Bonjour à tous, Je suis ravi d'être ici aujourd'hui pour partager mon expérience de stage. Je m'appelle Zhentao XU et j'ai effectué mon stage au sein de Deltacad pendant 24 semaines. Au cours de cette présentation, Je vais me concentrer sur les principales contributions de mon stage et sur les enseignements que j'en ai tirés.

Le sujet de mon stage était initialement Développement et Intégration de fonctions de vision par ordinateur pour l'inspection automatisée. Plus précisément, Il s'appuie sur deux cadres, ETREL et TEMIS, et a pour objectif de concevoir un logiciel permettant de résoudre le problème de l'identification des défauts des pièces sur la chaîne de production.

Au cours du processus de développement et de test, je conçois principalement sur la base des données des parties du système AML. Comme le montre la figure, la partie central est la pièce à inspecter.

Sur cette base, je vais commencer à présenter le principale composant de mon stage, un logiciel appéle PowerEye. C'est un logiciel développé pour des scénarios industriels afin de réaliser la fonction susmentionné.

Tout d'abord, je vais présenter le PowerEye à partir de l'architecture globale. En ce qui concerne la disposition de l'interface graphique, l'ensemble du logiciel est divisé en trois parties, à savoir la barre des moduele, la barre des fonctions et la fenêtre principale. En sélectionnant le module, différentes barres de fonctions s'affichent, puis la fenêtre principale est affichée en cas de sélection de la fonction. Grâce à la caractéristique du Contentcontrol, je peux injecter l'interface en tant que contrôle, ce qui permet le développement indépendant des différentes parties.

Dans la partie de la fenêtre principale, compte tenu du fait que la réalisation de nombreuses fonctions nécessite la sélection du chemin d'accès au ficher et d'autres configurations, L'Expander est utilisé pour diviser la fenêtre en deux parties, la partie supérieure pour les choix de configuration et la partie inférieure pour les fonctionnalités. La figure suivant montre spécifiquement la fenêtre principale.

En développant le logiciel, j'ai choisi MVVM comme modèle de conception., qui se compose de trois éléments: le modèle, la vue et le modèle de vue. L'interface image que je viens de présenter constitue la composante vue. Dans la partie modele, j'ai conçu la structure de données globale destinée à contenir les informations relatives à l'interface créée. Comme le montre l'image, pour chaque module, j'ai conçu une classe de base à partir de laquelle des interfaces dotées de différentes fonctions peuvent être héritées et modifiées selon les besions. Enfin, pour la partie "modele de vue", qui contient l'implémentation de fonctions et d'instances spécifiques du modèle utilisé pour se lier à l'interface.

Ensuite, je vais présenter en détail les fonctions du logiciel de chaque module.

Le premier module est la préparation des données, pour laquelle j'ai conçu et développé deux interfaces graphiques fonctionnelle, l'extraction d'images et l'ajout de bruit de perlin.

Pour l'interface d'extraction d'images, l'objectif de cette interface est d'extraire le contour de la pièce sur le photo de la ligne d'assemblage. Lorsque j'ai commencé à développer PowerEye, une fonction qui utilise le feature matching pour réaliser cet objectif a déjà été développée. Donc ma tâche principale a consisté à l'intégrer dans l'interface et à y apporter des améliorations. Donc j'ai conçu une barre de progression et une fenêtre pour montrer l'image avant et après le traitement comme le montre l'image ci-dessous. En ce qui concerne l'intégration de la fonctionnalité d'extraction développée, j'ai résolu deux problèmes principaux. D'une part, la fonction est développée en C++, tandis que l'interface est développée en C#, j'ai donc appliqué le ficher dll pour l'importation et l'exportation des fonctions. D'autre part, c'est comment exécuter la fonction sans bloquer l'interface graphique, le thread est utilisé pour résoudre ce problème.

Dans la partie d'ajout de bruit de perlin, L'objectif de cette interface est d'ajouter un masque de bruit à l'image pour générer des pièces comportant des défauts. En plus de l'interface graphique présentée, j'ai également utilisé la classe process en c# pour permettre l'exécution de scripts python.

Le deuxième module est une fonction d'apprentissage automatique. Les scripts python sur l'apprentissage automatique étant développés par les laboratoires collaborant au projet, donc je ne les présenterai pas. De plus, il utilise encore la classe de processus c# et affiche la valeur de sortie de l'exécution dans une zone de texte.

Ce module, qui est l'application la plus importante de tout le logiciel, permet de détecter les images des pièces sur la ligne d'assemblage et de sauvegarder les données. D'abord pour la surveilllance de la ligne d'assemblage, L'idée de cette fonction est d'utiliser le modèle entraîné pour inspecter chaque image de pièce et d'afficher les résultats de l'inspection dans l'interface. Dans la conception de l'interface, j'ai choisi d'utiliser différentes couleurs pour indiquer si le pièce est qualifiée ou non, un peu comme le voyant d'avertissement, et afficher l'imagede ce pièce. Les pièces dont le score est inférieur à 90 sont considérées comme non qualifiées, celles dont le score est supérieur ou égal à 95 sont considérées comme qualifiées, et celles dont le score est compris entre 90 et 95 doivent être testées à nouveau manuellement. Une fois la pièce analysée, ses données sont sauvegardées dans l'interface de stockage des données. Dans cette interface, j'ai conçu trois listes pour stocker les données des différents types de pièces, y compris son temps de production, le nombre de points gagnés et le chemin où l'image est sauvegardée. Et pour les pièces dont la qualification n'est pas certaine, vous pouvez les vérifier manuellement en cliquant sur la liste. Une nouvelle fenêtre apparaît alors et l'état de la pièce peut être vérifié en faisant glisser l'image vers la gauche ou la droite. Ces trois interfaces constituent l'ensemble du processus du contrôle de la chaîne d'assemblage.

Le quatrième module est le suivi des données relatives aux pièces. C'est conçu pour une utilisation à long terme et retracer les données de la pièce précédente enregistrées dans des fichiers Excel. Dans la conception de l'interface, J'utilise une grille de données pour afficher les données d'un fichier Excel, une barre de sélection pour choisir le type de données à filtrer et une zone de texte pour saisir des mots-clés. Afin de lire le fichier Excel, une extension ClosedXML est utilisée, ce qui me permet d'obtenir le contenu du fichier par chemin et de le stocker dans la classe DataTable. Et pour obtenir cette fonction de filtrage, la propriété RowFilter de DataView est tout à fait appropriée, il suffit de changer sa valeur en fonction de différents mots-clés pour modifier le contenu de l'affichage.

Le dernier module est la configuration des données globales. Le module est principalement conçu pour permettre aux développeurs de compléter la sélection de la configuration de plusieurs modules directement par l'importation de fichiers de configuration afin de faciliter les tests. Dans la partie précédente de l'architecture globale, j'ai expliqué la structure de données qui compose PowerEye, consistant en les listes d'exemples de composants pour chaque interface de chaque module. Dans ce module, j'ai utilisé la sérialisation et la désérialisation xml pour importer et exporter la configuration globale. Grâce à l'affichage de l'interface, l'utilisateur peut voir les informations relatives à la configuration actuelle de manière très intuitive. le processus de développement est largement facilité et l'individualisation pour chaque utilisateur est possible.

Après la partie précédente de la présentation, je voudrais faire un résumé de toute mon expérience de stage. En fait, Avant de commencer mon stage, le logiciel était dans sa phase de prototypage. Il contient des fonctionnalités développées par le stagiaire précédent, ainsi qu'un module d'apprentissage automatique développé par les laboratoires d'UTC. Lorsque j'ai terminé mon stage, il disposait d'une interface graphique mature, de modules clairement différenciés et de fonctionnalités rationalisées. Plus précisément, au cours de mon stage, j'ai

non seulement achevé le développement d'une toute nouvelle IHM pour PowerEye, mais j'ai également transféré toutes les fonctionnalités de l'interface prototype vers la nouvelle

interface et j'en ai amélioré une grande partie. Avec ma contribution, PowerEye passe progressivement du stade du prototype à celui de l'optimisation. Au cours de mon stage, j'ai également mis en évidence certaines insuffisances. En raison de mon manque d'expérience dans le développement WPF, j'ai utilisé un cadre jugé a posteriori trop complexe au début de mon développement, ce qui m'a fait perdre du temps à résoudre ce problème. Mais de toutes façons, cette expérience de stage m'a beaucoup touchée. Ce stage

m'a donné l'occasion de travailler dans une entreprise et de participer à un projet à long terme, ce qui m'a permis de comprendre comment organiser mon travail et discuter des

détails avec mes collègues. D'autre part, en tant qu'étudiant qui vient d'arriver en France pour un an, cette expérience me sert également d'expérience d'échange international, me

permettant de sentir l'environnement de travail français et de communiquer avec les autres avec plus d'assurance.

C'est la fin de ma présentation de stage，Merci de votre attention!

Bonjour à tous,

C'est avec grand plaisir que je me tiens devant vous aujourd'hui pour partager mon expérience de stage au sein de Deltacad. Je me présente, Zhentao XU, et pendant 24 semaines, j'ai eu l'opportunité de contribuer au sein de cette entreprise. Mon exposé se concentrera sur mon travail de stage ainsi que les enseignements que j'en ai tirés.

Le sujet central de mon stage était le "Développement et Intégration de fonctions de vision par ordinateur pour l'inspection automatisée". Mon travail s'est articulé autour de deux cadres, ETREL et TEMIS, avec pour objectif la conception d'un logiciel résolvant le défi crucial de l'identification des défauts sur les pièces au sein de la chaîne de production.

Au cours du processus de développement et de test, je conçois principalement sur l'ensemble de données des pièces du AML système. Comme le montre la figure, la partie central est la pièce à inspecter.

Sur cette base, je vais commencer vous présenter l’élément clé de mon travail. Mon principal accomplissement a été la développement d'un logiciel nommé PowerEye. Ce logiciel, conçu spécifiquement pour des scénarios industriels d'accomplir les tâches évoquées précédemment.

Permettez-moi de vous présenter PowerEye sous l'angle de son architecture globale. Sa conception de disposition est divisée en trois parties principales : la barre des modules, la barre des fonctions et la fenêtre principale. Lorsqu'un module est sélectionné, de barre de fonctions spécifiques s'affichent, et en choisissant une fonction, la fenêtre principale présente le contenu correspondant.

Dans la fenêtre principale, j'ai utilisé une approche de conception où la partie supérieure est dédiée aux choix de configuration, tandis que la partie inférieure est consacrée aux fonctionnalités.

Lors du développement du logiciel, j'ai adopté le modèle MVVM, composé du modèle, de la vue et du modèle de vue. L'apparance d’interface représente la vue. Le modèle définit la structure globale de données liée à l'interface. Enfin, le modèle de vue contient les implémentations des fonctions et des instances du modèle pour la liaison avec l'interface.

Passons à présent les fonctions spécifiques de chaque module du logiciel.

Le premier module est dédié à la préparation des données. J'ai développé deux interfaces fonctionnelles : l'extraction d'images et l'ajout de bruit de Perlin. Pour l'extraction d'images, mon rôle consistait à intégrer une fonction existante basée sur le correspondance des caractéristiques. J'ai mis en place une barre de progression et une fenêtre permettant de visualiser les images avant et après traitement.

Pour l'ajout de bruit de Perlin, L'objectif de cette interface est d'ajouter un masque de bruit à l'image pour générer des pièces comportant des défauts. En plus de l'interface graphique présentée, j'ai également integré des scripts Python exécutés via la classe Processus de C#.

Le deuxième module concerne l'apprentissage automatique. Les scripts Python pour cet apprentissage étaient développés en collaboration avec les laboratoires partenaires. donc Je ne vais pas en parler ici. De plus, j’ai utilisé encore la classe de processus c# et affiche la valeur de sortie de l'exécution dans une zone de texte.

Le troisième module est la pierre angulaire du logiciel, permettant la détection des défauts sur les pièces de la chaîne d'assemblage et la sauvegarde des données. J'ai mis en place un processus de surveillance qui utilise un modèle entraîné pour inspecter les images des pièces et afficher les résultats dans l'interface. Les codes de couleur est utilisé pour indiquer si une pièce est conforme ou non. Les pièces avec des scores inférieurs à 90 sont considérées comme non conformes, celles au-dessus de 95 comme conformes, et les autres entre 90 et 95 nécessitent une vérification manuelle. Les données des pièces analysées sont stockées dans une interface dédiée, y compris leurs scores, le temps de production et le chemin des images. Et pour les pièces dont la qualification n'est pas certaine, vous pouvez les vérifier manuellement en cliquant sur la liste. Une nouvelle fenêtre apparaît alors et l'état de la pièce peut être decidé en faisant glisser l'image vers la gauche ou la droite. Ces trois interfaces constituent l'ensemble du processus du contrôle de la chaîne d'assemblage.

Le quatrième module est conçu pour le suivi des données. Il permet de retracer les données précédentes enregistrées dans des fichiers Excel. J'ai utilisé la bibliothèque ClosedXML pour la lecture des fichiers Excel. Après avoir sélectionné le fichier Excel, le filtrage est effectué en sélectionnant le type de colonne et en saisissant les mots-clés.

Enfin, Le cinquème module est la configuration des données globales. Le module est principalement conçu pour permettre aux développeurs de compléter la sélection de la configuration de plusieurs modules directement par l'importation de fichiers afin de faciliter les tests. Dans ce module, j'ai utilisé la sérialisation et la désérialisation xml pour importer et exporter la configuration globale. Grâce à l'affichage de l'interface, l'utilisateur peut voir les informations relatives à la configuration actuelle de manière très intuitive. le processus de développement est largement facilité.

Pour conclure mon exposé, je souhaite résumer mon expérience de stage. J'ai contribué au passage du logiciel de sa phase de prototypage à une interface mature et fonctionnelle. J'ai également identifié des domaines où j'aurais pu gagner en efficacité, notamment dans le choix initial d'un cadre de développement. Cependant, cette expérience m'a énormément enrichi, me permettant de travailler au sein d'une entreprise, de comprendre l'organisation professionnelle et d'améliorer mes compétences en communication au sein d'une équipe.

Je vous remercie de votre attention.

Ceci conclut ma présentation de stage.