse, composée des modules « PathKit », « ProfileKit » et « MonitorKit » associés à la base RobotStudio™, fournit une solution logicielle complète pour résoudre les problèmes de projection thermique pour des pièces complexes.

Il est à nota que, certaines technologies de développement logiciel ont été nécessaires pour le développement du « Thermal Spray Toolkit », et notamment :

- ü RPC (Remote Procedure Call) – obtention des données du robot,
- ü OpenGL – visualisation de la trajectoire robot sur l'écran,
- ü Add-in – extension logiciel pour RobotStudio™,
- ü Compteur de haute précision – obtention des intervalles de temps,
- ü MultiThread – réalisation de la multifonction logicielle simultanée.

Afin de garantir la qualité du logiciel, les méthodes du génie logiciel ont été employées dans touts les développements logiciels du T.S.T.

En termes de perspectives, l'un des axes principaux d'amélioration du résultat de cette étude pourrait être d'étendre les fonctionnalités du « PathKit » pour l'adapter à l'ensemble des formes rencontrées en projection thermique, y compris les pièces hétérotypiques. En outre, le post-processeur de la trajectoire pourrait être amélioré afin d'optimiser la trajectoire robot. Une procédure de détection ou de modification de toute variation singulière de l'angle d'orientation de la torche doit être établie afin d'éviter tout mouvement inopiné de reconfiguration des axes du robot afin de protéger le montage de la torche. De même, les opératoires mobilisant des axes externes doivent être considérés et introduites dans la procédure.

Une autre perspective possible pour le module « ProfileKit » serait de considérer de

plusieurs manières la caractérisation du profil de la surface pour évaluer globalement ses propriétés et prendre en compte la relation entre les profils du dépôt et sa qualité.

Enfin, le module « MonitorKit » pourrait être intégré à RobotStudio™ comme une extension logicielle. Pour ce faire, une technique de communication (avec le robot) sous l'environnement de RobotStudio™ doit être étudiée.

D'une façon générale, notre travail sur l'élaboration d'une application CAO-Robotique pour l'aide à la programmation hors-ligne en projection thermique montre tout l'intérêt que l'on peut porter à ce type de solution. Le gain en rapidité et en qualité de programmation est indéniable ; le temps de production est considérablement réduit de même que le nombre de test de qualification.

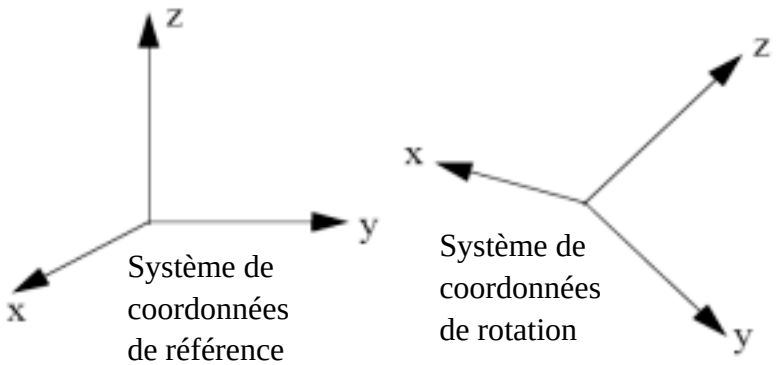
Chapitre 6

Annexes

Annexe 1 Quaternion

Dans système du robot, le paramètre d'***orientation*** est généralement décrit sous forme d'un quaternion qui comporte donc quatre éléments : *q1*, *q2*, *q3* et *q4*.

L'orientation d'un système de coordonnées (par exemple celui d'un outil) peut être décrite par une matrice de rotation qui décrit la direction des axes du système de coordonnées par rapport à un système de référence.



Les axes du système de coordonnées rotatif (x, y, z) sont des vecteurs qui peuvent être exprimés dans le système de coordonnées de référence comme suit :

$$x \mid /x_1, x_2, x_3 0$$
$$y \mid /y_1, y_2, y_3 0$$
$$z \mid /z_1, z_2, z_3 0$$

Ceci signifie que la composante x du vecteur x dans le système de coordonnées de référence sera x_1, la composante y sera x_2, etc.

Ces trois vecteurs peuvent être réunis dans une matrice, une matrice de rotation, où chaque vecteur forme une des colonnes :

$$\begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{pmatrix}$$

Un quaternion est tout simplement une façon plus concise de décrire cette matrice de rotation ; les quaternions sont calculés en se basant sur les éléments de la matrice de rotation :

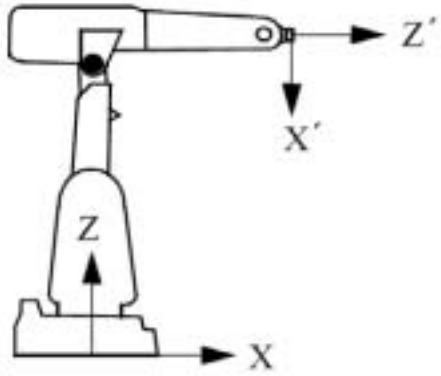
$$q1 = \frac{\sqrt{x_1 + y_2 + z_3 + 1}}{2}$$

$$q2 = \frac{\sqrt{x_1 - y_2 - z_3 + 1}}{2} \qquad sign(q2) = sign(y_3 - z_2)$$

$$q3 = \frac{\sqrt{y_2 - x_1 - z_3 + 1}}{2} \qquad sign(q3) = sign(z_1 - x_3)$$

$$q4 = \frac{\sqrt{z_3 - x_1 - y_2 + 1}}{2} \qquad sign(q4) = sign(x_2 - y_1)$$

Un outil est orienté de façon à ce que son axe **Z** soit tourné directement vers l'avant (dans la même direction que l'axe **X** du système de coordonnées de base). L'axe **Y** de l'outil correspond à l'axe **Y** du système de coordonnées de base. Comment l'orientation de l'outil est elle définie dans les données de position ?



L'orientation de l'outil dans la position programmée correspond normalement au système de coordonnées de la pièce utilisée. Si les coordonnées de base sont celles du système de coordonnées universelles, l'orientation correspond au système de coordonnées de base et la relation entre les axes sera comme suit :

$$x' = -z \quad /0,0,-1,0$$
$$y' = y \quad /0,1,0,0$$
$$z' = x \quad /1,0,0,0$$

Ce qui correspond à la matrice de rotation suivante :

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 4\,1 & 0 & 0 \end{pmatrix}$$

La matrice de rotation fournit un quaternion correspondant :

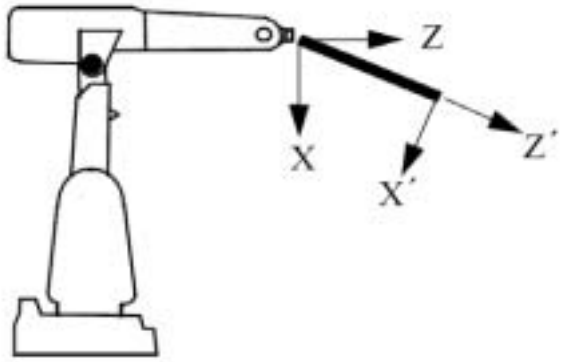
$$q1 = \frac{\sqrt{0\,2\,1\,2\,0\,2\,1}}{2} = \frac{\sqrt{2}}{2} = 0.707$$

$$q2 = \frac{\sqrt{0\,4\,1\,4\,0\,2\,1}}{2} = 0$$

$$q3 = \frac{\sqrt{1\,4\,0\,4\,0\,2\,1}}{2} = \frac{\sqrt{2}}{2} = 0.707 \qquad sign(q3) = sign(1\,2\,1) = 2$$

$$q4 = \frac{\sqrt{0\,4\,0\,4\,1\,2\,1}}{2} = 0$$

Si la direction de l'outil fait l'objet d'une rotation de 30° autour des axes **X** et **Z** par rapport au système de coordonnées du poignet, l'orientation de l'outil est définie dans les données de l'outil par la relation suivante entre les axes :



La suit :

$$x' = /\cos 30\forall, 0, 4 \sin 30\forall 0$$
$$y' = /0, 1, 0 0$$
$$z' = /\sin 30\forall, 0, \cos 30\forall 0$$

Ce qui correspond à la matrice de rotation suivant :

$$\begin{pmatrix} \cos 30\forall & 0 & \sin 30\forall \\ 0 & 1 & 0 \\ 4 \sin 30\forall & 0 & \cos 30\forall \end{pmatrix}$$

La matrice de rotation fournit un quaternion correspondant :

$$q1 := \frac{\sqrt{\cos(30°) + 2 \cdot 1 + 2 \cdot \cos(30°) + 2 \cdot 1}}{2} = 0.965926$$

$$q2 := \frac{\sqrt{\cos(30°) - 4 \cdot 1 - 4 \cdot \cos(30°) + 2 \cdot 1}}{2} = 0$$

$$q3 := \frac{\sqrt{1 - 4 \cdot \cos(30°) - 4 \cdot \cos(30°) + 2 \cdot 1}}{2} = 0.258819 \qquad sign(q3) := sign(\sin(30°) + 2 \cdot \sin(30°)) = 2$$

$$q4 := \frac{\sqrt{\cos(30°) - 4 \cdot \cos(30°) - 4 \cdot 1 + 2 \cdot 1}}{2} = 0$$

Dans les autre cas de développement, il faut transformer le quaternion à la matrice de rotation pour calculer l'orientation dans un système de référence. La transformation s'établit comme suit :

$$R := \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2 \cdot (q_1 \cdot q_2 - q_0 \cdot q_3) & 2 \cdot (q_1 \cdot q_3 + q_0 \cdot q_2) \\ 2 \cdot (q_1 \cdot q_2 + q_0 \cdot q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2 \cdot (q_2 \cdot q_3 - q_0 \cdot q_1) \\ 2 \cdot (q_1 \cdot q_3 - q_0 \cdot q_2) & 2 \cdot (q_2 \cdot q_3 + q_0 \cdot q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix}$$

Annexe 2 RobotStudio™ API

Using the RobotStudio API

The RobotStudio objects are structured in a hierarchical fashion, with the Application object as the topmost object. The view of this hierarchical structure is referred to as the Object Model. The Object Model shows you which object provides access to the next level of objects. All methods and properties use SI units

Objects and Collections

Objects are the fundamental building block of Visual Basic; nearly everything you do in Visual Basic involves modifying objects. Most of the elements of RobotStudio are exposed to Visual Basic as objects. For example:

- ü Graphical objects such as solids and lines are objects.
- ü Organizational structures, such as assemblies, parts, and entities are objects.
- ü Simulation tools such as paths, joints and simulations are objects.
- ü The station and the RobotStudio application are considered as objects too.

A Collection is an object that contains several other objects, usually of the same type; for example, all the assembly objects in a station are contained in a single collection object, the assemblies' collection. Using properties and methods, you can modify a single object or an entire collection of objects.

Properties

A property is an attribute of an object or an aspect of its behavior. For example, properties of a path include its name, its velocity and its position. To change the characteristics of an object,

you change the values of its properties.

To set the value of a property, follow the reference to an object with a period, the property name, an equal sign, and the new property value. The following example changes the name of the first target to StartTarget and then moves it in the x-direction 1000 mm:

```
Application.Workspace.Station(1).Targets(1).Name = “StartTarget:1”
newTarget.Transform.x = 1
```

You can return information about an object by returning the value of one of its properties. The following example returns the name of the first target in the path Path1:

```
firstName = Paths(“Path1”).TargetRefs(1).Target.Name
```

Methods

A method is an action that an object can perform. For example, just as a target can be created by the “Target” button, the station object has a CreateTarget method. A path has an Insert method to add a target into the path. The following example creates a target, a path and inserts the target in the path. Then it moves the target to z=1000 mm.

```
Dim newTarget As Target
Dim newPath As Path
Set newTarget = Application.Workspace.Stations(1).Targets.Add()
Set newPath = Application.Workspace.Stations(1).Paths.Add
Call newPath.Insert(newTarget)
newTarget.Transform.z = 1
```

In most cases, methods are actions and properties are qualities. Using a method causes something to happen to an object, while using a property returns information about the object or it causes a quality about the object to change.

Procedures and Events

Visual Basic code is divided in Procedures and a procedure run only when a specific Event occurs. Objects can have several different events. For example the Application object has an event, the StationBeforeSave event. This event is triggered before a station is saved in RobotStudio. A procedure for the event looks like this:

```
Private Sub Application_StationBeforeSave(StationName,bCancel)
End Sub
```

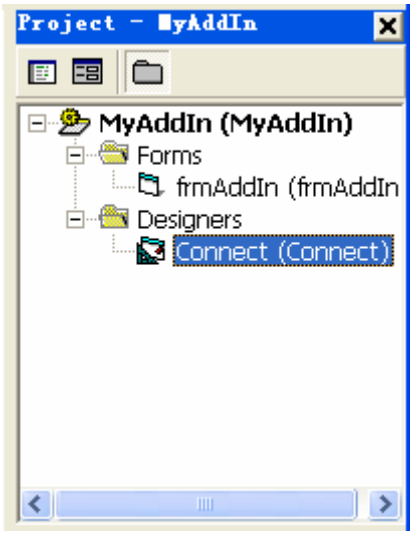
The procedure above is called when the event “Application_StationBeforeSave” occurs.

Annexe 3 Développement d'Add-in

Writing an Add-in consists of three steps, first start an Add-in project in VB, then setting up the AddInDesigner and finally filling in the code.

Part 1: Starting the Add-in Project

Start up Visual Basic, the "New Project" window is displayed; select the Add-in wizard in VB. The wizard creates two forms, a form and an ActiveX-designer. As seen in the Project window.



Project window.

Some sample code has already been filled in. This code is based on making an Add-in for Visual Basic. To make an Add-in for RobotStudio you will change some of the information and delete most of the code in the following two steps.

Part 2: Setting up the AddInDesigner

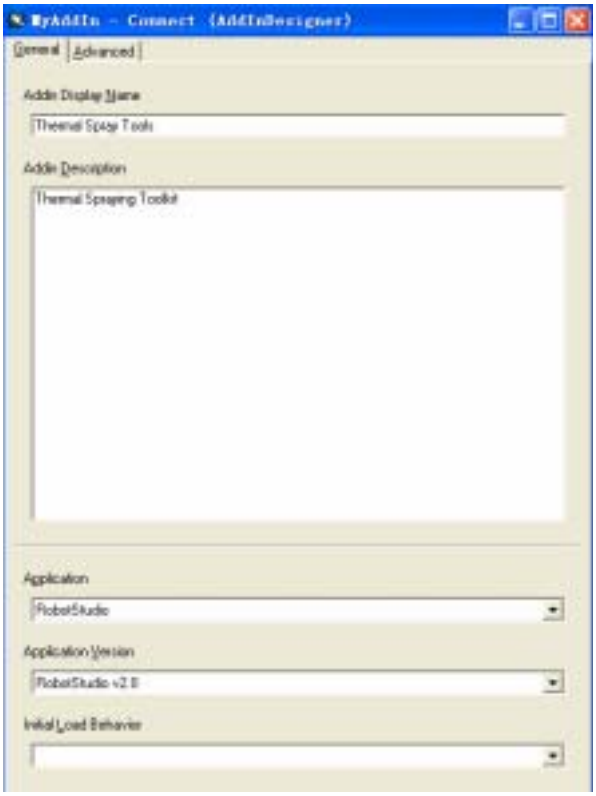
1. Double-click on the Connect ActiveX-designer in the Project Window. The AddInDesigner window is displayed.
2. In the Add-in Display Name field, fill in the name that will be displayed in RobotStudio

Add-in manager when you add the Add-in to RobotStudio.

3. If you want to add more descriptive text of what your application is supposed to do, enter it in the Add-in Description text field. This descriptive text is only displayed if you use the following VBA-API call:

```
Dim Description as String
Description = Application.COMAddins("Your Addin Display Name").Description
```

4. In the Application field select the application you are writing the Add-in for. Applications that are registered in the system as being able to handle COMAddin functionality are displayed in the Application field. In this case, select RobotStudio.
5. In the Application Version field, select the correct version for the chosen application
6. In the Initial Load Behavior field, select the default value, Startup.



Add-in Designer

Part 3: Filling in the Code

1. In this example only the ActiveX-designer is useful. Therefore, delete the frmAddin form and all the code in the Connect designer. Check that only an empty Connect is remaining.

The Connect designer is the connection to the main application. Our addin and RobotStudio communicates with each other using something called an interface. The Connect designer implements the IDTExtensibility2 interface. This is the standard interface defined by Microsoft for applications and addins to communicate. The IDTExtensibility2 interface has been implemented as five events. They are:

OnConnection( )
OnStartupComplete( )
OnBeginShutdown( )
OnDisconnection( )
OnAddinsUpdate( )

2. Add a reference in VB to the RobotStudio type library by selecting references from the project menu. Browse to “ABB RobotStudio 1.0 Type Library” make sure it’s checked and click OK.

3. Make sure you have the code window for the Connect-designer as your active window. Do these by right-click on the Connect designer, and in the context menu select View Code.

4. Declare some useful variables, like the following examples:

```
Option Explicit
Public WithEvents TheApp As RobotStudio.Application
```

Note: Make sure you use the WithEvents keyword as this makes it possible to receive events from objects.

5. Write the code for the eventhandlers, starting with OnConnection. In the code window, select AddinInstance in the object dropdown menu. Then select OnConnection from the procedure dropdown menu. The OnConnection event is the first event that should arrive to the Add-in. This occurs when the Add-in is loaded. There are some arguments that the main application adds to this event, described in the table below.

Name:	Description:
Application	Application is a reference to the application that activated the Add-in. This is useful if there is more then one active instance of the Application to communicate with.
ConnectMode	ConnectMode tells how the Add-in was loaded. The different modes are ext_cm_Startup, ext_cm_AfterStartup. The ext_cm_Startup mode is set if the addin is activated when the main application is starting up. This is followed by the OnStartupComplete event when the main application has initialized completely. If the addin is activated using the add-in manager, the ConnectMode will be ext_cm_AfterStartup.In this case the main application is already up and running and not followed by a OnStartupComplete.
AddInInst	AddInInst is a reference to the COMAddIn object that refers to this Add-in in the COMAddIns Collection for the host application.
Custom	Custom is an array of Variants that RobotStudio could use for application specific data. At the time of writing it is unused and is just an empty array.

Therefore the OnConnection event will look as follows:

```
Private Sub AddinInstance_OnConnection(ByVal Application As Object, _
        ByVal ConnectMode As AddInDesignerObjects.ext_ConnectMode, _
        ByVal AddInInst As Object, custom() As Variant)
    Set TheApp = Application
```

```
    If ConnectMode = ext_cm_AfterStartup Then
    'Robotstudio is up and running - add stuff to menus and such here
    'Call EnsureMenuStuffAdded
End Sub
```

6. If the Add-in is activated during startup of Robotstudio, the OnStartupComplete event will fire when Robotstudio is fully initialized. The OnStartupComplete event could be implemented as follows:

```
Private Sub AddinInstance_OnStartupComplete(custom() as Variant)
    'If we get here, the OnConnection will never have been called with
    'ext_cm_AfterStartUp so if menus should be added, do it here.
    'Call EnsureMenuStuffAdded
End Sub
```

7. Add your code to the Add-in that responds to the user selecting something in Robotstudio. The event that fires when the selection changes is called SelectionChanged and belongs to the Application object. The code looks as follows:

```
Private Sub TheApp_SelectionChanged( )
    MsgBox "SelectionChanged"
End Sub
```

8. Add functionality to handle the shutdown. This is done by adding the following code to the OnBeginShutdown event.

```
Private Sub AddinInstance_OnBeginShutdown( )
'No need to clean up the menus since RobotStudio does this for you
'automatically on shutdown. Note: Menus and toolbars added through
'the RSE are not persisted.
```

End Sub

9. When the OnDisconnection event occurs, prepare to deactivate and unload.

The OnDisconnection event has two arguments, RemoveMode and custom. Make sure that any and all references to objects from RobotStudio are explicitly released. Failure to do so will most likely result in a crash on shutdown.

Name:	Description:
RemoveMode	RemoveMode tells us the reason for disconnection. If RemoveMode is ext_dm_HostShutdown then the Add-in is disconnected because the main application is shutting down. Otherwise RemoveMode is ext_dm_UserClosed in which case the user has selected to unload the Add-in via the add-in manager.
Custom	Custom is an optional array of variants that is not used by RobotStudio.

The implementation of OnDisconnect could look as follows:

```
Private Sub AddinInstance_OnDisconnection(RemoveMode As ext_DisconnectMode, _custom() As Variant)
    Set TheApp = Nothing
    ‘Note: If you called with ext_dm_UserClosed you will want to
    ‘delete stuff added to the RSE here since this is only done
    ‘automatically on shutdown.
    ‘Call RemoveMenuStuff
End Sub
```

10. From the File menu select make MyAddin.dll to compile our code to an ActiveX dll. Compiling the AddIn will also register it for use with RobotStudio. If the AddIn is moved, it will have to be registered again. This can be done using RegSvr32 from the command line.

11. To try out your AddIn, just fire up RobotStudio and select something in the browser. A message box should pop up saying “Selection Changed!”

Below follows the code needed for this example Add-in:

```
‘AddIn Sample. Listen to selection changes.

Option Explicit
Public WithEvents TheApp As RobotStudio.Application
Private Sub AddinInstance_OnConnection(ByVal Application As Object, _
        ByVal ConnectMode As AddInDesignerObjects.ext_ConnectMode, _
        ByVal AddInInst As Object, custom() As Variant)

    Set TheApp = Application
End Sub

Private Sub AddinInstance_OnDisconnection(ByVal RemoveMode As
AddInDesignerObjects.ext_DisconnectMode, custom() As Variant)

    Set TheApp = Nothing
End Sub

Private Sub TheApp_SelectionChanged()
    MsgBox "Selection Changed!"
```

Annexe 4 Thread

Thread Priority

Every thread has a base-priority level determined by the thread's priority value and the priority class of its process. The operating system uses the base-priority level of all executable threads to determine which thread gets the next slice of CPU time. Threads are scheduled in a round-robin fashion at each priority level, and only when there are no executable threads at a higher level will scheduling of threads at a lower level take place.

Threads are scheduled to run based on their scheduling priority. Each thread is assigned a scheduling priority. The priority levels range from 0 (lowest priority) to 31 (highest priority). Only the zero-page thread can have a priority of zero. (The zero-page thread is a system thread responsible for zeroing any free pages when there are no other threads that need to run.)

The system treats all threads with the same priority as equal. The system assigns time slices in a round-robin fashion to all threads with the highest priority. If none of these threads are ready to run, the system assigns time slices in a round-robin fashion to all threads with the next highest priority. If a higher-priority thread becomes available to run, the system ceases to execute the lower-priority thread (without allowing it to finish using its time slice), and assigns a full time slice to the higher-priority thread. For more information, see Context Switches.

The priority of each thread is determined by the following criteria:

- The priority class of its process
- The priority level of the thread within the priority class of its process
- The priority class and priority level are combined to form the base priority of a thread.

Each process belongs to one of the following priority classes (Tableau 6.1):

Tableau 6.1 Thread priority class

Priority Classes Names	**Priority Classes**	**Level**
Idle	IDLE_PRIORITY_CLASS	4

Normal	NORMAL_PRIORITY_CLASS	9/7
High	HIGH_PRIORITY_CLASS	13
Real-time	REALTIME_PRIORITY_CLASS	24

The following are priority levels within each priority class (Tableau 6.2):

Tableau 6.2 Thread priority level

priority levels Names	**Priority Levels**	**Description**
Idle	THREAD_PRIORITY_IDLE	Thread priority fix on 16
Lowest	THREAD_PRIORITY_LOWEST	Process priority level -2
Below Normal	THREAD_PRIORITY_BELOW_NORMAL	Process priority level -1
Normal	THREAD_PRIORITY_NORMAL	Same as Process priority level
Above Normal	THREAD_PRIORITY_ABOVE_NORMAL	Process priority level +1
Highest	THREAD_PRIORITY_HIGHEST	Process priority level +2
Real-time	THREAD_PRIORITY_TIME_CRITICAL	Thread priority fix on 31

The priority level of a thread is determined by both the priority class of its process and its priority level. The priority class and priority level are combined to form the base priority of each thread.

Synchronizing Multithreading

Each thread has its own stack and its own copy of the CPU registers. Other resources, such as files, static data, and heap memory, are shared by all threads in the process. Threads using these common resources must be synchronized. Win32 provides several ways to synchronize resources, including semaphores, critical sections, events, and mutexes.

Synchronizing resource access between threads is a common problem when writing multithreaded applications. Having two or more threads simultaneously access the same data can lead to undesirable and unpredictable results. For example, one thread could be updating

the contents of a structure while another thread is reading the contents of the same structure. It is unknown what data the reading thread will receive: the old data, the newly written data, or possibly a mixture of both. MFC provides a number of synchronization and synchronization access classes to aid in solving this problem. This article explains the classes available and how to use them to create thread-safe classes in a typical multithreaded application.

A typical multithreaded application has a class that represents a resource to be shared among threads. A properly designed, fully thread-safe class does not require you to call any synchronization functions. Everything is handled internally to the class, allowing you to concentrate on how to best use the class, not about how it might get corrupted. The best technique for creating a fully thread-safe class is to merge the synchronization class into the resource class. Merging the synchronization classes into the shared class is a straightforward process.

Annexe 5 Remote Procedure Call - Distinct ONC RPC/XDR

Remote Procedure Calls (RPCs) are similar to local procedure calls. In a local procedure call, the program places the parameters in a well-known place (usually the stack), transfers control to the procedure and, when the procedure completes, resumes execution. There is an inherent assumption here that the local system is capable of executing a procedure correctly and efficiently, which is often not correct. For example, a small computer running Windows may not have the resources to execute complicated floating point operations or have the disk space to store and query a large database. With remote procedure calls, any computer can make procedure calls to another remote machine.

To achieve this, the program makes a normal function call to a local procedure called the client stub with the necessary arguments. The client stub, after any necessary local operations, locates the server and sends the message to the server stub, which executes a local procedure call with the given parameters. When done, the server stub contacts the client stub with the results and the client stub returns this result to the program which can at this point resume execution.

The client portion of the Distinct ONC RPC/XDR library contains both a high level and a low level interface to RPC calls. The high level interface consists of rpc_call and callrpc (the two functions only differ in how the server address is specified). Both functions default to the UDP/IP transport protocol. To use the low level interface, the application must first call clntudp_create or clnttcp_create to set up an RPC client connection using either the UDP/IP or the TCP/IP transport protocol. Then one or more RPC calls can be issued by calling clnt_call. Finally, the application must free all resources allocated to the RPC client connection by calling clnt_destroy. Both interface levels include the same functionality. The caller must provide pointers to the argument ('out') and result ('in') buffers (usually containing basic data types or structures, sometimes arrays or even linked lists) along with pointers to the routines used to convert data to and from a host independent network format: the external data representation (XDR). The outgoing data is then encoded into network format using the XDR routine specified and the RPC call is sent to the server. When the RPC reply is received, the

incoming data is decoded into host (local) format using the given XDR routine. All necessary byte swapping is performed automatically by the XDR routines. RPC calls do not return to the caller until an answer has been received or until a timeout has occurred. A detailed description of the above mentioned client RPC routines are given in the reference section of this manual.

The Distinct RPC/XDR-library includes an RPCBIND daemon, which is dynamically loaded when the first RPC server registers a TCP/IP or UDP/IP service. The RPC Information utility (RPCINF32.EXE) can be used to obtain a listing of all registered services and their ports either on the local machine or on other hosts on the network. Please refer to the section 'RPCBIND and RPC Information' for more details. Server applications can register a service using pmap_set and can unregister a service using pmap_unset. The pmap_getport function can be used to look up the port number of an RPC service registered on any host (including the local machine). With the pmap_getmaps function, an application can obtain a list of all the RPC services registered on the local or any remote machine. A detailed description of the four above mentioned port mapper routines is given in the reference section of this manual.

The server portion of the Distinct RPC/XDR-library allows a Windows application or DLL to register one or more services for the UDP/IP or TCP/IP transport protocol. Each such service has associated with it an XDR encode routine, an XDR decode routine and a service routine to be called by the port mapper daemon whenever a request for the service arrives. A server process associates its service number (e.g. 100002 for the rusers service) with an actual port on the local machine by first allocating the necessary resources with a call to svctcp_create or svcudp_create depending on the desired transport protocol. Many servers provide the same service for both UDP/IP and TCP/IP. In this case, the same service routine and the same XDR routines can be used for both transport protocols. Next, the port mapper is instructed to add the new service to the list of available services on the local machine by calling svc_register. Every time an RPC request for a particular service arrives on the local port, the port mapper calls the associated service routine. Upon completion of the request, the service routine must call svc_sendreply to transmit the reply buffer to the client (in case of an error, the svcerr_ functions may be used to return an error message). Before the server terminates, it must call svc_destroy for all of the services it registered to free all resources allocated to the services. A detailed description of the above mentioned server RPC routines

are given in the reference section of this manual.

External data representation (XDR) is used to encode data into a host independent network data format and decode network data back into the host specific data format. Data sent as an argument to an RPC call and the data making up the reply part of the call are transferred in network data format. Before this data can be used, the correct XDR routine must be called for decoding. Any byte swapping or differences in the length of data elements (for example, two versus four byte integers) are handled transparently in the XDR routines. The XDR primitives included in this library can be used to encode or decode call arguments and results directly or to build more complex custom XDR routines. Short descriptions and function prototypes of all XDR primitives are given in the section 'External Data Representation'. For information on how to generate C source code for custom XDR routines which handle arbitrary data structures, please refer to the section - RPC Protocol Compiler (RPCGEN)'.

The Distinct ONC RPC/XDR library also allows encoding and decoding of data to and from XDR memory buffers ('streams') without actually making a remote procedure call. Applications can, for example, encode and decode data that is transmitted over sockets with a protocol other than RPC.

Annexe 6 OpenGL

OpenGL (pour Open Graphics Library) est une bibliothèque portable de rendu graphique 2D et 3D créée en 1992 par SGI (Silicon Graphics). [http://www.sgi.com/]

Elle se compose de plusieurs centaines de fonctions qui permettent au programmeur la création d'images de haute qualité. OpenGL est utilisé dans de nombreux domaines, aussi bien dans les moteurs de jeux vidéo (la série Quake est un bon exemple), dans des logiciels de modélisation 3D professionnels payants (3D Studio Max, Lightwave) ou gratuits (Blender), ou encore dans l'imagerie médicale.

OpenGL est maintenu par un groupe indépendant : l'ARB (Architecture Review Board), composé des plus grands acteurs du marché (Compaq, Microsoft, SGI, IBM, 3DLabs, HP, Intel, Evans & Sutherland). C'est l'ARB qui contrôle donc les propositions d'évolution de l'API, les étudie et les approuve.

Programmation avec OpenGL

La première chose à faire est d'inclure le fichier d'en tête (header file) pour les applications Windows.

```
#include <windows.h>
```

Celui-ci contient les déclarations pour les fonctions qu'on va utiliser. Il faut ensuite créer la fonction WinMain, qui sert de point d'entrée dans le programme.

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int CmdShow)
{
    return 0;
}
```

On déclare la fonction WinMain, retournant une valeur int. Le premier paramètre correspond en gros au handle pour les fichiers. C'est l'ID du programme, qui permet à Windows de le reconnaître. Le second ne sert à rien. Il est utilisé pour la compatibilité avec les vieux programmes 16-bits qui, eux, l'utilisait. Le troisième donne les paramètres passés sur la ligne de commande. Enfin, le dernier sert à définir l'état de la fenêtre au démarrage.

Aucun de ces paramètres n'a de réelle utilité pour nous. Il nous sera en effet plus simple de modifier l'état de la fenêtre plus tard.

Nous avons maintenant écrit le minimum pour créer une application windows. Autrement dit, il est possible de compiler ce programme. Bien sur, son exécution ne donnera aucun résultat, même pas la plus petite fenêtre.

Nous allons donc maintenant créer une fenêtre windows. Pour cela, il faut créer une classe de fenêtre, l'enregistrer puis créer la fenêtre elle-même à partir de la classe enregistrée. Il doit déclarer une variable pour stocker la classe de fenêtre avant WinMain :

```
WNDCLASS wc;
wc.style = CS_OWNDC;
wc.lpfnWndProc = WindowProc;
wc.cbClsExtra = 0;
wc.cbWndExtra = 0;
wc.hInstance = hInstance;
wc.hIcon = LoadIcon(NULL, IDI_APPLICATION);
wc.hCursor = LoadCursor(NULL, IDC_ARROW);
wc.hbrBackground = (HBRUSH)GetStockObject(LTGRAY_BRUSH);
wc.lpszMenuName = NULL;
wc.lpszClassName = "OGL";
```

Le premier paramètre correspond au style de la fenêtre : c'est à dire sa place dans l'écran, les divers menus disponibles, il ne nous sert pas pour le moment. Le second spécifie la procédure de gestion de message. C'est cette procédure qui s'occupera de la gestion des entrées utilisateurs. Les troisième et quatrième paramètres ne nous intéressent pas, pas la peine de vous

encombrer avec. Le cinquième est déjà connu, c'est le fameux hInstance qui désigne le programme actuel. Les sixième, septième, huitième et neuvième paramètres permettent de spécifier divers paramètres, comme une icône, une couleur de fond, etc. Le dernier correspond au nom de la classe. Vous pouvez mettre ce que vous voulez, mais vous devrez vous en resservir ensuite.

Il faut maintenant enregistrer la classe. Si on ne le fait pas, on ne pourra pas créer la fenêtre.

```
RegisterClass(&wc);
```

Pour finir, on crée la fenêtre en elle-même avec tous ses paramètres.

```
hWnd = CreateWindow("OGL", "Fenetre OpenGL",
                    WS_CAPTION | WS_POPUPWINDOW | WS_VISIBLE,
                    0, 0, 640, 480, NULL, NULL, hInstance, NULL );
```

Il faut aussi déclarer une variable :

```
HWND hWnd;
```

Elle nous permet d'identifier notre fenêtre. Le premier paramètre correspond au nom de la classe de la fenêtre. Le second est le texte qui sera affiché dans la barre de titre. Vous pouvez mettre ce que vous voulez. Le troisième contient tous les paramètres de la fenêtre, séparés par des « | ». Les troisième et quatrième paramètres définissent la position de la fenêtre dans l'écran, respectivement en x et en y. Les cinquième et sixième paramètres définissent la taille de la fenêtre en largeur et en hauteur. Les deux paramètres suivants n'ont pas d'intérêt pour nous, je les laisserais donc tranquilles.

Nous allons donc créer la fonction de traitement des messages maintenant. Cette fonction s'appelle WindowProc et on a déjà utilisé son nom lorsque l'on a crée la classe de la fenêtre. Il doit placer cette fonction avant WinMain (mais après les variables globales).

```
LRESULT CALLBACK WindowProc (HWND hwnd, UINT uMsg, WPARAM wParam,
LPARAM lParam)
{
    switch (uMsg)
    {
        case WM_CLOSE:
            PostQuitMessage (0);
            break;
        default:
            return DefWindowProc (hwnd,uMsg,wParam,lParam);
            break;
    }
    return 0;
}
```

La partie intéressante se situe dans le switch. Si le message envoyé par la boucle de message est un message de fermeture (autrement dit, si l'utilisateur veut quitter le programme), cette fonction envoie un message d'arrêt à windows, ce qui permet de fermer le programme proprement. Si ce message n'est pas un message de fermeture, la fonction de gestion de message par défaut est appelée.

Cependant, notre but n'est pas de créer. C'est ici qu'OpenGL prend le relais et va nous permettre de créer une surface OpenGL remplissant la fenêtre.

Jusqu'ici, nous avons créé une application Win32. Maintenant nous allons donc inclure les deux headers nécessaires à l'utilisation des fonctions OpenGL.

```
#include <gl/gl.h>
#include <gl/glu.h>
```

Ce code vient juste après la déclaration du header "windows.h" et pas avant, sinon on ne pourra pas compiler.

Device Context (DC) et Rendering Context (RC) sont les deux choses qu'il va nous falloir obtenir pour notre fenêtre OpenGL. Un device context ou contexte de périphérique, désigne le périphérique sur lequel on va écrire. Dans ce cas, c'est l'écran et plus précisément une fenêtre. Nous allons donc créer un DC avec le handle de notre fenêtre.

```
HDC DC;
HGLRC RC;
DC=GetDC (hwnd);
```

Comme d'habitude, ces variables prennent place au début du programme. On fait donc appel à la fonction GetDC avec comme paramètre le handle de notre fenêtre : hWnd.

Voici à quoi elle doit ressembler après modification :

```
LRESULT CALLBACK WindowProc (HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    switch (uMsg)
    {
        case WM_CREATE:
            DC=GetDC (hwnd);
            break;
        case WM_CLOSE:
            PostQuitMessage (0);
            break;
        default:
            return DefWindowProc (hwnd,uMsg,wParam,lParam);
            break;
    }
    return 0;
}
```

Ainsi, le DC est initialisé lorsque la fenêtre est crée. Il faut maintenant initialiser le PixelFormat, c'est à dire définir les caractéristiques du DC pour l'affichage. Rajoutez donc le code suivant après la création du DC :

```
InitPixelFormat (DC);
```

Ce code appelle la procédure InitPixelFormat qui va servir à définir le DC.

```
void InitPixelFormat (HDC hDC)
{
    PIXELFORMATDESCRIPTOR pfd =
    {
        sizeof (PIXELFORMATDESCRIPTOR),
        1,
        PFD_SUPPORT_OPENGL | PFD_TYPE_RGBA | PFD_DRAW_TO_WINDOW |
        PFD_DOUBLEBUFFER,
        16,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        16,
        0, 0, 0, 0, 0, 0, 0
    };
    SetPixelFormat (hDC, ChoosePixelFormat (hDC, &pfd), &pfd);
}
```

Le premier paramètre indique la taille du PIXELFORMATDESCRIPTOR. Pour ne pas avoir à s'en occuper, il est habituellement mis à sizeof (PIXELFORMATDESCRIPTOR). Le second spécifie la version de la structure, on la met toujours à 1. Le troisième paramètre reçoit des drapeaux (flags) spécifiants certaines caractéristiques du PixelFormat.

PFD_SUPPORT_OPENGL indique le support d'OpenGL par le PixelFormat.
PFD_DRAW_TO_WINDOW indique la possibilité d'écrire sur un périphérique comme l'écran.
PFD_DOUBLEBUFFER indique que nous utilisons 2 buffers (double buffering).
PFD_TYPE_RGBA, indique que les couleurs seront représentées en RGBA.

La procédure SetPixelFormat définie le PixelFormat et prends comme paramètres le DC actuel, un PixelFormat valide et un pointeur vers le PIXELFORMATDESCRIPTOR. Le PixelFormat valide est choisie grâce à ChoosePixelFormat.

Maintenant, il nous reste à obtenir un RC et à le définir comme sortie OpenGL courante.

```
RC = wglCreateContext (DC);
wglMakeCurrent (DC, RC);
```

Il faut aussi veiller à libérer le RC et le DC lorsqu'ils ne sont plus utilisés. Pour cela, WindowProc va encore subir un changement, cette fois si dans WM_CLOSE.

```
wglMakeCurrent (NULL, NULL);
wglDeleteContext (RC);
ReleaseDC (hwnd,DC);
PostQuitMessage (0);
```

Maintenant, il ne manque plus que l'initialisation de l'affichage. Il doit rajouter ceci à WindowProc :

```
case WM_SIZE:
glViewport (0,0,LOWORD (lParam),HIWORD (lParam));
glMatrixMode (GL_PROJECTION);
glLoadIdentity ();
gluPerspective (45,(float)(LOWORD(lParam))/(float)(HIWORD (lParam)),1,100);
```

```
break;
```

Ce code à lieu lorsque la fenêtre est ouverte et à chaque fois qu'elle est redimensionée. glViewPort fixe la taille de la zone d'affichage. Ici, le mot faible (LOWORD pour low word) de lParam correspond à la largeur et le mot fort (HIWORD pour high word) à la hauteur. glMatrixMode définie ici une matrice de projection qui va nous permettre de réaliser un effet de perspective. glLoadIdentity réinitialise la matrice pour nous permettre de la modifier. J'en reparlerais un peu plus loin. gluPerspective va nous permettre d'obtenir l'effet de perspective. Le premier paramètre est l'angle de vue, ici 45. Le second est le ratio pour le champ de vue. Sans ce paramètre, la vue est déformée. Les troisième et quatrième paramètres concernent le clipping. Le clipping permet de supprimer les objets se trouvant trop près ou trop loin pour être vus. Ici, lorsqu'un objet se trouve plus proche que le point 1 (nos yeux) ou plus loin que le point 100, il soit supprimé.

Ensuit il doit créer la procédure dans laquelle nous mettrons tout le code relatif à l'affichage OpenGL dans la procédure WindowProc :

```
case WM_PAINT:
RePaint ();
break;
```

et créer la procédure correspondante :

```
void RePaint ()
{
    glClear (GL_COLOR_BUFFER_BIT);
    glMatrixMode (GL_MODELVIEW);
    glLoadIdentity ();

    /* Ici sera placé tout le code relatif à l'affichage */
```

```
    SwapBuffers (DC);
}
```

glClear sert à effacer le frame buffer, autrement dit l'écran que nous avons sous les yeux.

glMatrixMode définie cette fois une matrice de visualisation, pour que nous puissions afficher quelque chose à l'écran.

glLoadIdentity réinitialise la matrice afin de nous permettre de la modifier. Il est nécessaire de réinitialiser la matrice entre chaque modification pour que nous puissions voir un changement à l'écran. C'est pour cela que cette procédure est appelée depuis WM_PAINT. Ainsi, l'affichage est réactualisé en permanence.

SwapBuffers intervertit le frame buffer et le back buffer. Autrement dit, il met le contenu de l'écran en mémoire, et affiche ce qui était en mémoire à l'écran. C'est ce que l'on appelle la technique du "double buffering". Cette technique permet de ne pas voir l'effet de "déchirement" entre deux réactualisations d'écran.

Résumé

Cette étude s'inscrit dans le cadre de l'amélioration et de la maîtrise de l'utilisation d'un robot appliquée à la projection thermique sur des formes de géométrie complexe. Différentes voies ont été étudiées pour mener à bien ce projet. Sur la base d'une structure logicielle dénommée (RobotStudio™), une trousse à outil dédiée à la projection thermique dite « Thermal Spray Toolkit » a été développée. Cette trousse à outil comporte trois modules (« PathKit », « MonitorKit », « ProfileKit ») qui fournissent respectivement les créations de la trajectoire robot, le contrôle du robot en ligne et l'analyse du profil du dépôt dans un environnement de projection thermique.

Dans un premier temps, les principes de l'extension logicielle pour RobotStudio™ ont été étudiés et la structure globale du programme d'extension a été établie. Puis, des méthodes et protocoles de communication ont été analysés et testés pour la connexion avec le robot. Plusieurs expérimentations ont été effectuées pour tester les fonctionnalités et la stabilité du logiciel développé. Les résultats présentés montrent que ce développement est bien adapté à la génération et l'optimisation de trajectoires robot hors-ligne pour la projection thermique.

Mots clés : Projection thermique, Programmation hors-ligne, Extension logicielle, Trajectoire robot, Contrôle en temps réel, Communication, Qualification du dépôt

Abstract

The aim of this study was to improve robot trajectories control during the thermal spraying process on complex geometric objects. Many researches have been done to support this project. A commercial software (RobotStudio™) has been installed and the robot off-line programming has been tested and achieved. A toolkit dedicated to thermal spray called "Thermal Toolkit Spray" was developed on the platform "RobotStudio™". This toolkit consists of three modules ("PathKit", "MonitorKit", "ProfileKit") which respectively provide the generation of the robot trajectory, the real-time monitoring of the robot and the analysis of the profile of the deposit.

Initially, the principles of the add-in program for RobotStudio™ were studied and the structure of the software was established. Then, the methods and communication protocols were analyzed and tested for the connection with the robot. Several experiments were carried out to test the software functions and its stability. The results show that this development can perfectly be used to create, optimize and monitor the robot program in the whole process of thermal spraying.

Key words: Thermal spray, off-line programming, add-in program, Robot trajectory, Real-time monitoring, Communication, Coating qualification