Analysis of Dynamic Scenes: Application to Driving Assistance

Tay C. Tay Christopher

INSTITUT POLYTECHNIQUE DE GRENOBLE

N° attribué par la bibliothèque
|_|_|_|_|_|_|_|_|_|_|

Thèse

pour obtenir le grade de

DOCTEUR DE L'Institut polytechnique de Grenoble

Spécialité : *Informatique*

préparée au laboratoire LIG et l'INRIA Rhône-Alpes,
dans le cadre de **l'École Doctorale** *Mathématiques, Sciences et Technologies de l'Information, Informatique (MSTII)*

présentée et soutenue publiquement
par

TAY Meng Keat, Christopher

le 4 Sept 2009

Titre :

Analysis of Dynamic Scenes : Application to Driving Assistance

DIRECTEUR DE THÈSE :

Christian Laugier

JURY :

M. James Crowley	Président
M. Raja Chatila	Rapporteur
M. Michel Pasquier	Rapporteur
M. Hiromichi Yanagihara	Examinateur
M. Kamel Mekhnacha	Examinateur
M. Christian Laugier	Directeur de thèse

Abstract

The development of autonomous vehicles garnered an increasing amount of attention in recent years. The interest for automotive industries is to produce safer and more user friendly cars. A common reason behind most traffic accidents is the failure on the part of the driver to adequately monitor the vehicle's surroundings. In this thesis we address the problem of estimating the collision risk for a vehicle for the next few seconds in urban traffic conditions.

Current commercially available crash warning systems are usually equipped with radar based sensors on the front, rear or sides to measure the velocity and distance to obstacles. The algorithms for determining the risk of collision are based on variants of time-to-collision (TTC). However, it might be misleading in situations where the roads are curved and the assumption that motion is linear does not hold. In these situations, the risk tends to be underestimated. Furthermore, instances of roads which are not straight can be commonly found in urban environments, like the roundabout or cross junctions.

An argument of this thesis is that simply knowing that there is an object at a certain location at a specific instance in time does not provide sufficient information to asses its safety. A framework for understanding behaviours of vehicle motion is indispensable. In addition, environmental constraints should be taken into account especially for urban traffic environments.

A bottom up approach towards the final goal of constructing a model to estimate the risk of collision for a vehicle is presented. The simpler case of "free" motion is first presented. The thesis then proposes to take collision risk estimation further by being more "environmentally aware" where environmental structures and constraints are explicitly taken into account for urban traffic scenarios.

This thesis proposes a complete probabilistic model motion at the trajectory level based the Gaussian Process (GP). Its advantage over current methods is that it is able to express future motion independently of state space discretization. Driving behaviours are modelled with a variant of the Hidden Markov Model. The combination of these two models provides a complete probabilistic model for vehicle evolution in time. Additionally a general method of probabilistically evaluating collision risk is presented, where different forms of risk values with different semantics can be obtained, depending on its applications.

Résumé

Le développement des véhicules autonomes a reçu une attention croissant ces dernières années, notamment les secteurs de la défense et de l'industrie automobile. L'intérêt pour l'industrie automobile est motivé par la conception de véhicules sûrs et confortables. Une raison commune derrière la plupart des accidents de la circulation est due au manque de vigilance du conducteur sur la route.

Cette thèse se trouve dans le problématique de l'estimation des risques de collision pour un véhicule dans les secondes qui suivent en condition de circulation urbaines.

Les systèmes actuellement disponibles dans le commerce sont pour la plupart conçus pour prévenir les collisions avant, arrières, ou latérales. Ces systèmes sont généralement équipés d'un capteur de type radar, à l'arrière, à l'avant ou sur les côtés pour mesurer la vitesse et la distance aux obstacles. Les algorithmes pour déterminer le risque de collision sont fondés sur des variantes du TTC (time-to-collision en anglais). Cependant, un véhicule peut se trouver dans des situations où les routes ne sont pas droites et l'hypothèse que le mouvement est linéaire ne tient pas pour le calcul du TTC. Dans ces situations, le risque est souvent sous-estimé. De plus, les instances où les routes ne sont pas tout droit se trouve assez souvent dans les environnement urbain ; par exemple, les rond point ou les intersections.

Un argument de cette thèse est que, savoir simplement qu'il y ait un objet à une certaine position et à une instance spécifique dans le temps ne suffit pas à évaluer sa sécurité dans le futur. Un système capable de comprendre les comportements de déplacement du véhicule est indispensable. En plus, les contraintes environnementales doivent être prises en considération.

Le cas le plus simple du mouvement « libre » est d'abord traité. Dans cette situation il n'ya pas de contraintes environnementales ou de comportement explicite. Ensuite, les contraintes environnementales des routes sur trafic urbain et le comportement des conducteurs des véhicules sont introduits et pris en compte explicitement.

Cette thèse propose un modèle probabiliste pour les trajectoires des véhicules fondé sur le processus gaussien (GP). Son avantage est le pouvoir d'exprimer le mouvement dans le futur indépendamment de la discrétisation d'espace et d'état. Les comportements des conducteurs sont modélisés avec une variante du modèle de Markov caché. La combinaison de ces deux modèles donne un modèle probabiliste de l'évolution complète du véhicule dans le temps. En plus, une méthode générale pour l'évaluation probabiliste des risques de collision est présentée, où différentes valeurs de risque, chacun avec sa propre sémantique.

Contents

List of Figures

Part I

Summary In French

Introduction

Contexte général

Le développement des véhicules autonomes a reçu une attention croissant ces dernières années, notamment les secteurs de la défense et de l'industrie automobile. L'intérêt de la défense se mettre en évidence par le « DARPA Urban Challenge ». Les véhicules autonomes peuvent sen servir comme les véhicules télécommandés pour faire de combat ou comme une multiplicateur de force technologique. L'intérêt pour l'industrie automobile est motivé par la conception de véhicules sûr et confortables. Une raison commune derrière la plupart des accidents de la circulation est due au manque de vigilance du conducteur sur la route. Un véhicule équipe avec un système qui permettre de prévenir le conducteur sur le risque potentiel de collision peut réduire un grand nombre d'accidents mortels.

(a) DARPA Urban Challenge (Carnegie Mellon)

(b) Project INTERSAFE

Figure 1: Exemples d'assistance et des véhicules autonomes

Bien qu'il n'existe actuellement aucun véhicule autonome disponible sur le marché, la technologie de conduit assisté telle que le contrôle de vitesse, l'alerte de franchissement de voie ou le parking automatique sont disponibles aux consommateurs en ce moment. Ces technologies peuvent être vu comme une progression naturelle vers les véhicules autonome.

Cette thèse se situe dans le cadre de l'assistance à la conduite. Un système capable destimer le risque de collision est souhaitable, surtout pour prévenir les conducteurs des véhicules sur les collisions qui peuvent arriver dans les prochains seconds, mais qui ne sont pas conscient de ces dangers. L'objectif sera de pouvoir donné ces indications de risque passivement. Le système ne prend pas le contrôle actif de véhicule.

Déscription du problème

Le problème principal de cette thèse concerne l'estimation du risque de collision d'un véhicule. Du point de vue du conducteur, le conducteur peut obtenir une indication générale du risque de collision pour les prochains secondes qui suivent, afin de prévenir le conducteur de ces risques. Le risque estimé peut être aussi utilisés pour aider un véhicule autonome doptimiser sa trajectoire afin de minimiser son risque de collision.

Pour un véhicule complètement autonome, ou pour un système de l'estimation de collision, l'estimation du risque de collision est une composante du système complet. L'estimation du risque prend en entrée un ensemble des données provenant des capteurs traitées par d'autres modules. Le valeur en sortie est un valeur probabiliste, qui peut être interprété de façon diffèrent en fonction de contexte actuel.

Dans cette thèse, on suppose que l'ensemble des donnés suivant est disponible:

1. **Géométrie de la route:** Afin de prendre en compte le contrainte géométrique de la route pour estimer le risque, les informations géométrique comme la courbure et largeur de la route sont pertinent. Ces informations peuvent être obtenu à partir des algorithmes spécifiques qui traitent les données brutes de caméra ou lidars. Alternativement, il est aussi possible dobtenir l'information sur la géométrie en utilisant les systèmes dinformation géographique, en utilisant une carte de l'environnement et un GPS pour se localiser.

2. **Le suivi des cibles:** L'estimation des risques de collision nécessite la détection et le suivi des obstacles mobiles. Le suivi des obstacles inclure l'estimation de position et vélocité des objets mobiles.

3. **Capteurs spécifique:** Il y a des informations supplémentaire qui ne sont pas crucial mais assez importante. Par exemple, la détection des clignotant pour les véhicules donnes les indicateurs forts sur l'intention de véhicules. Un autre capteur « virtuelle » qui est aussi important est la distance au bord de voie. La distance au bord indique l'intention de véhicule de tourner ou de changer de la voie. Ces informations sont très informatives pour la reconnaissance du comportement des véhicules.

Un véhicule pour que nous appellerons l'ego véhicule est supposé dêtre équipé avec les capteurs appropriés afin d'obtenir l'ensemble des entrées des capteurs mentionnés ci-dessus. Dans cette thèse, le risque est estimé est une valeur numérique qui exprime quantitativement le risque du ego véhicule qui est probablement en collision avec un autre véhicule dans les secondes qui suivent.

L'estimation du risque de collision dans le futur implique la construction de modèles qui représente le mouvement du véhicule dans le futur. En plus, ce modèle devrait être capable de prévoir raisonnablement les états futur véhicule. C'est seulement avec une prédiction sur les états futurs de véhicule qu'il est possible d'estimer le risque de collision dans le futur.

En faisant raisonnement sur le futur, il est logique de décrire le futur en utilisant le probabilité. Un argument pour l'utilisation de probabilité sera discuté dans le chapitre 3. En bref, il ny a que les fondations de probabilités qui donne une méthode raisonnable et consistent pour manipuler les croyances de façon cohérente.

Nous présentons un modèle dévolution de véhicule entièrement probabiliste pour obtenir et inférer les croyances sur les états futurs des véhicules dans les environnements urbains. Par conséquent, le risque de collision estimé peut être obtenus à partir des modèles en termes de probabilité de manière théoriquement cohérente.

Une meilleure prédiction de risque de collision

Les systèmes actuellement disponibles dans le commerce sont pour la plupart conçus pour prévenir les collisions devant, arrière, ou latérales. Ces systèmes sont généralement équipés d'un capteur de type radar, arrière, devant ou les côtés pour mesurer la vitesse et la distance aux obstacles. Les algorithmes pour déterminer le risque de collision sont fondés sur des variantes de TTC (time-to-collision en anglais)(LEE76). Le TTC est essentiellement un fonction de deux objets, qui donne le temps avant un objet rentre en collision avec l'autre en faisant l'hypothèse que les deux objets maintient la même vitesse linéaire. Certains systèmes ne sont pas passifs, mais au contraire, elle intervient en contrôlant directement les freins et éventuellement la volant pour qu'il puisse prend des actions correctives nécessaires. Les systèmes fondé sur le TTC sont dépendent du fait que les observations sont effectuées à une fréquence assez élevé afin de s'adapter à l'évolution de l'environnement.

Les systèmes commerciaux courant donne des performances assez raisonnables sur les autoroutes ou certains partie de la ville où les routes sont tout droit. Cependant, il y a des situations comme les routes avec des courbures, où les mouvements sont pas linéaire (voir figure 2). Dans ces situations, les risques sont souvent sous-estimés. En plus, les instances où les routes ne sont pas tout droit se trouve assez souvent dans les environnement urbain ; par exemple, les rond point ou les intersections.

Plusieurs projets de recherche ont été créés pour surmonter ces problèmes en prenant en compte la structure de l'environnement en particulier les intersections où il ya un taux plus élevé des accidents. Ces projets donne les risque de collision aux intersections où il ya des communication sans-fils, soit entre les véhicule ou en utilisant les infrastructures installé dans l'environnement (PJL$^+$00) (Adm04) (FJ07). Ces systèmes ont des véhicules équipé d'une paire de capteurs (soit les radar ou les lidars) aux coins avant le véhicule pour détecter des véhicules en direction orthogonale. Les vitesses et donc le TTC des obstacles sont ensuite évaluées pour déterminer le risque de collision. Même les structures sont prises en considération lors de l'évaluation du risque de collision, le calcul du risque de collision est encore fondamentalement fondé sur l'hypothèse d'un mouvement linéaire. L'horizon

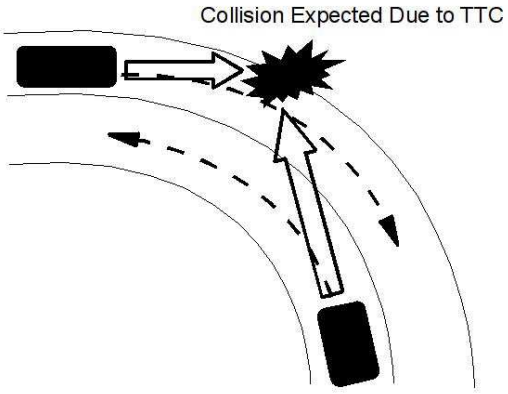


Figure 2: Un exemple d'une mauvaise estimation du risque à cause de supposition linéaire dans les systemes TTC.

temporel de prévision du risque est court et information cruciale sur l'environnement et les informations des capteurs ne sont pas bien utilisées.

Un discours de cette thèse est que, connaissant simplement qu'il y ait un objet à une certaine position et à une instance spécifique dans le temps ne fournissent pas d'informations suffisantes pour évaluer son sécurité dans le futur. Un système pour comprendre les comportements de déplacement du véhicule est indispensable. En plus, les contraintes environnementales doivent être prises en considération, en particulier pour la circulation en trafic urbaine. Une vue densemble de l'approche proposée et présentée dans la section suivante.

Approche

Une approche pour estimer le risque de collision d'un véhicule est présentée. Le cas le plus simple du mouvement « libre » est d'abord traité. Dans cette situation il n'ya pas des contraintes environnementales ou de comportement explicite.

Ensuite, les contraintes environnementales des routes de trafic urbain et le comportement des conducteurs des véhicules sont introduits. Le modèle probabiliste pour la réalisation physique de trajectoire est fondé sur le modèle du mouvement « libre » sans contrainte. Ce modèle est adapté et étendu pour le cas de mouvement avec contrainte sur la route. Une modélisation sur le comportement des conducteurs est introduite pour arriver à un modèle complet de mouvement de véhicule probabiliste. L'estimation du risque de collision peut ensuite se calculer a partir de modèle de mouvement de véhicule probabiliste.

Sans contraintes

Pour le cas simple sans contraintes, le mouvement dans un environnement comme une grande salle ouverte est considérée. Plus précisément, un modèle du mouvement humain est construit, qui peut être considéré comme une version réduite du modèle probabiliste de l'évolution des véhicules sans contraintes et sans comportement.

Le modèle de mouvement est fondé sur le processus stochastique gaussien (GP). Grace au modèle GP, il n'est plus nécessaire de traiter les problèmes de discrétisation, tout en conservant mathématiquement la représentation probabiliste cohérente pour chaque modèle de mouvement.

Le GP est une distribution gaussienne sur les fonctions où une fonction échantillonnée à partir de ce GP peut être considéré comme une séquence de mouvement. Ces mouvements sont supposés de varier autour de motion prototype (moyenne de la distribution GP) selon la distribution Gaussien. Ainsi, un modèle de mouvement peut être considéré comme une distribution gaussienne sur une espace multidimensionnelle. Un scénario composé de plusieurs schémas de mouvement peut être facilement modélisés comme un modèle de mélange de GP.

L'ajout des contraintes

Figure 3 donne une vue d'ensemble sur l'architecture de notre approche pour estimer le risque de collision pour un ego véhicule. Le contexte se trouve dans la boîte pointillée. Les détails complets du fonctionnement interne concernant le contenu de la zone en pointillés peuvent être trouvés au chapitre 7.

Chaque véhicule dans la scène est associé avec un modèle probabiliste sur son évolution et suivi. Un modèle complètement probabiliste sur l'évolution de véhicule est composé de l'estimation de comportement et la réalisation de comportement. L'estimation de comportement consiste à utiliser les données des capteurs traitées, par exemples les clignotants ou la distance au bordure de la route, d'estimer si le véhicule suivi est en train deffectuer les comportement comme tourner, doubler ou aller tout droit. Les comportements sont estimés en décomposant les comportements où chaque comportement est défini par une succession des sous états plus fine au niveau inférieur qui décrit le comportement. Les états de haut niveau et bas niveau de comportement sont ensuite estimées en utilisant une variante de la modèle de markov caché (HMM).

Pour chaque comportement de haut niveau, le module sur la réalisation de comportement décrit sa trajectoire correspondante. La distribution probabiliste sur la réalisation physique de trajectoire correspondant à chaque comportement est fondée sur le processus gaussien (GP).

Une implémentation naïve du GP similaire à celui utilisé pour les espaces ouverts (section I) fonctionne bien que pour un environnement limitée en taille géographique. Quand l'environnement devient plus grand, ou si la topologie du réseau routier devient plus compliquée, le nombre de modèle GP augmentera. Nous vous proposons de résoudre ce problème d'échelle en exploitant des structures répétitives des routes. Un GP que nous

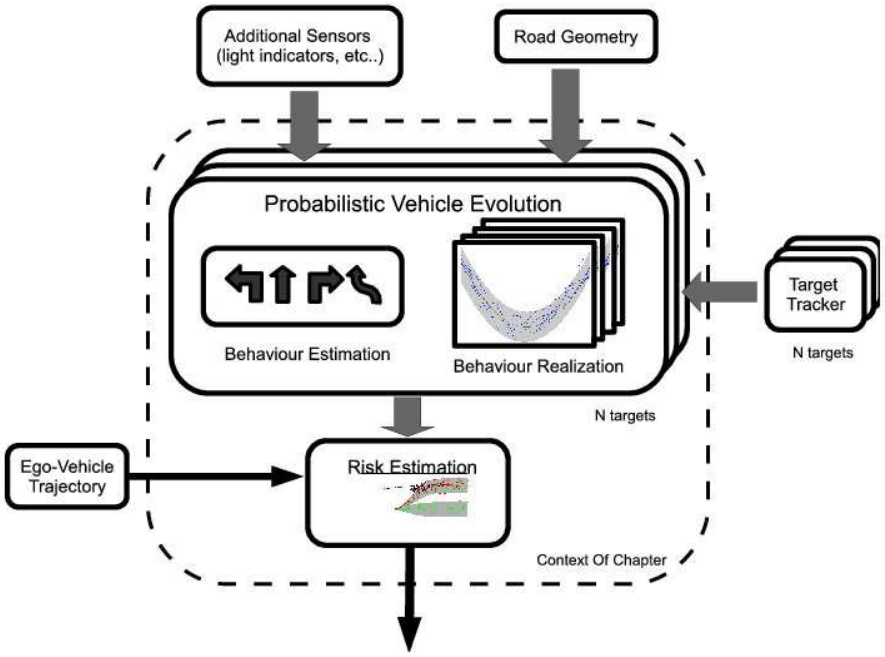


Figure 3: Overall View Of Vehicle Risk Estimation

appelons le GP canonique est définie dans un espace séparé. Ce GP canonique est alors transformé pour s'adapter à la contrainte géométrique La méthode « Least Squares Conformal Map » (LSCM) est utilisée pour cette transformation avec les bonnes propriétés comme la minimisation des distorsions locales.

Une distribution de probabilité sur les comportements et une distribution de probabilité sur la réalisation trajectoire des comportements donne un modèle complet et probabiliste sur l'évolution de mouvement de véhicule. Ce modèle probabiliste de l'évolution du véhicule est utilisé pour l'estimation du risque. La valeur de risque obtenue est une mesure de probabilité. Nous considérons que les des risques pour un certain véhicule que nous appellerons le véhicule ego.

A la base, le risque d'une trajectoire pré-déterminé du ego véhicule est estimé, en appliquant le modèle probabiliste de l'évolution des véhicules pour tous les autres véhicules autour de ego véhicule. Une application d'une telle valeur du risque peut être utilisé dans la boucle de rétroaction dun véhicule autonome robotique par exemple. Le calcul du risque peut être généralisé où une variété de valeurs de risques différents avec des sémantiques différentes peut être calculée, selon les exigences dont les valeurs de risque vont être utilisés. Par exemple, nous pouvons calculer le risque contre des véhicules individuels, le risque dépendant sur l'horizon temporel avant la collision ou le risque associé avec un certain comportement.

Intuition

Un scénario exemple possible à deux voies dans la même direction est illustré dans la figure 4. Deux véhicules A et B se déplacent sur chaque voie et le risque de collision doit être estimée pour le véhicule B. De point de vue de véhicule A, la structure locale des voies sont implicitement décrite par des manouvres; tout droit, tourner à droite / gauche, changement de voie. Ces manœuvres sont dénommés les comportements. L'ensemble total de possibles comportements sont prédéfinis. Cependant, tous les comportements sont disponibles dans tous les cas. Par exemple, il n'est pas possible de tourner à gauche à l'intersection suivante, car il n'y a pas de à gauche. L'ensemble des comportements possibles à chaque instance est un sous-ensemble de tous les possibles comportements.

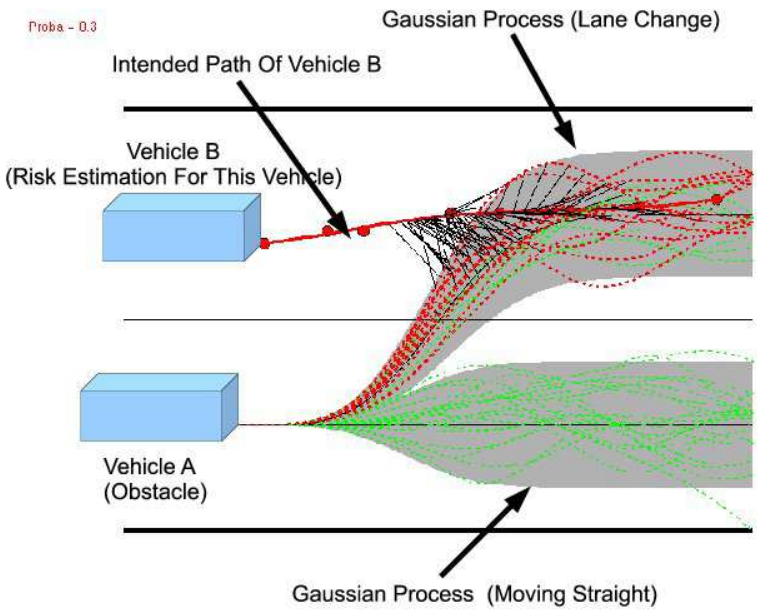


Figure 4: Risque estimée pour une trajectoire à prendre par le véhicule B. La prédiction de mouvement pour le véhicule A (obstacle) est obtenue par échantillonnage du distribution GP (un pour chaque comportement; tout droit et changement de voie). Le risque de collision est calculé par une somme pondérée des trajectoires échantillonnée en collision..

Pour chaque comportement possible, il y a plusieurs manières d'exécuter physiquement la réalisation trajectoire du comportement. Les humains ne conduire pas d'une manière absolument tout droit en suivant précisément le centre de la voie. Il est donc raisonnable de supposer que la routine normale de conduite suit approximativement le centre de la voie. La conduit en suivant la voie pour un comportement donné est représenté par une distribution GP qui donne une distribution de probabilité sur les réalisations physiques des trajectoire dans le futur. La moyenne de cette distribution sera le trajectoire qui suis exactement le centre de la voie. Le GP et ses variations sont présentées comme des régions

d'ombre dans la figure 4 qui correspond aux comportements changement de voie et aller tout droit. Les lignes pointillées représentent le chemin échantillonné à partir du GP. Pour les cas où la route a une courbure non nulle ou pour les comportements qui consiste a tourner, le GP sera adapté de manière appropriée en fonction de la géométrie de la route.

L'ensemble des GP pour chacun des comportements réalisables dans la scène, en combinaison avec la probabilité que le véhicule A exécute un certain comportement, donne un modèle probabiliste pour l'évolution future du véhicule A dans la scène.

Comme pour l'évaluation du risque de collision par TTC, l'évaluation du risque de collision seront pour B véhicule par rapport au véhicule A. Contrairement à TTC où la collision est estimée en supposant l'hypothèse d'une trajectoire linéaire pour les deux véhicules A et B, nous évaluons le risque de collision de la trajectoire à prendre par véhicule B contre tous les possibles trajectoires à prendre par véhicule A. La valeur du risque sera alors une combinaison pondérée de la trajectoire unique à prendre par le véhicule B par rapport aux tous trajectoires possibles pour les véhicules A. Les pondérations sont attribuées selon le modèle probabiliste de l'évolution futur des comportement pour véhicule A.

Défis

La modélisation des schémas de mouvement en utilisant le GP avec des contraintes présente une série de défis différents et ses difficultés contrairement aux cas où les contraintes ne sont pas présentes:

1. Comme sera présenté dans le chapitre 6, les schémas de mouvement ont été modélisées dans un cadre fixe de référence pour les espaces sans contrainte. Si les mêmes méthodes sont naïvement appliquées à la problème de la modélisation de schémas de mouvement sur les routes, la complexité du modèle agrandissent avec la complexité du réseau routier. Comme les mouvements peuvent commencer à partir de n'importe quel point au long de la route, et en plus, à partir de n'importe quel point du réseau routier à l'autre, le nombre de schémas de mouvement possibles augments de façon combinatoire.

2. En modélisant les phénomènes de mouvement dans une référence fixe, cela signifie également qu'il est valide que pour un certain groupe de réseaux routiers dans une zone géographique spécifique. Le même schéma de mouvement ne peut pas être utilisé dans une nouvelle zone géographique avec une configuration du réseau différent.

3. Un référence fixe signifie également que les calculs appropriés doivent être effectués dans le référence globale. Il y a conséquemment un besoin de se localiser qui peut être compliquée et parfois pas nécessaire puisque le risque de collision est toujours évaluées par rapport à un véhicule.

4. Les structures routières ne sont pas bien exploitées. La description des schémas de mouvement dans une référence globale explique aussi implicitement la configuration

du réseau routier. Les structures locales des routes peuvent être décrites en utilisant des verbes comme tout droit, tourner à gauche, tourner à droite par exemple. Ayant une description de la route locale est compact. En plus, une description routière mondiale nest pas vraiment nécessaire comme les capteurs installés sur les véhicules pour la détection d'obstacles ont des limites sur la distance de détection et des limites sur le champ de vision.

Contribution

La contribution principale de la thèse est sur l'estimation de risque de collision pour un ego véhicule, tout en en tenant le comportement des autres véhicules et en respectant les contraintes géométriques des scénarios de circulation urbaine. Les points suivants mettent en évidence les points clés:

1. L'estimation du risque est porté un peu plus loin en prennent en compte les contrainte géométrique imposé par les structures de l'environnement de manière hiérarchique.

2. Les comportements de conduite ont une grande influence dans la précision sur l'estimation du risque de collision. Elle est prise en compte de façon explicite dans notre modèle d'estimation du risque.

3. Notre approche donne une sémantique qui correspond à des notions intuitives de risque. Ceci est particulièrement utile lorsque les interfaces homme-machine (IHM) sont considérées dans l'assistance à la conduite.

4. Sémantique intuitive, comme tourner à gauche /ou à droite, sont convertis en géométrie, en tenant en compte la structure et la topologie.

5. Notre méthode permettre le passage aux grands environnements en exploitant les structures de l'environnement qui sont sémantiquement répétitives. En plus, notre méthode permettre l'adaptation dynamique en fonction de la géométrie de l'environnement.

6. Quand on parle de l'estimation du risque de collision, il ya une référence implicite à prédiction dans le futur. La notion de « peut-être » est rigoureusement codée dans le calcul des probabilités.

Une autre contribution est un modèle probabiliste de mouvement fondé sur les processus gaussien (GP). Ses points clés sont:

- Les modèles états-espace nécessite la discrétisation de l'espace dans lequel l'objet se déplace. Le choix de la discrétisation n'est pas évident et il est souvent prédéterminé et fixé en avance. Mais le modèle GP n'a pas ce genre de problème.

- Le GP est entièrement probabiliste et nous permet d'exprimer la moyenne et les écarts type à tout moment en suivant le schéma de mouvement. Par conséquence, il peut faire des inférences probabilistes sur la motion dans le futur ou de calculer la probabilité d'une observation de la trajectoire appartenant à un modèle de mouvement d'une manière cohérente. Ceci est aux méthodes existant (voir chapitre 2) où les distances entre les deux trajectoires sont définies dans un manière ad hoc.

Organisation du document

Première partie: Contexte et l'état de l'art

Modèle de Motion et Prédiction. Cette thèse est centrée sur la modélisation probabiliste de schémas de mouvement. Chapitre 2 présente un panorama de l'état actuel.

Les Modèles Probabilistes. Chapitre3 présente le formalisme des probabilités et des modèles bayésienne , qui sont les bases pour des modèles probabilistes de mouvement.

Le Processus Gaussien et Modèle de Markov Caché. Comme notre modèle probabiliste pour le mouvement des véhicules est fondé sur le processus gaussien et modèle de markov caché, nous consacrons un chapitre à présenter ces deux modèles au chapitre 4.

Deuxième partie: Modèles Proposé et Algorithmes

Introduction à la Partie II. Chapitre 5 explique la structure pour la partie II. Dans la partie II, une approche ascendante est adoptée lorsque le problème simple sur le modélisation de mouvement sans contraintes est d'abord présentée. Ce modèle de mouvement de base est alors adaptée à la contraintes structurelles des conditions de circulation urbaine dans l'évaluation des risques de collision.

Motion sans contraintes. Le problème de la motion sans les contraintes imposées par le structure de l'environnement est présentée dans le chapitre 6. Une intuition sur comment le GP peut modéliser les mouvements et ses généralisations à des situations plus complexes, impliquant plusieurs schémas de mouvement sont présentées. À partir de ce modèle, nous présentons une algorithme d'inférence variational pour récupérer les schémas de mouvement à partir d'un ensemble de données, qui consiste d'un ensemble d'observations correspondant à des séquences de mouvement.

L'estimation des risques de collision. Un modèle de modélisation dans les espaces structurés est donnée dans le chapitre 7, avec un importance particulier sur l'estimation des risques de collision d'un véhicule dans le trafic urbain. Nous présentons une méthode complète et entièrement probabiliste sur l'adaptation de GP pour modéliser l'évolution des véhicules. L'évaluation des risques et de ses généralisations sera discutée.

Troisième partie: Les Expériments

Mouvement dans les espaces ouverts sans contraintes. Nous présentons au chapitre 8 des résultats sur la récupération de schémas de mouvement dans un espace ouvert, notamment sur la base de données sur l'ensemble mouvements humains dans le hall d'entrée de l'INRIA Rhône-Alpes.

Planification du Mouvement avec l'estimation du risque. Un exemple de l'application du GP comme un outil de prédiction pour guider un robot du type voiture à naviguer en toute sécurité est présenté au chapitre 9. L'algorithme de planification de trajectoire est fondé sur une extension probabiliste de RRT (Randomly-exploring Random Trees) développées par (FTSL08) et les expérimentes sont fait.

L'estimation des risques de collision en assistance à la conduite. Deux séries d'expériences sont présenté au chapitre 10, qui se déroule dans le contexte de l'estimation des risques de collision dans l'assistance à la conduite. La première expérience est un simulation monte carlo afin d'évaluer et valider l'exactitude et la fiabilité de modèle GP dans l'estimation des risques de collision. La deuxième expérience porte l'évaluation des risques à un environnement plus réaliste. Pour des raisons pratiques dans l'évaluation des collisions, un monde virtuel du trafic urbain est créé lorsque les véhicules sont conduits par l'homme dans cet environnement virtuel pour ajouter du réalisme. L'estimation des risques est ensuite évaluée dans ce cadre expérimental. Cette expérience se passe dans une échelle relativement grande avec le joint coopération avec Toyota Motors Europe (TME) et l'entreprise Probayes.

Quatrième partie: Conclusion

Ce manuscrit se termine par une résumé des travaux et une discussion sur notre approche dans le chapitre 11. Nous discutons quelques faibless, et nous donnons des suggestions pour les extensions dans le futur.

L'estimation du Risque

Organization

Ce chapitre commence par l'architecture du système expliquée dans la section 7.1. Le modèle probabiliste d'évolution du mouvement de véhicule est constitué de deux sous-modules permettant une estimation de comportements et leurs réalisations. Le modèle probabiliste de l'évolution du mouvement de véhicule est ensuite utilisé pour estimer des risques de collision. Ces sous-modules seront présentés dans les sections I, I et I, respectivement.

L'architecture globale

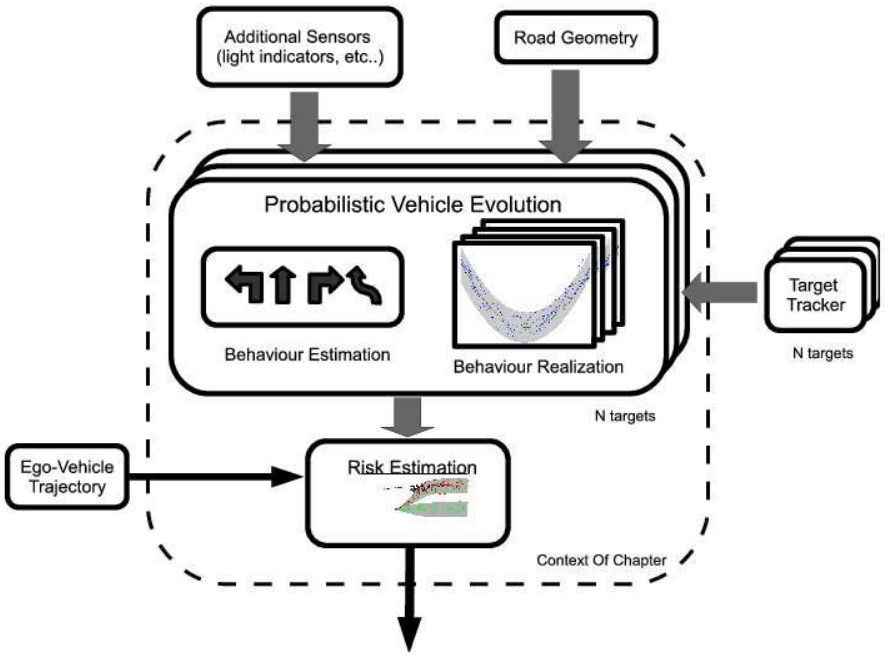


Figure 5: L'architecture sur l'estimation du risque

La figure 5 représente une vue globale des divers éléments dans le contexte de l'estimation

de risque. Le problème qui nous intéresse est associé à lensemble des sous-modules contenus dans la zone en pointillés :

1. **Reconnaissance de comportement.** L'objectif de la reconnaissance de comportement est d'estimer la probabilité pour un véhicule d'exécuter l'un des comportements possibles. Par exemple, il pourrait donner une valeur de probabilité $P(tourner\ gauche)$, qui représente la probabilité pour un véhicule observé d'effectuer une manœuvre de type « tourner à gauche ». Comme il etait mentionné précédemment, les comportements sont des représentations sémantiques de haut niveau de la structure de route. La distribution de probabilité sur les comportements est réalisée par un modèle de Markov caché (HMM). Notre modèle actuel consiste de 4 comportements : tout droit, tourner à gauche, tourner à droite et changements de voie. Ces comportements seront décrits plus en détail dans la section I.

2. **Réalisation du Comportement.** Lévaluation de risque devrait prendre en compte la géométrie de la route. La réalisation de comportement est représentée par la distribution de la probabilité GP, qui est une représentation probabiliste de l'évolution possible du mouvement d'un véhicule. L'adaptation de GP en fonction du comportement est effectuée en utilisant une transformation géométrique connue sous le nom "Least Squares Conformal Map"(LSCM). Tous les détails pertinents seront décrits dans la section I

3. **Evaluation de risque.** Un modèle probabiliste complète sur l'évolution du mouvement d'un véhicule est représenté par la distribution de probabilité sur la reconnaissance de comportement et la distribution de probabilité sur la réalisation du comportement. Le risque de collision est calculé en utilisant ce modèle probabiliste complèt.

En général, le calcul du risque de collision peut être encapsulé dans la notion intuitive de « risque de collision dans les secondes qui suivent ». Cependant, sa définition mathématique précise dépend de son application. Notre modèle pour l'estimation du risque est compatible avec une variété de notion du risque selon les besoins des applications. Il sera expliqué en détail dans la section I

Reconnaissance de comportement et la modélisation

L'objectif de la reconnaissance de comportement est dattribuer un label et une mesure de probabilité sur une séquence de données. Dans ce contexte, les données séquentielles sont les observations reçues des capteurs. Quelques exemples sont la distance aux bord de la voie, les signaux clignotant ou la distance jusqua prochaine intersection, etc. Parce que nous souhaitons obtenir la distribution de probabilité sur les comportements, ces comportements sont les variables cachées. Il y existe des modèles différents pour résoudre ce problème.

Un modèle probabiliste bien connu pour inférer les comportements cachés est le Modèle de Markov Cachée (HMM) (Rab89). Les extensions de HMM inclurent les HMM paramétrisés

(WB98b), HMM entropique (BK00), HMM de longueur variable (GJH01), HMM couplé (BOP97) et HMM structuré (HBN00). Ces modèles étendrent le HMM standard pour la modélisation des activités complexes et leurs interactions.

Nous présentons dans cette section un modèle des comportements de véhicules dans les conditions normales de circulation, qui est un étape intermédiaire pour lestimation du risque de collision. Le HMM en couche (LHMM, Layered HMM en anglais) décompose l'espace des paramètres tels que la robustesse du système et il est amélioré par une réduction dentrainement et du réglage. Son architecture est très appropriée pour la modélisation du comportement de véhicule. Chaque couche a un lien direct déquivalence sémantique qui est directement modélisée.

Les comportements sont modélisés sur deux couches. Chaque couche est composée d'une ou plusieurs HMM. La couche supérieure est un HMM unique où ses états cachés représentent les comportements à un niveau élevé, tels que changement de voie, tourner à gauche, tourner à droite, ou aller tout droit. Pour chaque état caché (ou le comportement dans le HMM de la couche supérieure), il y a un HMM correspondant dans la couche inférieure qui représente la séquence des transitions d'état plus fine du comportement unique. La figure 6 montre le schéma de LHMM.

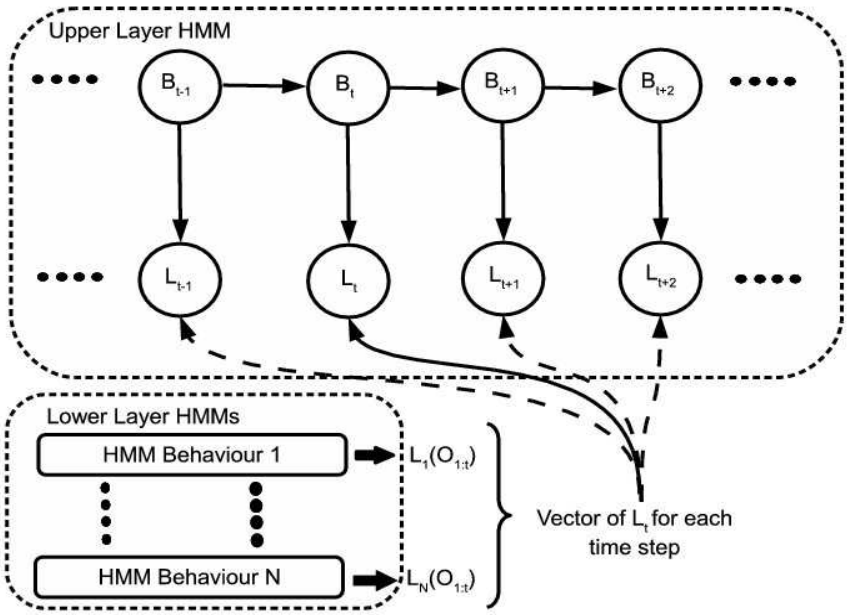


Figure 6: LHMM. La vraisemblance de HMM de couche inférieure est calculée et réutilisée comme observation pour le HMM de la couche supérieure.

Dans notre modèle, nous définissons la sémantique d'état suivant caché dans la couche inférieure HMM pour chaque HMM de couche supérieure :

- *Continuer tout droit* (1 état caché) : avancer.

- *Doubler* (4 états cachés) : changement de voie, accélérer (et en dépassement des véhicules), revenir à la voie originale et la vitesse normale.
- *Tourner à gauche / droite* (3 États caché) : décélération avant le tourne, l'exécution, et la reprise de la vitesse normale.

Afin de déduire le comportement des véhicules dans notre contexte, nous souhaitons maintenir un distribution de probabilité sur les comportements qui sont représentés par les États cachés de HMM dans la couche supérieure. Les observations des véhicules interagissent avec le HMM dans la couche inférieure et l'information est ensuite propagée à la couche supérieure. Dans la couche inférieure, il y a un HMM correspondant pour chaque comportement. Chaque HMM dans la couche inférieure, indexée par h, met à jour son état actuel:

$$P(S_{t,h}O_{1:t}) \propto P(O_t|S_{t,h}) \sum_{S_{t-1,h}} P(S_{t-1,h})P(S_{t,h}|S_{t-1,h}), \tag{1}$$

où les variables probabilistes O_t correspondent à des observations à l'instant t, $S_{t,h}$ est la variable pour l'état caché de HMM h à l'instant t. Pour chaque HMM h dans la couche inférieure, son vraisemblance d'observation, $L_h(O_{1:t})$, peut être calculé:

$$L_h(O_{1:t}) = \sum_{S_{t,h}} P(S_{t,h}O_{1:t}). \tag{2}$$

Chaque vraisemblance d'observation $L_h(O_{1:t})$ serve comme « observations » pour le HMM de la couche supérieure. L'inférence des comportements au niveau supérieur prend une forme similaire:

$$\begin{aligned} P(B_t|O_{1:t}) &= P(O_{1:t}|B_t) \sum_{B_{t-1}} P(B_{t-1})P(B_t|B_{t-1}) &(3)\\ &= L_{B_t}(O_{1:t}) \sum_{B_{t-1}} P(B_{t-1})P(B_t|B_{t-1}). &(4)\end{aligned}$$

où B_t est la variable d'état caché du HMM au niveau supérieur à l'instant t, $P(B_t|B_{t-1})$ est la matrice de transition de comportement au niveau supérieur. Pour la plupart du temps, il est raisonnable de supposer qu'une transition de comportement du niveau supérieur arrive plus souvent à la fin du niveau inférieur de la séquence de comportement, plutôt qu'au milieu de la séquence de comportement au niveau inférieur. Par exemple, si un véhicule est en train de doubler, cela correspond au comportement d'un niveau supérieur. Ce comportement, doubler, est composé des comportements au niveau inférieur : changements de voie, une accélération en passant autre véhicule, le retour à la voie originale et la reprise de sa vitesse normale. La chance d'un changement de comportement de haut niveau pour un véhicule (doubler vers tourner à gauche) lors le véhicule est dans un état de comportement de changements de voie au niveau inférieur, est plus faible.

Pour tenir compte de ces effets, il y a deux matrices de transition différentes pour le HMM au niveau supérieur. Une matrice de transition correspond à la transition lorsque le

comportement de niveau inférieur est complètement exécuté (T_{final}). L'autre matrice de transition, $T_{not-final}$ correspond aux cas où le comportement au niveau inférieur n'est pas à l'état final. Donc, pour le niveau supérieur, la matrice de transition de comportement peut être calculée comme une fonction des états au niveau inférieur:

$$P(B_t|B_{t-1}) = \sum_{S_{t,B_{t-1}}} P(S_{t,B_{t-1}})P(B_t|S_{t,B_{t-1}}B_{t-1}), \tag{5}$$

où $S_{t,B_{t-1}}$ est l'état à l'instant t du HMM au niveau inférieur, qui correspond au comportement précédent B_{t-1}, et $P(B_t|S_{t,B_{t-1}}B_{t-1})$ est par définition:

$$P(B_t|S_{t,B_{t-1}}B_{t-1}) = \begin{cases} \mathbf{T}_{final} & S_{t,B_{t-1}} \text{ is a final state,} \\ \mathbf{T}_{not-final} & \text{otherwise.} \end{cases} \tag{6}$$

A chaque pas de temps, les distributions de probabilités sur les comportements au niveau supérieur $P(B_t|O_{1:t})$ sont mises à jour itérativement. Ceci sera utilisé dans l'estimation du risque dans la section I. Le LHMM est mis à chaque pas de temps:

Algorithm 1: Layered HMM Updates

Input: observation courante O_t
Output: $P(B_t|O_{1:t})$

```
foreach HMM au niveau inférieur h do
    Mettre à jour P(S_{t,h}O_{1:t}) (eqn. 1);
    Calculé la Log-vraisemblance L_h(O_{1:t}) (eqn. 2);
end
Mise à jour de la couche supérieure P(B_t|O_{1:t}) (eqn. 4);
```

Réalisations des comportements

Un comportement est une représentation abstraite du mouvement d'un véhicule. Une distribution de probabilité sur la réalisation physique de la trajectoire de véhicule (sachant son comportement) est indispensable pour une estimation du risque. La distribution de probabilité sur la réalisation physique du mouvement futur de véhicule est modélisée à l'aide d'un GP.

Rappelons que le GP représente la routine normale de la conduite où un conducteur suit à peu près la voie et il ne dérive pas trop loin vers gauche ou droite. Sur la route toute droite, cela peut être facilement représenté par un GP où la moyenne correspond au milieu de la voie (voir figure 7).

Comme mentionné au chapitre I section 1.3.4, une représentation compacte du point de vu de GP ne s'agit pas d'apprendre les GP différents spécifiquement pour le réseau routier. Pour résoudre les cas où il y a des variations de la courbure des voies ou pour des comportements tels que tourner à gauche ou à droite, nous proposons une procédure d'adaptation, ce que nous appellerons un *GP canonique*.

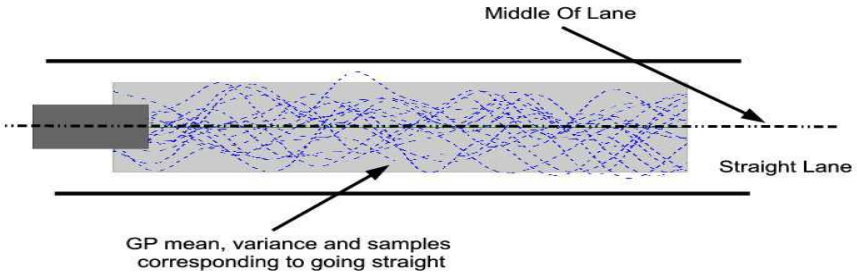


Figure 7: Exemple trivial du modèle GP d'une voie parfaitement rectiligne.

Un GP canonique correspond graphiquement à la figure 7 où un véhicule se déplace sur une route d'une ligne droite. Pour s'adapter à la géometrie de la route, ce GP canonique est déformé, ce qui permet d'avoir une représentation compacte et une flexibilité d'adaptation à la géométrie des voies différentes. En plus, un seul GP peut être calculé qu'une seule fois, puis, il peut être réutilisé pour la géometrie différente, en réduisant le temps de calcul.

Lorsque les situations non linéaires sont rencontrées, une déformation sera effectuée sur le GP canonique pour s'adapter à la géométrie de la voie, comme un exemple sur la figure 8 en illustre.

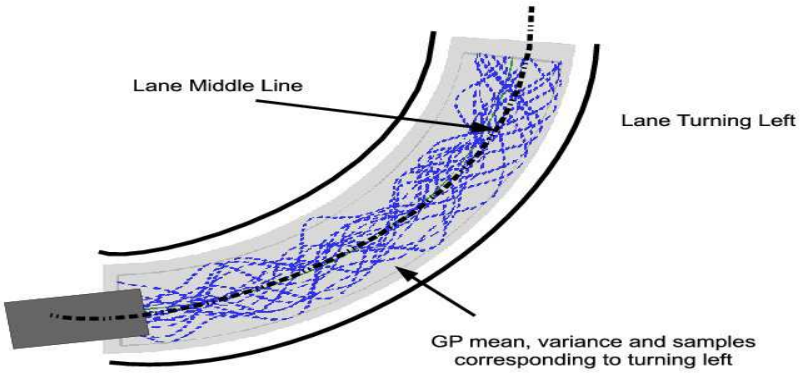


Figure 8: Exemple d'un modèle du GP déformé pour un virage à gauche.

Processus de déformation du modèle gaussien

L'objectif de la déformation de GP est d'adapter le GP canonique à la géométrie de la voie. Une façon naturelle de regarder le GP adapté consiste à considérer le GP adapté comme le même GP canonique mais définie en coordonnées curvilignes. Le problème de l'adaptation du GP peut être formulé ainsi comme la transformation inversible, $\mathcal{U} : (x, y) \mapsto (u, v)$, qui applique chaque point du GP canonique (x, y) définie en coordonnées cartésiennes vers le point correspondant (u, v) en coordonnées curvilignes. $\mathcal{U}$ est une application où $\mathcal{U}^{-1}$ existe (voir sur la figure 9).

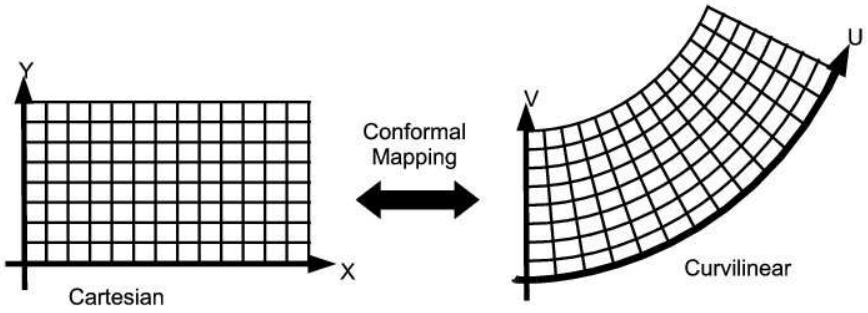


Figure 9: Transformation conformal inversible.

Les coordonnées curvilignes apparaissent dans de nombreux problèmes techniques tels que la dynamique des fluides, l'électromagnétisme où une grille basée sur les coordonnées curvilignes est utilisée pour résoudre des équations aux dérivées partielles numériquement. Les méthodes utilisées dans ces domaines nécessitent des calculs couteaux et la spécification des conditions aux frontières. Une technique courante pour la construction de coordonnées curvilignes est la *transformation conformal*.

Definition 0.0.1. Une **transformation conformal** est une fonction de variables complexes $\mathcal{U} : (x, y) \mapsto (u(x, y), v(x, y))$, qui est analytique dans le voisinage ouvert qui contient (x, y). Les fonctions analytiques sont connues afin de satisfaire l'équation du Cauchy-Riemann:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \qquad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}. \tag{7}$$

En faisant la différentielle de l'équation 7 par rapport à x et y, et vice versa, l' équation de Laplace est ainsi obtenue:

$$\Delta u = 0, \qquad \Delta v = 0, \tag{8}$$

où $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ est l'opérateur de Laplace. Comme l'application satisfait les équations de Laplace, elle est également connue comme une application harmonique.

Propriété

Une transformation conformal sur la grille de coordonées minimise la distorsion au niveau local et est utilisée pour effectuer la déformation du GP. Cette transformation est non seulement lisse, mais, il y existe une application inverse qui est indispensable pour faire d'une prédiction dans le GP canonique. En plus, la déformation locale est minimale parce que la jacobienne de $\mathcal{U}$ est partout une rotation et à l'échelle près.

Problème associé avec l'implantation

La transformation conformal a été initialement définie dans le domaine continu, mais elle est coûteuse au niveau du temps de calcul. Les techniques de la cartographie conforme discrete permettant d'une approximation de ce processus effectuent la transformation linéaire par morceaux entre les triangles du maillage. La plupart des méthodes existants font une approximation de la transformation conformal par discrétisation de l'opérateur de Laplace aux sommets des triangles de maillage. Telles solutions nécessitent généralement la spécification des conditions aux frontières (EDD$^+$95 ; PJP93).

Implémentation choisie

Dans le contexte de notre problème, la spécification de la frontière n'est pas nécessaire. La frontière de la voie est implicitement définie par la courbe représentant le milieu de la voie et sa largeur. En plus, la spécification des limites de la voie qui n'est toute droite n'est pas évidente, en particulier dans les portions des voies à courbure élevée. Idéalement, il suffit simplement de générer les coordonnées curvilignes en utilisant uniquement la ligne ou courbe représentant le milieu de la voie et sa largeur.

Une approche duale qui évite la spécification des conditions aux frontières, le « Least Squares Conformal Map » LSCM, a été proposée (LPRMt02). Au lieu de discrétiser l'opérateur de Laplace aux sommets des triangulations, LSCM propose d'adhérer autant que possible la condition de conformalité dans chaque triangle. Le problème est simplifié avec une minimisation quadratique sans contraintes qui peut être efficacement résolu numériquement.

Prédiction de mouvement du véhicule

La section précédente I a décrit une transformation isomorphe entre le GP adapté à la géométrie de la route et le GP canonique. Cette section présente la procédure sur l'utilisation de la transformation pour prédire le mouvement du véhicule.

Une prédiction informée sur le mouvement du véhicule utilise l'observation de l'état actuel et les états passés. A chaque pas de temps t, une séquence ordonnée temporellement d'observation courante et passée $O = \{O_t, O_{t-1}, \ldots, O_{t-K}\}$ est maintenue où chaque

observation $O_{t-k} = (x_{t-k}, y_{t-k})$ est un vecteur qui contient les positions du véhicule. Ensuite, les observations sont transformées par LSCM vers l'espace du GP canonique, où le mouvement futur peut être inféré. La distribution de probabilité sur le mouvement futur est ensuite transformée à nouveau au repère global. Nous montrons chaque étape de cette procédure permettant de prévoir le mouvement du véhicule.

Transformation conformal entre l'espace du monde et l'espace canonique.

La correspondance entre l'espace global et l'espace canonique (l'espace où le GP canonique réside) est discrétisée et représentée comme une transformation isomorphe entre deux maillage en utilisant LSCM.

Le calcul de transformation nécessite la spécification d'un certain nombre de points fixés et leurs coordonnées après la transformation. Les points fixés sont choisis de manière déterministe; un ensemble de points discrétisés et situés au milieu de la voie, où chaque point correspond à un point sur l'axe horizontal du GP canonique (voir sur figure 10).

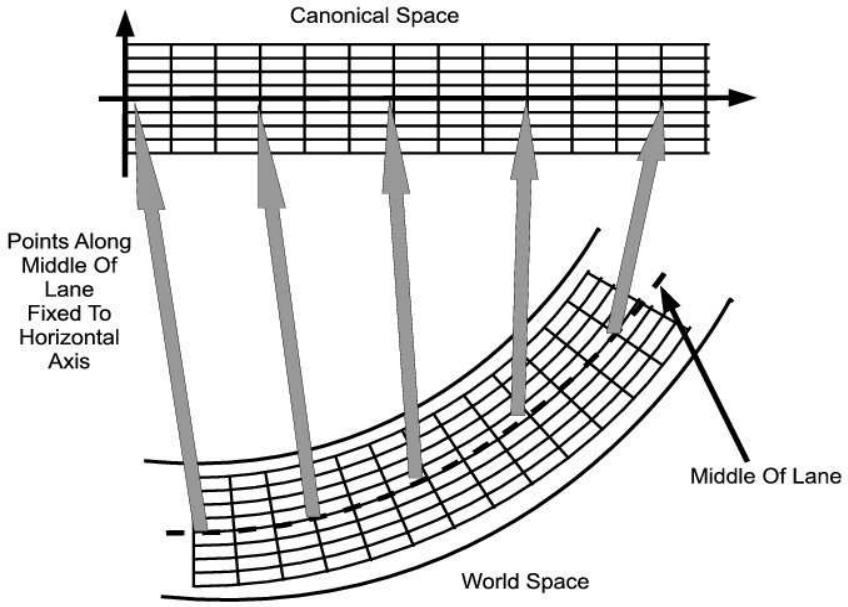


Figure 10: Transformation conformal entre l'espace canonique et l'espace global. Les grosses flèches montrent les points fixés où les points de la ligne au milieu de la voie dans l'espace global correspondant à l'axe horizontal de l'espace canonique.

Inférence de la distribution de probabilité sur le mouvement futur.

Les observations dans le repère global doivent être transformées à l'espace canonique avant que l'inférence sur le mouvement futur puisse être effectué. Le LSCM donne les morceaux discrets de transformation affine entre les deux espaces. Les observations

dans les coordonnées globales peuvent être transformées à l'espace canonique par $\mathcal{U}^{-1}(O_i) = (x_i, y_i)$.

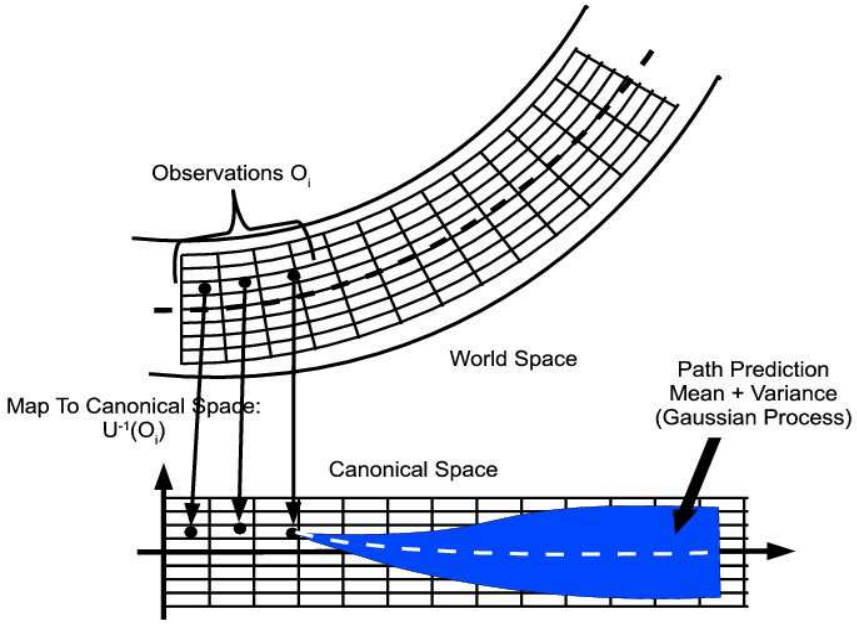


Figure 11: Les observations sont transformées dans l'espace canonique avant le conditionnement du GP sur les observations, ce qui permet d'obtenir une distribution de probabilité sur le mouvement futur.

La cartographie $\mathcal{U}$ est discrétisée et se manifeste sous la forme d'un maillage. $\mathcal{U}^{-1}(O_i)$ peut être calculé en trouvant d'abord le triangle de maillage qui contient O_i dans l'espace global, puis par une transformation O_i au triangle correspondant dans l'espace canonique en calculant les coordonnées barycentriques.

La transformation des observations n passées pour les positions des véhicules dans le repère global donne un ensemble des valeurs $\{(x_i, y_i)\}_{i=1}^{n}$ dans l'espace canonique. La distribution de probabilité sur le mouvement futur du véhicule observé correspond à la distribution de probabilité donnée par le GP:

$$P(Y_*|X_*, X, Y) = \mathcal{GP}(\mu_{Y_*}, \Sigma_{Y_*}) \quad (9)$$

$$\mu_{Y_*} = K(X_*, X)\left[K(X, X) + \sigma^2\mathbf{I}\right]^{-1} Y \quad (10)$$

$$\Sigma_{Y_*} = K(X_*, X_*) - K(X_*, X)\left[K(X, X) + \sigma^2\mathbf{I}\right]^{-1} K(X, X_*), \quad (11)$$

où $X = (x_1, \ldots, x_n)^T$, $Y = (y_1, \ldots, y_n)^T$ sont les observations, $X_* = (x_1^*, \ldots, x_K^*)$ est le vecteur des valeurs à prédire représenté par $Y_* = (y_1^*, \ldots, y_K^*)$, chaque $x_i^* > \max X$.

Transformation inverse au repère global.

La distribution de probabilité sur le mouvement futur est un GP dans l'espace canonique spécifié par l'équation 11 et cette distribution doit être retransformée vers l'espace global pour pouvoir évaluer les risques. Cependant, la transformation conformal d'un GP n'est pas triviale. Mais, l'échantillonnage à partir dune distribution gaussienne est facile. Donc, on choisi une représentation de Monte Carlo (section 3.6.2) de la distribution échantillonnée $P(Y_*|X_*, X, Y)$. Les échantillons seront ensuite utilisés pour évaluer les risques (section I). Intuitivement, chaque échantillon est une réalisation possible sur le mouvement futur du véhicule, qui est représentée comme une séquence de position $S_i = ((x^*_{i,1}, y^*_{i,1}), \ldots, (x^*_{i,K}, y^*_{i,K}))$. Comme les échantillons sont dans l'espace canonique, ils sont transformés vers l'espace global en utilisant le LSCM.

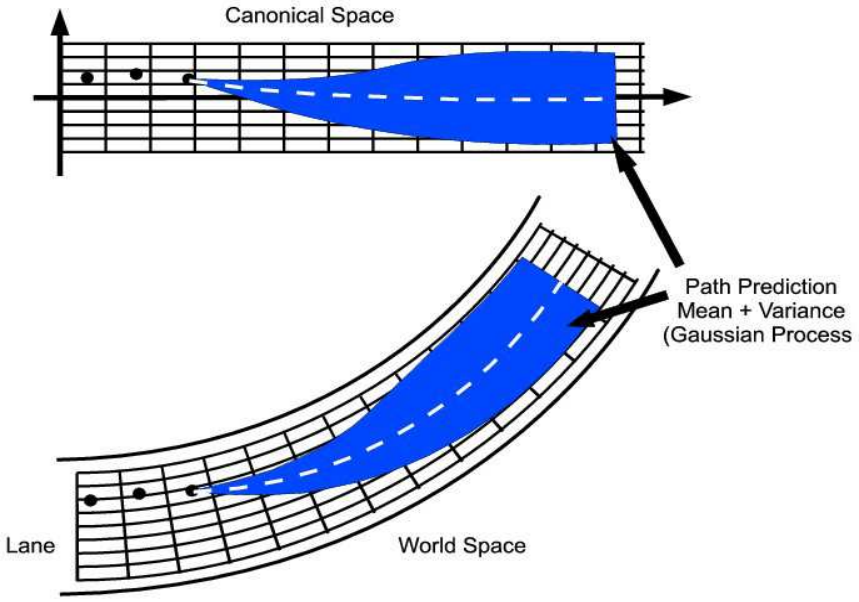


Figure 12: Le *GP canonique* est transformé à nouveau à l'espace global. Les régions ombrées représentent la moyenne et la variance. La transformation peut être approximée par échantillonnage à partir du GP dans l'espace canonique avant la transformation des échantillons.

Evaluation des risques

L'approche LHMM (section I) permet de calculer une distribution de probabilité sur les comportements à chaque pas de temps. Pour chaque comportement, un GP donne la probabilité de distribution sur sa réalisation de trajectoire. Comme la sémantique du comportement est propagée à partir du LHMM jusqu'au niveau physique, il est donc possible d'attribuer une sémantique à la valeur de risque.

Il est important de noter que la définition du risque peut prendre des formes diverses dépendant largement de la manière dont la valeur du risque va être utilisée. Un risque pourrait être une valeur scalaire qui est suffisante pour un système dalerte de collision, ou une application qui a besoin de savoir les valeurs du risque contre chaque véhicule dans la scène. Le champ des applications utilisant ces valeurs de risque peut être classé en 2 catégories.

La première catégorie d'applications implique un degré variable de contrôle du véhicule où les valeurs de risque peuvent être utilisés pour conduire un véhicule autonome, ou simplement pour prendre le contrôle d'un véhicule pour eviter une situation dangereuse.

A) **Risque dune trajectoire calculé en prenant en compte le comportement d'un véhicule**

Nous commençons avec l'exemple simple d'un véhicule autonome dans un environnement dynamique, où le véhicule se déplace en évitant les collisions avec autres entités en mouvement dans cet environnement. Généralement, ces véhicules autonomes sont équipés avec un système de navigation. Nous considérons un module de contrôle qui prend en compte le risque de collision. Il n'est pas difficile d'imaginer que ce module de contrôle évalue l'ensemble de trajectoires possibles à prendre par le véhicule autonome et que le véhicule autonome choisira la trajectoire avec le niveau de risque le plus faible.

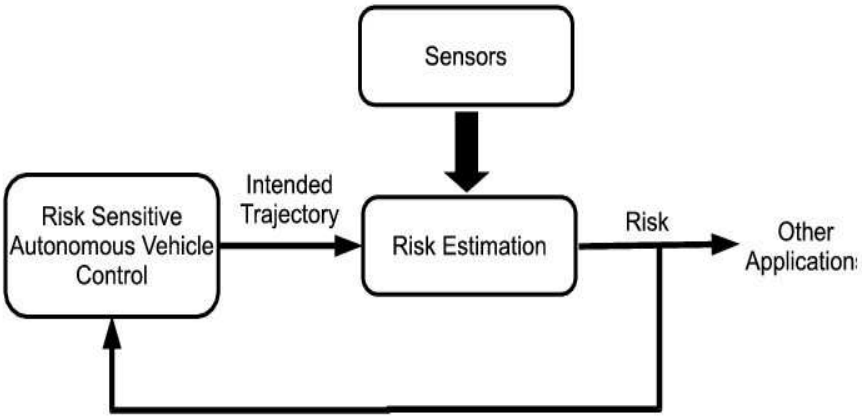


Figure 13: L'architecture de contrôle d'un véhicule autonome.

Dans ce cas, nous calculons le risque d'une trajectoire unique en considération. Dans une scène, il peut y avoir plusieurs véhicules présents. Prenons le cas simple d'un véhicule V_1, (excluant les véhicules autonomes, V_A). Le risque d'une trajectoire considérée par le véhicule V_A, c'est-à-dire la trajectoire T_A, contre le comportement b du véhicule V_1 est donné par:

$$P(C|T_A\, B_{V_1}\, V_1) = \sum_{T_{V_1}} P(C|T_A\, T_{V_1}\, B_{V_1}\, V_1) P(T_{V_1}|B_{V_1}\, V_1), \tag{12}$$

où C est une variable booléenne probabiliste indiquant s'il y a une collision, B_{V_1} est la variable qui correspond aux comportements du véhicule V_1. Ces comportements se trouvent dans les états cachés du HMM au niveau supérieure. T_A et T_{V_1} sont les trajectoires de V_A et V_1, respectivement. $P(T_{V_1}|B_{V_1}\,V_1)$ est la réalisation physique du comportement B_{V1} et, donc, est représentée par les trajectoires échantillonnées à partir du GP mentionné précédemment dans la section I. $P(C|T_A\,T_{V_1}\,B_{V_1}\,V_1)$ évalue s'il existe une collision entre des trajectoires T_A et T_{V_1} .

B) **Risque d'une trajectoire contre un véhicule avec les comportements agrégés**

Le risque d'une trajectoire contre un autre véhicule peut être obtenu en agrégeant les risques calculés précédemment. L'agrégation est essentiellement une somme pondérée de $P(C|T_A\,B_{V_i}\,V_i)$ pour chaque comportement B_V du véhicule V_i:

$$P(C|T_A\,V_i) = \sum_{B_{V_i}} P(C|T_A\,B_{V_i}\,V_i)P(B_{V_i}|V_i). \tag{13}$$

La somme pondérée a été effectuée sur le terme $P(B_{V_i})$ et ses valeurs viennent du LHMM (voir la section I, l'équation 6).

C) **Agrégation des risques par rapport aux véhicules**

Le risque d'un trajectoire T_A lors de la prise en compte d'un seul véhicule V_i est représenté par $\mathcal{R}_i = P(C|T_A\,V_i)$. Il y a plusieurs choix possibles pour l'agrégation des risques, qui est largement dépendente de la manière dont la valeur agrégée des risques soit être utilisée ou interprétée. La fonction d'agrégation des risques est une fonction des valeurs de risque de tous les véhicules, i.e. $\mathcal{F}(\mathcal{R}_1, \ldots, \mathcal{R}_N)$ pour les N véhicules dans la scène:

- **Marginalisation par rapport aux véhicules.** Un moyen direct d'agréger les risques sera de marginaliser sur les probabilités a priori des véhicules. Les probabilités a priori sur les véhicules peuvent provenir d'un module de reconnaissance des objets qui exprime la confiance que l'objet est un véhicule par exemple. Sans aucune information, un prior uniforme peut être utilisé à la place et est équivalent de prendre le risque moyen de tous les véhicules:

$$\begin{aligned} \mathcal{F}(\mathcal{R}_1, \ldots, \mathcal{R}_N) &= P(C|T_a) \\ &= \sum_{V_i} P(C|T_A\,V_i)P(V_i) \\ &= \sum_{V_i} \mathcal{R}_i P(V_i). \end{aligned} \tag{14}$$

- **Risque maximale.** La marginalisation par rapport aux véhicules pourrait sous estimer le risque dans certains cas. Cela est particulièrement vrai quand un

véhicule unique pose un danger imminent dans une scène avec nombreux véhicules dans la scène. La moyenne des risques donne une estimation basse dans ce cas. En prenant la valeur de risque maximale pourrait représenter le risque plus précisément:

$$\mathcal{F}(\mathcal{R}_1, \ldots, \mathcal{R}_N) = \max_{V_i} P(C|T_A\, V_i). \tag{15}$$

- **Risque temporellement plus proche.** L'évaluation des risques de collision, $P(C|T_A\, B_{V_1}\, V_1)$, ne prend pas en compte le temps explicitement. Par exemple, la vérification de collision entre T_A et une trajectoire échantillonnée T_{V_i} indique seulement s'il y a une collision dans un certain horizon temporel dans le futur, indépendamment de la longueur de l'horizon.

$$Collide(T_A, T_{V_i}) = \begin{cases} 1.0 & \text{If any collision exists in time horizon,} \\ 0.0 & \text{otherwise.} \end{cases} \tag{16}$$

L'intégration du temps dans l'évaluation des risques est utile dans certains cas. Pour les applications tels que la prévision de collision, il est moins probable que si le conducteur maintient l'accélération actuelle, un accident se produit dans 30 secondes inévitablement, avec une probabilité de 1.0. Les conducteurs de véhicules impliqués ont suffisamment de temps pour réagir à la situation. Dans ce cas, il pourrait être souhaitable d'exprimer des risques à plus long terme dans le temps comme ayant moins "d'importance". Cela peut être pris en compte par une modification où le risque est pondéré par une fonction décroissante avec le temp:

$$Collide^*(T_A, T_{V_i}) = \begin{cases} \exp^{-\alpha t^2} & \text{If collision between } T_A \text{ and } T_{V_i}, \\ 0.0 & \text{otherwise.} \end{cases} \tag{17}$$

D) **Risque associé au comportement de la conduite**

Jusqu'à présent, la valeur du risque d'une trajectoire T_A pour un véhicule autonome est calculée. Pour les applications où les valeurs de risque sont passivement utilisées, surtout si le conducteur nest pas une machine, cest moins pratique dévaluer le risque pour une seule trajectoire T_A . L'alternative sera d'évaluer le risque associé avec aux comportements de conduite ou la valeur générale de risque pour l'ego-véhicule.

- **Risque associé aux comportements.** Au lieu d'évaluer pour une unique trajectoire, le risque est évalué pour lensemble des trajectoires associées au comportement. Essentiellement, il suffit de faire les échantillonnages sur le GP associé au comportement, et la valeur de risque est calculée en trouvant le proportion de ces trajectoires en collision.
- **Valeur générale de risque.** Une valeur générale de risque entre deux véhicules peuvent être obtenue en évaluant le risque associé aux comportements décrits dans le paragraphe précédent et, ensuite, une marginalisation sur les comportements.

Ceci est équivalent à la somme pondérée de risque associé au comportement par la distribution de probabilité sur les comportements, qui est estimée par le LHMM.

Plusieurs exemples de risques avec des sémantiques différentes sont présentés. Le nombre de façons différentes d'évaluation des risques est combinatoire. Le risque peut être évalué entre les échantillons de trajectoire, des comportements, des véhicules ou tous les véhicules. Notre modèle qui combine le LHMM pour identifier les comportements est l'utilisation des GP pour des réalisations de comportement, est suffisamment général pour le calcul de valeurs différentes de risque, tout en prenant en compte la géométrie de la route et la topologie.

Conclusion

Nous avons montré les différents modules impliqués dans l'estimation du risque dans une situation structurée de la route. Le LHMM (section I) est utilisé pour l'estimation du comportement des véhicules dans la scène. Les comportements peuvent être divisés en deux niveaux hiérarchiques, ce qui correspond à l'architecture de LHMM. Les comportements au niveau supérieur sont les comportements tels que "aller tout droit," "tourner à gauche," etc. Chaque comportement de niveau supérieur est composé des sous comportements.

Pour chaque comportement au niveau supérieur, il existe un GP correspondant à une distribution gaussienne sur la trajectoire typique pour ce comportement (section I). La distribution gaussienne sur la trajectoire future (section I est obtenue en transformant d'abord les observations vers un espace canonique dans lequel le GP canonique réside. La transformation est conformal qui utilise une approche moins carrée pour approximer la transformation conformal sous la forme dun maillage 2D où la transformation de chaque triangle du maillage est affine. La distribution de probabilité sur le mouvement futur est obtenue dans l'espace canonique à partir du GP canoniques. La transformation inverse est, ensuite, appliquée pour obtenir la distribution de probabilité sur le mouvement dans le futur dans le repère global pour l'évaluation de risque.

Le risque est évalué à basé de la distribution de probabilité gaussienne sur le mouvement futur pour des comportements différents, et les comportements sont estimés à partir du LHMM. Il a été également souligné que, grâce à la combinaison de processus gaussien et le LHMM pour modéliser le mouvement aux différents niveaux sémantiques, il existe de nombreuses manières différentes d'évaluation de risque, chacun ayant ses sémantiques associées et qui dépend aussi de l'application nécessitant la valeur du risque.

Part II

Introduction

Chapter 1

Introduction

1.1 General Context

The development of autonomous vehicles garnered an increasing amount of attention in recent years. Currently, the application of autonomous vehicles has been receiving particular attention from the defense and automotive industry. The interest from the defence industry is obvious from the DARPA Urban Challenge. Autonomous vehicles provides unmanned or remotely controlled combat ground vehicles as a technological force multiplier. The interest for automotive industries is to produce safer and more user friendly cars. A common reason behind most traffic accidents is the failure on the part of the driver to adequately monitor the vehicle's surroundings and consequently make the correct decision. An in vehicle system capable of warning or intervening with the appropriate actions can potentially reduce a large number fatal accidents.

(a) DARPA Urban Challenge (Carnegie Mellon)

(b) Project INTERSAFE

Figure 1.1: Examples of advanced driver assistance or autonomous vehicles

Although there is currently no commercially available autonomous vehicle, driving assistance technologies such as cruise control systems, lane departure warning or automated parking features are now available for consumers. Such driving assistance systems can be viewed as a natural progression towards fully autonomous vehicles.

This thesis is situated within the context of driving assistance. A crash warning system is desirable where it acts as a watchdog, warning drivers of potentially collisions which might be overlooked by human drivers. The aim is thus to passively warn drivers if the vehicle they are driving is about to collide within the next few seconds. It will not actively take control of the vehicle to avert the potential crash.

1.2 Problem Description

The main problem of this thesis concerns the estimation of the risk of collision of a vehicle. From the driver's point of view, the driver can obtain a general indication of the risk of collision for the next few seconds, warning the driver of unnoticed risks. The estimated risk of collision can also be used to aid an autonomous vehicle in choosing a suitable trajectory to minimize its risks.

For a completely autonomous vehicle, or even for a crash warning system, estimation of the risk of collision is a component of the complete system. The estimation of the risk of collision receives a set of processed sensor information from other modules of the complete system and it outputs risk values, which is to be interpreted by the application in context.

Throughout this thesis, the following set of processed sensor inputs are assumed to be available:

1. **Road geometry:** In order for the risk estimation to be aware of the road constraints, it must have geometrical information such as the width of the road and its curvature. Such information can be obtained from specific algorithms which processes raw information from camera images or lidars. Alternatively, it is also possible to obtain road geometry information given a Geographic Information System (GIS) with a pre-built map and a localization device such as the GPS.

2. **Target tracking:** The estimation of collision risk necessitates the detection and tracking of moving obstacles. The position and velocity of the moving obstacles can then be obtained.

3. **Detailed specified sensors:** Some examples of additional information which are not crucial but desirable are information such as the detection of the status of the signal lights of other vehicles as it is a strong indicator of the intention of other moving vehicles. It is also possible to have additional "virtual" sensors coming from further processed raw sensory data. An example would be the distance of the vehicle to the left or right lane border, which might indicate intentions to perform a lane change. Such information is highly informative in driving behaviour recognition.

A vehicle for which we shall call the ego vehicle is assumed to be equipped with the appropriate sensors so as to obtain the set of processed sensor inputs mentioned above. In this thesis, the estimated risk is a numerical value which expresses quantitatively the risk of the ego vehicle going into collision with another vehicle in the next few seconds.

Estimating the risk of collision in the future involves the construction of models describing vehicle motion in the sensor visibility range of the ego vehicle. Furthermore, this model should be capable of reasonably predicting future vehicle states. It is only with a prediction on future vehicle states that it is possible to estimate the risk of collision in the future.

When reasoning about the future, it is sensible to describe the future in terms of probability. A strong argument for using probabilities, as will be discussed in chapter 3, is that the foundations of probability provides the only reasonable way of manipulating beliefs in a coherent and consistent manner.

In this thesis, we present a fully probabilistic vehicle evolution model for obtaining and inferring beliefs on the future states of vehicles in urban traffic environments. Consequently, the estimated risk of collision can be obtained from the models in terms of probability in a theoretically consistent manner.

1.2.1 Towards Better Collision Warning

Current commercially available crash warning systems are mostly aimed at preventing front, rear, or side collisions. Such systems are usually equipped with radar based sensors on the front, rear or sides to measure the velocity and distance to obstacles. The algorithms for determining the risk of collision are based on variants of time-to-collision (TTC) (LEE76). TTC is basically a function of two objects, giving the time remaining before an object enters into collision with the other assuming that the two objects maintains the same linear velocity. Some systems are not passive but rather, it intervenes by directly controlling the brakes and possibly the steering to effectuate the necessary corrective actions. Systems based on TTC are based on the fact that observations are made at a reasonably high frequency in order to adapt to potentially changing environments.

Current commercial systems works reasonably well on automotive highways or certain sections of the city where roads are straight. However, it might be misleading in situations where the roads are curved and thus the assumption that motion is linear does not hold (see figure 1.2). In these situations, the risk tends to be underestimated. Furthermore, instances of roads which are not straight can be commonly found in urban environments, like the roundabout or cross junctions.

Several research projects were created to overcome such problems by taking into account the structure of the environment especially in intersections where there is a higher rate of accidents. These projects aims to provide intersection collision warning systems where there wireless communications are either between vehicles, or by using road side infrastructure such as traffic light information (PJL$^+$00) (Adm04) (FJ07). Each of these systems have vehicles equipped with a pair of detectors (either radar sensors or laser scanners) at the left and right front corners of the vehicles in order to detect cross-traffic vehicles at intersections.

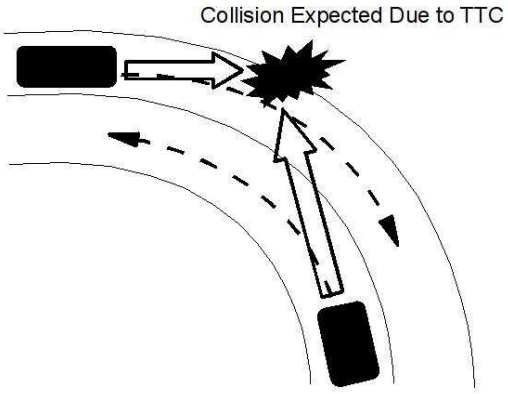


Figure 1.2: Example of false collision alarm due to linear assumption of TTC based systems. Actual path of vehicles as dotted arrows.

The speeds and hence TTC of the obstacles are then evaluated to determine the risk of collision. Although the environmental structures are taken into consideration when evaluating the risk of collision, the actual calculation of the risk of collision is still based on the assumption of linear motion. The time horizon of risk prediction is short and crucial environmental information and information on sensors are not fully utilized.

An argument of this thesis is that simply knowing that there is an object at a certain location at a specific instance in time does not provide sufficient information to asses its safety. A framework for understanding behaviours of vehicle motion is indispensable. In addition, environmental constraints should be taken into account especially for urban traffic environments. An overview of the proposed approach is presented in the next section.

1.3 Approach

A bottom up approach towards the final goal of constructing a model to estimate the risk of collision for a vehicle is presented. The simpler case of "free" motion is first presented. In this situation there are no environmental constraints or explicit behaviour to be taken into account unlike urban traffic road conditions.

Next, the environmental constraints of urban traffic roads and vehicle driving behaviour are introduced. The simpler model provides the foundation for describing physical realization of trajectories. From this basis, a method of adapting the simpler model to road constraints and driving behaviour modelling is introduced to arrive at a complete probabilistic vehicle evolution model. The estimated risk of collision can then be calculated from the probabilistic vehicle evolution model.

1.3.1 Without Constraints

For the simple case without constraints, motion in environments such as a wide open hall is considered. More specifically, a human motion model is constructed which can be viewed as a reduced version of the probabilistic vehicle evolution model without constraints and behaviours.

The motion model is based on the stochastic Gaussian Process (GP). Thanks to the GP model, there is no longer a need to deal with discretization issues, while retaining a mathematically consistent probabilistic representation for each motion pattern.

The GP is basically a Gaussian distribution over functions where a sampled function from this GP can be analogously viewed as a motion sequence. The prototype motion pattern (or mean function of the GP) is assumed to vary according to the Gaussian distribution. Thus, a motion pattern can be viewed as a Gaussian distribution in a higher dimensional space. A scenario consisting of several motion patterns can be easily modelled as a Gaussian Mixture Model.

1.3.2 Adding Constraints

Figure 1.3 gives an architectural overview of our approach for estimating risk of collision for an ego vehicle. The context in which the work is situated can be found within the dotted box. The complete details of the inner workings concerning the contents of the dotted box can be found in chapter 7.

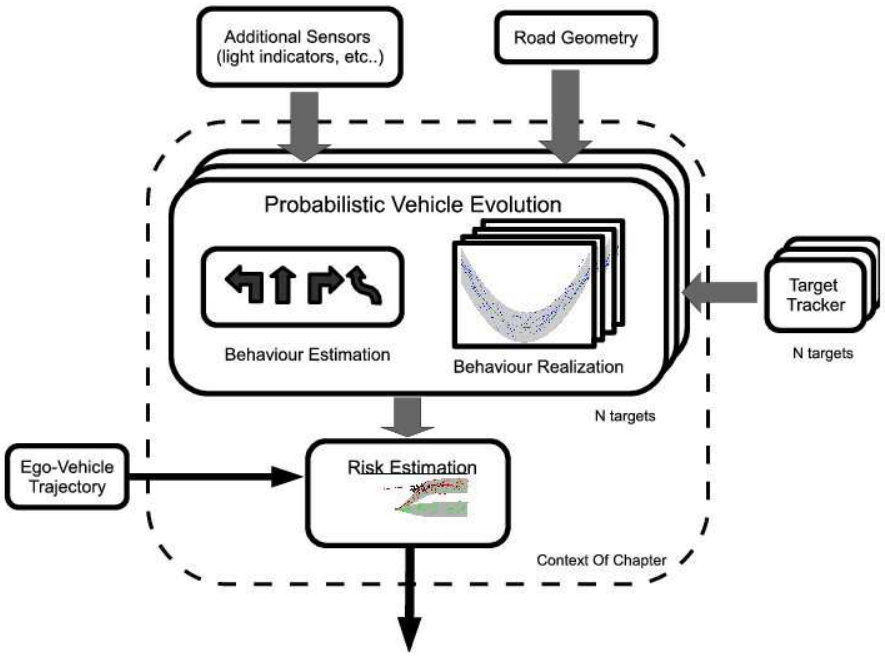


Figure 1.3: Overall View Of Vehicle Risk Estimation

For each vehicle in the scene, we have a probabilistic vehicle evolution model, and an associated target tracker. A complete probabilistic vehicle evolution model consists of behaviour estimation and behaviour realization. Behavior estimation consists of making use of processed sensor data, i.e. light indicators, distance of vehicle to lane borders, to estimate if the tracked vehicle is executing behaviours such as turning, overtaking or going straight. The behaviours are estimated by decomposing the behaviours where each behaviour is defined by a sequence of finer lower level states which describes the behaviour. The high and low level behaviour states are then jointly estimated using a variant of the hidden markov model.

For each high level behaviour, the behaviour realization module describes the physical trajectory realization of the behaviour. The distribution over physical trajectory realization corresponding to each behaviour is represented using a Gaussian Process (GP).

A naive implementation of GP similar to the one for open spaces (section 1.3.1) works well only for an environment of limited geographical area. As the environment gets larger, or even if the road network topology gets more complicated, the number of GPs will grow. We propose to cope with this scaling issue by taking advantage of the repetitive structures of roads. A GP which we call the canonical GP is defined in a separate space with a fixed frame of reference. This canonical GP can then be transformed to fit the geometrical constraints of the road. The least squares conformal map (LSCM) is used to perform this transformation to minimize distortion locally.

A probability distribution over behaviours and a probability distribution over physical trajectory realizations of behaviours gives the full probabilistic vehicle evolution model for tracked vehicles in the environment. The probabilistic vehicle evolution model is used in the estimation of risk. The risk value obtained is a probability measure. We consider the risk for a certain vehicle which we shall call the ego vehicle.

At the very basic level, the risk of a pre-determined trajectory of the ego vehicle is estimated, using the probabilistic vehicle evolution model for all other vehicles around the ego vehicle. An application of such a risk value can be used in the feedback loop of an autonomous robotic ego vehicle for example. However, the computation of risk can be generalized where a variety of different risk values with varying semantics can be computed, depending on the requirements on how the risk values are going to be used. For example, we can compute the risk against individual vehicles, risk dependent on the time horizon before collision or behaviour related risks such as the risk associated with the ego turning left, overtaking etc.

1.3.3 Intuition

An example of a possible scenario with two lanes both in the same direction as illustrated in figure 1.4. Two vehicles A and B are travelling on separate lanes and the risk of collision is to be estimated for vehicle B. From the driver of vehicle A's point of view, the local road structure is implicitly described by manoeuvres such as going straight, turning right/left, lane change. Such manoeuvres shall be referred to as behaviours. The total set of possible behaviours are pre-defined. However, not all behaviours are available at all instances. For

example, it might not be possible to turn left at the next intersection because there is no road turning left. The set of feasible behaviours at each instance is a subset of all the possible behaviours.

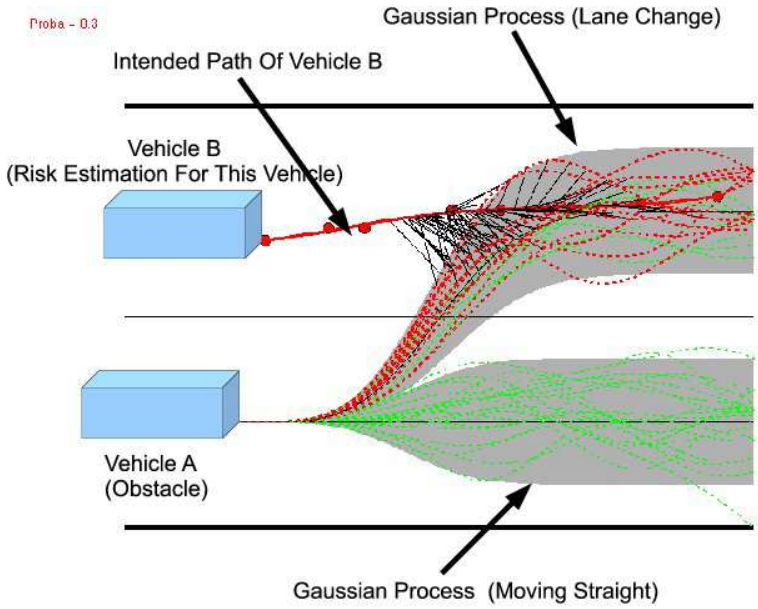


Figure 1.4: Risk to be estimated of a trajectory to be taken by vehicle B. Path prediction for vehicle A (obstacle) is obtained by sampling from the GPs (one each for going straight and lane changing). The risk of collision is calculated by a weighted sum of trajectories in collision.

For each feasible behaviour, there are a number of different ways of physically executing the behaviour. Humans do not drive in an absolutely straight manner, precisely following the middle of the lane. However, it is reasonable to assume that a normal driving routine approximately follows the lane. The lane following for a given behaviour is represented using a GP which gives a probability distribution over the possible future physical realizations of the paths where the mean will be the path following exactly the middle of the lane. The GPs and its variances are shown as grey regions in figure 1.4 corresponding to behaviours lane changing and going straight. Dotted lines represents the path sampled from the GP. For cases where the road has a non zero curvature or for turning behaviours, the GP will be appropriately adapted according to the geometry of the road.

The set of GPs for each of the feasible behaviour in the scene, in combination with the probability that vehicle A executes a certain behaviour, gives a probabilistic model of the future evolution of vehicle A in the scene.

Similar to the TTC evaluation of risk of collision, the evaluation of the risk of collision will be for vehicle B against vehicle A. In contrast to TTC where the collision is estimated

assuming a single linear future trajectory for both vehicles A and B each, we evaluate the risk of collision of the intended trajectory to be taken by vehicle B against all possible trajectories to be taken by vehicle A. The risk value will then be a weighted combination of the single intended trajectory of vehicle B against the possible trajectories for vehicle A. The weights are assigned according to the probabilistic model for the future evolution of vehicle A behaviours.

1.3.4 Challenges

Modelling motion patterns using GPs with constraints presents a set of different challenges and difficulties in contrast to when the constraints are not present:

1. As presented in chapter 6, motion patterns were modelled in a fixed frame of reference for unconstrained spaces. If the same methods were to be naively applied to the problem of modelling motion patterns on roads, the model complexity grows as the size of the road network grows. As motion patterns can begin from any point along a stretch of road, and additionally, from any point of the road network to another, the number of possible motion patterns grows combinatorially.

2. Describing motion patterns in a fixed frame of reference also means that it is valid only for a certain group of road networks within a geographical zone. The same motion patterns cannot be used in a new geographical area with a different road network configuration.

3. A fixed frame of reference also means that relevant calculations must be performed in the global fixed frame. There is consequently a need to perform localization which can be complicated and sometimes unnecessary since the risk of collision is always evaluated with respect to a vehicle.

4. Road structures are not fully taken advantaged of.The description of motion patterns in a fixed global reference frame also describes implicitly the road network configuration. The local road structures can be described using verbs such as going straight, turning left, turning right etc.. Having a local road description is compact. Furthermore, a global road description might not be really necessary as the sensors installed on vehicles for the detection of obstacles have a limited range and field of view.

1.4 Contribution

The main contribution of the thesis is the estimation of collision risk of an ego vehicle, while taking into the behaviour of the other vehicles and respecting the geometrical constraints of urban traffic scenarios. The following highlights its key points:

1. Collision risk estimation is taken one step further by being more “environmentally aware” where environmental structures and constraints are explicitly taken into account in a hierarchical way.

2. Driving behaviours have a great influence in estimating collision risk more accurately. It is taken into account explicitly in our risk estimation model.

3. Our approach is able to give semantics which corresponds to a human user's intuitive notion of risk. This is especially useful when Human Machine Interfaces (HMI) are taken into account for driving assistance applications.

4. Intuitive semantics such as turning left/right, are translated into geometry, taking into account road structure and topology.

5. Our framework is able to scale to large environments by exploiting the environment structures which are semantically repetitive. Furthermore, the framework is able to adapt dynamically to the geometry of the environment.

6. When talking about estimating the risk of collision, there is an implicit reference to prediction. The notion of "maybe" is rigorously encoded in the calculus of probability.

Another contribution is a probabilistic motion model based on Gaussian Process. Its key points are:

- State-space models requires the discretization of the space in which motion takes place. The choice of discretization is not evident and is often predetermined and fixed. However, GP does not have this issue.

- The GP is fully probabilistic and allows us to express mean and variances at any time along the motion pattern. As a result, it is able to perform probabilistic inferences on future motion or calculate the likelihood of a trajectory observation belonging to a motion pattern in a probabilistically consistent way. This is contrast to existing methods (see chapter 2) where distances between two trajectories are defined in an ad-hoc manner.

1.5 Document Organization and Outline

Part One: Background & State Of The Art

Motion Models and Prediction. This thesis is focused on modelling motion patterns. Chapter 2 presents a survey of the current state of the art .

Probabilistic Models. Chapter 3 presents the probability formalism and Bayesian models which are the foundations for probabilistic motion models.

The Gaussian Process and Hidden Markov Model. As our probabilistic vehicle evolution model is based on the Gaussian Process and Hidden Markov Model, we consecrate a chapter to introduce these two models in chapter 4.

Part Two: Proposed Models & Algorithms

Introduction to Part II. Chapter 5 explains the structure for part II. In part II, a bottom up approach is adopted where the easier problem of motion modelling without constraints is first presented. The basic motion model is then adapted to the structural constraints of urban traffic conditions in evaluating collision risk.

Motion Without Constraints. The problem of motion without constraints imposed by the environment is presented in chapter 6. An intuition of how GPs describe a motion pattern and its generalization to more complex situations involving several motion patterns are presented. Based on this model, we present a variational inference algorithm to recover the motion patterns from a set of training data, consisting of a set of observations corresponding to motion sequences.

Collision Risk Estimation. Motion modelling in structured spaces is given in chapter 7, with particular emphasis on estimating collision risk of a vehicle in an urban traffic environment. We present a complete and fully probabilistic framework for adapting GPs to model vehicle evolution. The evaluation of risk and its generalizations are discussed.

Part Three: Experiments

Motion In Open Spaces Without Constraints. We present in chapter 8 experimental results on recovering motion patterns in an open space, notably based on the data set of human motion in the entry hall of INRIA Rhône Alpes.

Path Planning With Risk Estimation. An example of the application of GPs as a prediction model to guide a car-like robot in navigating around safely is presented in chapter 9. The path planning algorithm is based on a probabilistic extension to Rapidly-exploring Random Trees (RRT) developed by (FTSL08) and experiments are jointly conducted with the author of the path planning algorithm.

Collision Risk Estimation in Driving Assistance. Two sets of experiments are provided in chapter 10, which takes place within the context of estimating collision risks for driving assistance. The first experiment is a monte carlo based simulation for evaluating and validating the accuracy and reliability of GPs in estimating collision risks. The second experiment brings the risk evaluation to a more realistic environment. Due to practical considerations in evaluating collisions, a virtual urban traffic environment is created where vehicles are driven by humans to add realism. Our risk estimation framework is then evaluated within this experimental setting. The second experiment requires a lot of effort and is jointly conducted with Toyota Motors Europe (TME) and technology enterprise ProBayes.

Part Four: Conclusion

The thesis ends with a summary of the work and a discussion of our approach in chapter 11. We also discuss some of the shortcomings of our work, giving suggestions for possible future extensions.

Part III

Background & State Of The Art

Chapter 2

Motion Models and Prediction

An important component of estimating collision risk in the future involves constructing a motion model and its ability to predict its position in the future. This chapter presents the current state of the art related to motion modelling and prediction.

2.1 Introduction

Motion prediction as understood by most of the literature takes on a different meaning from motion prediction in the probabilistic vehicle evolution model of this thesis. To compare and contrast, we start by first describing motion prediction in the traditional sense.

Motion Model In The Traditional Sense

For the majority of the current literature, the prediction of movement takes place on a time scale of several seconds at the most. These predictions often models the kinematics and dynamics of the moving object in question in order to arrive at an estimate on the state of an object in the very near future. Its purpose is mainly to serve as an initial guiding estimate in applications such as target tracking especially when there are occlusions, or when filtering and control is required.

Prediction in this form often manifests itself in the form of the *Kalman Filter* (Kal60) or its extensions such as the *Extended Kalman Filter* (JU97) and the *Unscented Kalman Filter* (WM01). Its *Sequential Monte Carlo* (cf section 3.6.2) counterpart, the *particle filter* (AMGC02) or more commonly known in the computer vision community as the *condensation filter* (IB98) have became increasingly popular in recent years.

The *Bayesian Filter* is a generalization of these filtering methods for prediction and illustrates clearly the short time scale of the prediction. A Bayesian Filter is iterative and can be decomposed in two steps, the prediction and estimation step (see figure ref-fig:bayesfilter).

Prediction Step: The prediction is performed by "propagating" the state based on the motion model, $P(x_t|x_{t-1})$ and then marginalize over the probability distribution of

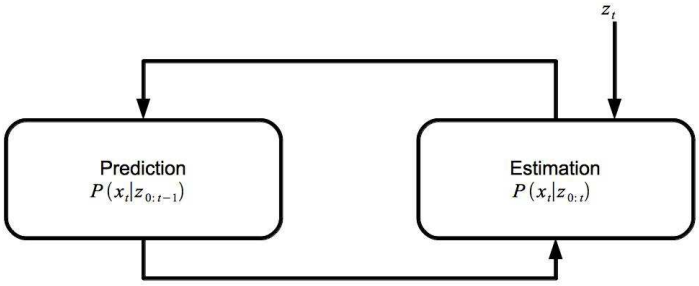


Figure 2.1: Schema of the iterative process of the Bayesian filter. Consists of the filtering step and estimation step.

the state of the previous instance. This is also commonly known as the *Chapman-Kolmogorov* equation (Pap91).

Estimation Step: Given the current prediction of the state of the object from the previous state of the object, it can now be updated by factoring in the observation at the current time t by the product of $P(z_t|x_t)$ which is the observation model which gives the likelihood of the current observation z_t when at the predicted state x_t.

Prediction is performed only for the next time step. It is possible to iterate the prediction T times successively in order to produce a prediction T time steps ahead. However, this is clearly insufficient as it is based on the assumption that the kinematics and dynamics will not change. Unfortunately, this assumption does not hold in most realistic environments as T increases. There are many applications in tracking or prediction which violates this assumption as there is often an agent (robots or humans) "in control" of the motion of the associated object.

Motion Model As Implied In The Thesis

Current commercial Time To Contact (TTC) based solutions when estimation collision risk takes on a motion model as described previously. Such solutions assumes that the velocity or acceleration remains constant. However, this assumption is violated in urban traffic scenarios because collision risks can come from situations such as around a corner or along a curved stretch of road.

This thesis proposes a probabilistic vehicle evolution model which overcomes these limitations. Doing so implies that the motion model describes the future motion on a longer time scale. Such motion model can either be obtained algorithmically from a set of

motion data sequence, automatically inferring typical motion, or by exploiting the structure of the space containing the motion.

This thesis is hence concerned with motion on a longer time scale, where influences on motion are not kinematically or dynamically driven. Rather the interest is on motion driven by intentions often by a driving agent, virtual or human. Our proposed probabilistic vehicle evolution model used to evaluate collision risk is based on this motion model. This chapter presents the current literature for describing these types of motion.

2.1.1 Organization

A review of the different methods for longer term motion model is presented. Existing methods for recovering such models relies on a wide variety of representation and learning methods but can generally be classified into two categories; by extracting motion patterns or what will be called *trajectory exemplars* in this thesis; and a *state space model* representation of the environment.

The organization of this chapter is as follows:

- Section 2.2 describes methods related to motion models at the trajectory level where it is represented as a motion pattern (or exemplars) consisting of a motion prototype.
- Section 2.3 describes methods where a motion model is represented as transition between states. The states can be low level, where states are physical location or high level, where states takes on semantics such as a room or a kitchen.
- Section 2.4 presents methods that do not fit into the previous two categories.
- The chapter ends with a discussion.

2.2 Trajectory Exemplars

As objects probably do not move in a totally random fashion, such motions exhibit patterns and it is natural to represent each motion pattern with a trajectory exemplar.

2.2.1 Trajectory Exemplar Representations

Most typical trajectory representations takes the form of a list of a sequence of points $x_t \in \mathbb{R}^n$ which are usually samples of observed positions along the trajectories at time instant t.

However, there were several alternative representation of a trajectory exemplar proposed. The main difference amongst the different representations lies in the representation of the different representative exemplars and the description of the possible variations of trajectories belonging to the same exemplar.

Primitive Descriptions

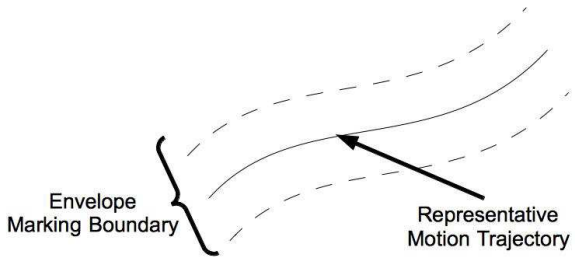


Figure 2.2: Schema of primitive description of motion based on a representative trajectory and its boundary marked by an envelope.

The simplest representation takes the form of a central spline (usually a polyline) which is the representative trajectory. Variations about the central spline are demarcated by an envelope representing the extrema of the observed trajectories belonging to the exemplar (ME02) (see figure 2.2). A similar representation was used in (JJS04). The envelopes are defined as being the normal direction of the nodes of the central spline. Although its advantage lies in its simplicity, it does not induce representations of trajectory densities within the envelope. Neither does it describe the characteristics of the trajectories within the envelope, i.e. if it is wavy etc .

Another simple way of describing trajectory exemplars will be to have only representative trajectories and merging them when they are sufficiently close together (KmG01). A way of looking at it is as if the trajectory densities of a certain scene are represented by samples from the trajectory densities.

Merging is beneficial especially when short fragmented trajectories are observed separately due to problem such as occlusions or bad object results. The merging is done by taking the "averages" of the two trajectories overlapping when the overlapping regions are sufficiently close together. (KmG01) has shown better tracking results with the aid of these representative trajectories by assuming that an object will always follow one of these representative trajectories. However, such a method does not scale well as the number of representative trajectories generally increases with number of trajectories observed.

Cluster Based

Clustering algorithms can be naturally applied in settings where there are several trajectory exemplars. The problem of clustering in this context will be to group similar trajectories together to form sets of trajectory exemplars.

The *hierarchical clustering* algorithm is used to perform such clustering in (Buz04) using the longest common subsequence to measure similarity; and (VGF04) using the euclidean distance. The *k-means* clustering algorithm was applied in a hierarchical fashion in (HXF+06) to create a hierarchical structure. Observed trajectories are first clustered based on spatial information before the same k-means clustering is used to create sub-clusters, this

time based on temporal information, within each cluster. Each trajectory is sub-sampled in order to produce trajectory representations of equal length for all trajectories.

The *k-medoids* algorithm has also been used to recover trajectory exemplars (RRRS). (CCP07) also presented a different representation of trajectory where each trajectory is defined as a sequence of headings. Each trajectory is then modeled using a mixture Von Mises distribution. The Von Mises distribution (Bis06) is sometimes called "circular normal" and is more suitable for modelling directions than the Gaussian distribution. The k-medoids algorithm is used to cluster the trajectories based on the Bhattacharyya distance which measures the distance between two distributions. Each of the headings which defines a trajectory is assumed to be independent and identically distributed, i.e. $P(T) = \prod_i \theta_i$ where T is trajectory and θ_i its headings.

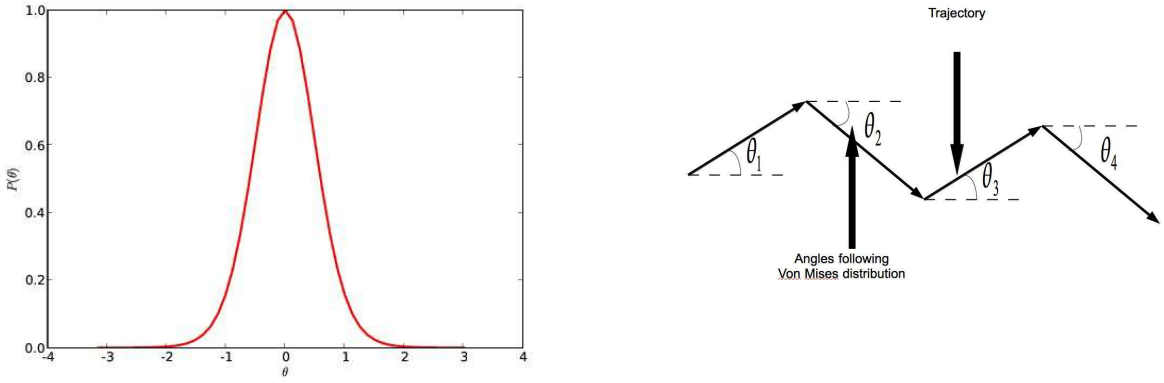


Figure 2.3: (a) Example of a probability density plot of a von mises distribution. θ is the random variable for heading. (b) Motion described probabilistically with independently and identically distributed heading .

However, having the headings independently sampled from the mixture of Von Mises distribution leads to degenerate cases due to the fact that the distribution of the headings θ_i are not dependent on time. An example of a degenerate case will be to have a trajectory drifting left then right. However, another trajectory which drifts right first then left will give the same probability distribution of headings for the two trajectories.

(JJS04) proposed a method of performing clustering based on *graph-cut algorithms* (BK01). A *complete graph* (whenever two nodes of the graph are connected via an edge) is constructed where each node represents an observed trajectory (a sequence of positions in time) and the edges indicates the *Hausdorff distance* between the two trajectories. The min-cut max-flow algorithm is then used to recursively split the trajectories into clusters. The envelope representation (c.f. section 2.2.1) is used to define each clusters where the extreme trajectories in the cluster defines the envelope.

The graph cut formulation provides a discriminative method for performing clustering. However, the criteria for when to stop the recursive splitting is difficult to define

methodically.

(IG07) proposed a similar clustering methodology based on the complete graph representation. However, instead of the min-cut max-flow algorithm it uses the *Normalized Cuts (NC)* spectral clustering algorithm (FBCM04) to group tracks into clusters. In (IG07), the NC spectral clustering algorithm uses the spectrum of the data similarity matrix to perform dimensionality reduction before the k-means algorithm is used to cluster the data points in the spectral embedding space. The k-means clustering algorithm is also used in (SHT+07) where trajectories are represented using the Hidden Markov Model (see also sect. 2.3.1 for more details).

2.3 State-Space Models

State-space models are characterized by a discretization of the intermediary states of a trajectory motion. Very often, each quanta of the discretization is a state and a trajectory can be probabilistically represented as a *Markov chain*. In a Markov chain, transitions between states are probabilistic and described by a *transition probability matrix* for discrete states with assumptions on the dependence of the temporal states known as the *markov assumption*. The markov assumption basically states that the values in any states are only influenced by the states that directly precedes it. The number of preceding influencing states are known as the *order* of the markov chain. Due to computational reasons, most practical implementations, such markov chains are of order 1.

As markov chains discretize the states, it does not include the notion of uncertainty. More elaborate state-space models probabilistically associates a certain observation to a state-space. In this case, the state of the markov chain is not observed directly and is thus hidden. Such markov chain models with probabilistic association of observations to hidden states are called *Hidden Markov Models (HMM)* (Rab89). Such models have been popular in recent years due to the fact that it is able to express the uncertainty of movements in a probabilistic framework and are mostly based on variants of markov models.

State-space models in trajectory modelling falls generally into two sub-categories. The first is the low level state space models which usually associates an observation of a position along a trajectory with a discretization of the state-space. The second state-space based approach has a more abstract notion of state, i.e. intention, places etc.

2.3.1 Low Level State-Space Models

The most direct method of modelling will be to discretize the states which are positions in the real world. And the transition between these states gives a trajectory (see figure 2.4). A HMM can be used in this case where observations of a position along the trajectory can be probabilistically associated with the hidden states. In (SHT+07) a trajectory is defined in such a way. *Multi-dimensional scaling* is used to perform dimensionality reduction before a k-means clustering is used with a distance function (JR85) which measures the probability distance between HMMs.

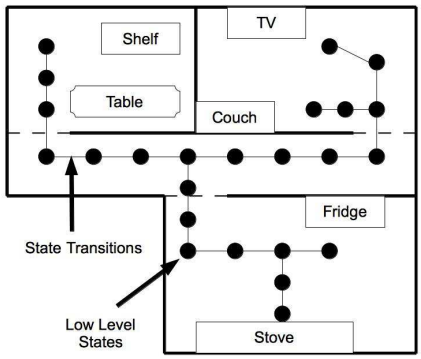


Figure 2.4: Low level state space model. Each state represents a discretization of space and a motion consists of transition between the states.

(BBCT05) adopts an approach in the inverse manner where clustering is performed before learning the HMM parameters. Each trajectory is represented by a fixed length sequence of positions obtained from real data via interpolation.The *expectation-maximization (EM)* algorithm (DLR77) is used to cluster the trajectories using the *Bayesian Information Criterion (BIC)* to score each model in order to obtain the best number of clusters according to the BIC. A HMM is then learnt for each cluster where each HMM state is a discrete point position in space and the probability of an observation corresponding to the HMM state is Gaussian distributed and is the same for each state.

2.3.2 High Level State-Space Models

High level state-space models have states which are more abstract semantically instead of having the states representing discretized positions. Some high level state-space models are hierarchical with a different semantic at each level of hierarchical structure. Such models are motivated by the observation that complex human activities possesses a natural hierarchical decomposition (see figure 2.5).

(NPVB05) proposed a two level *Hierarchical Hidden Markov Model (HHMM)* for modelling activities in a room. The top level consist of complex behaviours which are actions of visiting a sequence of landmarks, i.e. a complex behaviour "short meal" includes visiting landmarks such as the cupboard, go to the fridge then the stove. The bottom level consist of primitive behaviours which are actions involving going from one landmark to another. However, there is no attempt at explicitly modeling trajectories as they only work on the semantic level, requiring that a movement is just a physical displacement from one landmark to another.

A similar hierarchical model with semantics attached at each level was also proposed by

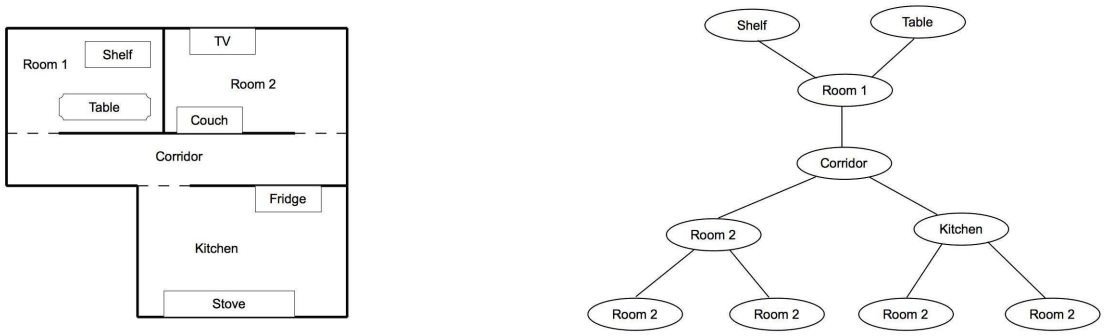


Figure 2.5: (a) A schema of an indoor environment. (b) A hierarchical representation of the states for the indoor environment.

(LPFK07), with the application of inferring transportation routines in a city. It is divided into 3 layers where the top layer models the markov transitions between goals in a city, the second layer models the switching of the mode of transportation along the journey and the bottom layer models the location and velocity of the person which is tracked by a GPS. In this work, the motion of the person is always assumed to be along pre-defined edges which represents the segments taken during the journey to the goal.

The semantic of states in high level state-space models are not just restricted to the behaviours in a hierarchical fashion. The states can also represent a geometric entity. In (HBLC08), a trajectory is represented as a sequence of point positions which are basically observations from a tracking module. Kernel smoothing is applied to the set of point positions to obtain a smooth approximation of the trajectory from which a sequence of curvature values can be obtained from the smooth trajectory approximation. A HMM is then defined where state spaces are the set of discretized curvature values and the class of trajectories can be described by the markov transitions between the curvature states. The number of hidden curvature states are determined empirically and the observation probability distribution of a curvature given its state is Gaussian.

2.4 Other Models

The two previous sections (2.2 and 2.3) introduced two common frameworks for representing motion in spaces. However, the existing literature contains a variety of different methods for performing trajectory learning and prediction which might at times fall partially into one of the two previously mentioned common frameworks. But this chapter highlights certain different characteristic.

2.4.1 Structured Environment Approximation

Sometimes, it might be possible to simply the representation of trajectories normally as edges or lines connecting two places. The reasons might be that learning and prediction at the trajectory level in detail in order to predict future position coordinates is not required like in (LPFK07) (c.f. 2.3.2).

The other reason for having a simplistic trajectory representation is due to the topology of the environment where such environments are highly structured, i.e. roads or inside buildings. (WN08) represented a network of roads with a graph structure. Vehicles are always assumed to travel along the edges of the graph. A probability distribution function representing velocity profiles along each edge of the graph are collected from historical data. The collision risk or Time To Interaction (TTI) is computed by taking current position of vehicles and propagating the positions of these vehicles while making the assumption that they respect the velocity profile of the edges.

A similar model is proposed by (LFH$^+$03) for learning the motion model of people to improve tracking in the context of the interior of buildings. For the case of interior of buildings, the motion space is highly structured as it consists mostly of rooms and corridors. Such a topological representation can be easily constructed and represented using a voronoi graph of the environment (see figure 2.6). The expectation-maximization algorithm (c.f. section 3.6.2) is then used to learn the parameters of the motion model.

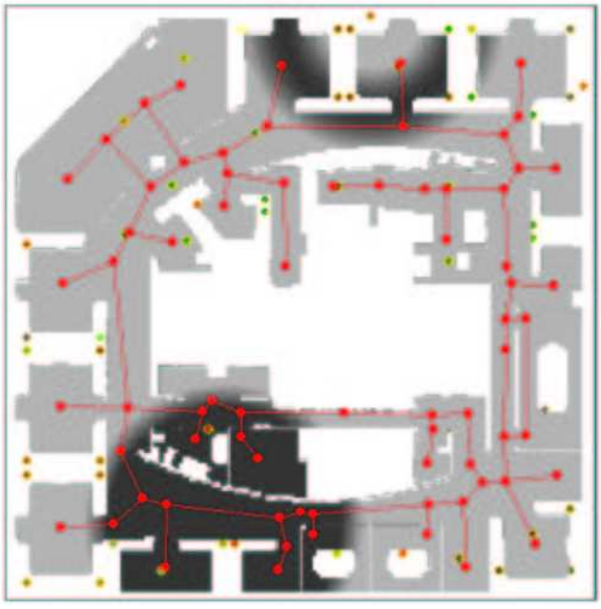


Figure 2.6: A voronoi graph (in red) of the indoor environment (from (LFH$^+$03)).

2.4.2 Classification Models

Some methods for learning trajectories performs classification instead of clustering. Essentially, both classification and clustering seek to partition the data set coherently where every pair of members from each partition are similar. Classification belongs to the class of supervised learning algorithms and learns from a data set consisting of labels.

However clustering is a form of unsupervised learning and does not usually have a set of pre-specified labels in advance and the aim of clustering is to to group similar data points together based on a similarity measure.

Most trajectory learning approaches which performs classification are mainly for applications involving motion pattern recognition and usually as no facility for prediction. A classification method based on the SVM was proposed in (PF07) (c.f. section 4.3.4). Each observed trajectory is evenly sub-sampled at its positions to produce vectors of the same dimension. The SVM is then trained on these trajectories which are deemed normal and the trained SVM can be used to detect anormal trajectories. SVMs have also been used in (HBLC08) (c.f. section 2.3.2) where the vectors for classification are the parameters of the HMM which describes a trajectory. Before training of the SVM, the set of labels (each label representing a predetermined trajectory class) and the training data labels were specified in the training data.

(SF08) performed classification using the Gaussian Mixture Model. Each observed trajectory is described by a vector of control points of a B-Spline curve where the B-Spline curve is fitted to the sequence of observed positions of the trajectory. A semi-supervised learning method was used where at any time during the training sequence, a user can intervene to indicate if the observed trajectory is normal or not. The learning algorithm incrementally models the distribution over the observed trajectories as a Mixture of Gaussians over the vector space of B-Spline control points. The occasionally user labelled trajectories are used to update the Mixture of Gaussians. A threshold is set such that if an observed trajectory is too far away from the Mixture of Gaussians, it will be then recognized as an anormal trajectory.

Working with camera based images permits a richer description of trajectories from the norm. In (SEG00), each trajectory is represented as a sequence of location, speed, size, direction. Furthermore, the trajectory description includes description information such as the object and the binary motion silhouette. Rich trajectory description of such forms is able not only to differentiate movements in space, but associate categories of objects to the kind of movements each type of object makes.

With the set of rich trajectory descriptions each as a vector, it is partitioned into its different classes by *Vector Quantization (VQ)*. VQ produces a codebook where all trajectory vectors from the same partition are considered similar and hence represented by a prototype trajectory vector. A hierarchical classification tree is then constructed by searching for co-occurrences between the rich trajectory vectors. The resulting classification tree constructed is able to refine classification based on different criterias found within each rich trajectory description. For example, it might be able to perform a classification at the first level of the tree based on the silhouette then going further down the tree based

on the size, then color histogram etc. This is in contrast to state-space models where the appearance of an object is able to narrow down the class of motion patterns it might execute in the future without observing a partial sequence of its motion.

2.4.3 Neural Network Inspired Models

Neural Networks are made of interconnecting artificial neurons which mimics the biological neurons. They are biologically inspired architectures for solving AI problems without necessarily creating a model of a real biological system. (HXT04) proposed a hierarchical self-organizing neural network for learning trajectories. The first network learns the distribution of flow vectors from the trajectories where a flow vector consist of position and displacements, i.e. (x, y, dx, dy). The distribution of flow vectors are then fed into the second network with output neurons corresponding to trajectories. Side links were introduced between neurons to the first network which can be self organized during learning. It chains up the neurons each representing a flow vector to represent a trajectory.

(JH96) used the same flow vector representation with competitive neural networks. The first network learns the distribution of flow vectors by performing vector quantization. The output trajectory distribution is modelled using another competitive neural network with a leaky neuron layer (WA90) connecting the two networks.

A drawback with such architectures by having neurons representing trajectories or flow vectors is that the number of neurons must be fixed a priori before training, which is a fairly strong assumption. To overcome this limitation, several authors proposed the use of a self organizing neural network known as the *Growing Neural Gas* (GNG) (Fri95). GNG is represented as a graph with the advantage that it is able to adapt itself to the topology of the space it is embedded within and adjusts the number of nodes automatically according to some pre-specified threshold.

The GNG is used to perform vector quantization in order to generate a cookbook, i.e. a collection of prototypes. Each prototype can correspond to a position, a position and velocity or its variants. (BBSP06) represented prototypes as positions. And paths are defined as a sequence of prototypes. The paths are formed by drawing observed trajectories randomly without replacement and then associate the observed trajectories to a sequence of prototypes. Similar prototype sequences below a certain threshold are then merged together.

Instead of representing paths as a sequence of prototypes, (VGF07) represented motion as transitions along the edges of the graph of the GNG from one prototype (node) to another. Transitions between the nodes are stochastic and the HMM algorithm is adapted where the hidden states of the HMM are the nodes of the GNG and edges of the GNG are constraints on the transition between the nodes which is markovian (c.f. section 2.3). Learning is performed on-line which basically amounts to counting the number of transitions between nodes via edges.

2.5 Discussion

The methods discussed previously each has its own merits and disadvantages. We consider several issues related to models for describing motion of vehicles in urban traffic environments; namely uncertainty, discretization and scalability.

2.5.1 Representing Uncertainty

In constructing a model for describing future motion, it is sensible to reason in terms of uncertainty. Furthermore, it is important to be able to express the uncertainty of future motion given previously observed motion. Unfortunately, most of the existing methods are unable to do so.

Structured approximations of the environment is clearly unable to meet the criteria. This is due to the fact that the environment is often approximated by exploiting its structure to obtain a reduced topological representation.

Exemplar based methods by its definition are not capable of expressing uncertainty as its motion model is represented with a prototype trajectory and the boundaries to trajectory variation about the prototype are often marked geometrically. In the limit, exemplars obtained via clustering are able to represent uncertainty of a certain motion example belonging to one of the clusters. However, such a representation is unable to express the uncertainty of future motion based on previously observed motion. The situation is the same for classification based methods.

In contrast, state space models are able to this criteria. A probability distribution can be placed around each low level state space node and the transitions between nodes can be stochastic. They unfortunately suffer from the problem of discretization which will be presented next.

2.5.2 Discretization Issues

The issue of discretization is an issue when predicting future motion. State space models involves the discretization of state space (also called nodes) and motion is expressed as transition between nodes. The problem of discretization surfaces when we wish to obtain the position of an object at a certain time instance in the future to evaluate the collision risk, but its position at that particular instance falls in between the nodes.

Interpolating estimated risk or positions between nodes might solve the problem. However, the choice of interpolation is highly dependent on the size of between each discrete step. The size between discrete steps are often chosen arbitrary or experimentally.

2.5.3 Scalability

Scalability is about model complexity as the environment gets increasingly large or complex. A simple case of bad scalability is the flow vector representation mentioned in section

2.4.3. As the environment grows, the number of discretized positions to represent the flow vector grows as well.

The other categories presented grows in model complexity as the environment gets increasingly complicated. An example of a complicated environment is a city with a complicated network of roads and many combination of roads leads from a certain point in the city to another. It is the number of ways to travel between two points in the city that adds complexity to the model.

2.5.4 Comparison

The following table summarizes the different properties of the current state of the art with respect to the three issues mentioned previously:

	Uncertainty	Discretization Required	Scalability
Exemplars	✗	✗	✗
State Space	✓	✓	✗
Environment approximation	✗	✓	✗
Classification	✗	✗	✗
Neural Network	✗	✓	✗

Table 2.1: Comparison of the different categories of methods.

In the current state of the art, only state space models are capable of representing uncertainty. We consider that exemplar and classification based methods do not require discretization. This is because these methods are dependent on the choice of trajectory model and includes cases without discretization. We consider models such as splines to contain no discretization. Scalability is by far an issue which has not been addressed by any of the current methods.

2.5.5 Our Proposed Approach

Our proposed probabilistic vehicle evolution model addresses the issues mentioned. It consists of a mixed state space and exemplar based representation. The states are high level states corresponding to vehicle related behaviours i.e. turning left, overtaking. The transition between behaviours are modelled as a variant of the Hidden Markov Model (HMM). The trajectory realizations for each of the possible behaviours is modelled as a Gaussian Process (GP). Each GP is an exemplar motion pattern for its corresponding behaviour. The proposed methods for addressing each of the issue are:

- **Discretization:** The GP representation does not face issues of discretization and is capable of inferring the probability distribution over future positions in a consistent way.

- **Uncertainty:** The probabilistic vehicle evolution model is fully grounded on probability and hence expresses uncertainty about the state of vehicles in the future. The uncertainty over future motion given previously observed motion can be obtained with out model.
- **Scalability:** Our model exploits the repetitive structure present in urban traffic environments. These structures are implicitly encoded by the notion of behaviours. At the lower geometrically level, we introduce a method of adapting the GP accordingly. Doing so enables the model to be flexible enough to represent the environment at the local level, and as a result scale to large or complicated environments.

Chapter 3

Probabilistic Models

The probability formalism and Bayesian models forms the core of the techniques used in this thesis. This chapter presents an introductory overview of the related techniques. The interested reader is invited to consult the pedagogical book by Bishop (Bis06) and E.T.Jaynes (Jay03) for additional details.

3.1 Introduction

Estimating the risk of collision involves predicting the future. As one can ever be sure about the future, the probability formalism offers a theoretically grounded and consistent way to reason about the future. Our probabilistic vehicle evolution model, as the name implies, is based on probability which quantifies uncertainty about the future and is hence Bayesian.

We have already seen an example of a Bayesian model in chapter 2 in the form of a Bayesian filter. The Bayesian filter updates its uncertainty on the state of an object upon repeated observations. The probabilistic vehicle evolution model consists of behaviour estimation and trajectory realizations of behaviour. Behaviour estimation is similar to Bayesian filters where observations are used to update behaviour uncertainty. The Gaussian Process model for trajectory realizations expresses its uncertainty as a Gaussian distribution over trajectory realization.

3.1.1 Organization

This chapter introduces the background in probability theory and related Bayesian models required for our proposed models of motion and for estimating collision risks. The organization of this chapter is as follows:

- Section 3.2 describes the Bayesian interpretation of probability and its contrast to the frequentist interpretation.

- The choice of using probabilities to represent uncertainty is not ad-hoc. It is justified in section 3.3.
- Section 3.4 introduces the basics of probability and its rules.
- Section 3.5 presents graphical models. It is useful for visualizing probabilistic models. Furthermore, efficient algorithms of inference are devised with the help of graphical models. Graphical models are also known as Bayesian networks in some literature.
- Section 3.6 describes the various algorithms for probabilistic inference. Notably, an overview of exact and approximate inference methods which are used in our probabilistic motion models are presented. The section also provides the foundation for approximate variational methods which are used for modelling motion without constraints (chapter 6).

3.2 Subjective & Objective Probability

Bayesian probability is different from an orthodox point of view in that probabilities are interpreted as a measure of uncertainty or "state of knowledge" instead of the interpretation that probability is the limit of its relative frequency of an event occurring in a large number of trials. Hence, it is also commonly known as being *subjective* whereas the frequentist approach is *objective*.

The point about frequentist probability is that as it is impossible to observe an event an infinite number of times, different relative frequencies appears in different series of observations. Sometimes, it is even difficult or nearly impossible to repeat an experiment.

For example, it might be difficult to estimate the probability of the existence of extraterrestrial lifeforms. However, we might have some opinion on it and is subjective from person to person according to the person's state of knowledge. Such preconceptions from the Bayesian point of view are known as priors. Furthermore, the state of knowledge of a person should be updated accordingly in light of new evidence. The Bayesian interpretation of uncertainty as probability provides an elegant and rational way.

3.3 Logic as Probabilities

It has been shown by E.T.Jaynes (Jay03) that by respecting the elementary desiderata of consistence in inference proposed by Cox (Cox61), the only reasonable way of manipulating these beliefs and to guarantee coherence is by conforming to the laws of probability. The elementary desiderata by Cox is also known as the Cox axioms. It can be stated informally as follows:

- Degrees of beliefs are represented as real numbers.
- The beliefs and its updates do not contradict common sense.

- Consistency:
  - If there are more than one ways of arriving at a conclusion, then each way should give the same conclusion.
  - All relevant evidence are taking into account of.
  - Equivalent states of knowledge are represented by equivalent degrees of beliefs.

Another way of justifying the use of probabilities to represent the degrees of beliefs in a consistent manner comes in the form the *Dutch Book Theorem*. Suppose that if one is willing to place bets according to one's degrees of beliefs. And if the beliefs and its manipulation do not respect the laws of probability, there exists a set of simultaneous bets ("Dutch Book") which will be acceptable but is guaranteed to lose money, whatever the outcome it may be.

3.4 Probability Fundamentals

As stated previously in section 3.3, the manipulation of beliefs are consistent only when beliefs respect the laws of probability.

Definition 3.4.1. A **probability space** is an ordered triple (Ω, ε, P) where

(i) Ω is a set. It is also known as the sample space.

(ii) ε is a family of subsets of Ω. ε is a σ-algebra over Ω; The elements of ε are stable under complementation and over countable unions.

(iii) P is a function which maps an element of ε to values in $[0, 1]$

and that the following conditions hold:

- $P(\varnothing) = 0$
- $P(\Omega) = 1$
- $\forall e \in \varepsilon,\ P(e) \geq 0$

3.4.1 Basic Probability

$P(x)$ represents the belief or plausibility in the proposition x. When $P(x) = 0$ it simply means that x is definitely not true and $P(x) = 1$ means that x is definitely true. A conditional dependence $P(x|y)$ is the belief that x is true given that y is true.

One point to take note of with conditional independence is that $P(x|y)$ does not imply the logical statement "y implies x" with a certain belief. The logical statement applies regardless of other information in hand. However, the probability statement is even more precise in the sense that y implies x with belief $P(x|y)$ when the only thing we know is y.

If any other information z is available which might influence x, the probability statement should be $P(x|y,z)$. The only exception is when z is conditionally independent of x given y i.e. $P(x|y,z) = P(x|y)$. This point was emphasized by Pearl (Pea88).

Independence. Two random variables X and Y are independent (sometimes written as $X \perp Y$) if and only if:

$$P(X,Y) = P(X)P(Y) \tag{3.1}$$

Marginal Probability.

$$P(X) = \sum_{y \in Y} P(X,y) \tag{3.2}$$

Conditional Probability.

$$P(X|Y) = \frac{P(X,Y)}{P(Y)} \tag{3.3}$$

Where $P(y) \neq 0$ for otherwise $P(X|y)$ will be undefined.

Product Rule can be easily obtained from conditional probability (eqn 3.3). It is also known as the chain rule.

$$P(X,Y) = P(X|Y)P(Y) = P(Y|X)P(X) \tag{3.4}$$

Bayes' Theorem. From the product rule (eqn 3.4) and conditional probability (eqn 3.3) we get Bayes' Theorem.

$$\begin{aligned} P(Y|X) &= \frac{P(X|Y)P(Y)}{P(X)} \\ &= \frac{P(X|Y)P(Y)}{\sum_{y' \in Y} P(X|y')P(y')} \end{aligned} \tag{3.5}$$

3.5 Graphical Models

Graphical models are a fusion of probability theory and graph theory. It expresses dependencies between random variables in a probabilistic model where nodes normally representing random variables with links between the nodes representing their dependencies. There are mainly three advantages in using graphical models. Firstly, a graph representation gives an intuitive model for visualizing the probabilistic model. Secondly, efficient message passing algorithms for inference in probabilistic models are devised with the aid of the graph representation. Finally, it embeds the notion of modularity and a complex system can be built by composing the final model with its simpler sub-modules.

There are three different kinds of graphical models found in the literature; Undirected, directed and factor graphs. We present the undirected and directed graphical models which are the most common.

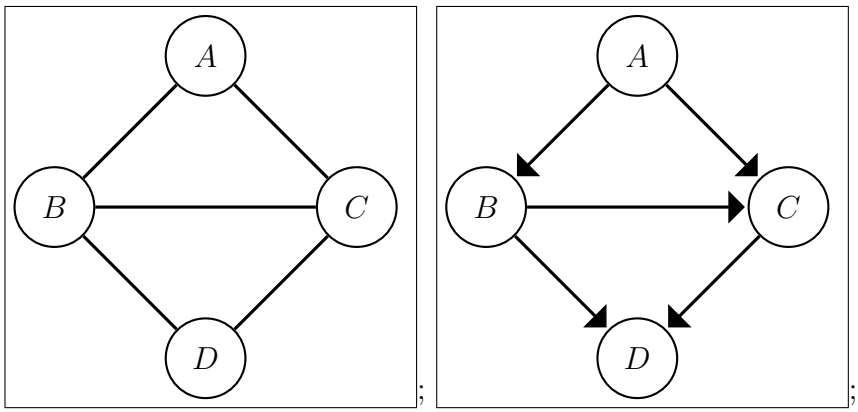


Figure 3.1: An undirected (left) and directed (right) graphical model

1. **Undirected Graphical Model:** In an undirected graphical model, each node represents a random variable and the edges indicate conditional independence relationships:

   Definition 3.5.1. (Conditional Independence) Two sets of random variables $\mathbb{X}$ and $\mathbb{Y}$ in an undirected graphical model are conditionally independent given random variables $\mathbb{Z}$ ($\mathbb{X} \perp \mathbb{Y}|\mathbb{Z}$) only if all paths leading from $\mathbb{X}$ to $\mathbb{Y}$ go through nodes in $\mathbb{Z}$.

   Using the example in figure 3.1, definition 3.5.1 indicates that $A \perp D|\{B, C\}$. The factorization of the joint probability distribution of all random variables in an undirected graph can be written as the product of non-negative functions of variables in the maximal cliques for the graph (Hammersley-Clifford Theorem (Bes74)). As in the example, the probability distribution can be written as:

   $$P(A, B, C, D) = cf_1(A, B, C)f_2(B, C, D)$$

   Where f_1 and f_2 are known as potential functions. c is constant which ensures that the probability distribution is normalized.

2. **Directed Graphical Models** Directed graphical models are sometimes also known as directed acyclic graphs *(DAGs)*. Like undirected graphical models, DAGs have nodes which represents random variables but its edges are directed and represent statistical dependencies. In general its factorization of the joint probability distribution is the product of conditional probabilities of each node given its parents:

   $$P(X_1, \ldots, X_N) = \prod_{i=1}^{N} P(X_i|X_{pa_i}) \tag{3.6}$$

   where X_{pa_i} denotes the random variables which are parents of X_i in the DAG. Conditional independence relationships in DAGs are less apparent than directed graphical

models. The separation between the variables and the direction of the edges have to be taken into account. A graphical test for evaluating the conditional independence of two sets of random variables given a third set of random variables is performed using *d-separation* (Pea88). A minimal set of random variables that d-separates variable X from the rest of the graph consists of the parents, children, and the parents of the children of X. This minimal set is also known as the Markov boundary of X.

3.6 Probabilistic Inference

This section starts with by presenting a simple framework for inference under the probabilistic framework. Presented with a data set $D = \{x_1, x_2, \ldots, x_N\}$, a statistical *model* is constructed to explain the probability distribution of D. Very often, the data D is assumed to be independent and identically distributed *(iid)*, and hence there are no ordering presented in the data D. A model simply explains the distribution of the data points $\mathbf{X}$, i.e. $P(\mathbf{X}|D, \theta)$ the probability distribution of the random variable $\mathbf{X}$ given the vector of parameters for the model θ and data D.

A well defined Bayesian model requires a prior distribution over the model parameters $P(\theta)$ and the *predictive distribution* of random variable $\mathbf{X}$ is weighted by the posterior distribution over the parameters:

$$P(\mathbf{X}|D) = \int P(\mathbf{X}|D, \theta)P(\theta)d\theta \tag{3.7}$$

Sometimes, it might be troublesome or difficult to represent the entire distribution over the parameters, $P(\theta)$. Instead, a point estimate of the parameter $\hat{\theta}$ can be chosen:

$$\begin{aligned} P(\mathbf{X}|D) &= \int P(\mathbf{X}|D, \theta)\delta(\hat{\theta} - \theta)d\theta \\ &= P(\mathbf{X}|D, \hat{\theta}) \end{aligned} \tag{3.8}$$

where $\delta(x)$ is the Dirac delta function. There are two common methods in the search for the point estimate $\hat{\theta}$. The first is known as the *maximum a posteriori* where the point estimate is:

$$\begin{aligned} \hat{\theta}_{MAP} &= \arg\max_{\theta} P(\theta|D) \\ &= \arg\max_{\theta} \left[\log P(\theta) + \sum_{n} \log P(x_n|\theta) \right] \end{aligned} \tag{3.9}$$

Another choice would be the *maximum likelihood (ML)* estimate where the prior is not

taken into account:

$$\begin{aligned} \hat{\theta}_{MAP} &= \arg\max_{\theta} P(\theta|D) \\ &= \arg\max_{\theta} \sum_{n} \log P(x_n|\theta) \end{aligned} \tag{3.10}$$

The ML estimate (eqn. 3.10) is a frequentist method as it does not require a prior over the parameters. The problem with ML estimation is over-fitting. Over-fitting is when a model becomes over complex to fit the training data but is not desirable as it does not generalize well. In order to overcome over-fitting, frequentist sometimes add a *regularization function* to equation 3.10 which penalizes a certain characteristic such as model complexity (GJP95). If this regularization term is a proper probability distribution, then it is equivalent to the MAP procedure. One difference between the MAP and ML procedure is that the MAP estimate is not invariant to reparametrization unlike for ML. This is because the penalized ML is a function and not a probability distribution and does not change with the Jacobian of the reparametrization.

3.6.1 Exact Inference

One advantage of using a statistical model is the ability to ask questions on any subset random variables $\mathbf{X_Q}$ conditioned on observed values of another subset of random variables $\mathbf{X_{obs}}$. This can be expressed by conditional probability (eqn. 3.3) while marginalizing out (eqn. 3.2) out other variables $\mathbf{X_{oth}}$:

$$P(\mathbf{X_Q}|\mathbf{X_{obs}} = x_{obs}) = \frac{\sum_{\mathbf{X_{oth}}} P(\mathbf{X_{obs}} = x_{obs}, \mathbf{X_{oth}}, \mathbf{X_Q})}{\sum_{\substack{\mathbf{X_{oth}} \\ \mathbf{X_Q}}} P(\mathbf{X_{obs}} = x_{obs}, \mathbf{X_{oth}}\mathbf{X_Q})} \tag{3.11}$$

Probabilistic inference is thus essentially the computation of large sums or integrals. It is exponential in the number of variables. Even if the probability distribution is continuous, there is no guarantee that the integrals are tractable.

Most algorithms for computing these sums and integrals exploit the structure of the graphical model representation to compute the solution efficiently. However, the problem is fundamentally NP-hard. There are three common algorithms in exact inference; variable elimination, belief propagation and junction trees. Belief propagation and junction trees are special cases of variable elimination. We present variable elimination here. The remaining algorithms can be found in appendices A.1 and A.2.

Variable Elimination Taking the directed graphical model of figure 3.1 as an example, we can ask for the distribution of a certain random variable D for example:

$$
\begin{aligned}
P(D) &= \sum_{B,C,A} P(B|A)P(C|A,B)P(A)P(D|B,C) \\
&= \sum_{B,C} P(D|B,C) \sum_{A} P(B|A)P(C|A,B)P(A) \qquad (3.12)
\end{aligned}
$$

From equation 3.12, we can see that the number of sums and products can be reduced simply by "pushing" in the sums as far as possible. This property of the distributivity of sums and of products is the underlying mechanism behind variable elimination (MA00). The elimination of random variables via the sums transforms the graph by eliminating the node being summed and adding edges between all nodes in the markov boundary of that same node. The order of elimination does not change the answer, but its computational complexity. The choice of elimination order is NP-complete and most implementations resort to heuristics in elimination order such as number of neighbours of node (*maximum cardinality search* or least numbers of edges added (*minimum deficiency search*).

3.6.2 Approximate Inference

In slightly more complicated graphical models, inference becomes quickly intractable. There are several reasons why models becomes intractable quickly. Sometimes, models with discrete hidden variables which can have an exponential number of combinations. A latent variable model is rather common and unfortunately intractable. For example, given observations y and latent variables x, the marginal distribution of y would be:

$$P(y) = \int \int P(y, x|\theta)P(\theta)dxd\theta \qquad (3.13)$$

The marginalization over latent variables x and model parameter θ can possibly be high dimensional. Latent variables can also be combinatorial in nature and there are often no analytical solutions to the integrals. For these cases, approximate inference algorithms are needed to approximate the solutions. There are two main branches of approximate inference algorithms.

Monte Carlo Approaches

Monte Carlo approaches have been popular for evaluating integrals in high dimensions. As the name suggests, Monte Carlo methods are about stochastic approximations of an integral using sampling:

$$\int F(x)P(x)dx \approx \frac{1}{T}\sum_{t=1}^{T} F(x^{(t)}) \qquad (3.14)$$

where $x^{(t)}$ are *iid* samples drawn from the distribution $P(x)$. Monte Carlo methods are unbiased and the variance of its evaluation is inversely proportional to the number of particles and is independent of dimension. However, the main difficulty in practice is that the distribution $P(x)$ of eqn. 3.14 is difficult or impossible to sample from directly. Methods such as *rejection sampling* or *importance sampling* are some of the methods to overcome such limitations by approximating $P(x)$ with an alternative distribution $Q(x)$ which one can easily sample from and adjusting the samples accordingly.

Another set of Monte Carlo methods to approximate $P(x)$ are *Markov Chain Monte Carlo (MCMC)*. Unlike other approaches where samples are draw independently, MCMC methods performs sampling on a markov chain where each sample $x^{(i)}$ depends on the previous value in the sample $x^{(i-1)}$. The aim is to construct a sequence samples $x^{(i)}$ from a distribution $P_i(x)$ such that it converges to some desired distribution $P^*(x)$.

The transition probability between the states along the markov chain is defined by a transition kernel from state x to state x'; $T(x \rightarrow x')$ with the property':

$$P_t(x') = \sum_x P_{t-1}(x) T(x \rightarrow x') \tag{3.15}$$

The *detailed balance* condition is important in assuring that the markov chain convergences to a stationary/equilibrium distribution:

$$P^*(x')T(x' \rightarrow x) = P^*(x)T(x \rightarrow x') \tag{3.16}$$

Another property of such markov chains in MCMC is that it should converge to the same stationary distribution regardless of initial conditions; $\lim_{t\rightarrow\infty} P_t(x) = P^*(x)$. Such a markov chain is thus *ergodic*. A sufficient condition for ergodicy in markov chains is when:

$$T^K(x \rightarrow x') > 0 \; \forall x, x' \; where P^*(x') > 0 \tag{3.17}$$

There are a variety of MCMC methods in the literature. Some notable examples includes Metropolis-Hastings (Has70) Gibbs sampling (GG84), Reversible Jump MCMC (Gre95) and Hybrid Monte Carlo (Nea99).

Variational Approaches

Variational methods are another class of methods to perform inference. Instead of performing stochastic approximations by sampling, variational methods approximates the true posterior distribution with another set of distributions that are deemed appropriate. The aim of Variational inference is to search for the minimum *Kullback-Leibler divergence* between the two distributions.

Definition 3.6.1. (Kullback-Leibler(KL) Divergence) The KL divergence sometimes known as *relative entropy* between two probability distributions p and q is given by:

$$KL(p\|q) = -\int p(x) \ln \left\{ \frac{q(x)}{p(x)} \right\} dx \tag{3.18}$$

Variational Bayes The Variational Bayes method works by maximizing the lower bound of the marginal likelihood. Given a probabilistic model with model parameters and latent parameters given by the set of random variables Z, the set of observed variables X, the marginal log probability of the model can be expressed as:

$$\ln P(X) = \mathcal{L}(Q) + KL(Q\|P) \tag{3.19}$$

$$\mathcal{L}(Q) = \int Q(Z) \ln \left\{ \frac{P(X,Z)}{Q(Z)} \right\} dZ \tag{3.20}$$

$$KL(Q\|P) = \int Q(Z) \ln \left\{ \frac{P(Z|X)}{Q(Z)} \right\} dZ \tag{3.21}$$

Where $\mathcal{L}(Q)$ (eqn. 3.20) is a lower bound for the log marginal probability (eqn. 3.19). Since the KL divergence is never negative, $\mathcal{L}(Q) \leq \ln P(X)$. In fact, Variational Bayes is a generalisation of the *Expectation-Maximization (EM)* algorithm (DLR77). In the EM algorithm with hidden variables and parameters, the lower bound expressed as:

$$\mathcal{L}(Q,\theta) = \int Q(Z) \ln \left\{ \frac{P(X,Z|\theta)}{Q(Z)} \right\} dZ \tag{3.22}$$

EM is an iterative algorithm which alternates between the expectation step and maximization step till it converges:

- **E-Step:** In the expectation step, the lower bound (eqn. 3.22) is optimized with respect to the distribution over hidden variables, $Q(Z)$ while holding the parameters θ fixed. Similarly to equation 3.19, the lower bound $\mathcal{L}(Q,\theta)$ has largest value only when$Q(Z)$ is equal to $P(Z|X,\theta)$ and the KL distance is zero.
- **M-Step:** In the maximization step, the lower bound from the E-step is maximized with respect to the parameters θ while holding the distribution over hidden variables, $Q(Z)$, fixed. This has the effect of increasing the lower bound and correspondingly increases the marginal log likelihood.

The EM algorithm is thus an iterative procedure which optimizes the probability distribution over hidden variables and then increases the lower bound via optimization over parameters. It has also been proven that the EM iterations never decrease the log likelihood (DLR77).

Variational Bayes provides a framework for generalizing the EM algorithm even for cases when $P(Z|X,\theta)$ is computationally intractable. The most common form of approximations is to partition the hidden variables into M disjoint sets Z_i thus given the factorization:

$$Q(Z) = \prod_{i=1}^{M} Q_i(Z_i) \tag{3.23}$$

This is also sometimes called a *mean field* approximation. The main difference is that the factored variational E-Step itself is iterative. In each iteration of this E-Step, the

lower bound is maximized with respect to a particular $Q_i(Z_i)$. The different $Q_i(Z_i)$ can be updated iterative during the E-Step till convergence before going on to the M-Step. However, it is also possible to just make a single update for each $Q_i(Z_i)$ before the M-Step. The maximization of the lower bound with respect to a single $Q_i(Z_i)$, gives the following update equation:

$$\begin{aligned} Q_i^*(Z_i) &\propto \int \ln P(X, Z|\theta) \prod_{i \neq j} Q_j(Z_j) dZj \\ &= E[\ln P(X, Z|\theta)]_{\prod_{i \neq j} Q_j(Z_j)} \end{aligned} \tag{3.24}$$

The final algorithm for the factored variational EM is as follows:

Algorithm 2: Factored Variational EM Algorithm

```
1 while Not Converged do
2     forall Q_i(Z_i) do
          /* Maximize L(Q, θ) with respect to Q_i(Z_i)        */
3         Q_i^*(Z_i) ∝ ⟨ln P(X, Z|θ)⟩_{∏_{i≠j} Q_j(Z_j)};
      /* Maximize the parameter θ with respect to L(Q, θ)     */
4     θ^new = arg max_θ L(Q, θ);
```

Another related method for performing variational inference is known as Expectation Propagation (EP) (Min01b) (Min01a). Informally speaking, EP minimizes the KL divergence $KL(q\|p)$ instead of $KL(p\|q)$. However properties and effects of EP versus variational Bayes are quite different. More details can be found in appendix A.3 and Bishop (Bis06).

Chapter 4

The Gaussian Process and Hidden Markov Model

4.1 Introduction

The schematic of our probabilistic vehicle evolution model is illustrated in figure 4.1. It consists of a behaviour estimation and a behaviour realization sub-module. Behaviour estimation is based on a variant of the Hidden Markov Model. Behaviour realization is on the representation of the probability distribution over trajectories in the form of Gaussian Process. As such, we devote this chapter to introduce the foundations to Hidden Markov Model and Gaussian Process.

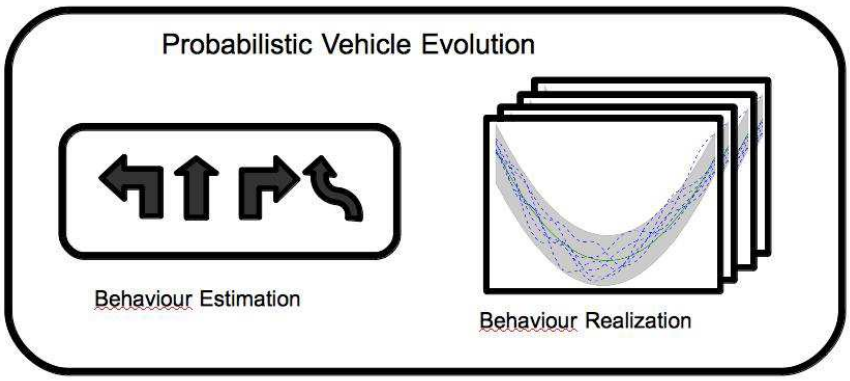


Figure 4.1: Schematic diagram of the probabilistic vehicle evolution model.

Organization

This chapter is organized as follows:

- Section 4.2 presents the basic Hidden Markov Model and typical inference and learning algorithms for it in sections 4.2.1 and 4.2.2.

- Section 4.3 gives the background behind Gaussian Process, and its relations to other models (section 4.3.4). The Gaussian Process regression model used in learning motion from a data set is presented in section 4.3.3. Finally,

4.2 The Hidden Markov Model

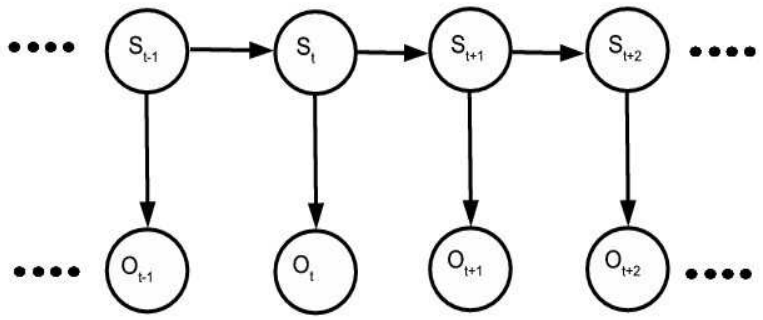


Figure 4.2: Standard Hidden Markov Model (HMM). O_t are observation variables at time t. S_t are hidden variables at times t.

A Hidden Markov Model (HMM) (Rab89) is a stochastic finite automaton, where each hidden state produces an observation and is not directly observable. The hidden states are discrete and finite, $S_t \in \{1, \ldots, K\}$. A complete HMM can be described with the following:

- *Transition Model:* The transition model describes the probability of entering a certain state given the previous states. It can be represented as a *stochastic matrix* $\mathbf{A}$. An example of a first order HMM where each $i-j$ entry of the stochastic matrix has the value $\mathbf{A}_{ij} = P(S_t = j|S_{t-1} = i)$. In general, a HMM of order N has transition probabilities conditional on the N previous hidden states, i.e. $P(S_t|S_{t-1}S_{t-2}\ldots S_{t-N})$.
- *Observation Model:* The observation model gives the probability distribution over observations for each hidden states, i.e. $P(O_t|S_t)$.
- *Initial State Distribution:* The *a priori* probability distribution over the initial states. It is represented as a vector π where each element $\pi_i = P(S_0 = i)$

The joint probability distribution of the HMM can thus be written as follows:

$$P(S_{0:T}O_{1:T}) = \pi \prod_{i=1}^{T} P(S_t|S_{t-1})P(O_t|S_t) \tag{4.1}$$

4.2.1 Inference In The HMM

HMMs have been employed in diverse fields such as speech recognition (Rab89) or text processing.(GY04) The majority of the different applications of HMMs can be summarised into a few "standard" inference problems:

1. *State Estimation:* Computes the belief over the current hidden state given a sequence of observations, $P(S_t|O_{1:t})$. State estimation can be performed in a recursive manner using the Chapman-Kolmogorov equation at each recursion:

$$P(S_t|O_{1:t}) \propto P(O_t|S_t) \sum_{S_{t-1}} P(S_t|S_{t-1}) P(S_{t-1}|O_{1:t-1}) \tag{4.2}$$

   Equation 4.2 has a similar form to that of a Bayesian filter (see chapter 2). The terms within the summation of equation 4.2 is equivalent to the prediction step and the multiplication with the term $P(O_t|S_t)$ constitutes the estimation step of the Bayesian filter.

2. *State Prediction:* As the name suggests, state prediction calculates the probability distribution of hidden state in the future, i.e. $P(S_{t+k}|O_{1:t})$ where $k > 0$. The predictive distribution over states can also be computed recursively:

$$P(S_{t+k}|O_{1:t}) = \sum_{S_t} P(S_{t+k}|S_t) P(S_t|O_{1:t}) \tag{4.3}$$

   where

$$P(S_{t+k}|S_t) = \sum_{S_j, j=t+1 \ldots t+k-1} \prod_{i=1}^{k} P(S_{t+}|S_{t+i-1}) \tag{4.4}$$

   $P(S_t|O_{1:t})$ corresponds to state estimation (eqn. 4.2) and equation 4.4 can be pre-computed as it is translation invariant in t.

3. *State Smoothing:* Smoothing serves to better estimate the probability distribution of a certain state, S_k given a set of observations in the future, $O_{k+1:t}$ as well as the past, $O_{1:k}$, $1 < k < t$:

$$P(S_k|O_{1:t}) \propto P(S_k|O_{1:k}) P(O_{k+1:t}|S_k) \tag{4.5}$$

   Each term of the decomposition of equation 4.5 can be defined recursively as well.:

$$\begin{aligned} \alpha(S_k) &= P(S_k|O_{1:k}) && (4.6) \\ &= P(O_t|S_t) \sum_{S_{t-1}} P(S_t|S_{t-1}) \alpha(S_{k-1}) && (4.7) \end{aligned}$$

$$\begin{aligned} \beta(S_k) &= P(O_{k+1:t}|S_k) && (4.8) \\ &= \sum_{S_{t+1}} P(O_{k+1}|S_{k+1})P(S_{k+1}|S_k)\beta(S_{k+1}) && (4.9) \end{aligned}$$

Hence for smoothing, two recursive passes are required, the forward pass (eqn 4.7), and the backward pass (eqn 4.9). A recursive formulation can be translated to a dynamic programming solution which avoids recalculating sub-solutions via memoization. The overall cost for evaluating the whole chain is $O(K^2N)$ for a chain of length N with K hidden states.

4. *Most Probable Hidden State Sequence:* One of the most popular questions posed in a HMM model is to find the most likely hidden state sequence given its observations, i.e. $\arg\max_{S_{1:t}} P(S_{1:t}|O_{1:t})$. It can be solved with the *viterbi* algorithm. The *viterbi* algorithm works analogously to the forward pass, $\alpha(S_k)$ of equation 4.7. Instead of marginalizing over the previous hidden states, the maximum value over previous states is taken:

$$\begin{aligned} \delta(S_k) &= \max_{S_{1:k}} P(S_{1:k}|O_{1:k}) && (4.10) \\ &\propto \max_{S_{1:k}} P(S_{1:k}O_{1:k}) && (4.11) \\ &= \max_{S_{k-1}} P(S_k|S_{k-1})P(O_k|S_k)\delta(S_{k-1}) && (4.12) \end{aligned}$$

Like the forward and backward pass methods, the *viterbi* algorithm is implemented using dynamic programming. The sequence of most likely hidden state can be reconstructed by tracing back through the dynamic programming matrix. Similarly to state smoothing, the computational complexity is $O(K^2N)$.

4.2.2 Learning The HMM

The learning problem in HMM is about adjusting the parameters to the HMM, so a given training set is best represented by the model and its parameters in the best way for the intended application. A typical training set consists of a sequence of hidden states and observations. The goal is to find the parameters of the HMM, more specifically, the transition matrix of the HMM hidden states and the observation model. The EM algorithm is popularly used for learning HMM parameters and it is also commonly known as the *Baum Welch* algorithm (Rab89).

The learning of the lower level HMMs can be performed separately by running the *Baum Welch* algorithm against a different training data set each. For example, the lower level HMM corresponding to overtaking is trained on a set of data containing overtaking

sequences, the lower level HMM corresponding to turning left is trained on a set of data containing turning left sequences, etc.

Thanks to the layered HMM, learning in the HMM can be decoupled. The *Baum Welch* algorithm can be used for training the lower layer HMMs and the upper layer HMM separately. An advantage to doing so will be that the lower layer HMMs, which are more sensitive to environmental changes can be retrained, without the need to retrain the upper layer HMM.

4.3 Gaussian Process

The *Gaussian Process (GP)* is a probabilistic model applied to a wide range of problems in regression and classification. It permits the Bayesian use of kernels for learning and is simple to implement. However a serious problem with GP is that a naïve implementation has a complexity of $O(n^3)$ where n is the number of training input points. This is mainly due to calculations involved in inverting the covariance matrix of a Gaussian distribution.

Historically, GP is not a recent topic. It has been around since the middle of the 20^{th} century under time series analysis (Wie64). Its applications are well known in the geostatistics field notably in spatial statistics (Whi84) (Rip81) (Cre93). Under the geostatistics field, it is commonly known as *kriging*. GP within the context of probabilistic models is only described recently (RW05).

4.3.1 Basics

A Gaussian Process is a generalization of the Gaussian probability distribution. It assumes a GP prior over functions which are Gaussian distributed.

Definition 4.3.1. (Gaussian Process) A Gaussian Process (GP) is a collection of random variables, any finite number of which have a consistent joint Gaussian distribution. □

Take for example a Gaussian distribution on functions of the form $f : \mathbb{R} \mapsto \mathbb{R}$. The collection of random variables $\{f(x_1), f(x_2), \ldots, f(x_N)\}$ are then Gaussian distribution. A GP is completely specified by its mean and covariance function $k(x, x')$:

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \tag{4.13}$$
$$m(x) = E[f(x)] \tag{4.14}$$
$$k(x, x') = E[(f(x) - m(x))(f(x') - m(x'))] \tag{4.15}$$

Where $E[.]$ denotes the expectation operator. The GP remains mathematically the same as the standard Gaussian distribution. However the covariance matrix of a GP is actually a matrix where each ij-entry of the covariance matrix is $k(x_i, x_j)$. A more detailed description of the covariance functions can be found in section 4.3.2.

The consistency as stated in definition 4.3.1 is also known as the marginalization property. It basically means that the probability distribution on a set of variables Y does not change the probability distribution of the subset of variables $y \subset Y$. Indeed, consistency can be easily verified for the case of the Gaussian distribution:

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim \mathcal{G}\left(\begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{bmatrix} \right) \tag{4.16}$$

$$x \sim \mathcal{G}\left(\mu_x, \Sigma_{xx}\right) \tag{4.17}$$

Where $\mathcal{G}(\mu, \Sigma)$ denotes the Gaussian distribution with mean μ and covariance matrix Σ. GP is a *nonparametric* probabilistic model. A statistical model is nonparametric when the number of parameters increases with data and extends to models over infinite dimensional function. It might seem strange to be able to deal with infinite dimensional spaces but it is possible to work in a finite dimensional subspace where it is tractable and the extension to infinite dimensional spaces is natural thanks to the consistency property.

4.3.2 Covariance Functions

As can be seen in the previous section, the covariance function is important in GPs. Covariance functions defines the similarity between two points. Intuitively a covariance function $k(x, x')$ gives an idea how much influence $f(x)$ has over $f(x')$. Most covariance functions assumes that input points close together are likely to have outputs which are similar.

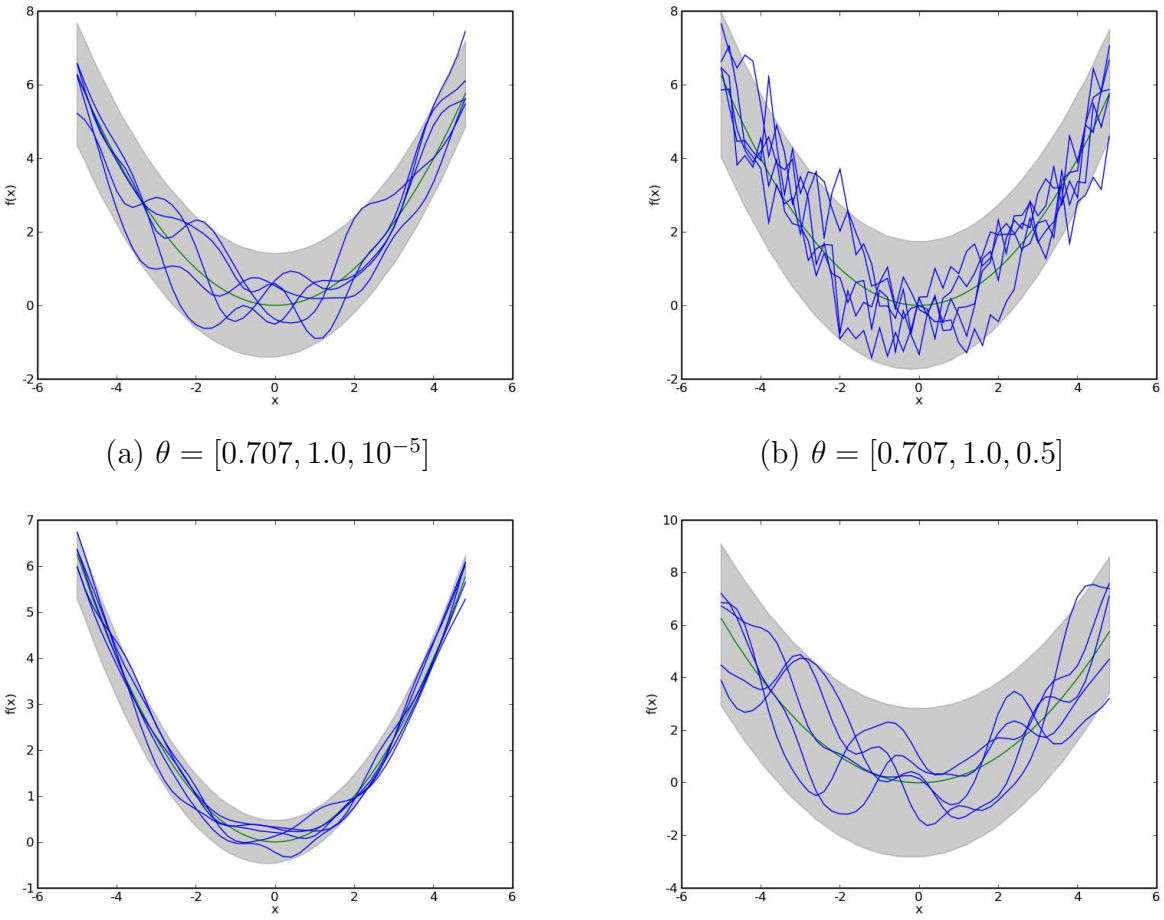
The term covariance function has a direct analogue in the Support Vector Machine (section 4.3.4 for brief introduction) literature and is also known as kernels. A covariance function $k(x, x')$ is a function which maps a pair of inputs $x, x' \in \mathcal{X}$ to $\mathbb{R}$ i.e. $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$. The term kernel originates from the theory of integral operators where such operators map one space of functions to another. For a function to be a valid kernel, it has to be symmetric i.e. $k(x, x') = k(x', x)$ and positive semi-definite. Its justification originates from its duality to reproducing kernel Hilbert spaces. Appendix B.2 provides more details and the interested reader is referred to the monograph by Wahba (Wah90).

Definition 4.3.2. (Positive Semi Definite Kernels) A kernel k is positive semi-definite ($\succeq 0$) if

$$\int \int k(x, x') f(x) f(x') d\mu(x)\, d\mu(x)' \geq 0, \ \forall f \in L_2(\mathcal{X}, \mu) \tag{4.18}$$

where μ is a measure in $\mathcal{X}$. □

The integral of equation 4.18 can be intuitively viewed as a generalization of the definition of a PSD matrix i.e. $x^T A x \geq 0 \ \forall x$. Hence, a kernel k gives a PSD *gram matrix* where each entry $K_{ij} = k(x_i, x_j)$ given a set of input points $\{x_i | i = 1 \dots N\}$. The PSD kernel matrix is also consistent with that of a covariance matrix in Gaussian distributions.



(a) $\theta = [0.707, 1.0, 10^{-5}]$

(b) $\theta = [0.707, 1.0, 0.5]$

(c) $\theta = [0.707, 3.0, 10^{-5}]$

(d) $\theta = [1.414, 1.0, 10^{-5}]$

Figure 4.3: Examples of different parameters of covariance function (eqn. 4.19) on a GP. See section 4.3.2 for a detailed explanation.

PSD matrices are known to have a spectral decomposition. Its analogue for the continuous case exists as well and it is known as *Mercer's Theorem* (see appendix B.1.

Figure 4.3 illustrates several examples of different parameters for a commonly used covariance function:

$$k(x, x') = \theta_1^2 \exp\left(-\frac{(x - x')^2}{\theta_2^2}\right) + \delta_{x,x'}\theta_3^2 \tag{4.19}$$

where the parameters are $\theta = [\theta_1, \theta_2, \theta_3]$, and $\delta_{x,x'}$ is the kronecker delta function which has value 1 if and only if $x = x'$. Modifying the parameters θ of $k(x, x')$ has the effect of modifying the influence of $f(x)$ over $f(x')$.

To see this, we shall compare the effects of the different parameters with reference to fig. 4.3(a). Fig. 4.3(b) shows the effect of θ_3 which indicates the level of noise and is hence has more jitter. θ_2 is also known as the characteristic length-scale which determines the degree of influence between $f(x)$ and $f(x')$ given two fixed points x and x'. θ_2 in fig. 4.3(c) is higher and thus exerts greater influences which gives less varying forms. θ_1 has a scaling effect vertically on $f(x)$ as in fig. 4.3(d).

Common Covariance Functions

There are basically two types of covariance functions. The covariance functions dependent on $(x - x')$ are invariant to translations in the inputs space, $\mathcal{X}$. These covariance functions are *stationary covariance functions*. The input space $\mathcal{X}$ can be multidimensional in general. If the covariance function is a function of $\|x - x'\|$, it is invariant to orientation of $(x - x')$ and are *isotropic*. The following lists several examples of common stationary covariance functions:

a) **Squared Exponential (SE):** The squared exponential function is one of the most commonly used kernel in the machine learning literature. It has the form

$$k(x - x') = \exp\left(-\frac{(x - x')^2}{2\ell^2}\right) \tag{4.20}$$

where the parameter ℓ is the *characteristic length-scale*. The SE function is infinitely differentiable and the resulting mean of the GP with the SE covariance function is infinitely differentiable.

b) **γ-exponential:** The γ-exponential covariance function includes the SE exponential function and has the form

$$k(x - x') = \exp - \left(\frac{x - x'}{\ell}\right)^{\gamma} \tag{4.21}$$

where $0 < \gamma \leq 2$. However this class of functions are not mean square differentiable except the case where $\gamma = 2$.

c) **Rational Quadratic:** The rational quadratic (RQ) covariance function can be seen as superpositions of an infinite number of SE kernels with different length scales ℓ according to a distribution $P(\ell)$, i.e. $k(x-x') = \int \exp(-(x-x')^2/2\ell^2)P(\ell)d\ell$. In the case of the RQ covariance function, the distribution $P(\ell)$ is a gamma distribution $P(\tau|\alpha,\beta) \propto \tau^{\alpha-1}\exp(-\alpha\tau/\beta)$ where $\tau = \ell^{-2}$. The RQ can thus be obtained by an infinite sum (integral). Letting $r = (x-x')$,

$$k(r) = \int P(\tau|\alpha,\beta)k_{SE}(r|\tau)d\tau \tag{4.22}$$

$$\propto \int \tau^{\alpha-1}\exp\left(-\frac{\alpha\tau}{\beta}\right)\exp\left(-\frac{\tau r^2}{2}\right)d\tau \tag{4.23}$$

$$\propto \left(1+\frac{r^2}{2\alpha\ell^2}\right)^{-\alpha} \tag{4.24}$$

where $\beta^{-1} = \ell^2$. When $\alpha \to \infty$, we can see that equation 4.24 converges to the SE covariance function. As the RQ covariance function is a sum of SE functions which are infinite differentiable, RQ covariance functions are also infinitely differentiable.

d) **Periodic:** It is also possible to model a GP over periodic functions by using covariance functions which is periodic. An example (Mac98) shows such an example where the unknown period λ_i in the i^{th} input x_i is

$$k(x-x') = \theta\exp\left[-\frac{1}{2}\sum_i\left(\frac{sin(\frac{\pi}{\lambda_i}(x_i-x_i'))}{r_i}\right)^2\right] \tag{4.25}$$

where θ, λ_i and r_i are parameters of the periodic covariance function.

Very often stationary isotropic covariance functions involves $(x-x')^2$. In multiple dimensions, such isotropic covariance functions can be modified via a positive semi-definite matrix M such that the product becomes $(x-x')^T M(x-x')$. In (VW99), a linear dimensionality reduction can be performed by a low-rank representation of M. In linear dimensionality reduction, M can be factored similarly to *factor analysis* methods common in the linear dimension reduction literature.

$$M = \Lambda\Lambda^T + \Psi \tag{4.26}$$

Where Λ is a $D \times k$ matrix ($k < D$) with columns representing the vectors of the significant lower dimensions and Ψ a diagonal matrix.

The second type of covariance functions are non-stationary. Similarly, the following lists a few non stationary covariance functions.

Dot Product The simplest non-stationary covariance functions are the dot product covariance function exhibiting a linear trend.

$$k(x,x') = \sigma_0^2 + x^T\Sigma x' \tag{4.27}$$

where σ_0 is scalar offset term and Σ a general covariance matrix.

Warping Another method of introducing non-stationarity will be to warp the input space with an arbitrary non-linear mapping $x \rightarrow u(x)$. This produces a new covariance function $k(u(x), u(x'))$. The function $u(x)$ need not have the same dimensionality in its inputs and outputs. Furthermore, $u(x)$ does even need to be invertible. An example (Mac98) in a one dimensional space is $u(x) = (\cos(x), \sin(x))$ where one obtains periodic random functions.

Spatially Varying Length Scales Most standard covariance functions assumes a fixed length scale ℓ. (Gib97) derived a covariance function where length scales can be spatially varying as a function of its position x

$$k(x, x') = \prod_{d=1}^{D} \left(\frac{2\ell_d(x)\ell_d(x')}{\ell_d^2(x)\ell_d^2(x')} \right)^{\frac{1}{2}} \exp\left(-\sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{\ell_d^2(x)\ell_d^2(x')} \right) \tag{4.28}$$

where $\ell_d(x)$ is an arbitrary positive function. This covariance function is obtained by considering the limit of a finite discrete set of Gaussian basis functions with length scales in each dimension d varying according to positive function $\ell_d(x)$.

Generalized Construction Arbitrary non-stationary covariance functions can be obtained by further generalizing Gibb's construction of spatially varying length scale covariance functions 4.28. It has been shown in (PS04) that for any valid stationary, isotropic covariance function k, a general covariance function has the form

$$k_{gen}(x, x') = 2^{D/2}|\Sigma_i|^{1/4}|\Sigma_j|^{1/4}|\Sigma_i + \Sigma_j|^{-1/2}k(\sqrt{Q_{ij}}) \tag{4.29}$$

where $Q_{ij} = (x - x')^T((\Sigma_i + \Sigma_j)/2)^{-1}(x - x')$ and Σ_i is the covariance matrix of the kernel at x_i.

Covariance functions are not limited just to the "dictionary" of known functions. A covariance function can generally be constructed from previously defined kernels. Given valid kernels k_1 and k_2, the following gives a list of operations that constructs a valid kernel:

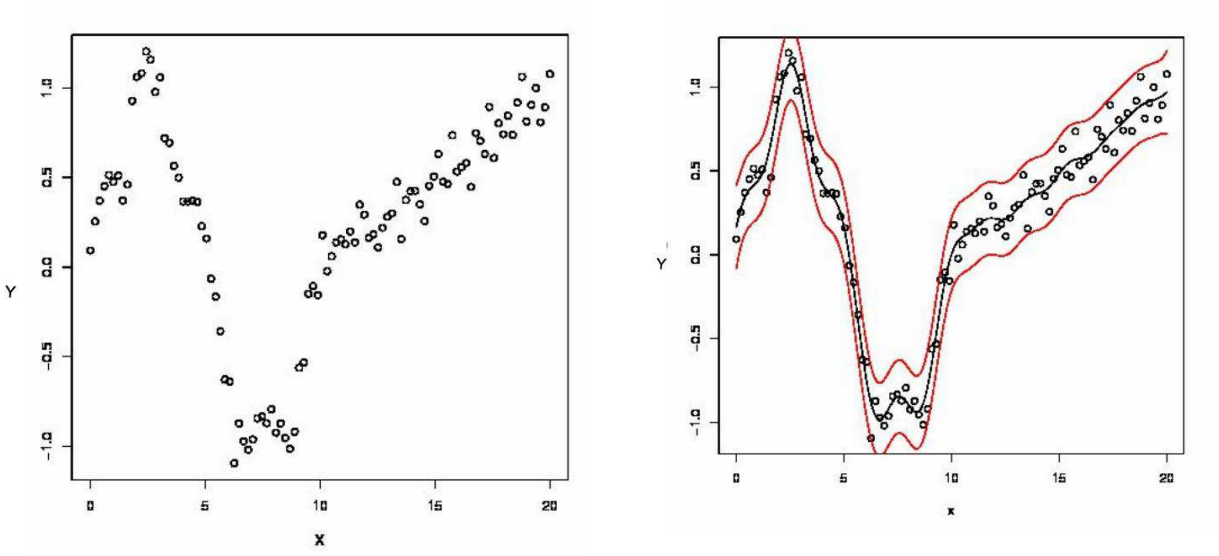
1. $k(u, v) = \alpha k_1(u, v) + \beta k_2(u, v)$ for $\alpha, \beta \geq 0$.
2. $k(u, v) = k_1(u, v)k_2(u, v)$
3. $k(u, v) = k_1(f(u), f(v))$ where $f : \mathcal{X} \mapsto \mathcal{X}$.
4. $k(u, v) = g(u)g(v)$ for $g : \mathcal{X} \mapsto \mathbb{R}$.
5. $k(u, v) = f(k_1(u, v))$ where f is a polynomial with positive coefficients.
6. $k(u, v) = \exp(k_1(u, v))$.
7. $k(u, v) = \exp\left(\frac{-\|u-v\|^2}{\sigma^2}\right)$

Kernels can be constructed according to the requirements of the concerned application. For example, we wish to probabilistically model smooth functions with additive white noise. It is given by the covariance function of eqn. 4.19 is the result of scaling the SE function and adding white noise to it by the first operation.

4.3.3 Regression

Two common applications of GPs is in regression and classification. This section presents the basic formulation and algorithms for the application of GPs in regression since it is used for learning motion in chapter 6. The formulation for classification can be found in appendix B.2.1

In typical regression problems, we are given a data set of observations and inputs $D = \{(y_i, x_i)|i = 1, \ldots, N\}$ where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$. Regression seeks a function $f : \mathcal{X} \mapsto \mathcal{Y}$ to predict values of $f(x_*)$ where $x_* \subset \mathcal{X}$ has never been observed previously in the training data set.



(a) Data points (b) GP regression

Figure 4.4: Mean and Variance in GP regression

From a probabilistic perspective, a model is constructed with a predictive distribution $P(y_*|x_*)$. This model is then capable of expression uncertainties in its predictions y_* given new input x_*. GP models the predictive distribution by the following assumptions:

$$
\begin{aligned}
y(x) &= f(x) + \varepsilon && (4.30)\\
f &\sim \mathcal{GP}(\mathbf{0}, k(x, x')) && (4.31)\\
\varepsilon &\sim \mathcal{G}(0, \sigma^2) && (4.32)\\
& && (4.33)
\end{aligned}
$$

Where the prediction function $f(x)$ has a GP prior with a zero mean function $\mathbf{0}$ and covariance function K . Additionally it is also assumed that the observed values $y(x)$ differ from the function values $f(x)$ by an *iid* Gaussian distributed white noise ε. The function $y(x)$ is thus Gaussian distributed with zero mean with covariance $Cov(y(x_i), y(x_j)) = k(x_i, x_j) + \sigma^2 \delta_{x_i,x_j}$ where δ_{x_i,x_j} is the Kronecker delta and has value 1 if and only if $x_i = x_j$.

From the consistency of GPs, the joint distribution of test outputs, Y_* training outputs, Y over test inputs, X_* and training inputs X according to the prior distribution is:

$$\begin{bmatrix} Y \\ Y_* \end{bmatrix} \sim \mathcal{GP}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma^2\mathbf{I} & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \tag{4.34}$$

Where $\mathbf{I}$ is the identity matrix. For n training points and n_* test points, $K(X, X_*)$ is an $n \times n_*$ covariance matrix evaluated at all pairs of training and test points. The dimensions and contents of $K(X, X)$, $K(X_*, X_*)$ and $K(X_*, X)$ are similar. Furthermore, $K(X_*, X) = K(X, X_*)^T$. The posterior distribution over function values at test outputs can be found by conditioning on the training data and test inputs:

$$\begin{aligned} P(Y_*|X_*, X, Y) &= \mathcal{GP}(\mu_{Y_*}, \Sigma_{Y_*}) && (4.35) \\ \mu_{Y_*} &= K(X_*, X)\left[K(X, X) + \sigma^2\mathbf{I}\right]^{-1} Y && (4.36) \\ \Sigma_{Y_*} &= K(X_*, X_*) - K(X_*, X)\left[K(X, X) + \sigma^2\mathbf{I}\right]^{-1} K(X, X_*) && (4.37) \end{aligned}$$

The distribution of equation 4.35 corresponds to conditioning the joint Gaussian distribution. Another important influencing factor in GPs is the covariance function. As seen in section]4.3.2, there are several different types of covariance functions and are mostly parametrized. Inference in GP whether by selecting a certain *hyper-parameter* (parameters of covariance function) or by placing a prior distribution over hyper-parameters and integrating them out are analytically intractable and approximations have to be used.

From a strict Bayesian point of view, the correct way will be to integrate the hyper-parameters out which can be relatively easily approximated using MCMC methods (see section 3.6.2). However, choosing a certain hyper-parameter value is usually more computationally cheaper. The choice of a hyper-parameter value in GPs is sometimes called *model selection*. Frequently, model section is performed by maximizing its marginal likelihood. The marginal likelihood in GPs is the integral of the likelihood times prior:

$$P(Y|X, \theta) = \int P(Y|f, X, \theta)P(f|X, \theta)df \tag{4.38}$$

Where θ is the vector of hyper-parameters. The product of two Gaussians is still a Gaussian. Simplifying the notation for the covariance matrix with covariance functions $k(x, x')$, we let $K_X = K(X, X)$:

$$\ln P(Y|X, \theta) = -\frac{1}{2}Y^T(K_X + \sigma^2\mathbf{I})^{-1}Y - \frac{1}{2}\ln|K_X + \sigma^2\mathbf{I}| - \frac{n}{2}\ln 2\pi \tag{4.39}$$

The maximization of the log marginal likelihood with respect to hyper-parameters θ can be obtained using a gradient based optimizer such as the conjugate gradient. The gradient of the log marginal with respect to hyper-parameters is thus:

$$\nabla_{\theta_i} \ln P(Y|X,\theta) = -\frac{1}{2} Y^T K_X^{-1} \frac{\partial K_X}{\partial \theta_i} K_X^{-1} Y - \frac{1}{2} tr(K_X^{-1} \frac{\partial K_X}{\partial \theta_i}) \tag{4.40}$$

Figure 4.4 shows an example of regression and maximization of the log marginal likelihood. The left sub-figure shows the plot of the data points for regression. On the right, the means and the variances of the GP regression.

The maximization of 4.40 is also known as MAP approximation (section 3.6). The search for the best $\hat{\theta}$ via maximization assumes that the hyper-posterior is sufficient peaked around $\hat{\theta}$. It has been shown to work well in many empirical studies. Moreover, MAP approximation in contrast to maximum likelihood methods (section 3.6) which integrates over the hidden variable, in this case the variable f and is less prone to over-fitting.

4.3.4 Relations to other models

Support Vector Machines (SVM) Kernel machines, in particular the SVMs (BGV92), have been popular since the mid 1990's (CT00) (SS01). It has been popularly used in performing classification and sometimes for regression. The key notion in SVM classification is to use the "kernel trick" to transform its input into a higher feature space where it can be easily separable and the classification decision boundary is chosen based on its *maximum margin* separation.

For the case of linear classification, the decision boundary in SVMs is defined by $f(x) = w \cdot x + w_0$ where w is a weight vector with scalar offset w_0. The vector w is the normal vector to the separating hyperplane. Hence what the decision boundary $f(x)$ does is to project x to the normal of the separating hyperplane, w, thus obtaining the distance to the hyperplane after adding w_0. The maximum margin in SVMs is obtained by minimizing the norm of the weight vector w while adhering to a set of constraints:

$$\begin{aligned} &\text{minimize } |w|^2 \text{ over } w, w_o \\ &\text{with constraints } y_i(w \cdot x_i + w_0) \geq 1 \ \forall i \end{aligned} \tag{4.41}$$

where x_i is the i^{th} input data with corresponding output label $y_i \in \{+1, -1\}$ (see figure 4.5). It is possible to have soft constraints when the data cannot be linearly separated. Such constraints take the form of a hinge loss function which penalizes violations of constraints. A hinge loss function is defined as $(z)_+ = z$ if $z > 0$ and 0 otherwise. Hence a soft margin SVM will minimize the following:

$$|w|^2 + C \sum_{i=1}^{N} (1 - y_i f(x_i))_+ \tag{4.42}$$

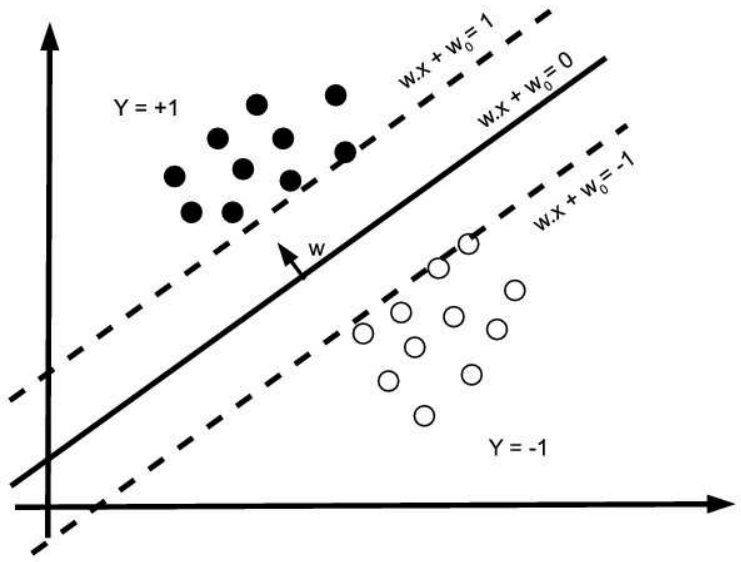


Figure 4.5: Diagram of a maximum separating margin corresponding to equations 4.41

Where C is a parameter that adjust the trade off between minimizing the hinge loss function and the norm of w. As $|w|^2 = \sum_{i,j} \alpha_i \alpha_j < x_i, x_j >$, its kernelized equivalent will hence be $f^T K^{-1} f$ where K is the covariance matrix of the equivalent GP and $K\alpha = f$ (cf. eqn B.7). This leads to the GP equivalent minimization function with a soft margin constraints:

$$f^T K^{-1} f + C \sum_{i=1}^{N} (1 - y_i f(x_i))_+ \tag{4.43}$$

Relevance Vector Machines (RVM) RVMs were introduced by Tipping (Tip01). Like SVMs, RVMs can be used for both classification and regression. But RVMs are more closely related to GPs than SVMs and is actually a special case of a GP. However the covariance function of a RVM is degenerate, i.e. it is made up of a finite number of basis components:

$$k(x, x') = \sum_{i=1}^{N} \frac{1}{\alpha_i} \phi_i(x) \phi_i(x') \tag{4.44}$$

where each α_i are the hyper-parameters. Usually, but not necessarily, the basis functions are squared exponential. Training in RVMs are similar to other GP models where the marginal likelihood is optimized with respect to the hyper-parameters $\{\alpha_i\}_{i=1}^N$. The original idea is to initialize the hyper-parameters to finite values. The

optimization process then "drives" a certain subset of the hyper-parameters to infinity, which effectively prunes away the corresponding basis functions with little influence. As these basis functions are not contributing sufficiently and are pruned, this leads to a sparse model. The remaining basis functions are then the *relevance vectors*.

The original algorithm by initializing the hyper-parameters to finite values did not prove as effective. An analysis (FT02) of the original algorithm showed that optimization with respect to a single α_i analytically leads to an accelerated training algorithm which initializes the hyper-parameters to infinity and adding in the basis functions.

Degenerate covariance functions has its drawbacks. As the degenerate covariance functions is made up of its basis functions which are dependent on the training data, a test point which has little or negligible support (far away from the ensemble of basis functions) will produce Gaussians with mean and variances which are close to zero. This is undesirable as it leads easily to false conclusions and should in principle have a high variance representing the uncertainty of its predictions, in contrast to reality where it has low variance. This leads to a conflict of interests between degeneracy for computational efficiency as a trade off for a good predictive model.

Regularization Regularization involves introducing additional information to a certain problem in order to reduce the complexity of the problem. Very often, regularization is used for "ill posed" problems (Tik63). Such problems often have no unique solutions and a penalty term is introduced in order to "reshape" the hypothesis space such that a unique solution exists and it encodes mathematically the intuition of a prior. It is sometimes viewed as a way of combating over-fitting. Regularization methods have also been introduced in the context of machine learning (GJP95).

For the case of GPs, inferring a function $f(x)$ based on a finite number of training data points is "ill posed". There are an infinite number of functions $f(x)$ that are able to fit a finite number of training points where $y_i = f(x_i)$ for $i = 1 \dots N$. Regularization often encodes the prior thus reducing complexity in terms of the smoothness of f:

$$J[f] = \lambda \|f\|^2_{\mathcal{H}} + Q(y, f) \tag{4.45}$$

where $J[f]$ is a functional rewarding smooth solutions f via the regularizing term $\|f\|^2_{\mathcal{H}}$ in reproducing kernel Hilbert space $\mathcal{H}$. The functional $Q(y, f)$ is a data-fit term which evaluates the negative log likelihood. $J[f]$ thus tries to fit data well (from the term $Q(y, f)$) while trying to keep it as smooth as possible (via the regularizing term). The balance between these two can be adjusted with an appropriate scaling parameter λ.

Although $f \in \mathcal{H}$ is potentially infinite dimensional, it has been shown by the *representer theorem* that the minimizer of $J[f]$ within the space of solutions $f \in \mathcal{H}$ can be represented in the form $f(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i)$. Thus, the solution of an infinite

dimensional solution lies in a finite dimensional solution subspace. It can be further proven (RW06) that for convex $Q(y, f)$, then there exists a unique minimize of $J[f]$.

The representer theorem equivalent for GPs can be seen by considering the predictive mean of a GP given training input points X and output points Y. With reference to equation 4.35 of section 4.3.3, the predictive mean $f(x_*)$ of testing point x_* is:

$$E[f(x_*)] = K(x_*, X)\left[K(X, X) + \sigma^2\mathbf{I}\right]^{-1} Y \tag{4.46}$$

Which can be alternatively represented as $E[f(x_*)] = \sum \alpha_i k(x_*, x_i)$ where the vector $\alpha = [\alpha_1, \ldots \alpha_N]^T = [K(X, X) + \sigma^2\mathbf{I}]^{-1} Y$.

Artificial Neural Network Models Artificial neural networks are the product of cognitive modelling approaches started from the late 1800s. It is a biologically inspired attempt to mimic biological neural systems consisting of a network of *neurons* connected via *synapses*.

An artificial neural network consists of an input layer of nodes which takes in inputs x, and has an output layer of nodes which linearly combines the outputs of a hidden layer of nodes:

$$f(x) = \sum_{j=0}^{N} \alpha_j h(x; u_j) \tag{4.47}$$

Where the parameters α_j gives the linear weight combination from the input nodes and $h(x; u_j)$ is the hidden node transfer function with u_j being the analogue of α_j but for the weighted linear combination of input nodes to hidden layer nodes. Figure 4.6 illustrates an example of this architecture. There are many different variants and architectures based on artificial neural networks and they are capable of performing classification as well as regression. It has been shown (Hor93) that the neural network model of equation 4.47 with a single hidden layer is capable of universal approximations as the number of hidden units increases till infinity.

The equivalence with a GP was given by (Nea96). It can be established by first letting the parameters α_j be zero mean Gaussian distributed with variances σ^2 and the input-to-hidden weight parameters u_j be independent and identically distributed. This thus gives the mean and covariances:

$$E_{\sigma,u}[f(x)] = 0 \tag{4.48}$$

$$E_{sigma,u}[f(x)\, f(x')] = \sum_{j=0}^{N} \sigma E_u[h(x; u_j)h(x'; u_j)] \tag{4.49}$$

$$= (N + 1)\sigma E_u[h(x; u_j)h(x'; u_j)] \tag{4.50}$$

As $h(x; u_j)$ is bounded, all moments of the distribution will be bounded. It is hence possible to apply the *central limit theorem* to show that as $N \to \infty$, $f(x)$ will converge to a GP.

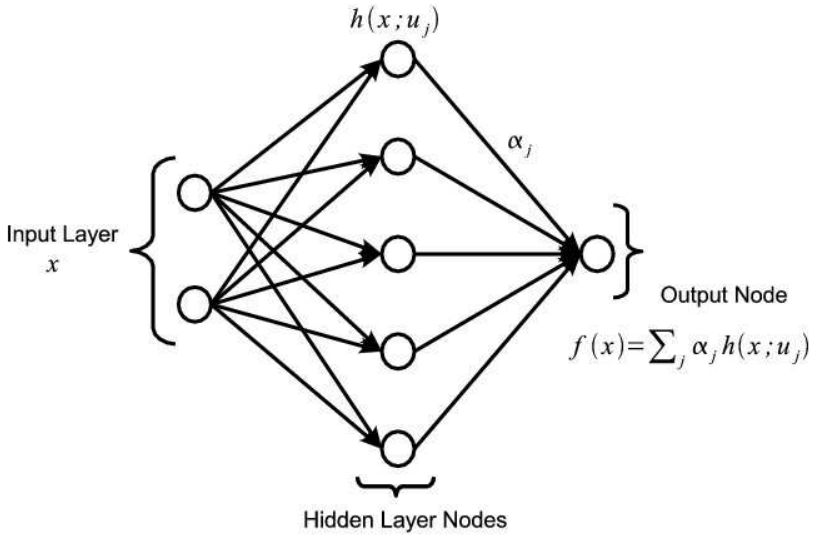


Figure 4.6: Single hidden layer artificial neural network

Part IV

Proposed Models & Algorithms

Chapter 5

Introduction to Part II

This chapter introduces the structure for part two of the thesis. In working towards the final problem of a probabilistic vehicle evolution model for collision risk estimation, the probabilistic model and algorithm for a simpler problem is first presented.

Motion Without Constraints. The simpler problem for motion without constraints is first considered in chapter 6. An example is human motion in a spacious entry hall of a large building on the ground floor. This is unlike environments such as indoor environments with a corridor connecting rooms or the network of roads where vehicles do not go off the road.

Motion in these environments are not entirely random and exhibits motion patterns. In the entry hall example, a person usually enters from the front door, looks for the receptionist, stays in the waiting area or going towards the lift. Chapter 6 presents the Gaussian Process model for describing motions in these environments.

More importantly, we show the viability of the Gaussian Process model for modelling motion, which makes it possible to represent paths as functions in a probabilistic manner. The problem of discretization is conveniently side stepped. Prediction on the future path taken can be performed in a theoretically proper probabilistic framework.

Collision Risk Estimation. The context of the problem in chapter 7 is on estimating the collision risk for a vehicle in urban traffic conditions. An indispensable component in estimating collision risk is the probabilistic motion model for predicting future motion.

A naive implementation of the model without constraints poses difficulties especially when scalability issues are concerned. Chapter 1 section 1.3.4 described the relevant issues in detail.

The scalability problem is addressed in chapter 7. The constraints imposed by road geometry are taken into account by adapting the Gaussian Process motion via conformal transformation (section 7.1.2). Combined with a motion behaviour model, we present a general framework for estimating collision risk.

Chapter 6

Motion Without Constraints

As part of the bottom up approach towards the final goal of constructing a probabilistic vehicle evolution model for collision risk estimation, the chapter presents the problems and models for the simpler problem of motion modelling in spaces without constraints

Organization

We present the chapter in the following order:

- The chapter starts off with a description of the problem context in section 6.1.
- Section 6.2 presents the intuition for the Gaussian Process. motion model.
- Section 6.3 presenting the model for the restricted case based on the assumption that there is only one exemplar motion in the scene.
- The model is further extended in section 6.4 where the generative model is based on the Gaussian mixture model to model a set of paths in a scene.
- Learning of the model and its parameters are described in section 6.5 using the factored variational expectation maximization algorithm.
- Motion prediction based on the Gaussian mixture model and Gaussian process are described in section 6.6.

6.1 Problem Definition

In the first step of the bottom up approach towards a fully probabilistic vehicle evolution model, we consider motion without constraints, unlike vehicles travelling on the road where vehicles do not run off the road. In this case, we are talking about modelling and learning motion in wide open spaces.

Motion pattern is extracted from a data set consisting of examples of motion in these environments. Such motions are a sequence of two dimensional positions in cartesian coordinates i.e $\{(x_i, y_i)\}_{i=1}^{N}$. The objective is to obtain probabilistic motion models describing the different motion patterns in the scene. Figure 6.1 provides an illustration.

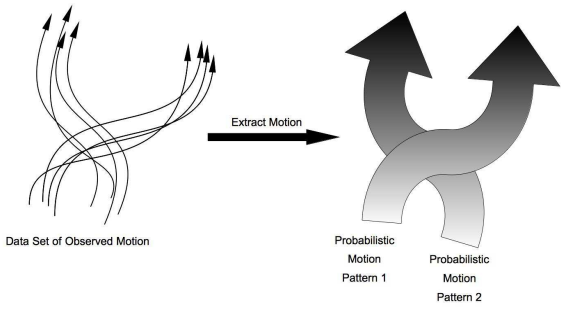


Figure 6.1: Given a data set consisting of motion sequence, probabilistic models of motion are extracted from this data set..

Our definition of motion patterns is similar to that of exemplars described in chapter 2, but with a probabilistic bent. A motion pattern is represented by a prototype motion and the allowable variations of motion centered around the prototype motion. This is quantifiable as a probability distribution where the prototype motion is the mean and the probability density function describes how likely the range of motions are going to be.

When presented with an observed motion, the predictive distribution on future motion can be inferred from the probabilistic motion models (figure 6.2).

In the formulation of our solution, an assumption that we make is that a typical motion pattern is Gaussian Process distributed. This means that a typical motion pattern can be probabilistically represented as a Gaussian distribution where realizations of this typical motion pattern does not deviate too far away from the mean motion pattern and this deviation is Gaussian distributed.

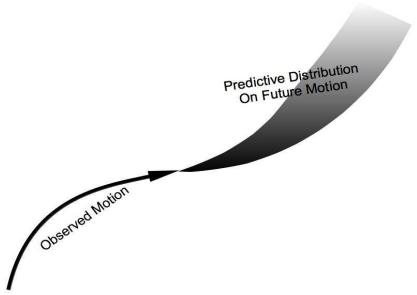


Figure 6.2: Given an observed motion give the prediction on future motion.

6.2 Intuition

To provide an intuition of how the Gaussian Process describes typical motion patterns, consider a simple one dimensional world consisting of a single motion pattern.

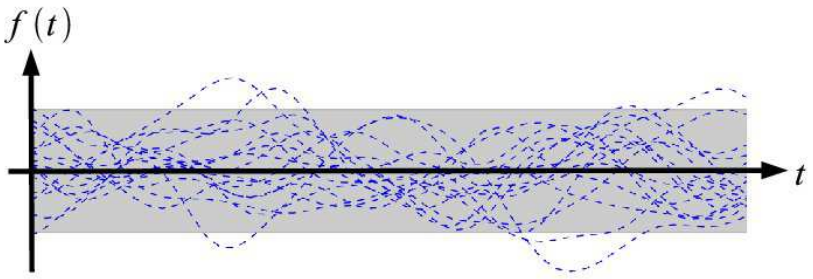


Figure 6.3: A simple 1-dimensional Gaussian Process which describes a typical motion pattern.

Figure 6.3 shows an example of a Gaussian Process with zero mean and shaded area represents the area with 2 standard deviation. Dotted lines are functions sampled from the Gaussian Process. It is a one dimensional motion pattern $f : \mathbb{R}_+ \mapsto \mathbb{R}$ which is a function of time t. $f(t)$, which represents the position at time t, has a Gaussian Process distribution. If we wish to express the probability distribution over motion at N different time instances, f is represented as an N dimensional Gaussian distribution. To see this, f as an N dimensional Gaussian takes the following form:

$$\begin{bmatrix} f(t_1) \\ \vdots \\ f(t_N) \end{bmatrix} \sim \mathcal{GP}(\mu_f, \Sigma_f) \tag{6.1}$$

Where μ_f and Σ_f are the $1 \times N$ mean vector and $N \times N$ covariance matrix of the N dimensional vector $[f(t_1), \ldots f(t_N)]^T$. The reason why a Gaussian distribution over function f has a finite N dimensional representation is due to the consistency of Gaussian Process (see chapter 4 section 4.3) where a finite number of points from f, $[f(t_1), \ldots f(t_N)]^T$, has a consistent joint Gaussian distribution. An alternative way of looking at this representation is that it represents the Gaussian distribution over functions at a finite number of points, with all other points marginalized away.

The advantage of GP is that the formulation of motion patterns as GP distributions simplifies the problem of learning a probabilistic distribution over a single typical motion pattern. It is reduced to the problem of Gaussian fitting.

Given a set of observations corresponding to various motion sequences coming from the same typical pattern. Each motion sequence corresponds to a point in N dimensional space.

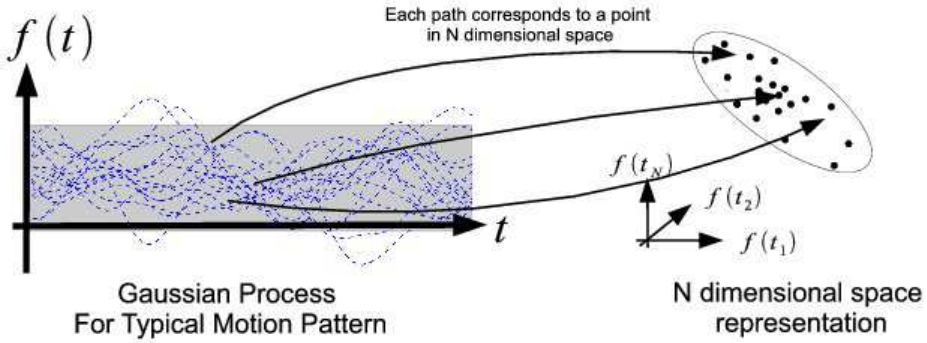


Figure 6.4: Each motion sequence (dotted lines) corresponds to a point in N dimensional space and the group of paths are Gaussian distributed. Mean is zero and the variance is shown in the shaded area.

The cluster of points from the set of motion sequence is Gaussian distributed (figure 6.4). Learning is essentially reduced to fitting a Gaussian distribution in this N dimensional space.

It is important to note that the Gaussian Process representation is not hindered by problems related to discretization unlike most approaches (detailed discussion can be found in chapter 2). It consists of a full probabilistic description of paths at all time instances due to the consistency property of the Gaussian Process. Suppose that we wish to obtain the probability distribution over the position$f(t^*)$ at time t^* which was not previously considered in the Gaussian distributed vector $[f(t_1), \ldots f(t_N)]^T$. The probability distribution over $f(t^*)$ can be easily obtained by conditioning on the points $\{f(t_1), \ldots f(t_N)\}$ since by consistency, $[f(t_1), \ldots f(t_N), f(t^*)]^T$ is Gaussian Process distributed.

Our proposed model enables one to pose questions in a probabilistic manner such as the predictive distribution of future motion. Figure 6.5 provides an illustration of this. The figure represents a 2D plane for which the motion is towards the right. A sequence of positions are observed (dots in the figure) and the predictive distribution of future motion is represented in the shaded area.

The model is not restricted to predictive distribution of future motion. The predictive distribution of motion in the past where observations might be missing can also be inferred from the Gaussian Process. Figure 6.6 shows a similar situation to that of figure 6.5. However, a certain portion of observations originally in figure 6.5 are missing in figure 6.6. The distribution over the missing areas is similar represented with its mean and variance in the shaded region.

The current model can easily be extended to more complicated situations involving multiple typical motion patterns. Since a single motion pattern is Gaussian distributed

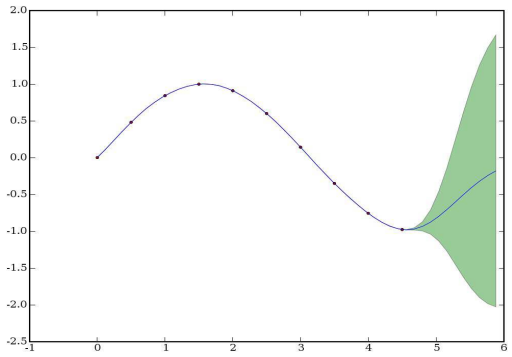


Figure 6.5: A 2D plane where motion is from left to right. Past motions are observed (dots) and the shaded region gives the predictive distribution for future motion.

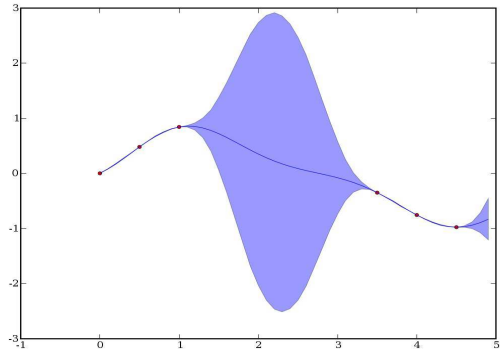


Figure 6.6: Similar situation to figure 6.5. However, some observations are missing in comparison to figure 6.5. Shaded regions show predictive distribution for missing observations.

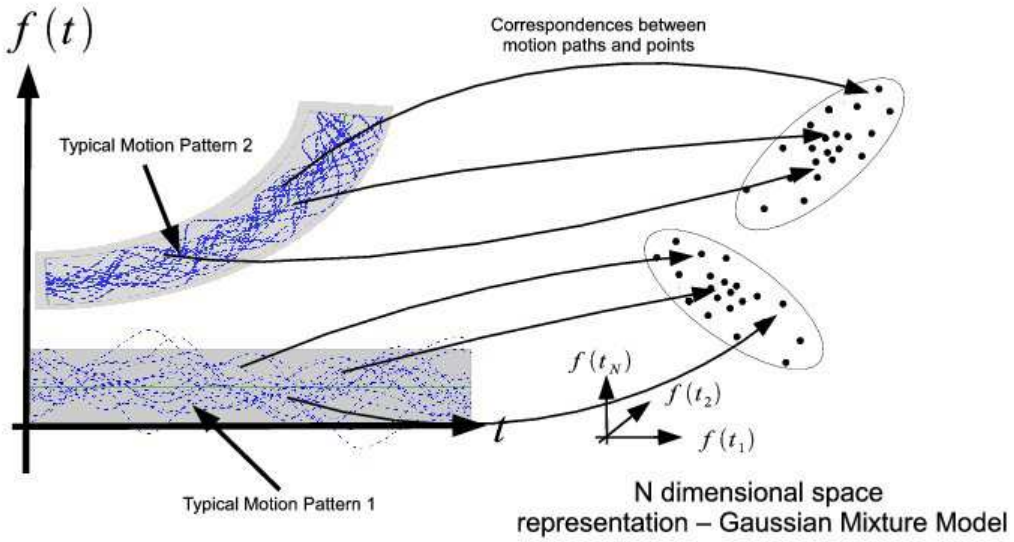


Figure 6.7: The Gaussian Mixture Model for several typical motion patterns.

in an N dimensional space, several motion patterns thus corresponds to several Gaussian distributions in the same N dimensional space. This probabilistic representation can be effectively modelled as a Gaussian Mixture Model. Learning the set of typical motion patterns given a set of motion sequences thus corresponds to fitting the parameters of the Gaussian Mixture Model in its corresponding N dimensional space (see figure 6.7). We present in this chapter a non supervised approach to automatically discover the number of typical motion patterns in a scene, as well as the parameters to the Gaussian distribution of each typical motion.

6.3 Gaussian Process as Exemplar Motion

A Gaussian process is used as a generative model to explain the observations of paths belonging to a typical exemplar path. In our model, each typical path is represented with two Gaussian processes, one each to represent the path in the x and y axes as we assume the movements in the x and y axes to be independent. Given a sample path (x_n, y_n) where $x_n = [x_{n,1} \dots x_{n,D}]^T$ and $y_n = [y_{n,1} \dots y_{n,d}]^T$, x_n and y_n are vectors containing the D positions observed along the sample path in the x and y axes respectively. Each sample path coming from a single typical exemplar path is distributed according to:

$$x_n \sim GP(\mu_x, C(\Theta)) \tag{6.2}$$
$$y_n \sim GP(\mu_y, C(\Theta)) \tag{6.3}$$

Where μ_x, μ_y, Σ_x and Σ_y are the mean vectors and covariances of the (x_n, y_n) path vectors. The mean of these Gaussian processes (μ_x, μ_y) is the mean of the typical path. Path covariances in the x and y axes are described by matrix $C(\Theta)$ where each entry of the matrix is defined by the covariance function:

$$k(x, x') = \theta_1^2 \exp\left(-\frac{(x - x')^2}{\theta_2^2}\right) + \delta_{x,x'}\theta_3^2 \tag{6.4}$$

The choice of covariance function is so that we model continuous motion with additive noise taken into account. The likelihood given a training set of N path observations for a single typical path can then be expressed as

$$L_x = \prod_{n=1}^{N} G(x_n|\mu_x, \Sigma_x) \tag{6.5}$$
$$L_y = \prod_{n=1}^{N} G(y_n|\mu_y, \Sigma_y) \tag{6.6}$$

In almost all cases, the sequence of observations of positions for each observed path are of different length. A fixed dimension D is chosen to be smallest number of path position

observations within all paths in the training data set. For each training data path, we then sample at D equal intervals along the path by interpolation to obtain the D dimensional vectors (x_n, y_n) corresponding to each of the N training data paths.

6.4 Mixture Model for Complicated Scenes

A single typical motion path is Gaussian process distributed and this can be easily extended to represent several typical motion paths with a mixture of Gaussian Processes model. In this model, each component of the Gaussian Process mixture corresponds to a single typical path.

Like the mixture of Gaussian model, we construct a hierarchical Bayesian generative model for the paths in a scene. Each path observation (x_n, y_n) is probabilistically associated with one of the mixture components k via the probabilistic variable $Z_{nk} = \{0, 1\}$ which has a multinomial distribution and the constraint $\sum_k Z_{nk} = 1$.

The graphical model representation of the generative model can be found in figure 6.8.

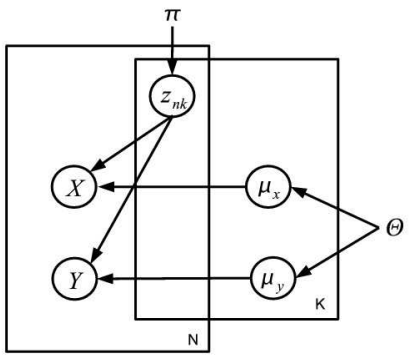


Figure 6.8: Graphical model of the generative model for trajectory clustering

The definitions of the variables are as follows:

- Size of training data: N
- Number of components: K
- $Z = \{z_{nk}\}$: z_{nk} associates each training data n with cluster component k
- $X = \{x_1, \ldots, x_N\}$: the set of x-axis training data observations
- $Y = \{y_1, \ldots, y_N\}$: the set of corresponding y-axis training data observations
- $\mu_x = \{\mu_{x1}, \ldots, \mu_{xk}\}$: the means of each cluster component along the x-axis

- $\mu_y = \{\mu_{y1}, \ldots \mu_{yk}\}$: the means of each cluster component along the y-axis
- Θ: The set of hyper-parameters for the Gaussian process
- $C(\Theta)$: The Gaussian process covariance matrix parametrized by hyper-parameters Θ

In figure 6.8, each of the N x-y pair of training data vectors is generated by one of the K clusters proportional to the variable z_{nk}. Z is multinomially distributed and is parametrized by π while the means μ_x, mu_y of each cluster is Gaussian distributed and indirectly parametrized by the Gaussian process hyper-parameters.

The prior distribution for Z, path observations and component means are thus:

$$P(Z|\pi) = \prod_{n=1}^{N}\prod_{k=1}^{K}\pi_k^{Z_{nk}} \tag{6.7}$$

$$P(X|Z,\mu_x,\Theta) = \prod_{n=1}^{N}\prod_{k=1}^{K}GP(x_n|\mu_{xk},C(\Theta))^{Z_{nk}} \tag{6.8}$$

$$P(Y|Z,\mu_y,\Theta) = \prod_{n=1}^{N}\prod_{k=1}^{K}GP(y_n|\mu_{yk},C(\Theta))^{Z_{nk}} \tag{6.9}$$

$$P(\mu_x|\Theta) = \prod_{k=1}^{K}G(\mu_{xk}|m_{xk},b_{xk}C(\Theta)) \tag{6.10}$$

$$P(\mu_y|\Theta) = \prod_{k=1}^{K}G(\mu_{yk}|m_{yk},b_{yk}C(\Theta)) \tag{6.11}$$

It is possible to construct a fully Bayesian hierarchical model for inference by assigning priors on the covariances of Gaussian processes. Covariances for the Gaussian process can be further extended via priors on Θ or more directly using a Wishart distribution. However, we would like to be able to obtain point estimates for some variables for practical purposes. As mentioned previously in section 6.3, the parameters d serves as indexes. However, there will be cases where we are interested in distributions of $(x_n(d), y_n(d))$ at locations in between indexes. To facilitate the fast and easy construction of the Gaussian process covariance matrix, it is always useful to have a parametric representation of Gaussian process covariance functions.

The decomposition of the joint distribution over the variables can be expressed as:

$$\begin{aligned}P(X,Y,\mu_x,\mu_y,Z|\pi,\Theta) = {} & P(X|Z,\mu_x,\Theta)P(Y|Z,\mu_y,\Theta) \\ & P(\mu_x|\Theta)P(\mu_y|\Theta)P(Z|\pi)\end{aligned} \tag{6.12}$$

6.5 Learning Exemplar Motions

The parameters for the probabilistic model in section 6.4 are learnt from the data using the expectation maximization (EM) algorithm. The learning algorithm used is primarily

motivated from the variational inference point of view and is then adapted for our model.

Consider the general model with data X, latent variables Z and model parametrized by Θ. The goal will be to maximize the log likelihood:

$$\begin{aligned} \ln P(X|\Theta) &= \ln \int_Z P(X, Z|\Theta) dZ \\ &= L(Q, \Theta) + KL(Q \parallel P) \end{aligned} \tag{6.13}$$

where $Q(Z)$ is the joint distribution over latent parameters. $KL(Q \parallel P)$ and lower bound L are defined as:

$$L(Q, \Theta) = \int_Z Q(Z) \ln \frac{P(X, Z|\Theta)}{Q(Z)} dZ \tag{6.14}$$

$$KL(Q \parallel P) = -\int_Z Q(Z) \ln \left\{ \frac{P(Z|X)}{Q(Z)} \right\} dZ \tag{6.15}$$

Maximization of eqn. 6.13 is equivalent to minimizing $KL(Q \parallel P)$. This occurs when $Q(Z) = P(X|\Theta)$. As the computation is intractable, $Q(Z)$ can be approximated using a constrained family of variational approximations by partitioning Z into disjoint subgroups Z_i along with the corresponding factor $Q_i(Z_i)$.

$$Q(Z) = \prod_i Q_i(Z_i) \tag{6.16}$$

In this case, the minimization of $KL(Q \parallel P)$ can be obtained by iteratively computing for each $Q_i(Z_i)$:

$$Q_i^*(Z_i) = \frac{exp\langle \ln P(X, Z|\Theta) \rangle_{i \neq j}}{\int exp\langle \ln P(X, Z|\Theta) \rangle_{i \neq j} dz_i} \tag{6.17}$$

Where the expectation in equation 6.17 is the expectation with respect to the distribution $Q(Z)$ for all variables z_i for $i \neq j$. $\langle . \rangle$ is the expectation operator.

$$\langle \ln P(X, Z|\Theta) \rangle_{i \neq j} = \int \ln P(X, Z|\Theta) \prod_{i \neq j} Q_i(Z_i) dZ_i \tag{6.18}$$

This variational maximization of the lower bound $L(Q, \Theta)$ constitutes the E-Step. It is also sometimes known as the mean-field equations because of its analogy to approximate methods from statistical physics.

During the M-Step, the log likelihood is maximized with respect to parameter Θ:

$$\Theta_{max} = \arg \max_{\Theta} \ln P(X|\Theta) \tag{6.19}$$

The E-Step and M-Step are iteratively computed till convergence as follows:

1. Repeat for each i,
   - Fix all $Q_j(Z_j)$ for $i \neq j$.
   - Calculate $Q_i^*(Z_i)$
2. Perform a maximization over the parameters Θ based on eqn. 6.19.

This EM procedure has guaranteed convergence because the bound is convex with respect n to each of the factors $Q_i(Z_i)$.

We shall next present in more detail the expectation and maximization step especially within the context of trajectory modelling.

6.5.1 Expectation Step

The mean-field update equations for the model is described in this section. A similar model can be found in (CB01). The variational approximation for the various distributions are defined as follows:

$$
\begin{aligned}
Q(\mu_x, \mu_y, Z) &= Q(\mu_x)Q(\mu_y)Q(Z) && (6.20)\\
Q(Z) &= \prod_{n=1}^{N}\prod_{k=1}^{K} r_{nk}^{z_{nk}} && (6.21)\\
Q(\mu_x) &= \prod_{k=1}^{K} G(\mu_{xk}|M_{xk}, B_{xk}C(\Theta)) && (6.22)\\
Q(\mu_y) &= \prod_{k=1}^{K} G(\mu_{yk}|M_{yk}, B_{yk}C(\Theta)) && (6.23)\\
& && (6.24)
\end{aligned}
$$

Where B_{xk} and B_{yk} are scalar parameters, M_{xk} and M_{yk} are means for the distribution over cluster means μ_{xk} and μ_{yk}. The mean field update equation from eqn. 6.17 is used to derive the update equations for each of the variables. The update equation for Z, which gives the optimized log factor:

$$
\begin{aligned}
\ln Q^*(Z) &= \langle \ln P(X|Z\mu_x\Theta)\rangle_{\mu_x} + \langle \ln P(Y|Z\mu_y\Theta)\rangle_{\mu_y} \\
&\quad + \langle \ln P(Z|\Pi)\rangle_{\mu_x,\mu_y} + const \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk} \ln P_{nk} + const && (6.25)
\end{aligned}
$$

where,

$$\begin{aligned}\ln P_{nk} &= -\ln|C(\Theta)| - D\ln 2\pi \\ &\quad -\frac{1}{2}\left\langle (x_n-\mu_{xk})^T C(\Theta)^{-1}(x_n-\mu_{xk})\right\rangle_{\mu_x k} \\ &\quad -\frac{1}{2}\left\langle (y_n-\mu_{yk})^T C(\Theta)^{-1}(x_n-\mu_{yk})\right\rangle_{\mu_y k} + \ln\pi_k \end{aligned} \tag{6.26}$$

$$r_{nk} = \frac{P_{nk}}{\sum_j P_{nj}} \tag{6.27}$$

Similarly, we can repeat the procedure for the other factorized distributions:

$$\begin{aligned}\ln Q^*(\mu_x) &= \langle \ln P(X|Z\mu_x\Theta)\rangle_z + \ln P(\mu_x|\Theta) + const \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K}\langle Z_{nk}\rangle \ln G(x_n|\mu_{xk}, C(\Theta)) + \sum_{k=1}^{K}\ln P(\mu_{xk}|\Theta) \\ &\quad + const \end{aligned} \tag{6.28}$$

$$\begin{aligned}\ln Q^*(\mu_y) &= \langle \ln P(Y|Z\mu_y\Theta)\rangle_z + \ln P(\mu_y|\Theta) + const \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K}\langle Z_{nk}\rangle \ln G(y_n|\mu_{xk}, C(\Theta)) + \sum_{k=1}^{K}\ln P(\mu_{yk}|\Theta) \\ &\quad + const \end{aligned} \tag{6.29}$$

$$\langle z_{nk}\rangle = r_{nk} \tag{6.30}$$

6.5.2 Maximization

The maximization involves the optimization of the marginal log likelihood given by:

$$\ln P(X,Y|\Theta) = \ln \sum_Z \int_{\mu_x}\int_{\mu_y} P(X,Y,\mu_x,\mu_y,Z|\pi,\Theta)d\mu_x d\mu_y \tag{6.31}$$

Since eqn 6.31 is not tractable, we can approximate it with the lower bound (refer also to eqn. 6.14):

$$\begin{aligned}L &= \langle \ln P(X|Z,\mu_x,\Theta)\rangle + \langle \ln P(Y|Z,\mu_y,\Theta)\rangle \\ &\quad \langle \ln P(\mu_x|\Theta)\rangle + \langle \ln P(\mu_y|\Theta)\rangle + \langle \ln P(Z|\pi)\rangle \\ &\quad -\langle \ln Q(\mu_x)\rangle - \langle \ln Q(\mu_y)\rangle - \langle \ln Q(Z)\rangle \end{aligned} \tag{6.32}$$

Optimization of the lower bound in equation 6.32 with respect to the Gaussian process parameters Θ. Optimization algorithms such as conjugate gradient or grid based search

can be used. The lower bound is also useful as it can be used in the test for convergence and is a valuable verification for the correctness of implementation as the lower bound never decreases.

By setting the first derivative of the lower bound with respect to π to zero and imposing the constraint of $\sum_i \pi_i = 1$ with Lagrange multipliers, π can be updated by:

$$\pi_i = \frac{N_i}{N} \tag{6.33}$$

6.5.3 Learning Algorithm

In the previous two subsections, we presented the EM algorithm as applied to the current context. In the expectation phase (sect. 6.5.1), the variational distribution corresponding to the variables Z and cluster means μ_x, μ_y are computed. However, the full variational update of all the variables were not taken into account. Namely the mixing coefficients π and the covariance hyper-parameters Θ.

A point estimate for mixing coefficients π is performed because of the relabelling problem as pointed out in (CB01) where a Bayesian model is unable to differentiate between permutations of cluster component parameters. This is a problem especially when the parameters are integrated out during Bayesian inference. Hence, a method suggested (CB01) is to perform a point estimate.

Another advantage of obtaining point estimates for mixing coefficients is the property of automatic relevance determination first introduced by Neal (Nea96). In automatic relevance determination, mixing coefficients that do not contribute significantly have very small mixing weights. This allows us to start with a reasonably large number of clusters and keep only the significant ones.

Given the data sets $\{(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)\}$, the following algorithm performs unsupervised learning to obtain typical exemplar paths.

1. **Repeat steps 2 and 3 until convergence**
2. **Expectation-Step**
   - Update $\ln Q^*(Z)$ (eq. 6.25). Taking the exponential of eq. 6.25 to yield $Q^*(Z)$ gives the form similar to the multinomial distribution. The distribution is normalized by equation 6.27 and $\langle z_{nk} \rangle = r_{nk}$.
   - Update $\ln Q^*(\mu_x)$ (eq. 6.28) and $\ln Q^*(\mu_y)$ (eq. 6.29). The exponential of eq. 6.28 yields a product of Gaussian. Since the product of Gaussian are Gaussian, the final Gaussian after the product gives the posterior on the mean of clusters.
3. **Maximization-Step**
   - Point estimate of mixing coefficients π (eq. 6.33).
   - Point estimate of Gaussian process hyper-parameters via gradient descent optimization/grid based search.

The lower bound can also be calculated after each EM iteration.

6.6 Motion Prediction

When performing motion prediction, the input is a partially observed path of dimension $M < D$. For the case of a D dimensional Gaussian with x_1 of dimension M and x_2 of dimension $D - M$:

$$P'(x_1, x_2) \sim G\left(\left[\begin{array}{c} \mu_1 \\ \mu_2 \end{array}\right], \left[\begin{array}{cc} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{array}\right]\right) \tag{6.34}$$

The probability of a partial path observation of dimension M belonging to a Gaussian of dimension D is evaluated by integrating over the $D - M$ dimensions of the Gaussian distribution to yield the marginal Gaussian distribution:

$$P'(x_1) \sim G(\mu_1, \Sigma_{11}) \tag{6.35}$$

The prediction of a path x_2 given observation x_1 can be obtained by the Gaussian conditional distribution for each cluster k:

$$P'_k(x_2|x_1) \sim G\left(\mu_2 + \Sigma_{21}\Sigma_{11}(x_1 - \mu_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{21}^T\right) \tag{6.36}$$

To choose the suitable clusters that corresponds to the observations made so far, the mahanalobis distance can be calculated and then gated based on the appropriate chi-square values. To account for variations in speed, the speed is converted into distance which is then normalized to fit into the indexes of the Gaussian process clusters.

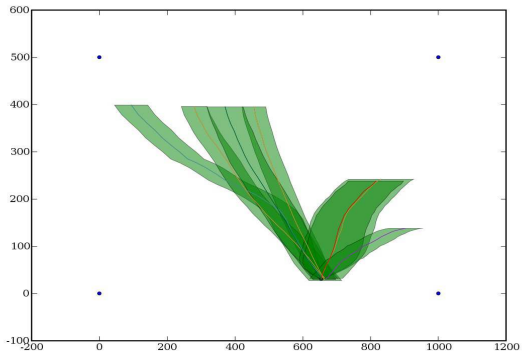


Figure 6.9: Prediction paths and path variance to 2 standard deviations. Example is based on the prediction of future paths given a single point observation, i.e. start of motion.

Figure 6.9 shows an example of the prediction where the predicted path mean and variance are represented by the 'bars' with only a single point at the beginning of motion observed. Clusters for prediction were selected according to the chi-square statistic

corresponding to the 95% confidence interval. For each cluster, the Gaussian distribution of the predicted path can be obtained using eq. 6.36. Some of the cluster component prediction has prediction that grows in uncertainty whereas some other components have a big uncertainty right from the start. The difference is attributed to the presence of noise component learnt from the data as part of the hyper-parameters of the Gaussian process.

Chapter 7

Collision Risk Estimation

In contrast to chapter 6, the learning and prediction of motion patterns with constraints presents a different set of challenges and difficulties (see chapter 1 section 1.3.4). This chapter presents the probabilistic vehicle evolution model and the estimation of collision risk.

Organization

This chapter, begins with the architecture of the system in section 7.1. The probabilistic vehicle evolution model consists of two sub-modules; behaviour estimation and realizations of behaviours. The probabilistic vehicle evolution model is then used in estimating the collision risk. They will be presented in sections 7.1.1, 7.1.2 and 7.1.4 respectively.

7.1 Overall Architecture

Figure 7.1 provides a global view of how the various components fit in within the global context. The problem in which we are interested in is associated with the sub-modules contained within the dotted box:

1. **Driving Behaviour Recognition:** The aim of behaviour recognition is to estimate the probability that a vehicle is executing one of the feasible behaviours. For example, it might give a probability value $P(turn\_left)$ that represents the probability that the vehicle observed will perform a turn left manoeuvre. As mentioned previously, behaviours are high level representations of road structure which contain semantics. The probability distribution over behaviours is performed by a Hidden Markov Model (HMM). Our current model has 4 behaviours; going straight, turning left, turning right and overtaking. These will be described in greater detail in section 7.1.1.

2. **Driving Behaviour Realization:** When evaluating the risk of collision, it has to be performed geometrically. Driving behaviour realization is represented as GPs which is a probabilistic representation of the possible future evolution of a vehicle given

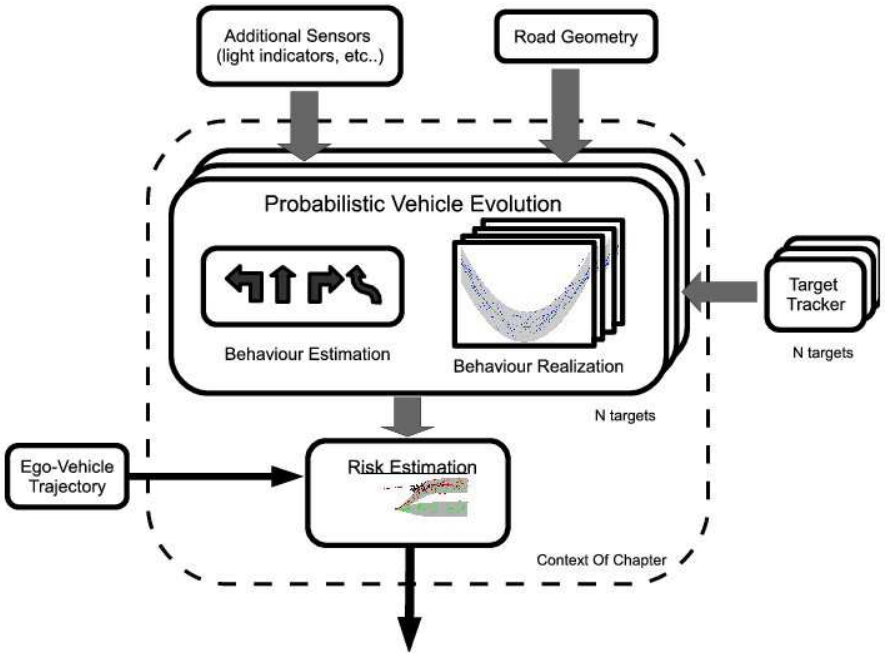


Figure 7.1: Overall View Of How The Risk Estimation Module Fits In

its behaviour. The adaptation of GP according to the behaviour is performed using geometrical transformation known as the Least Squares Conformal Map (LSCM). All relevant details will be described in section 7.1.2.

3. **Evaluation Of Risk:** A complete probabilistic model of the possible future evolution of a vehicle is given by the probability distribution over behaviours from *driving behaviour recognition* and *driving behaviour realization*. The risk of collision can be calculated based on this complete probabilistic model.

In general, the output of the risk of collision can be encapsulated under the intuitive notion of "risk of collision in the next few seconds". However, its precise mathematical definition is highly dependent on the application. Our model for risk estimation is compatible with a variety of risk estimation metrics according to the needs of applications. It will be described in detail in section 7.1.4.

7.1.1 Behaviour Recognition and Modelling

The aim of behaviour recognition is to assign a label and a probability measure to sequential data. In this context, the sequential data received are the observations coming from the sensors. Examples of sensor values are distance to lane borders, signaling lights or whether it is near an intersection etc. However, the output we wish to obtain are the probability

values over behaviours. As such, the behaviours are hidden variables. There are a variety of related models for solving the problem assigning labels to sequences.

A well known probabilistic model for inferring behaviours which are hidden based sequential observations is the *Hidden Markov Model* (HMM) (Rab89) (see figure 4.2). Extensions of the HMM includes Parametrized-HMM (WB98b), Entropic HMM (BK00), Variable-length HMM (GJH01), Coupled HMM (BOP97) and Structured HMM (HBN00). These models build upon the tradition HMM to model complex activities and interactions.

We present in this section a layered approach to model and estimate behaviours of vehicles under normal traffic conditions, as a means to the end within the context of estimating the risk of collision. The layered HMM (OHG02) decomposes the parameter space such that the robustness of the system is enhanced with the reduction of training and tuning requirements. Its architecture is very suitably applied to vehicle behaviour modelling. Each layer contains a direct semantic equivalence which can be directly modelled.

Behaviour is modelled in two layers. Each layer consists of one or more HMMs. The upper layer is a single HMM where its hidden states represents behaviours at a high level, such as overtaking, turning left, turning right, or going straight. For each hidden State or behaviour in the upper layer HMM, there is a corresponding HMM in the lower layer which represents the sequence of finer state transitions of a single behaviour. Figure 7.2 shows the schema for the layered HMM.

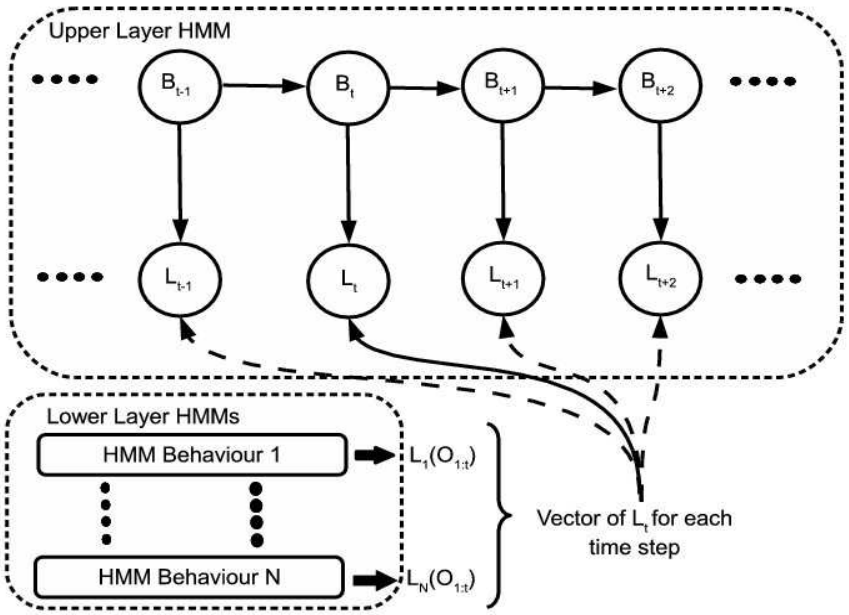


Figure 7.2: Layered HMM. Each lower layer HMM's likelihood is computed and serves as the upper layer HMM's observation.

In our model, we define the following hidden state semantics in the lower layer HMMs for each of the following behaviours of the higher layer HMM:

- *Go Straight (1 hidden state):* go forward.
- *Overtake (4 hidden states):* lane change, accelerate (while overtaking vehicle), lane change back to original lane, resume normal speed.
- *Turn left/right (3 hidden states):* Decelerate before turn, execute turn, resume normal speed.

For purposes of inferring behaviour of vehicles in our context, we wish to maintain a probability distribution over the behaviours represented by the hidden states of the HMM in the upper layer. Observations made on vehicles coming from sensors interact with the HMM in the lower layer and information is then propagated up to the upper layer. In the lower layer, there is a corresponding HMM for each higher level behaviour description. Each HMM in the lower layer, indexed by h, updates its current state by:

$$P(S_{t,h}O_{1:t}) \propto P(O_t|S_{t,h}) \sum_{S_{t-1,h}} P(S_{t-1,h})P(S_{t,h}|S_{t-1,h}) \tag{7.1}$$

where probabilistic variables O_t corresponds to observations at time t and $S_{t,h}$ is the variable for the hidden state of HMM h at time t. For each HMM h in the lower layer, its observation likelihood , $L_h(O_{1:t})$, can be computed:

$$L_h(O_{1:t}) = \sum_{S_{t,h}} P(S_{t,h}O_{1:t}) \tag{7.2}$$

Each of the observation likelihoods $L_h(O_{1:t})$ are the "observations" for the HMM of the upper layer. The inference of the upper level behaviours takes a similar form:

$$P(B_t|O_{1:t}) = P(O_{1:t}|B_t) \sum_{B_{t-1}} P(B_{t-1})P(B_t|B_{t-1}) \tag{7.3}$$

$$= L_{B_t}(O_{1:t}) \sum_{B_{t-1}} P(B_{t-1})P(B_t|B_{t-1}) \tag{7.4}$$

Where B_t is the hidden state variable of the upper level HMM at time t. $P(B_t|B_{t-1})$ is the upper level behaviour transition matrix. Most of the time, it is reasonable to assume that a change in higher level behaviour occurs more often after the end of the lower level behaviour sequence, rather than in the middle of the lower level behaviour sequence. An example is when a vehicle is executing the high level behaviour of overtaking. A high level behaviour of overtaking consists of lower level behaviours such as lane changing, accelerating past the other vehicle, return to original lane and resuming normal speed. Chances of a vehicle changing high level behaviour from overtaking to turning left, when the vehicle is at the lower level behaviour of lane changing, is lower.

To take into account these effects, there are two different transition matrix for the high level behaviour. One transition matrix corresponds to the behaviour transition when

the lower level behaviours are completely performed ($\mathbf{T}_{final}$). Another transition matrix, $T_{not-final}$ corresponds to the other case where lower level behaviours are not completely performed. Hence the higher level behaviour transition matrix can be calculated as a function of lower level states:

$$P(B_t|B_{t-1}) = \sum_{S_{t,B_{t-1}}} P(S_{t,B_{t-1}})P(B_t|S_{t,B_{t-1}}B_{t-1}) \tag{7.5}$$

where $S_{t,B_{t-1}}$ is the state at time t of the HMM at the lower level, corresponding to the previous behaviour B_{t-1}. $P(B_t|S_{t,B_{t-1}}B_{t-1})$ is by definition:

$$P(B_t|S_{t,B_{t-1}}B_{t-1}) = \begin{cases} \mathbf{T}_{final} & S_{t,B_{t-1}} \text{ is a final state} \\ \mathbf{T}_{not-final} & \text{otherwise} \end{cases} \tag{7.6}$$

At each time step, the probability distributions over high level behaviours $P(B_t|O_{1:t})$ is maintained iteratively. This will be used in the estimation of risk in section 7.1.4. The layered HMM is updated as follows in each time step:

Algorithm 3: Layered HMM Updates

Input: Current observation O_t
Output: $P(B_t|O_{1:t})$

```
foreach Lower layer HMM h do
    Update P(S_{t,h} O_{1:t}) (eqn. 7.1);
    Calculate log-likelihood L_h(O_{1:t}) (eqn. 7.2);
end
Update upper layer HMM P(B_t|O_{1:t}) (eqn. 7.4);
```

7.1.2 Realizations of Behaviours

A behaviour is an abstract representation of the motion of a vehicle. A probability distribution over the physical realization of the vehicle motion given its behaviour is indispensable to the estimation of risk. The probability distribution over the physical realization of future vehicle motion is modelled using a GP.

Recalling that the GP represents the normal driving routine where a driver approximately follows the lane and does not drift too far off to the left and right. On a straight road, this can be trivially represented with a GP where the mean of the GP corresponds to the middle of the lane (see figure 7.3):

As mentioned in chapter 1 section 1.3.4), a compact representation from the point of view of GPs does not involve learning separate GPs for the entire road network. To resolve cases where there are variations in curvature of lanes or for behaviours such as turning left or right, we propose a procedure of adapting, what we call a *canonical GP*, to the respective situations.

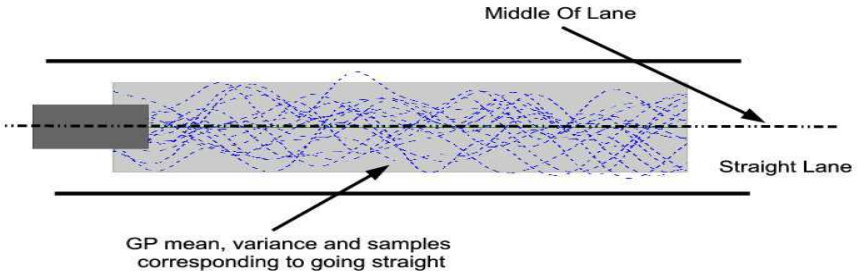


Figure 7.3: Trivial example of the GP model for a perfectly straight lane.

A *canonical GP* corresponds pictorially to figure 7.3 where it is the GP corresponding to a vehicle travelling along a perfect straight stretch of road. The *canonical GP* serves as a basis from which it will be deformed to fit the situation required. The advantage of doing so is the compact and flexible representation of the possible lane geometry. Furthermore, a single GP can be calculated once and then reused for the different situations, thus gaining in speed and computation.

When non linear situations are encountered, a deformation will be performed on the *canonical GP* to fit the geometry of the lane. An example is shown in figure 7.4 where the lane has a non zero curvature.

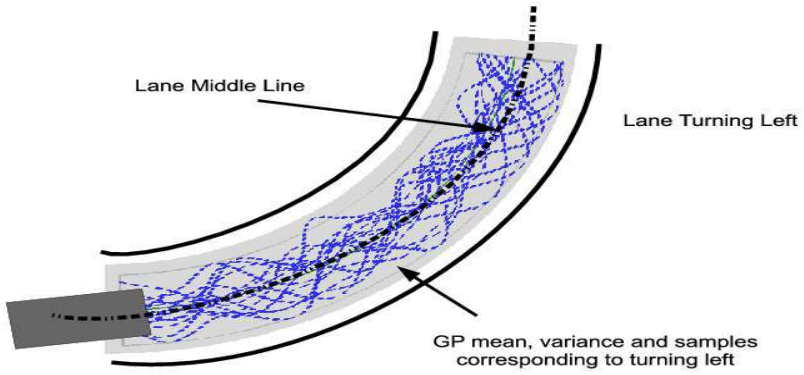


Figure 7.4: Example of the deformed GP model for a lane turning left.

Gaussian Process Deformation Model

The aim of GP deformation is to adapt the *canonical GP* to the geometry of the lane. A natural way of looking at the adapted GP is to view the adapted GP as the same *canonical GP* defined in curvilinear coordinates. Hence the problem of adapting the GP can be formulated as the invertible transformation, $\mathcal{U} : (x, y) \mapsto (u, v)$, mapping each single point of the canonical GP (x, y) defined in Cartesian coordinates to a single point (u, v) in curvilinear coordinates . $\mathcal{U}$ is a one-to-one mapping and $\mathcal{U}^{-1}$ exists (see figure 7.5).

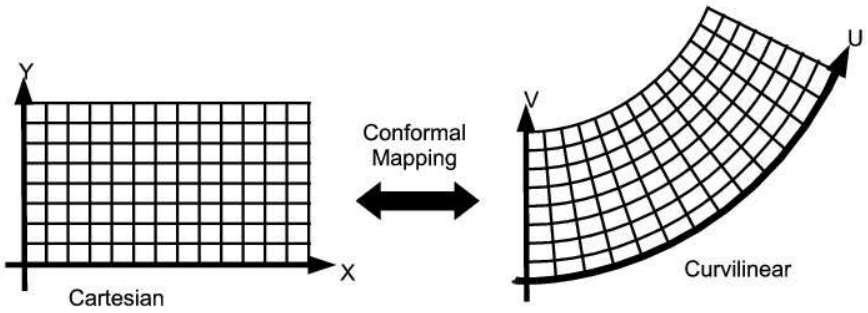


Figure 7.5: Invertible conformal map.

Curvilinear coordinates appears in many engineering problems such as computational fluid dynamics or electromagnetics where a grid based on the curvilinear coordinates are used to solve partial differential equations numerically. The methods employed in these domains are not only computationally expensive but requires the specification of the boundary. A common technique for the construction of curvilinear coordinates is *conformal mapping*.

Definition 7.1.1. A **conformal map** is a function of complex variables $\mathcal{U} : (x, y) \mapsto (u(x, y), v(x, y))$ which is analytic in the neighbourhood of the open set containing (x, y). Analytic functions are known to satisfy the Cauchy-Riemann equations:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \qquad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x} \tag{7.7}$$

By differentiating equations 7.7 with respect to x and y, and vice versa, the Laplace equation is obtained:

$$\Delta u = 0, \qquad \Delta v = 0 \tag{7.8}$$

where $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is the Laplace operator. Since the mapping satisfies the Laplace equation, it is also known as a harmonic mapping.

Property

A conformal map produces smooth and invertible mappings of coordinate grids which minimizes distortion at a local level. It is a good candidate for the deformation of GP as not only it is smooth, it has an inverse mapping which is essential for performing prediction within the canonical GP frame. Furthermore, the local deformation is minimal as the Jacobian of $\mathcal{U}$ is everywhere a rotation and scaling matrix.

Implementation Issues

Conformal mapping was originally defined in the continuous domain but is computationally demanding. Discrete conformal mapping techniques that approximate this process performs piecewise linear mappings between triangles of the mesh. Most current methods approximate the conformal map by discretization of the Laplace operator at the vertexes of the mesh triangles. Such solutions usually requires the specification of boundary conditions (EDD+95), (PJP93).

Chosen Implementation

In the context of our problem, the specification of the boundary is unnecessary. The lane boundary is implicitly defined by the curve representing the middle of the lane, and the width of the lane. Furthermore, specifying the boundary of the lanes is not straight forward especially in portions of lanes with high curvature. Ideally, it is simply sufficient to be able to generate the curvilinear coordinates based purely on the line or curve representing the middle of the lane and the width of the lane.

A dual approach which avoids the specification of all boundary solution, the LSCM was proposed (LPRMt02). Instead of discretizing the Laplace operator at the vertexes of the triangulation, LSCM proposes to adhere as much as possible the conformality condition in each of the triangles of the triangulation, reducing the problem into an unconstrained quadratic minimization problem which can be efficiently solved numerically.

Rewriting the conformal map $\mathcal{U}$ in $\mathbb{C}$ where $\mathcal{U} = u + iv$, the cauchy-riemann conditions can be written equivalent as:

$$\frac{\partial \mathcal{U}}{\partial x} + i\frac{\partial \mathcal{U}}{\partial y} = 0 \tag{7.9}$$

The LSCM seeks to minimize the violation of the conformality criterion of equation 7.9 on all triangles of the triangulation:

$$C(\mathcal{T}) = \sum_{T\in\mathcal{T}} \int_T \left| \frac{\partial \mathcal{U}}{\partial x} + i\frac{\partial \mathcal{U}}{\partial y} \right|^2 dA \tag{7.10}$$

$$= \sum_{T\in\mathcal{T}} \left| \frac{\partial \mathcal{U}}{\partial x} + i\frac{\partial \mathcal{U}}{\partial y} \right|^2 A_T \tag{7.11}$$

where $\mathcal{T}$ is the set of triangles in the triangulation, and A_T is the area of the triangle. Consider the mapping of a single triangle in complex space with points $\{(x_i, y_i)\}_{i=1..3}$ via $\mathcal{U}$ giving $\{(u(x_i, y_i), v(x_i, y_i))\}_{i=1..3}$ respectively. For a conformal mapping, the Jacobian is everywhere a scalar times rotation matrix, the mapping for a single triangle can be represented as a rotation and translation:

$$\begin{pmatrix} u_i \\ v_i \end{pmatrix} = \begin{pmatrix} \partial u/\partial x & \partial u/\partial y \\ \partial v/\partial x & \partial v/\partial y \end{pmatrix} \begin{pmatrix} x_i \\ y_i \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} \tag{7.12}$$

The rotation matrix can be obtained by solving a system of linear equations given 6 unknowns and the 3 point correspondences. Hence the gradient vector of $u(x, y)$ is:

$$\begin{pmatrix} \partial u/\partial x \\ \partial u/\partial y \end{pmatrix} = \frac{1}{D} \begin{pmatrix} y_2 - y_3 & y_3 - y_1 & y_1 - y_2 \\ x_3 - x_2 & x_1 - x_3 & x_2 - x_1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} \tag{7.13}$$

where D is twice the area of the triangle ($D = (x_1y_2 - y_1x_2) + (x_2y_3 - y_2x_3) + (x_3y_1 - y_3x_1)$) . The gradient vector of $v(x, y)$ is similar. Hence the cauchy-riemann equations 7.9 can be written as:

$$\frac{\partial \mathcal{U}}{\partial x} + i\frac{\partial \mathcal{U}}{\partial y} = \frac{i}{D}(W_1\ W_2\ W_3)(U_1\ U_2\ U_3)^T \tag{7.14}$$

where

$$\begin{aligned} W_1 &= (x_3 - x_2) + i(y_3 - y_2) && (7.15)\\ W_2 &= (x_1 - x_3) + i(y_1 - y_3) && (7.16)\\ W_3 &= (x_2 - x_1) + i(y_2 - y_1) && (7.17)\\ U_i &= u(x_i, y_i) + iv(x_i, y_i) && (7.18) \end{aligned}$$

The minimization of the violation of the the conformality criterion in equation 7.11 can be written in its discrete form:

$$\begin{aligned} C(\mathcal{T}) &= \sum_{T\in\mathcal{T}} C(T) && (7.19)\\ &= \sum_{T\in\mathcal{T}} \frac{1}{D_T} \left|(W_{1,T}\ W_{2,T}\ W_{3,T})(U_{1,T}\ U_{2,T}\ U_{3,T})^T\right|^2 && (7.20)\\ &= (\mathcal{M}\mathbf{U})^*(\mathcal{M}\mathbf{U}) && (7.21) \end{aligned}$$

where $W_{i,T}$ and $U_{i,T}$ are the values of W_i and U_i corresponding to triangle T respectively. $\mathbf{U} = (U_1, \ldots U_n)$ for n vertexes of the triangulated mesh. $\mathcal{M}$ is a sparse matrix, of dimension $n' \times n$ where rows corresponds to triangles and columns its vertexes. Each entry of $\mathcal{M}$, m_{ij}, contains the values:

$$m_{ij} = \begin{cases} \frac{W_{j,T_i}}{\sqrt{d_{T_i}}} & \text{if vertex j belongs to triangle } T_i \\ 0 & \text{otherwise} \end{cases} \tag{7.22}$$

Usually, a set of points, $p_i = (x_i, y_i)$ which makes up the vertexes of the triangular mesh in the original space are given. A subset of points U_i are fixed *a priori* which represents the user determined positions $\mathcal{U}(p_i) = U_i$. LSCM then computes the coordinates of the remaining *free points* given the user specified *fixed points*. Denoting the vector of free points as $\mathbf{U}_f$ and vector of fixed points as $\mathbf{U}_p$, the vector $\mathbf{U}$ and matrix $\mathcal{M}$ can be similarly decomposed in the following way:

$$\mathbf{U} = \left(\mathbf{U}_f^T \; \mathbf{U}_p^T\right) \tag{7.23}$$

$$\mathcal{M} = (\mathcal{M}_f \; \mathcal{M}_p) \tag{7.24}$$

where $\mathcal{M}_f$ and $\mathcal{M}_p$ are block matrices of dimensions $n' \times (n-p)$ and $n' \times p$ respectively. The equation to be minimized (eqn. 7.21) can be now written as:

$$C(\mathcal{T}) = \|\mathcal{M}_f \mathbf{U}_f + \mathcal{M}_p \mathbf{U}_p\|^2 \tag{7.25}$$

Equation 7.25 can be solved using the Moore-Penrose pseudoinverse:

$$\mathbf{U}_f = (\mathcal{M}_f^* \mathcal{M}_f)^{-1} \mathcal{M}_f^* (\mathcal{M}_p \mathbf{U}_p) \tag{7.26}$$

However, for large number of *free points*, the matrix $(\mathcal{M}_f^* \mathcal{M}_f)$ which is of size $(n - p) \times (n - p)$ will be large and involves a large number of multiplications. Furthermore, inversion of matrices has complexity $O(n^3)$. A faster method will be to use the conjugate gradient (HS52) to perform the inversion which reduces it to $O(n^2)$.

7.1.3 Predicting Vehicle Motion

The previous section 7.1.2 described an isomorphic mapping between the GP adapted to the road geometry and the *canonical GP*. This section presents the procedure on using the mapping for predicting vehicle motion.

An informed prediction on vehicle motion requires the observation of the current and past states. At every time instance t, a temporally ordered sequence of current and past observations $O = \{O_t, O_{t-1}, \ldots, O_{t-K}\}$ is maintained where each observation $O_{t-k} = (x_{t-k}, y_{t-k})$ is a vector containing the positions of the vehicle. The observations are then transformed via LSCM to *canonical GP* coordinates, where the future motion can be inferred. The probability distribution over future motion is then transformed back to real world coordinates. We detail each stage of the procedure for predicting vehicle motion:

Conformal transformation between world space and canonical space.

The mapping between world space and canonical space (the space where the *canonical GP* resides within) is discretized and represented as an isomorphic mapping between two meshes.This is done via LSCM (see section 7.1.2).

Obtaining the mapping requires the specification of a certain number of *fixed points* and their mapped coordinates. The *fixed points* are deterministically chosen; a discretized set of points lying along the middle of the lane, each corresponding to a point along the horizontal axis of the *canonical GP* frame (see fig 7.6).

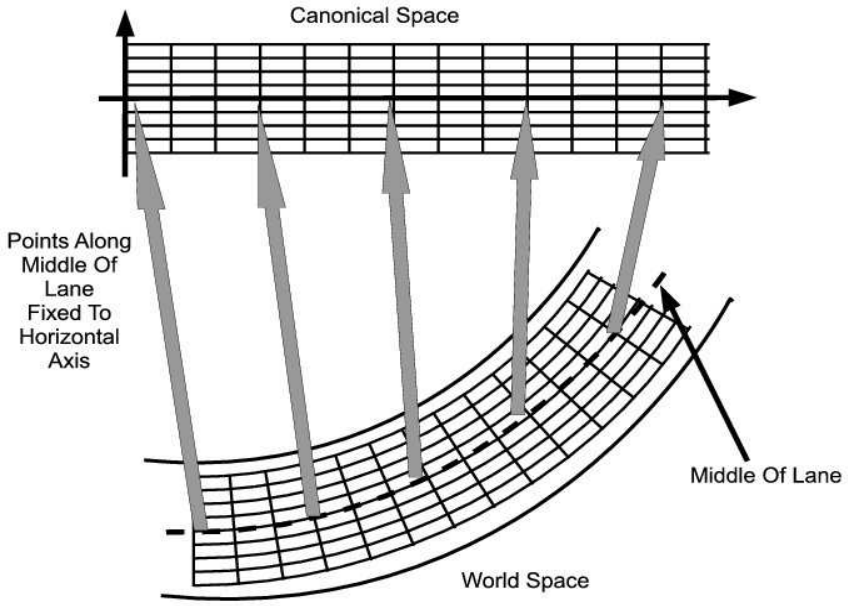


Figure 7.6: Conformal transformation between canonical space and world space. Fat arrows shows the fixed points where points along middle of lane in world space is mapped to horizontal axis of canonical space.

In our experiments, the middle of the lane is described as a poly-line. Such information can come from processed sensor data such as the lidar or camera. To determine the set of *fixed points*, the initial point is chosen by perpendicularly projecting the oldest observation, $O_l = \min_t O_t$, $O_t \in O$ on to the poly-line to obtain P_0. From P_0, a sequence of points, $P = \{P_1, \ldots, P_N\}$ along the poly-line are obtained such that for any two consecutive points, $\|P_i - P_{i+1}\|_2 = d$, where d is a constant. Each point P_i is associated with a vertex of the mesh in world space and is mapped to a vertex of the mesh in canonical space where $\mathcal{U}(Q_i) = P_i$, where Q_i has the coordinates $(d * i, 0.0)$ in canonical space.

The transformation is fully defined by having each vertex of the mesh in canonical space mapped to a vertex of the mesh in world space and vice versa. The vertexes of the mesh in canonical space is arranged in a grid and its points are known *a priori*.

Let P' be the set of vertexes in world space with free coordinates (unspecified so far). P' can then be obtained by solving equation 7.26 where $\mathcal{U}_f = P'$ and $\mathcal{U}_p = P$.

Inferring probability distribution on future motion.

The observations in world coordinates have to be mapped to canonical space before inference on future motion can be performed. The $LSCM$ gives the discrete piecewise affine mapping between the two spaces. Observations in world coordinates can be mapped to canonical space via $\mathcal{U}^{-1}(O_i) = (x_i, y_i)$.

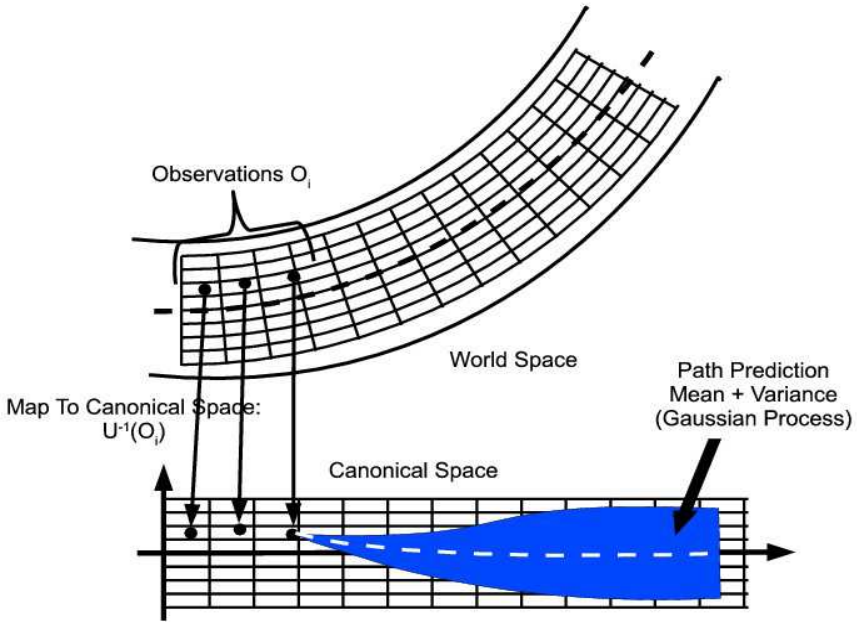


Figure 7.7: Observations are mapped into canonical space before conditioning Gaussian Process on observations to obtain probability distribution over future motion.

The mapping $\mathcal{U}$ is discretized and manifests in the form of a mesh. $\mathcal{U}^{-1}(O_i)$ can be calculated by first locating the mesh triangle which contains O_i in the world space mesh, and then transform O_i back to the corresponding mesh triangle in canonical space by calculating the corresponding barycentric coordinates.

The mapping of the past n observations of vehicle positions in world coordinates gives a set of values $\{(x_i, y_i)\}_{i=1}^{n}$ in canonical space. The probability distribution over future motion of the observed vehicle thus corresponds to the probability distribution given by the GP:

$$P(Y_*|X_*, X, Y) = \mathcal{GP}(\mu_{Y_*}, \Sigma_{Y_*}) \tag{7.27}$$

$$\mu_{Y_*} = K(X_*, X)\left[K(X, X) + \sigma^2\mathbf{I}\right]^{-1} Y \tag{7.28}$$

$$\Sigma_{Y_*} = K(X_*, X_*) - K(X_*, X)\left[K(X, X) + \sigma^2\mathbf{I}\right]^{-1} K(X, X_*) \tag{7.29}$$

where $X = (x_1, \ldots, x_n)^T$, $Y = (y_1, \ldots, y_n)^T$ are the observations. $X_* = (x_1^*, \ldots, x_K^*)$ is the vector of x values for which we wish to predict the values which is represented by $Y_* = (y_1^*, \ldots, y_K^*)$ and each $x_i^* > \max X$. In our experiments, the covariance function used is the squared exponential:

$$k(x, x') = \theta_1^2 \exp\left(-\frac{(x - x')^2}{2\theta_2^2}\right) \tag{7.30}$$

Mapping back to real world coordinates.

The probability distribution over future motion is a Gaussian Process in canonical space specified by equation 7.29 and has to be mapped back to world space in order to evaluate the risk of collision. However, the conformal transformation of a Gaussian Process is not trivially defined. Fortunately, sampling from a Gaussian distribution is trivial. Thus, we choose a monte carlo (section 3.6.2) representation of the distribution by first sampling from $P(Y_*|X_*, X, Y)$. The samples will later be used to evaluate the risk (section 7.1.4). Intuitively each sample is a possible realization of the future vehicle motion, represented as a sequence of position values $S_i = ((x_{i,1}^*, y_{i,1}^*), \ldots, (x_{i,K}^*, y_{i,K}^*))$. As the samples are in canonical space, it is transformed back to world space via LSCM.

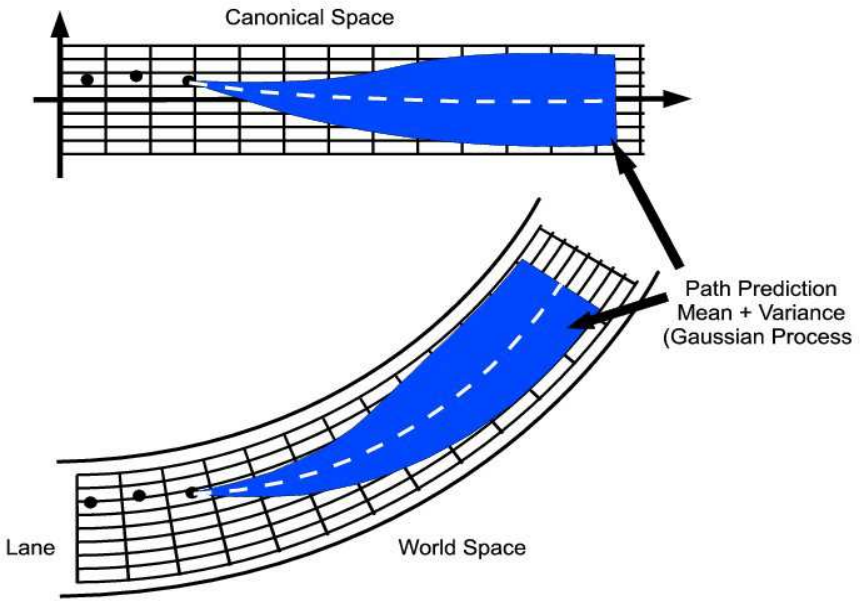


Figure 7.8: The *canonical GP* is transformed back to world space. Shaded regions display the mean and variance of the GP. Transformation can be approximated by sampling from the GP in canonical space before transforming the samples.

The procedure for the transformation is similar to that of mapping observations in

world space to canonical space, except that the mapping is in the inverse direction. Each point of sample S_i is mapped by locating the mesh triangle in canonical space containing the point and mapped to the corresponding mesh triangle in world space by calculating the barycentric coordinates.

7.1.4 Evaluation of risk

The layered HMM approach (section 7.1.1) assigns a probability distribution over behaviours at each time instance. And for each behaviour, a Gaussian Process gives the probability distribution over its physical realization. Because the behavioural semantics are propagated from the layered HMM right down to the physical level, it is now possible to assign semantics to risk values as well.

It is important to note that the definition of risk can take a variety of forms, which is largely dependent on how the risk output is going to be used. A risk scalar value might be sufficient for a crash warning system, or an application might require the risk values against each individual vehicle in the scene. The application scope using such risk values can be classified into 2 different categories.

The first category of applications involves a varying degree of vehicle control where risk values can be used to drive an autonomous vehicle, or simply to take control of a vehicle to avert the vehicle away from danger momentarily.

The second category of applications are passive in nature where no feedback into the control loop is involved. An example is a passive driving assistance system which warns drivers of possible danger ahead. We illustrate the risk evaluation in a generic bottom up manner with varying risk semantic in the following:

A) **Risk of Trajectory considering behaviour of one vehicle only:**

We start with the simple example of an autonomous vehicle navigating through a dynamic environment, avoiding collisions with the moving entities in the environment. Such autonomous vehicles usually has a feedback control or navigation system. Apart from low level control issues such as trajectory following, we consider a control module which takes into account the risk of collision as well. It is not difficult to imagine that this control module works by evaluating a set of potential trajectories to be taken by the autonomous vehicle and that the autonomous vehicle will choose the trajectory with the lowest risk within the considered set.

In this case we are calculating for the risk of a single considered trajectory. In a scene, there might be several vehicles present. Consider the simple case of only one vehicle present, vehicle V_1, (excluding autonomous vehicle, V_A). The risk of a trajectory considered by V_A, trajectory T_A, against behaviour b of vehicle V_1 is given by:

$$P(C|T_A\, B_{V_1}\, V_1) = \sum_{T_{V_1}} P(C|T_A\, T_{V_1}\, B_{V_1}\, V_1) P(T_{V_1}|B_{V_1}\, V_1) \tag{7.31}$$

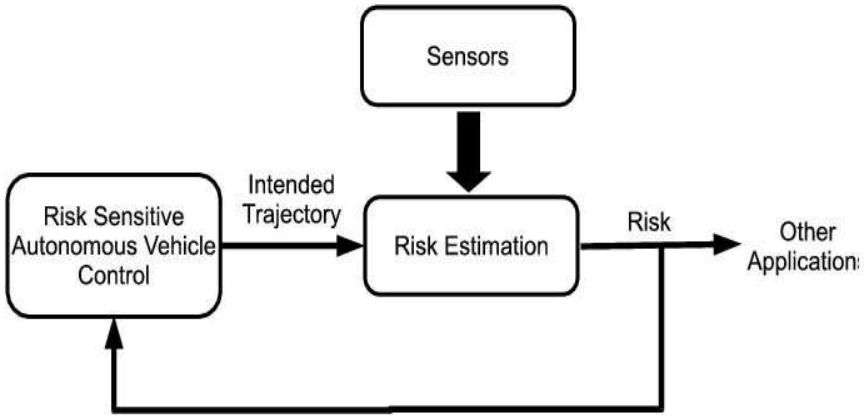


Figure 7.9: Architecture of a simple risk sensitive control of an autonomous vehicle.

Where C is a probabilistic boolean variable indicating if there is a collision, B_{V_1} is the variable corresponding to the behaviours for vehicle V_1, described by the hidden states of the upper layer HMM. T_A and T_{V_1} are the trajectories of V_A and V_1 respectively. $P(T_{V_1}|B_{V_1}\, V_1)$ is the physical realization of behaviour B_{V_1} and thus is represented by the trajectories sampled from the Gaussian Process mentioned previously in section 7.1.3. $P(C|T_A\, T_{V_1}\, B_{V_1}\, V_1)$ evaluates whether there is a collision between trajectories T_A and T_{V_1}.

In reality T_A and T_{V_1} is a list of points describing the path, i.e. $T_i = (P_1^i, \dots P_k^i)$, $P_j^i = (x_j^i, y_j^i)$. However we are able to obtain speed and accelerations of V_1 from a target tracker. Speed and acceleration on V_A can be obtained from its proprioceptive sensors. A constant acceleration model is used to compute if there is a collision. Based on the velocity and acceleration of V_A and V_1, their positions along trajectories T_A and T_i respectively can be easily calculated by linearly interpolation along the list of positions describing T_A and T_i. These positions are calculated in discrete time steps and at each time step, a collision detection is performed.

A subtle point when performing collision detection is that the geometry of the vehicles has a significant influence in the final risk values. It might be easy to think that a collision detection based on the L2-distance between two coordinates, coupled with the averaging effects over the sampled trajectories will yield a proper estimate. However, we have observed in our experiments that this is not so. Vehicles passing by each other on adjacent lanes consistently gives a false high collision probability. On the other hand reducing the L2-distance threshold will give overly optimistic estimates of collision probability when one vehicle is behind the other. This is a consequence of the geometry of the vehicle (rectangular) where the length of the vehicle is longer than its width. In our experiments, the assumption that all vehicles are of the same length and width were made. The collision detection between two rectangles representing

the geometry of the vehicles is performed by searching for an axis separating the two rectangles. The following pseudo-code (algorithm 4) gives a summary:

Algorithm 4: Evaluation of $P(C|T_A\, B_{V_1}\, V_1)$

Input: Trajectory T_A for vehicle V_A
Output: $P(C|T_A\, B_{V_1}\, V_1)$

```
ColCount = 0.0;
foreach Sampled path T_{V_1} ~ P(T_{V_1}|B_{V_1} V_1) do
    foreach Discretized time step t = step * Δt do
        X_A = Position of V_A at time t along polyline T_A;
        X_1 = Position of V_1 at time t along polyline T_{V_1};
        Θ_A = Orientation of line segment of T_A containing X_A ;
        Θ_1 = Orientation of line segment of T_{V_1} containing X_1 ;
        R_A = Rectangle centered at X_A and angle Θ_A;
        R_1 = Rectangle centered at X_1 and angle Θ_1;
        if Separating axis exist between R_A and R_1 then
            ColCount = ColCount + 1.0;
        end
    end
end
return ColCount / Number of Samples Paths;
```

B) **Risk of trajectory against one vehicle with behaviours aggregated:**

The risk of a trajectory against another vehicle, can be obtained by aggregating the risk previously (against one behaviour of another vehicle). The aggregation is essentially a weighted sum of $P(C|T_A\, B_{V_i}\, V_i)$ for each behaviour B_{V_i} of vehicle V_i.

$$P(C|T_A\, V_i) = \sum_{B_{V_i}} P(C|T_A\, B_{V_i}\, V_i) P(B_{V_i}|V_i) \tag{7.32}$$

The weighted sum was performed against the term $P(B_{V_i})$ in equation 7.32 and its values comes from the layered HMM (see section 7.1.1, eqn. 7.6).

C) **Aggregating risk with respect to all vehicle**

The risk of trajectory T_A when taking a single vehicle V_i into account is represented by $\mathcal{R}_i = P(C|T_A\, V_i)$. There are several possible choices for aggregating risk, which is largely dependent on how the aggregated risk value is going to be used or interpreted. The function of risk aggregation is a function of risk values with respect to all vehicles, i.e. $\mathcal{F}(\mathcal{R}_1, \ldots, \mathcal{R}_N)$ for N vehicles in the scene:

- **Marginalizing over vehicles:** A direct way of aggregating risks will be to marginalize over the prior probabilities of the vehicles:

$$\begin{aligned}\mathcal{F}(\mathcal{R}_1,\ldots,\mathcal{R}_N) &= P(C|T_a) \\ &= \sum_{V_i} P(C|T_A\, V_i)P(V_i) \\ &= \sum_{V_i} \mathcal{R}_i P(V_i) \end{aligned} \tag{7.33}$$

The prior probabilities over vehicles, $P(V_i)$, in equation 7.33 can come from an object recognition module which expresses the confidence that object V_i is a vehicle. Without any information, a uniform prior can be used instead and is equivalent to taking the average risk of all vehicles.

- **Maximum Risk:** Marginalizing over vehicles might be under conservative in some cases. This is especially so when a single vehicle poses an imminent danger in a scene with many vehicles and the average gives a low estimate. In this case, taking the maximum risk value might represent the risk more accurately:

$$\mathcal{F}(\mathcal{R}_1,\ldots,\mathcal{R}_N) = \max_{V_i} P(C|T_A\, V_i) \tag{7.34}$$

- **Temporally Nearest Risk:** The evaluation of collision risk, $P(C|T_A\, B_{V_1}\, V_1)$ (algorithm 4), does not explicit take into account time. For example, the check for collision between T_A and sampled trajectory T_{V_i} in algorithm 4 only indicates if there is a collision in a certain time horizon in the future regardless of the length of horizon:

$$Collide(T_A, T_{V_i}) = \begin{cases} 1.0 & \text{If any collision exists in time horizon} \\ 0.0 & \text{otherwise} \end{cases} \tag{7.35}$$

Incorporating time into risk evaluation is useful in certain cases. For applications such as crash warning, it is less probable that if the driver maintains the current acceleration, a crash 30 seconds in the future is unavoidable with probability 1.0. The drivers of the vehicles involved have reasonable time to react to the situation. In this case, it might be desirable to express risk further ahead in time as having less "importance". This can be taken into account by modifying algorithm 4 where the risk is weighted by a decreasing function with time:

$$Collide^*(T_A, T_{V_i}) = \begin{cases} \exp^{-\alpha t^2} & \text{If collision between } T_A \text{ and } T_{V_i} \\ 0.0 & \text{otherwise} \end{cases} \tag{7.36}$$

where t represents the amount of time before collision occurs and α is a constant which expresses the rate of risk decrease with time.

D) **Risk associated with driving behaviour:**

So far, the risk value of a single trajectory T_A for an autonomous vehicle is calculated. For applications where risk values are passively used, especially when the driver is a human and not a computer program, it is less practical to evaluate the risk of only a single trajectory T_A. The alternative will be to evaluate risks associated with behaviour or general risk value for the ego-vehicle (the vehicle for which the risk shall be evaluated for, but shall be named V_A still).

- **Behaviour related risk:** Instead of evaluating for a single T_A, the risk is evaluated for the collection of T_A associated with a behaviour of the ego-vehicle. For example, to obtain the risk of a certain behaviour of ego vehicle, against another vehicle, V_i:

$$P(C|B_{V_A}\, V_i) = \sum_{T_A, B_{V_i}} P(C|T_A\, B_{V_i}\, B_{V_A} V_i) P(T_A|B_{V_A}) P(B_{V_i}|V_i) \tag{7.37}$$

  where $P(T_A|B_{V_A})$ is the probability distribution over the future trajectory of the ego-vehicle with behaviour B_{V_A}. $P(C|T_A\, B_{V_i}\, B_{V_A} V_i)$ is the collision risk of trajectory T_A against vehicle V_i with behaviour B_{V_i}. The evaluation of this term is exactly the same as algorithm 4. Essentially, the algorithm for equation 7.37 will be to sample an ego-vehicle trajectory $P(T_A|B_{V_A})$ and each sample is evaluated against the sampled trajectories of vehicle V_i across all behaviours:

Algorithm 5: Evaluation of $P(C|B_{V_A}\, V_i)$

Input: Ego-vehicle behaviour B_{V_A}
Output: $P(C|B_{V_A}\, V_i)$

```
ColCount = 0.0;
foreach Sampled Trajectory T_A ~ P(T_A|B_{V_A}) do
    trajCol = Evaluate algorithm 4 with T_A as parameter;
    ColCount = ColCount + trajCol;
end
return ColCount / Number of sampled T_A;
```

- **General risk value:** A risk value between vehicle V_A and V_i can be obtained from 7.37 by marginalization over the estimated behaviour of the ego-vehicle:

$$P(C|V_i) = \sum_{B_{V_A}} P(C|B_{V_A}\, V_i) P(B_{V_A}) \tag{7.38}$$

  $P(B_{V_A})$ is the distribution over the behaviour of the ego-vehicle. This can be obtained by application of the layered HMM on the ego-vehicle.

Several examples of risk with different semantic are presented. The number of different ways of evaluating risk is combinatorial. Risk can be evaluated between trajectory samples, behaviours, vehicles or all vehicles. This is to highlight the flexibility of the current system

of using a HMM based object in identifying behaviours coupled with the use of GP for behaviour realizations, while taking the road geometry and topology into account.

7.2 Conclusion

In this chapter, we showed the various modules involved in the estimation of risk in a structured road traffic situation. To resume in a top down fashion, a layered HMM (section 7.1.1) is used to estimate the behaviours of the the vehicles in the scene. The behaviours can be separated into two hierarchical levels, corresponding to the architecture of the layered HMM. The high level behaviours are behaviours such as over taking, turning left etc. Each high level behaviour is composed of a sequence of sub-behaviours.

For each of the higher level behaviours, there is a corresponding Gaussian Process which is a Gaussian distribution over the paths of the typical pattern for each behaviour (section 7.1.2). The Gaussian distribution over the future motion path (section 7.1.2 is obtained by first transforming the observations to a canonical space in which the *canonical GP* resides. The transformation is conformal and uses a discretized least squares approach to approximate the conformal transform in the form of a 2D mesh where the transformation for each triangle mesh is approximately affine. The probability distribution over future motion is obtained in canonical space by conditioning the *canonical GP* on the transformations. The inverse conformal transformation is then applied to obtain the final probability distribution over future motion in world space for risk evaluation.

The risk is evaluated in a probabilistically sound manner (section 7.1.3, based on the Gaussian probability distribution over future motion for the various behaviours, and the estimated behaviours from the layered HMM. It has also been stressed that thanks to the combination of Gaussian Process and layered HMM to model motion at various semantic levels, there are many different ways of evaluating risk values each having its associated semantic and is dependent on the application requiring the risk value.

Part V

Implementation and Experiments

Chapter 8

Motion In Open Spaces Without Constraints

This chapter presents experimental results on recovering motion patterns based on the Gaussian Process model without constraints. For a brief recall, the problem is on obtaining motion patterns from a data set of motion sequence. The nature of motion is such that it usually takes place in open spaces where the environment does not limit motion. More details can be obtained from chapter 6.

Experiments were conducted based on both simulated and real data. The motion data sets have been kindly provided by Dizan Vasquez (VG07). The motion data sets consists of previously recorded simulated and real motion data. The context for both data sets takes place in the entrance hall of INRIA Rhône-Alpes.

8.1 Experiments on Simulated Data

The simulated data were data pre-generated in a simulator. It consists of the definition of 32 trajectory typical motion which were manually defined for the generation of the different motions. Figure 8.1 shows the ensemble of motion in the training data set.

Simulated data is generated by first choosing a motion pattern from the set of 32 pre-defined motion patterns. Each motion pattern consists of a set of control points. The motion is the simulated in discrete uniform time steps , with its direction drawn from a Gaussian distribution having the mean value of the destination as its mean.

It has been noted in (CB01) that the variational updates were initialized using the K-Means clustering algorithm. The same procedure was adopted in our experiments. The distance we used for the K-Means algorithm is the $L2$-distance with 100 clusters. The expectation maximization algorithms were performed with 100 clusters initialized with equal cluster component weights. Result of the k-means cluster algorithm is show in figure 8.2.

The number of cluster components can be determined from the weight of the cluster components. The learning algorithm on the training data set allowed the suppression of

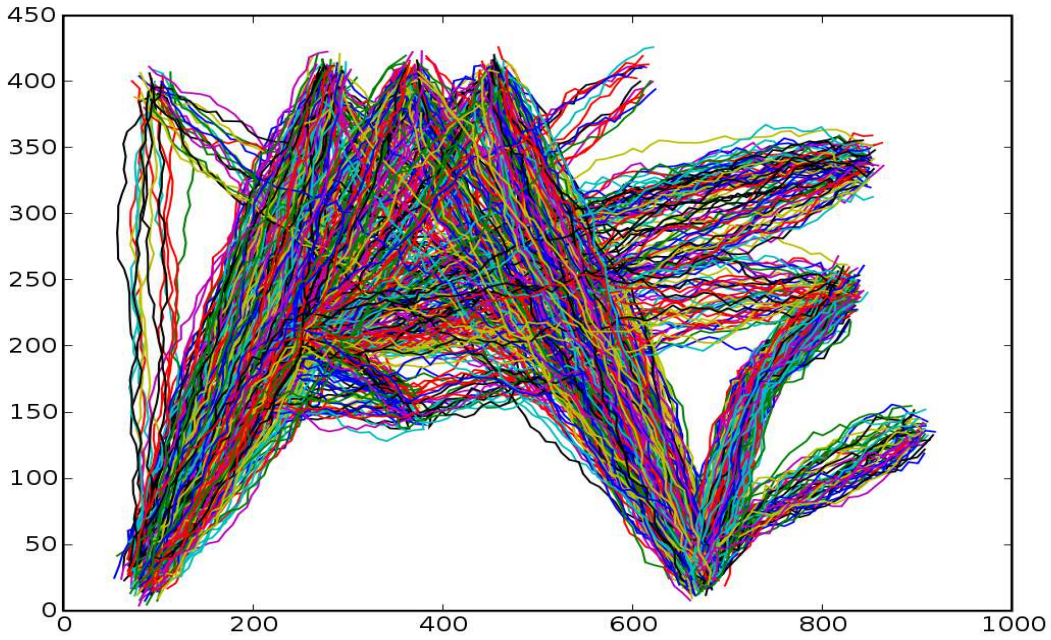


Figure 8.1: Training Data Motions. Each line represents a simulated path data and the ensemble constitutes the training data set

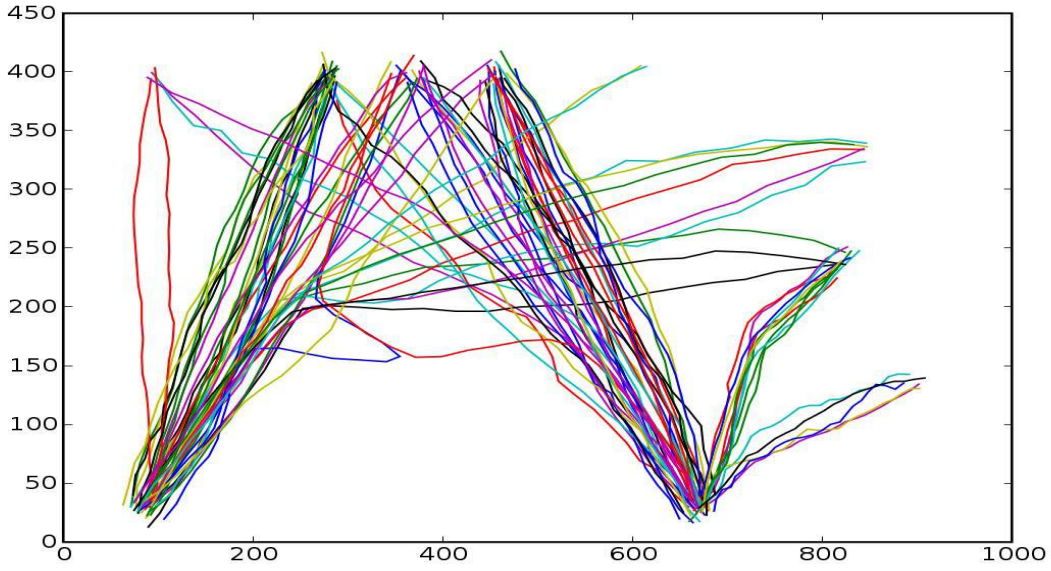


Figure 8.2: The output from the K-Means algorithm with 100 clusters

clusters with component weights of nearly zero ($< 10^{-5}$) very quickly early in the EM iterations. The learning algorithm allows the domination of Gaussian components over the others when clustering in the same regions. 26 Gaussian components were recovered after the clustering as shown in figure 8.3. The learning algorithm produces satisfactory results. The number of recovered components is less than the number of manually specified typical paths to initialize the simulations. This is due to the fact that during the human specification of example paths in the training data, there are some human input paths which are similar to another path and the learning algorithm is capable of grouping these similar paths together as a cluster.

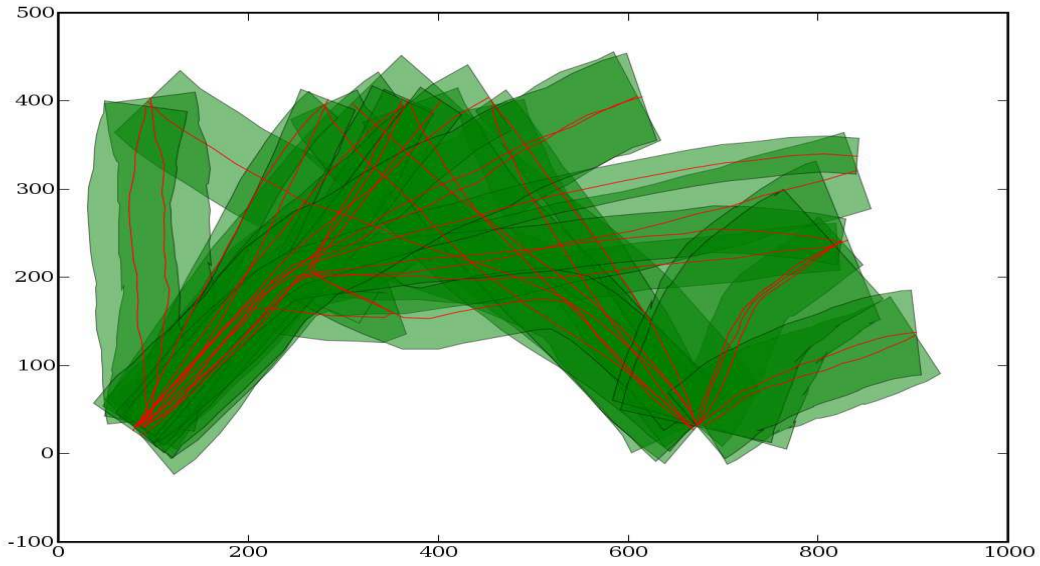


Figure 8.3: 26 typical motion recovered from the training data (different Gaussian process hyper-parameters for each cluster)

The suppression of unrepresentative cluster components with negligible weights works well only when Gaussian process hyper-parameters are different for each cluster. However, when the Gaussian process hyper-parameters are shared across clusters, the variance of the different clusters are not well expressed and as a result, it is difficult to obtain cluster components with negligible component weights (see figure 8.4).

8.2 Experiments Using Real Data Sets

The learning algorithm was also applied to real data sets. One such data set is from real data recordings from the camera on the same scene as the simulated INRIA entry hall scene. A camera has been mounted on the entry hall of INRIA (fig 8.5). Pre-processing

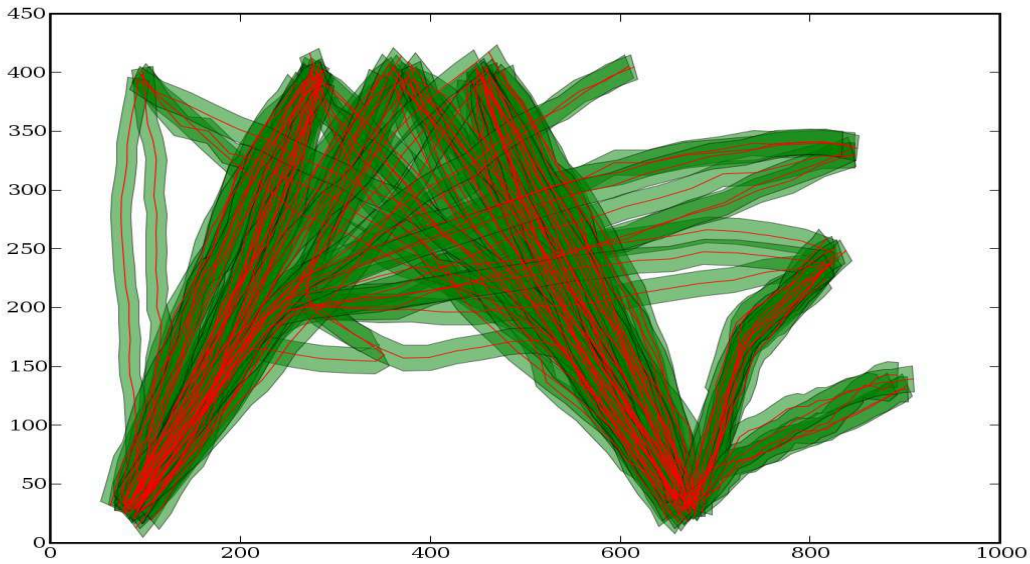


Figure 8.4: Typical motion recovered from the training data set (same Gaussian process hyper-parameters across all clusters)

of camera data includes the tracking with a camera and its homographic projections onto the ground plane to obtain ground plane coordinates and treatment of occlusion to obtain a set of well formed data. For more details on its treatment, refer to (VG07).

As can be seen from figure 8.6, the motion of objects in the scene is a lot more scattered than in the simulated data. The same experimental protocol was used starting with a Jena's clustering procedure (fig. 8.7). The real data set consists of more than 2000 paths.

A training data set of size 1000 was randomly chosen from the real data set for learning 49 components were recovered from the learning algorithm as illustrated in fig. 8.8.

The learning result from the real data showed limitations of the current implementation. The Gaussian components does not seem to generalize well in the middle regions of the scene. The middle region of the scene is characterized by having paths which begins and ends in relatively small regions, with large variations in the middle of the paths. The reason for why the middle regions of the scene data does not generalize well is due to the fact that the covariance function used is stationary. Hence it does not accommodate large variations of paths in certain regions while small variations of paths in other regions. A choice of non stationary covariance function might possibly overcome this problem and produce better results.

Figure 8.5: Video camera snapshot of INRIA entry hall scene data

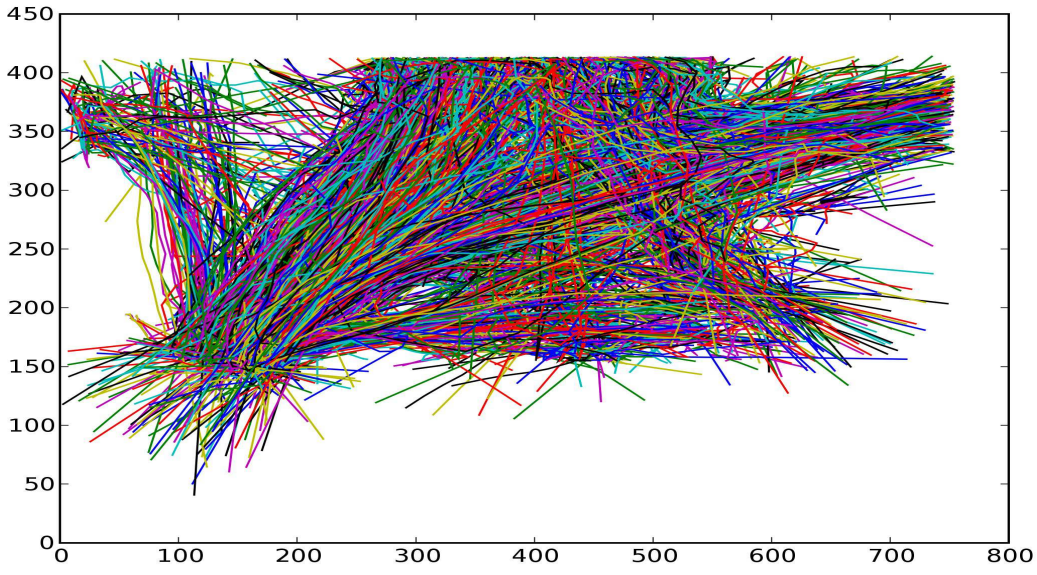


Figure 8.6: Real data of the same hall scene

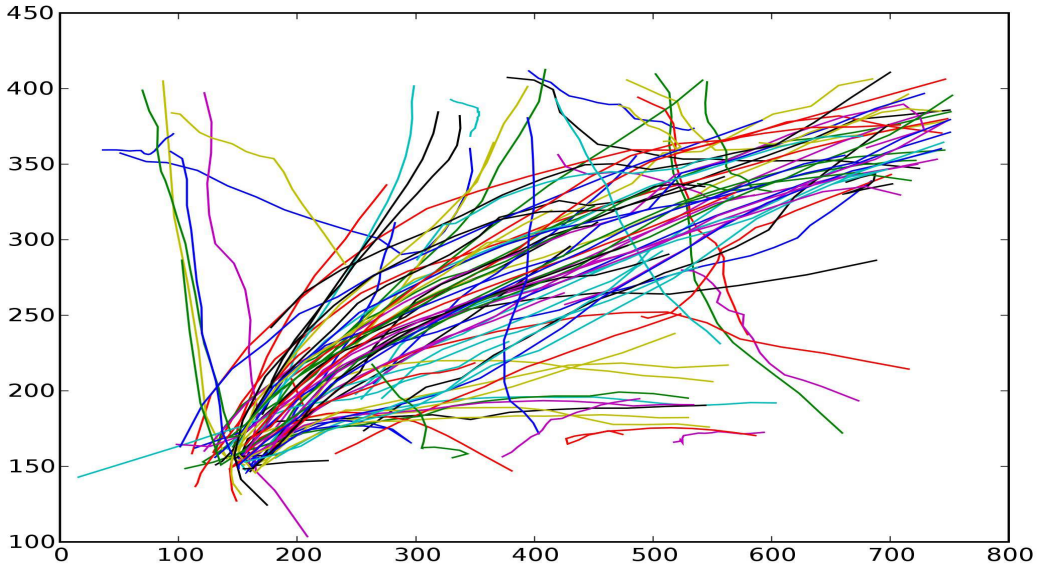


Figure 8.7: K-means plot of the 100 clusters from the real hall data

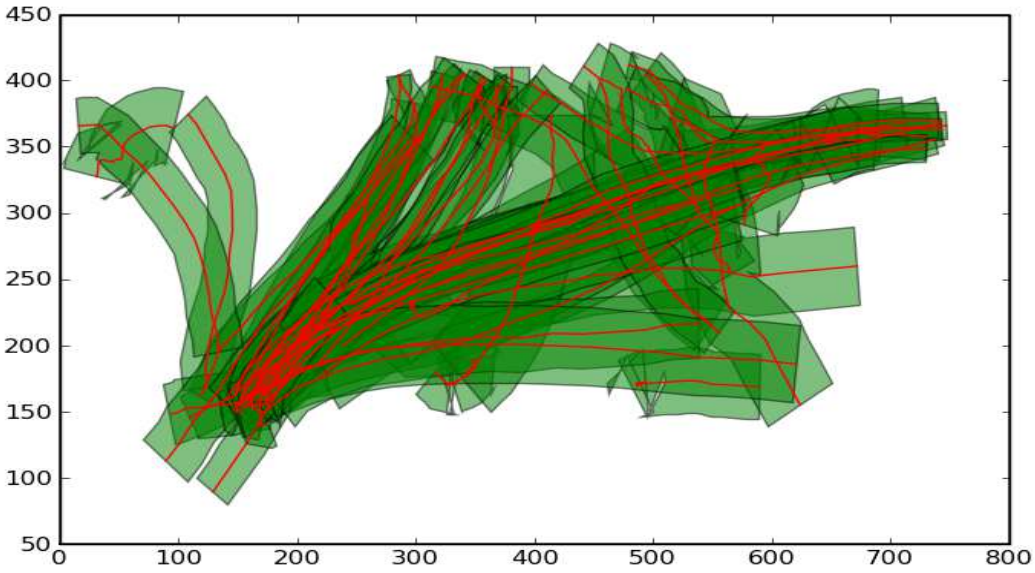


Figure 8.8: 49 typical motion recovered from the real hall scene data. Shaded bars show the approximate 2-sigma variance boundaries for the Gaussian Processes.

Chapter 9

Path Planning With Risk Estimation

This section presents an application of using Gaussian Processes to navigation in a dynamic uncertain environment. Moving obstacles are assumed to move on typical patterns which were learnt in a batch manner and are represented by Gaussian processes (see chapter 8).

The planning algorithm is based on an extension of the Rapidly-exploring Random Tree algorithm developed by Fulgenzi (Ful09) the likelihood of the obstacles trajectory and the probability of collision is explicitly taken into account. The algorithm is used in a partial motion planner, and the probability of collision is updated in real-time according to the most recent estimation. Results show the performance of the navigation algorithm for a car-like robot moving among dynamic obstacles with probabilistic trajectory prediction.

The context and overview of the approach is presented. Additional details can be found in (FTSL08). This work is done in collaboration with Fulgenzi (Ful09).

9.1 Risk Based RRTs

Consider a car-like robot moving in an unknown environment among static and moving obstacles. The task of the robot is to reach a given goal state avoiding collisions. The robot perceives its surroundings with a distance sensor (laser range finder) and is able to detect and track the moving obstacles in its view range. At each instance the robot knowledge about the world is incomplete and uncertain: incomplete in space because of the sensor range and the hidden areas and in time because of the limited validity of the motion models of dynamic obstacles and the unpredictability of new obstacles entering the scene. The uncertainty comes from the sensor error and accuracy and from the error of the motion models, the detection and tracking algorithms. Spatial uncertainty and incompleteness are represented by a probabilistic occupancy grid. In (BPPF06) the RRT algorithm is integrated in a *worst case* approach: obstacles are tracked in a dynamic environment and their kinematic model is used to bound the prediction of their future position. Also, partial planning is performed taking into account real-time constraints and the reliability of prediction.

We propose an algorithm to take into account a probabilistic representation of sensing

and prediction uncertainty which can be updated by new incoming information. Spatial uncertainty and incompleteness are represented by a probabilistic occupancy grid. Temporal uncertainty is represented using Gaussian mixture predictions (FTSL08).

9.1.1 Collision Risk

At a given instance, the robot knowledge about the state of the environment is represented by:

1. A list of pre-learned GPs which represent the typical patterns of the moving obstacles;
2. An occupancy grid, which represents the structure of the static environment around the robot, according to the previous observations;
3. A list of moving objects, their estimated position, velocity and previous observations;
4. An estimation of the state of the robot itself;
5. A goal state.

The configuration of the robot $q = (x, y, \theta, v, \omega, t)$ at a time instance t is given by the position (x, y) and orientation θ of the robot in the plane, the linear velocity v, the angular velocity ω. For each configuration q, a probability of collision $P_c(q)$ can be computed considering the static and moving obstacles and the perception limits.

The probability of collision with the static environment P_{cs} is computed considering the maximum probability of occupation among the cells of the occupancy grid touched by the robot in that configuration.

Consider M moving objects and K the number of GPs associated to the typical patterns learned. For an obstacle O_m, the predicted position at time t is estimated by a Gaussian Process mixture of K components. Considering each component k separately, the associated probability of collision $P_{cd}(k, m)$ is calculated considering the overlapping of the robot and the predictive distribution over future motion of O_m from the Gaussian Processes motion model (see chapter 6 section 6.6 on motion prediction). In practice, the probability of collision is computed in a set of points in the area occupied by the robot and the maximum is considered.

9.1.2 Traditional RRTs

The Rapid-exploring Random Tree (RRT)(LK99) is a randomized algorithm to explore large state space in a relatively short time. The algorithm chooses a point P in the configuration space and tries to extend the current search tree toward that point. P is chosen randomly, but generally in single-query planning, some bias toward the goal is applied in order to speed up the exploration. The nearest node neighbour of P within the

nodes of the search tree T, is chosen for extension. A new node is obtained applying an admissible control from the chosen node s toward P. If the node is collision free the new node is added to the tree. The algorithm can be stopped once the goal is found, or it can keep on running to find a better path. Once the goal state is reached, the path from the initial state to the goal is retrieved.

9.1.3 RRTs in Uncertain Environments

The traditional RRT assumes a deterministic representation of the environment, i.e. the algorithm knows a priori if a node is collision free or not. If the environment is not completely known, or if there are moving obstacles with unknown trajectories, the traditional RRT would not work.

We propose an extension of the basic RRT algorithm to take into account the probability of collision both during the exploration of the space and at the choice of the path. The probability of non-collision of a path becomes a measure of its feasibility. All the explored configurations are maintained in the tree and their probability of collision is updated by more recently acquired information. Our proposed extension gives an RRT algorithm which is risk averse.

Given a path $\pi(s_N) = \{s_0 ... s_N\}$ on the RRT search tree where each s_i is a node in the tree, the probability of collision for $\pi(s_N)$ is computed as probability of *not* having collision in each of the traversed nodes. For a single obstacle m and a Gaussian Process k :

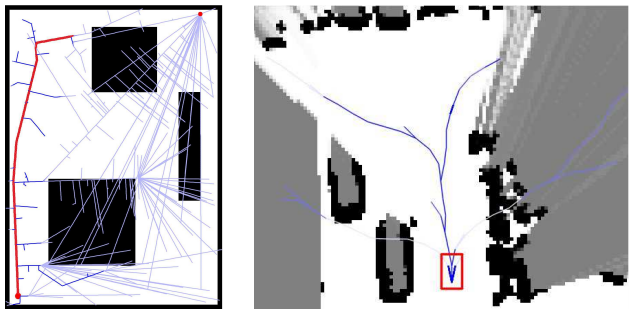
$$P_\pi(s_N, m, k) = 1 - \prod_{n=0}^{N} (1 - P_{cd}(s_n, m, k)) \tag{9.1}$$

The probability decreases exponentially with the length of the path. This is a sign that longer paths are more dangerous, as the uncertainty accumulates over subsequent steps. The probability of collision for $\pi(s_N)$ over all obstacles and motion patterns can be obtained by integrating over all objects and Gaussian Processes (see (FTSL08) for more details).

Figure 9.1(a) shows of the described algorithm for an holonomic robot within a static simulated environment. The initial position of the robot is in the bottom left corner, while the goal is in the upper right corner. Black rectangles are obstacles. The robot perceives the environment with a distance sensor, so that areas behind the obstacles are unknown. The colour of the edges of the tree depends on the probability of success of the associated path: the lighter the colour the lower the probability. In red, the path chosen. Figure 9.1(b) shows the described algorithm for a car-like robot within an observed occupancy grid.

9.2 Safe On-line Navigation

In a dynamic environment the robot has a limited time to perform planning which depends on the time-validity of the models used and on the moving objects in the environment.



(a) (b)

Figure 9.1: Probabilistic RRT in static environment. Figures show the search tree and the likelihood of the nodes (lighter colour is for lower likelihood). Observations are taken with a distance sensors, and there are occluded zones. (a) An example of a point holonomic robot in a simulated environment; (b) Another example for a car-like robot in an occupancy grid.

The conditions used for planning could be invalidated at execution time: for example an obstacle could have changed its behaviour or some new obstacle could have entered the scene.

The idea of Partial Motion Planning (PF05) is to take explicitly into account the real-time constraint and to limit the time available for planning to a fixed interval. After each planning cycle, the planned trajectory is generally a partial trajectory. The exploring tree is updated with the new model of the world and the final state of the previous trajectory becomes the root of the new exploring tree.

The planning algorithm works in parallel with execution. Each node of the tree is guaranteed to be not an Inevitable Collision State (ICS, (FA03)) by checking if there exists a collision free braking trajectory from the node. This is a conservative approximation that doesn't allow the robot to pass an intersection before an approaching moving obstacle. Our approach (FTSL08) presents an adaptable time horizon for planning. The time for the planning iterations depends on the length of the previous computed trajectory and on the on-line observations. Safety of a path is guaranteed studying braking trajectories only for the last state of the path.

9.2.1 Experimental Results

9.2.2 Laser Data Set

The algorithm has been tested with real data acquired on a car-like vehicle equipped with a laser range finder (Cycab (PHK$^+$05)). During the experiment, the robot is manually driven in an outdoor environment and perceives static obstacles and moving pedestrians. Using the algorithm developed in (TDV07 ; Bur07) the robot localizes itself, builds an occupancy grid map of the static environment and tracks the moving obstacles. The probabilistic

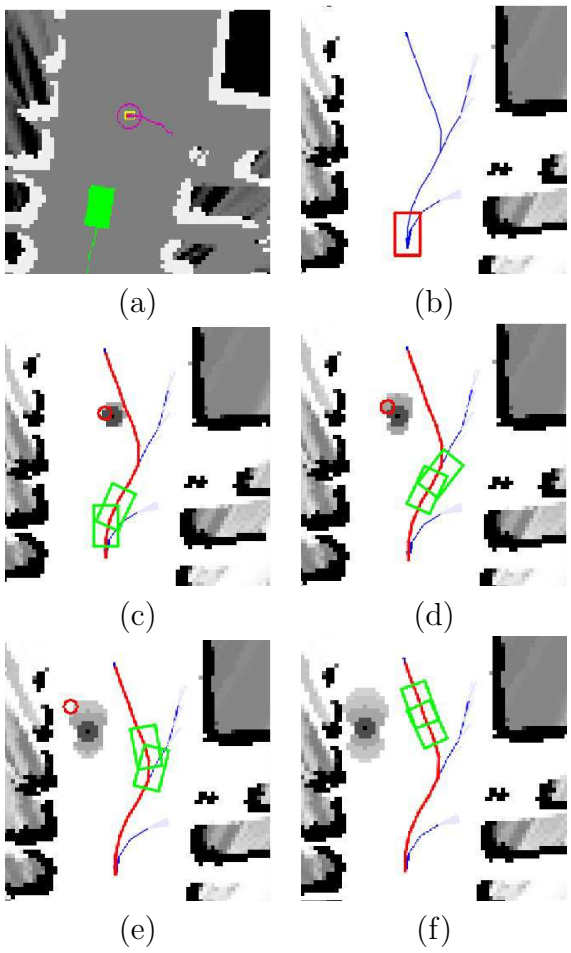


Figure 9.2: Results with a laser data set. (a) The static environment is mapped and the moving obstacles are tracked. (b) The algorithm explores the state space and chooses a path. (c-e) the path is compared with the real observations acquired.

predictions of the future state of the obstacles are computed using a constant velocity motion model and the position and velocity estimated, with their associated covariance. To test the planning algorithm we define a goal 20 meters ahead the robot at each observation cycle and let the algorithm run in parallel with the on-line mapping and tracking at 2Hz (fig. 9.2(b)). Each sequence is then tested with the real data, letting a virtual robot move through the map (fig. 9.2(c-e). Fig.9.2(a) shows the observed occupancy grid: free space is grey, occupied space white, non-explored space is black. The robot is the green rectangle and the circle is the tracked pedestrian with his trajectory. Fig.9.2(b) shows the tree of states explored in the available time: lighter blue is for higher probability of collision. The red line is the chosen path. Fig.9.2(c-f) shows subsequent positions of the virtual robot; on the background the occupancy grid predicted for planning; red circles represent instead the position of the moving obstacles as estimated in real-time by the tracking algorithm.

Results prove that the algorithm is able to compute safe trajectories in real time taking into account the static and moving obstacles perceived and the uncertainty in prediction of a real data set.

9.2.3 Simulation Results

Tests have also been conducted in a simulated environment. A set of 3000 trajectories has been simulated in a rectangular environment. A subset of 1000 trajectories have been used as training data set and, as a result, 26 Gaussian processes have been learned.
To test the probabilistic planning, we simulate the robot navigating among circular obstacles with trajectories that are chosen randomly from the simulated set. The static environment is supposed to be free and the perception of the robot is simulated. The time step chosen is of $0.5s$. Planning and execution run in parallel. Figure 9.3 shows some snapshot from the obtained results. The robot is the red rectangle and perceives the circular obstacle (red full point). The goal of the robot is at the bottom of the image (black circle). Green paths represent the mean of the Gaussian Processes: the likelihood of each GP is estimated at each time step on the basis of the previous observations; lighter colour is for lower likelihood. The tree explored by the robot is given by the blue lines. Again, lighter blue means lower likelihood. Circles represent the prediction for the obstacle for each associated Gaussian process and at subsequent time steps.

Fig. 9.3(a) shows the planning at the first time steps; in figure 9.3(b) the robot moves toward the goal, while the obstacle moves toward the upper left corner and the prediction gets better. around one GP only. In figure 9.3(c) the moving obstacle has disappeared behind the robot, while another one appears near the goal. Fig. 9.3(c), shows how the search tree is grown and the path is adapted to the new situation.

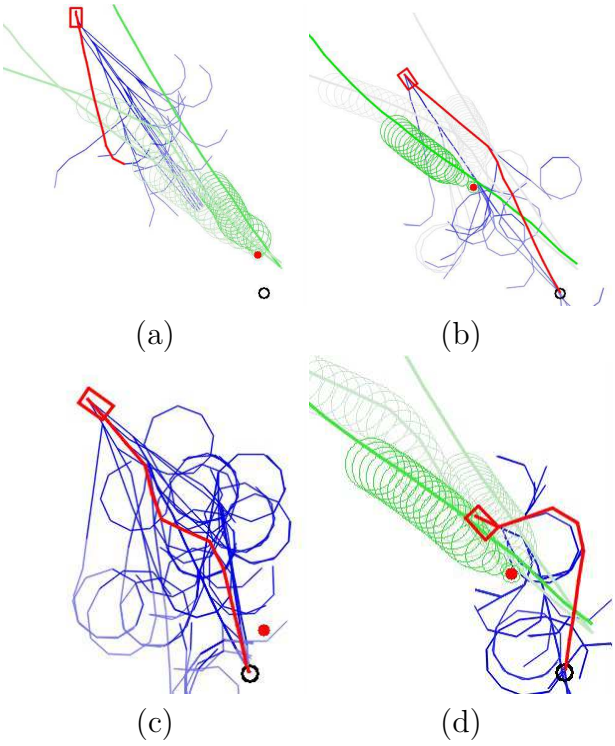


Figure 9.3: The robot moves in a simulated environment with a moving obstacle. The prediction of the obstacle is given by a Gaussian mixture based on the pre learned Gaussian processes (green). The exploring tree maintains an estimation of the likelihood of the path that adapts to the incoming observation.

Chapter 10

Collision Risk Estimation in Driving Assistance

The third experiment presented in this chapter is an application of motion prediction in urban road traffic scenes. The aim is the estimation of the risk of collision of a vehicle in a normal road traffic scene. For each of the other vehicles in the scene, its behaviours are inferred using the HMM and the probabilistic distribution over future paths for each behaviour are inferred with GPs. With a probabilistic representation of the future evolution of a certain scene, the risk of collision can be calculated. The details of the associated technique can be found in chapter 7.

Two different experimental validations within the context of driving assistance were conducted:

- The first is based on monte carlo simulations to validate its accuracy and reliability under a variety of situations and scenarios.
- Due to the nature of the application, it is impossible to produce real life crash testing. However, experiments based on an elaborate human driven scenario in a virtual environment were conducted and we present the results. The experiments were conducted in collaboration with Toyota Motors Europe and ProBayes.

10.1 Monte Carlo Simulation Validation

A pertinent question when performing experimental validation of the estimation of the risk of collision is its reliability. As a large number of different scenarios are required, it is infeasible in practice. A better method is to experimentally evaluate the estimation of risk of collision by randomly generating many scenarios under a variety of situations. To this end, a monte carlo based approach is adopted to sample the different scenarios for statistical evaluation.

The monte carlo simulations are performed over the space of different possible situations that can occur in a normal road traffic environment. The situations in this context includes

the different topology of the roads, the various configurations the vehicles in the scene are positioned and its associated dynamics.

Each sample thus represents a scene with a number of vehicles. Within each sample, one vehicle is identified as the ego-vehicle where the estimation of the risk of collision will be performed for the ego-vehicle.

10.1.1 Experimental Setup

One advantage of monte carlo based simulation is the ability to categorically analyze the algorithm under different specific situations. The situations are basically hierarchically organized in according to the topology of the road, and the behaviour of vehicle V_A which is the vehicle in question for which the risk of collision is to be estimated.

In the experimental setup, basically three basic road topology were identified; namely the parallel road, T - junction and cross -junction (see fig. 10.1):

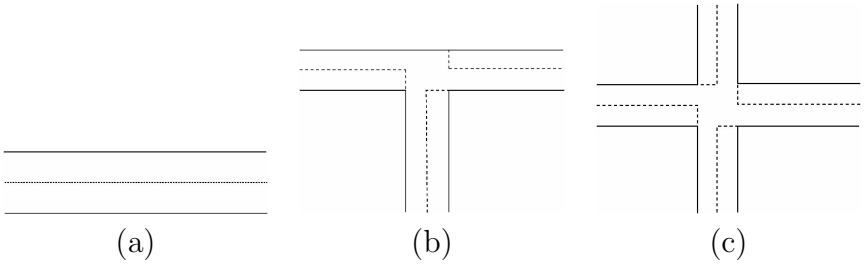


Figure 10.1: 3 different road topologies corresponding to Parallel, T-Junction and Cross-Junction

For each road topology, the situations can be artificially generated by generating a random number of vehicles in the scene. Each randomly sampled vehicle in the scene includes randomly chosen parameters such as the intended route (given indirectly from a randomly chosen behaviour), its starting position, its velocity and acceleration. One of the vehicles is designated as vehicle V_A for which the estimation of the risk of collision will be calculated. Each randomly sampled situation will then be simulated by evolving the random vehicles in the scene in time. At every time step, the risk of collision of the ego-vehicle is calculated and recorded.

In practice, the generation of the samples is performed in a hierarchical fashion (see fig. 10.2). The upper level of the hierarchy represents the different road topology. The middle level represents the different behaviours for the ego-vehicle within each road topology. For example, the middle level will represent the situation where the ego-vehicle is going straight, turning left and turning right for the road topology T-junction. The bottom level will then be each individual situation / samples. Additionally, a Gaussian noise added along the samples to test its robustness.

For the monte carlo experiments, we assume that the estimation of the behaviours, which in reality is to be estimated by the HMM, is known fully. The reason for this is to

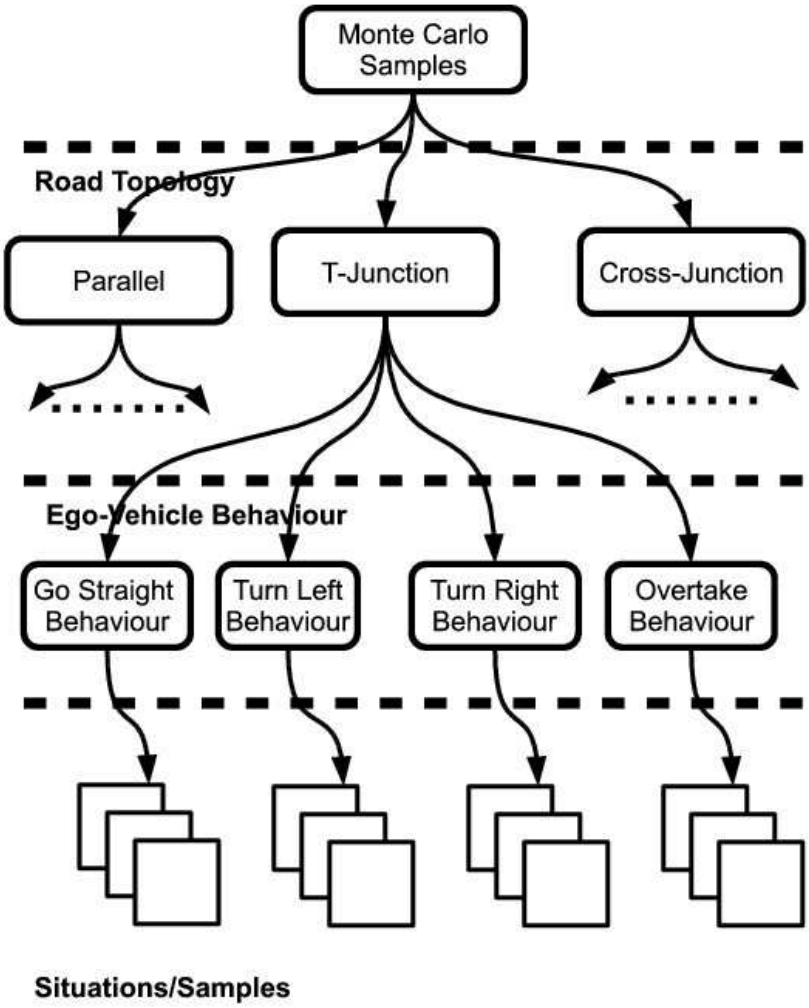


Figure 10.2: Organization Of Monte Carlo Simulation

evaluate the estimation of the risk of collision independently from the output coming from the layered HMM.

10.1.2 Results

The results in this sub-section presents the graph of the evolution of risk values 4 seconds before collision. 200 situations were randomly sampled from every combination of road topology and behaviour for vehicle V_A. For each instance in time where there is a collision for the ego-vehicle, the values of the estimation of risk of collision for the 4 second duration before collision is recorded. The plot of the mean and variance of risk values 4 seconds before each collision can be obtained.

In our experiments, the risk that we would like to evaluate for a vehicle V_A is the risk of its intended trajectory T_A. This risk value can be used as a feedback output to a risk sensitive vehicle control module, or it can be easily generalized to be a collision warning system, by evaluating different trajectories T_A as described in chapter 7. The risk of trajectory T_A against another vehicle V_i is given by:

$$P(C|T_A\, V_i) = \sum_{B_{V_i}} P(C|T_A\, B_{V_i}\, V_i) P(B_{V_i}|V_i) \tag{10.1}$$

Where $P(B_{V_i}|V_i)$ originally refers to the probability distribution over behaviours from the layered HMM corresponding to vehicle V_i. However, to evaluate the risk estimation independently from the layered HMM, $P(B_{V_i}|V_i)$ in the monte carlo experiments is a Dirac distribution centered at the known behaviour B_{V_i} for vehicle V_i.

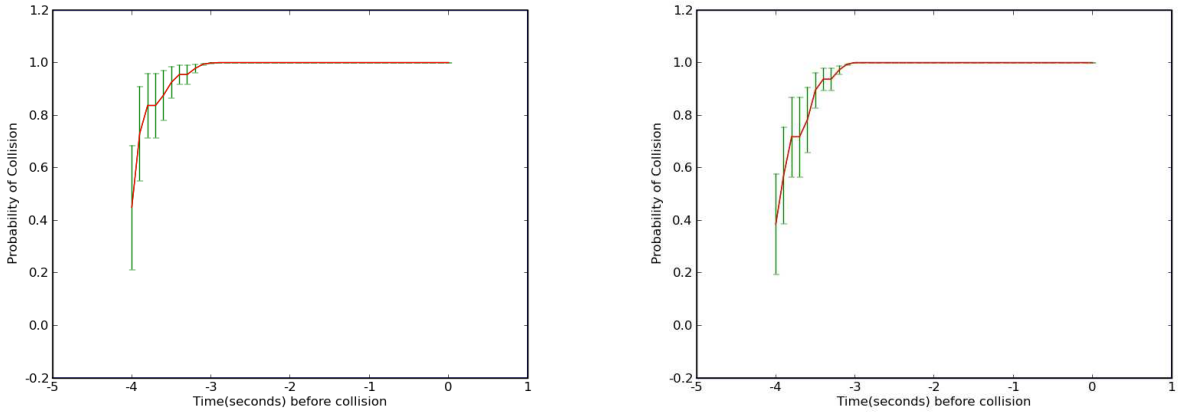
The time horizon of risk evaluation is for 3 seconds. Technically, this means that the prediction of future motion trajectory is limited to a distance of the maximum speed multiplied by 3 seconds. Finally, the risk amongst all vehicles is aggregated by taking the maximum risk of all vehicles.

Figure 10.3 shows the mean and variance of the risk values 4 seconds before collision, along a parallel stretch of road. Each sub-figure displays the corresponding plots when vehicle V_A is executing a certain behaviour.

Figures 10.4 and 10.5 shows the corresponding plots of risk means and variance in a T-junction and cross-junction respectively, along with its different behaviours for vehicle V_A in each of the T-junction and cross-junction.

It can be observed from all the graphs in figures 10.3, 10.4 and 10.5, the risk estimated 3 seconds before collision consistently achieves a value of approximately 1.0 across all graphs. As the risk value is a probability value, this means that the risk estimation for collision is consistent and reliable for the time horizon that we wish to compute for. However, these set of results are obtained knowing fully the behaviour of the vehicles in the scene thus masking the effects of the behaviour estimation from the layered HMM.

As mentioned previously, the distance for which the future motion trajectory is sampled from is the fastest speed of the vehicle multiplied by 3 seconds. As vehicles often travel below the assumed fastest speed, we are able to predict risk values beyond the 3 seconds,



(a) Lane Change (b) Go Straight

Figure 10.3: Plot of mean and variance of risk estimation for parallel road

which explains the rising risk values even beyond 3 seconds before collision. It has also been observed from the graphs that when vehicle V_A is going straight, the risk values beyond 3 seconds before collision rises slower and has greater variance compared to the other cases. It seems to suggest that under normal driving conditions, there is less of a chance of colliding with the vehicle in front. This can be explained by the fact that our model does not take into account behaviours such as sudden braking.

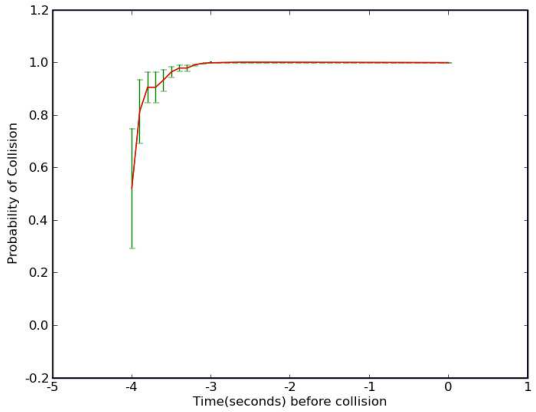
10.2 Driving Simulation

As it is difficult to perform experiments involving real life crash situations, experiments were performed in a virtual environment. The virtual environment is a virtual driving environment/simulator developed by Toyota Motors Europe.

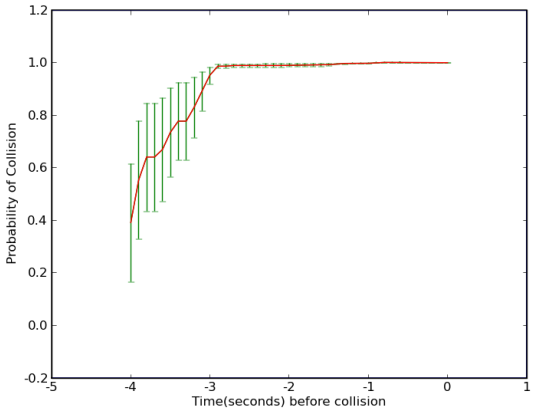
The virtual environment is a geometrical model of the world in three dimensions, consisting of a road network populated with vehicles. To increase the realism of this virtual environment, with respect to risk estimation, the vehicles populated in the scene are driven by a human. The experiments in the simulator is performed in collaboration with Toyota Motors Europe and ProBayes.

10.2.1 Experimental Setup

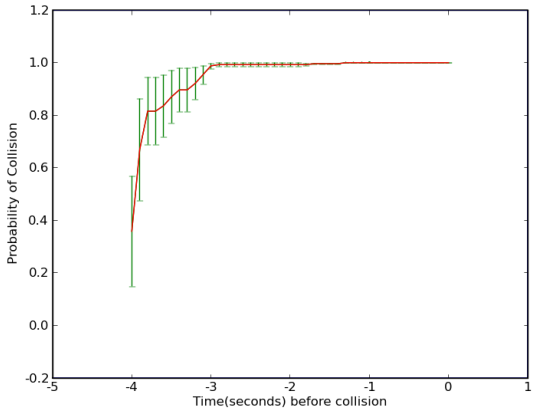
The virtual environment simulates traffic conditions consisting of a number of vehicles travelling along the road network. Each of the vehicles is driven by a human driver. Each human driver controls his virtual vehicle via a wheel joystick as if the human driver is in the driver's seat. Recording a large scenario with many vehicles driven simultaneously requires a large number of human drivers and wheel joysticks. Our scenarios are generated



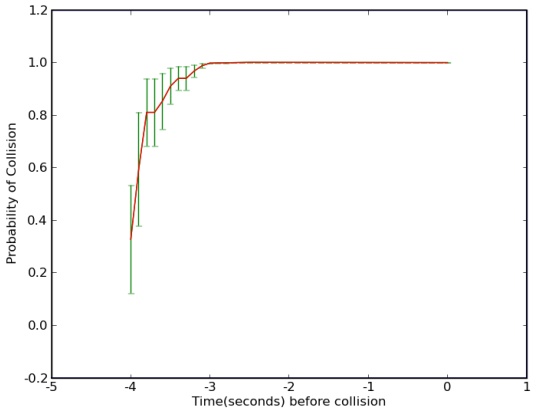
(a) Lane Change



(b) Go Straight

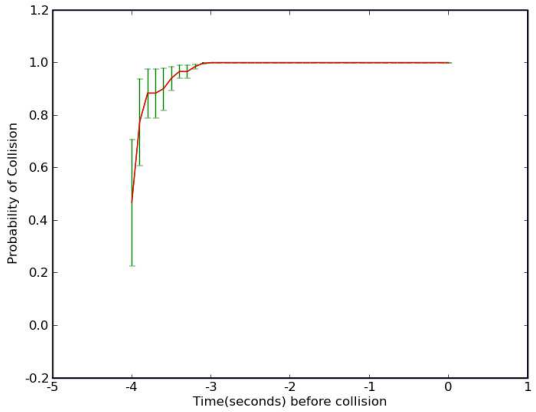


(c) Turn left

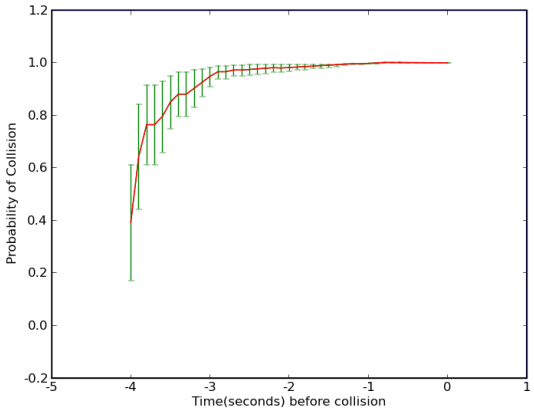


(d) Turn Right

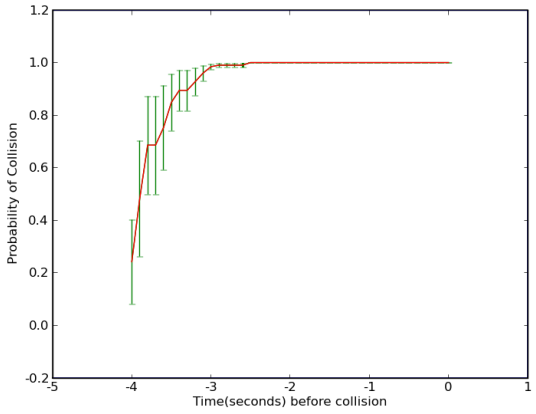
Figure 10.4: Plot of mean and variance of risk estimation for T-Junctions



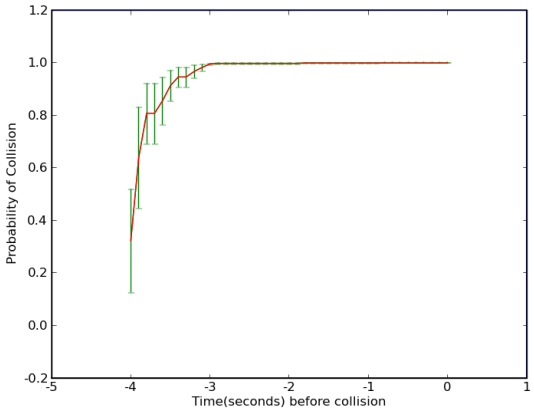
(a) Lane Change



(b) Go Straight



(c) Turn left



(d) Turn Right

Figure 10.5: Plot of mean and variance of risk estimation for Cross-Junctions

in an iterative fashion where only a single human driven vehicle is recorded at a time. In each iteration consisting of a single human driven vehicle, the previously recorded human driven vehicles are replayed. The entire scenario is generated by this process of iteratively "adding" human drivers into the scene. Because of the virtual environment, crashes can be easily and safely created.

In a scenario, the risk of a single designated vehicle, V_A will be evaluated. The risk to be evaluated is the same as that of the monte carlo experiments of section 10.1.2, equation 10.1. In contrast to the monte carlo experiments, where the behaviours of all vehicles are known, no behaviours on vehicles are known. A layered HMM evaluates the behaviour for every vehicle present in the scene except vehicle V_A. Different time horizons for the evaluation of risks were also performed.

(a) (b)

Figure 10.6: Driving Wheel in (a) and Annotation tool in (b)

The training of the HMM is performed as described in chapter 4 section 4.2.2. Training data was collected by collecting driving sequences from a number of human participants. Each participant uses the driving wheel (fig. 10.6(a)) as an interface to the virtual environment to simulate driving from the point of view of the driver in 3D. The set of collected driving sequences are then annotated manually using an annotation tool developed by ProBayes (fig. 10.6(b)) before being used as the training data to train the parameters of the layered HMM.

Figure 10.7: Screen Capture of Simulator

10.2.2 Results

Figure 10.7 shows the screen shot of the simulator. The simulator consists of a top down 2D view of the environment, and a 3D view from the point of view of the driver. This 3D view window is also used by the human drivers when the human drivers are used to record the different scenarios. In the simulator, the risk calculated will be for the yellow vehicle whereas all other vehicles are red. For our experiments, the convention is that vehicles drive on the right lane.

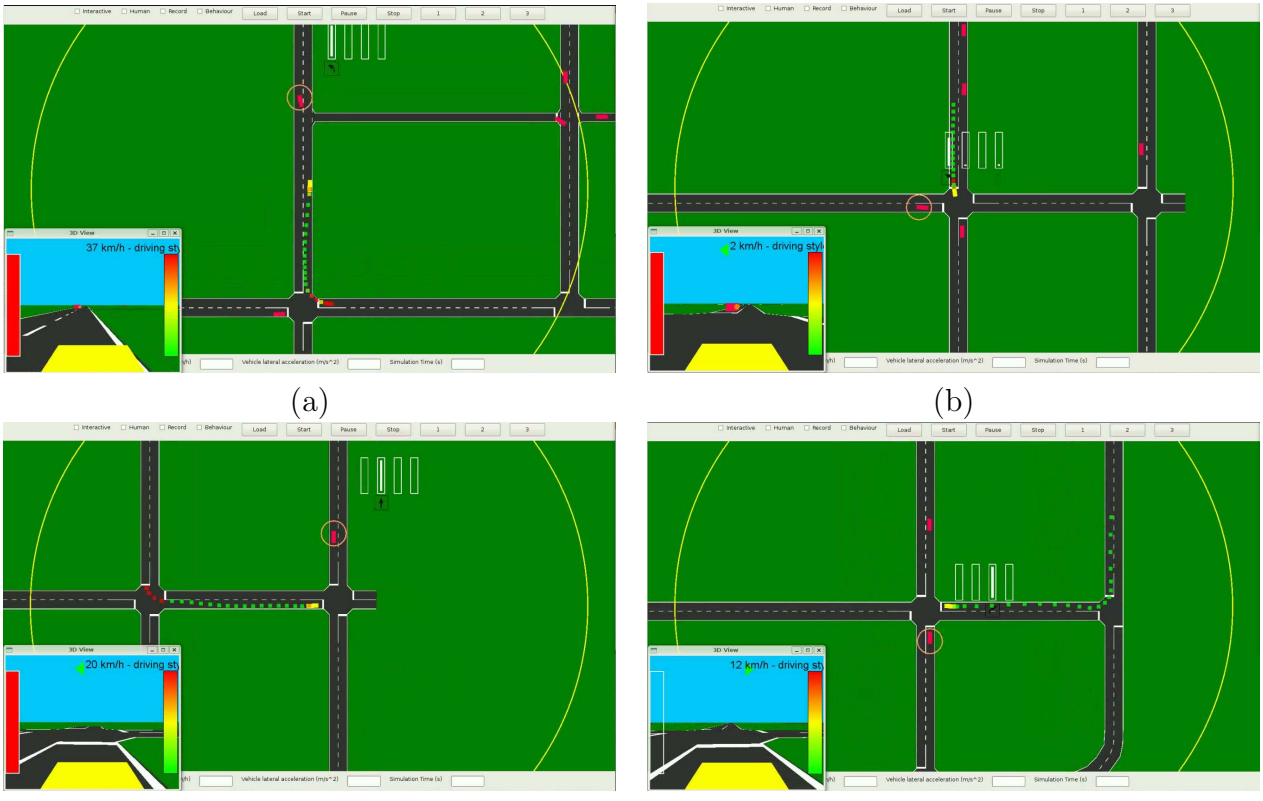
In the 2D view, a color coded trail behind the yellow vehicle indicates the estimated levels of risk previously. The big yellow circle indicates the radius in which the red vehicles are taken into account. At all moments, the red vehicle nearest to the yellow vehicle will have its estimated layered HMM behaviour probability displayed as vertical white bars. The 3D view indicates the speed of the yellow vehicle. The vertical color coded bar on the right gives the various risk value encoding from green being the least risky to red representing high risk. The vertical bar on the left indicates the current risk value for the yellow vehicle.

Current commercial crash warning systems are able to warn a driver if he is travelling too fast and about to collide with another vehicle in front. Our risk estimation algorithm is able to provide the same functionality. Figure 10.8 shows two such examples. In these examples, the red vehicles in front are estimated by the layered HMM to be turning left or right. The yellow vehicle has a relatively high speed with respect to the red vehicles and a high risk level is estimated as observed by the red vertical bar in the 3D view.

Our risk estimation algorithm is able to reason in more complicated situations such as the intersection. Figures 10.9(a) and 10.9(c) shows examples where an assumption on linear motion would not give a reasonable risk estimate when there are high risks of collision in reality. In these two situations, the layered HMM is able to correctly determine

(a) (b)

Figure 10.8: Collision risks with vehicles in front. Similar to current state of the art systems.



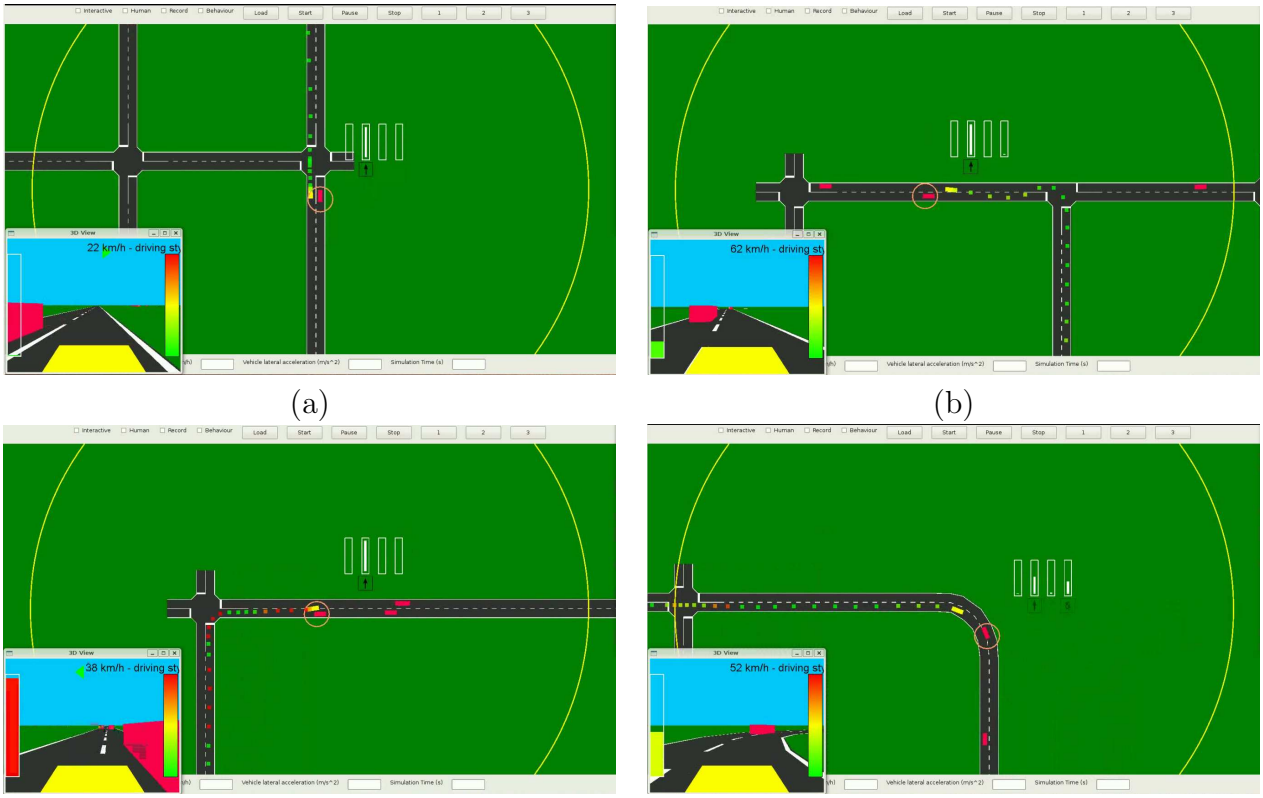
(a) (b)

(c) (d)

Figure 10.9: Taking into account intersection when evaluation risk

the behaviour of the red vehicles. The combination of the behaviour estimation from the layered HMM and taking into account the semantics (turning or going straight) at the geometrical level gives the appropriate high risk values.

Figures 10.9(b) and 10.9(d) shows very similar situation at the cross intersection where the yellow vehicle and its nearest red vehicle might enter into collision. Again, by appropriately taking into account behaviour probability distribution and geometry, reasonable risk probabilities are obtained. Figure 10.9(b) presents a situation with a high risk of collision. In fact, if we look more carefully, regardless of the behaviour of the red vehicle, chances of it going into collision is high. The situation for 10.9(d) looks very similar but it does not go into collision because the the layered HMM had correctly recognized a turning right behaviour for the red vehicle and does not enter into collision with the yellow vehicle. Current methods assuming linear motion will probably trigger a high false alarm in this case.



(a) (b)

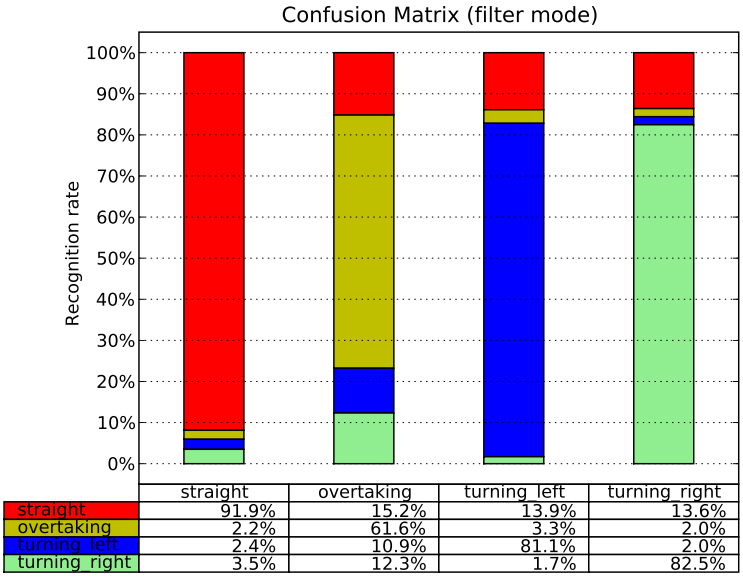
(c) (d)

Figure 10.10: Risks associated with passing by another vehicle along the adjacent lane.

The application of Gaussian Process in modelling deviation from the center of the lane gives intuitive risk values. Vehicles which deviates far from the center of the lane poses non negligible risks to other vehicles travelling on the adjacent lane. This effect can be seen from the sequence of figures 10.10(a) - (c). In general, the risk values for vehicles about

to pass by each other is higher as the vehicles passes by each other gets closer. As noted in chapter 7 section 7.1.4, the geometric configuration of the vehicle is taken into account. Using a simple threshold on the euclidean distance between two vehicles consistently gives false alarms of high collision probability each time a vehicle is about to go pass adjacent to another. Figure 10.10(d) shows another instance where an assumption on linear motion will give a high probability of collision which is inaccurate. By adapting the Gaussian Process to the curved lane geometry of the road, a more reasonable risk value is obtained as can be observed by the left vertical bar on the 3D view.

Figure 10.11 gives a quick summary on the recognition performance of the layered-HMM as a confusion matrix. The confusion matrix is a visualization tool where the columns represents the true class and the rows represents the predicted class. From the confusion matrix it is easy to see the percentage of mislabelling for each class and the diagonal of the confusion matrix represents the correctly predicted class. The highest recognition rate was for the going straight behaviour, followed by the turning left/right behaviour. The overtaking behaviour had a relatively low recognition rate (61.6%). Intuitively, this is because it is easier to confuse the overtaking behaviour which consists of lower level behaviours, lane changing, accelerating, lane change back to original and resuming normal speed. These lower level behaviours can easily be mixed up with the other behaviours.



	straight	overtaking	turning_left	turning_right
straight	91.9%	15.2%	13.9%	13.6%
overtaking	2.2%	61.6%	3.3%	2.0%
turning_left	2.4%	10.9%	81.1%	2.0%
turning_right	3.5%	12.3%	1.7%	82.5%

Figure 10.11: Confusion matrix on the performance of the layered HMM

Figure 10.12 shows the plot of the risk values 4 seconds before each collision. The means and variances were computed using the vector of risk values 4 seconds before each collision across 10 different scenarios. The time horizon of the risk evaluation for the 10 scenarios were fixed at 3 seconds.

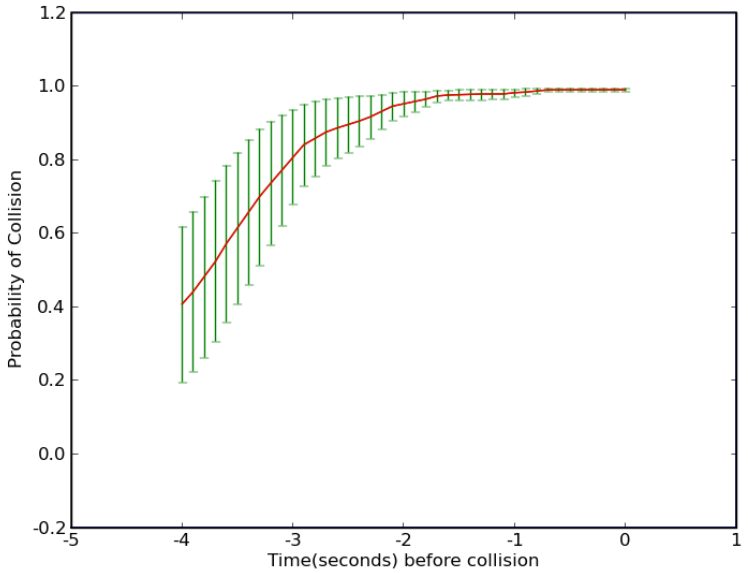
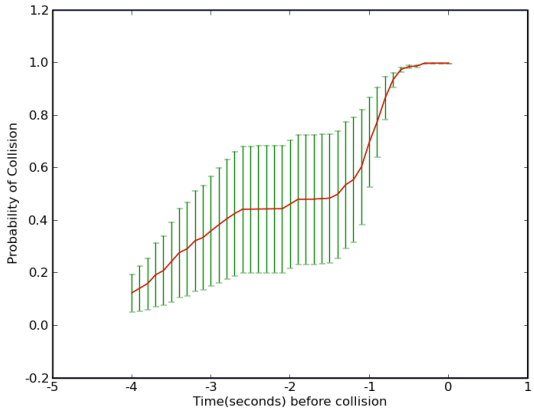
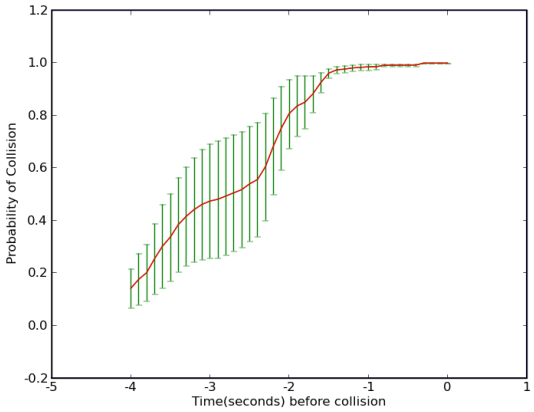


Figure 10.12: Aggregate risk mean and variance for 10 human driven scenarios. The time horizon for collision is 3 seconds.

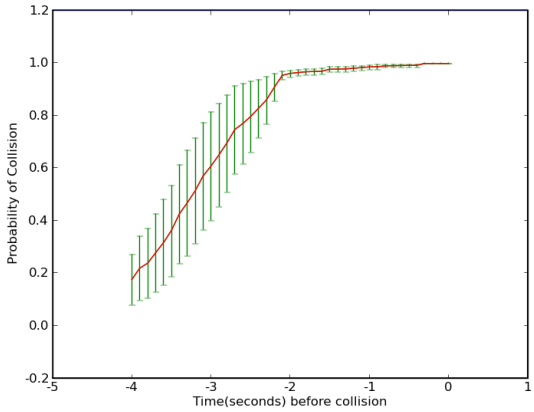
To evaluate the effects of different time horizons in the evaluation of risk, the experiments for the 10 scenarios were performed for different time horizons, for 1, 2, and 3 seconds. The means and variances 4 seconds before each collision across 10 scenarios are displayed in figure 10.13.



(a) 1 Second



(b) 2 Seconds



(c) 3 Seconds

Figure 10.13: Plot of mean and variance of risk estimation for different estimation time horizons

Part VI

Conclusion

Chapter 11

Conclusion and Perspectives

In this thesis, we focused on the motion of an agent or by an agent driving the motion. The underlying theme of this thesis is to learn and model such motion patterns in a consistent manner, thereby evaluating its risk, based on the language of probability. To this end, we presented problems and solutions for learning and modelling motion patterns with and without constraints imposed on motion by the environment.

The basic motion pattern model was presented in chapter 6 where Gaussian Processes (GP) were used to model motion patterns in an open environment. A single GP models a motion pattern and a probabilistic representation of the mean path and how much the path varies. Using GPs to model motion has the advantage of fully describing motion consistently and probabilistically. Furthermore, the issue of discretization was avoided unlike state-space models [eg. (BBCT05) (VGF07)].

We have also presented the extension of the GP as a motion pattern model to urban traffic environments in chapter 7. A fully probabilistic motion model that describes the probabilistic evolution of vehicles on urban traffic roads was applied to estimating the risk of collision of vehicles. However, the vehicle probabilistic evolution model is not limited to estimating risk of collisions. It can also be used as part of a fully autonomous vehicle for example.

The main drawback of using the mixture of GPs (the solution presented for open spaces) to model several motion patterns is its scalability to larger environments. We proposed to exploit the repeating structures existing in urban traffic environments to address this scaling issue. To do so, motion patterns are always relative to vehicles and is dependent on its local environment. A layered HMM estimates the probability distribution of the behaviour that the vehicle is executing, i.e. turning left/right, overtaking. And each behaviour consists of a GP that expresses the probability distribution over the physical trajectory realization of the corresponding behaviour. We introduced a method of adapting the GPs to the geometry of the road by performing a conformal mapping. The layered HMM and GP model gives a fully probabilistic vehicle evolution model.

In chapter 7 we also presented a variety of risk values that can be computed using the probabilistic vehicle evolution model. The risk values to be computed depends on how the risk values are going to be interpreted and used. Current risk estimation systems are

based on variants of time-to-collision (TTC), where a linear assumption was made. Hence, risk estimation around bends or corners are not properly taken into account. In contrast, our risk estimation based on the probabilistic vehicle evolution model is able to take these factors into account.

Experiments were conducted for both motion with and without constraints. Chapter 8 involves motion pattern modelling in open spaces where motion patterns were extracted from a set of human trajectory motion in the INRIA Rhône-Alpes entry hall. To further illustrate its uses with an application, motion patterns were applied to the problem of navigation in open environments in chapter 9. The path planning algorithm used is a probabilistic extension of Rapidly-exploring Random Trees (RRT) and experiments conducted with the author (FTSL08).

Further experiments were conducted for collision risk estimation. Monte carlo simulations were used to evaluate and validate the accuracy and reliability of GPs for estimating collision risk. As real life crash experiments could not be conducted, the risk estimation is evaluated in a virtual environment where vehicles were driven by humans in order for it to be as realistic as possible. We managed to show that reasonable risk warnings were given for the different time horizons and reported on the characteristics of the risk estimation based on experimental observations. The experiments in the virtual setting was made possible but the joint efforts with Toyota Motors Europe (TME) and French technology enterprise ProBayes.

11.1 Perspectives

The work presented in this thesis provides solutions for modelling motion patterns in in open and structured spaces. Nonetheless, it has a number of shortcomings. We highlight the shortcomings and provide several suggestions and extensions to our current work with its implications on the scope of possible applications.

Extracting Common Motion Patterns.

In the mixture of GPs approach to modelling motion patterns in open spaces in chapter 6, each GP represents an entire motion pattern sequence. However, this model does not account for similar subsequent between distinct motion patterns or what we shall call common motion sub-pattern. It is not difficult to imagine a more expressive model where motion patterns at the high level consists of stochastic transitions between common motion sub-patterns.

The advantage of doing so will be a more compact representation of the environment. Furthermore, this framework enables the modelling of transitions between different motion patterns. A similar model (WMNG08) used the Hierarchical Dirichlet Processes (HDP) (TJBB06) to extract these common motion sub-patterns and is analogous to hierarchical document clustering. However, the clustering takes places over the discretization of space and velocity. The motion pattern looses the ability to

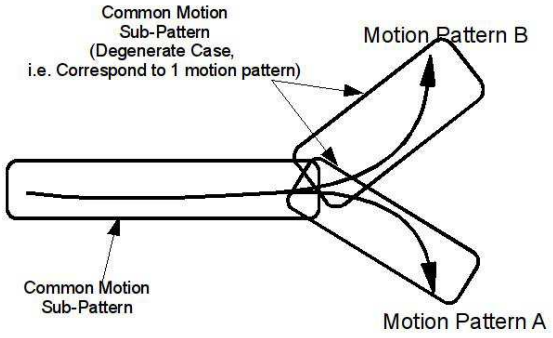


Figure 11.1: Given two motion patterns, common motion sub-pattern can be extracted. The degenerate case refers to motion sub-patterns that shares with only one motion pattern.

express mean and variances along motion pattern and faces increasing data complexity as the space-velocity discretization gets finer. An ideal solution consists of being able to slice the GP, where the sliced GP represents a common motion sub-pattern, and these sliced GP can be patched up combinatorially.

It is interesting to note that the method by Wang (WMNG08) can be directly used to extract motion patterns coming from the Bayesian Occupation Filter (BOF) (TMY$^+$08) and is a good fit. This is due to the fact that the basic principle of the BOF is to perform Bayesian filtering over discretized space-velocity and can easily be adapted for extracting motion patterns and its sub-patterns in a scene with a static frame of reference.

Online Learning.

Our current work requires a training stage where the algorithms learns the appropriate model and GP parameters when presented a data set. The robustness of the frameworks both in open and structured spaces can be increased by performing what is known as online learning. There are several definitions of online learning, but what we refer to is the learning of very large, potentially infinite, streaming data.

In open spaces, it is reasonable to expect that motion patterns might change over time. An online learning algorithm should be capable of "forgetting" previous motion patterns and adapt to new ones when required. A simple solution would be to add exponential decay on the weights of the GP clusters in the mixture model. A better solution would consist of adaptively removing clusters if motion patterns are not observed for some time and clusters are adaptively created as a number of new motion patterns are observed.

Another key obstacle to online learning in open spaces is the speed of learning the

parameters to the mixture model. A possible solution might be to perform for each iteration, one pass of variational EM updates and optimize the GP parameters with stochastic gradient descent (Bot04). Such stochastic gradient optimization methods have garnered attention in recent years due to its ability to avoid local minima and handle large or infinite data (BB08).

Online learning for structured spaces involve more complicated models for taking into account new behaviours and being robust enough to handle a wide variety of different geometrical structures.

Towards Non-holonomic motion.

In our model for collision risk estimation, the trajectory realization of behaviours of vehicles in the probabilistic vehicle evolution model does not have non-holonomic constraints. A non-holonomic motion is intuitively characterized by differential constraints in the kinematic motion model of an object. For example, a normal four wheeled vehicle is constrained to move in the direction of the wheels, but not perpendicularly. Modelling stochastic non-holonomic motion is a difficult topic and is still an open question.

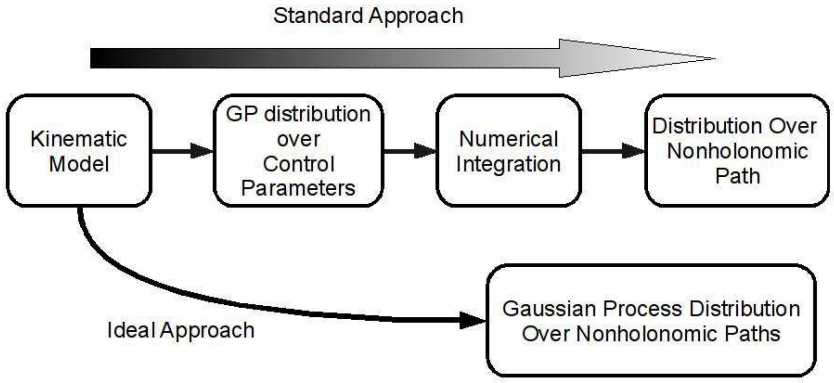


Figure 11.2: The ideal case will be given the kinematic model, one obtains directly a probability distribution over non-holonomic paths without requiring to perform costly numerical integration.

In non-holonomic motion, it is almost always required to perform numerical integration over the path in parameter space to obtain the path itself (see figure 11.2). Numerical integration is computationally expensive and real time robotics requires a much faster solution if non-holonomic motion is to be taken into account, in order to fit into our model for collision risk estimation. Instead of performing integration, it would be ideal to be able to omit the numerical integration to arrive at the motion path directly and express it probabilistically. Non-holonomic constraints presents

non trivial problems when adapting GPs geometrically (conformal mapping of GPs) as well.

However, we argue that the current GP model is sufficient for modelling vehicle motion in structured spaces. It is unclear even if non-holonomic constraints being taken into account would yield more accurate collision risk estimates. Furthermore, we have not observed any effects with our current model in our experiments.

Incorporating Location Dependent Priors.

The current collision risk estimation makes the assumption that the parameters generalizes on the global scale. However, it would also be interesting to take into account localized influences. For example, drivers in or near big cities usually exhibit more aggressive behaviours than in less populated smaller towns, or in a different country. GP and model parameters for the probabilistic vehicle evolution model can either be adapted in an online fashion as previously described to the new environment, or these parameters can serve as priors depending on the location.

Appendices

Appendix A

Inference Algorithms

A.1 Belief Propagation

Belief propagation (BP) algorithm is a message passing algorithm for exact inference in *single-connected* DAGs (Pea88).

Definition A.1.1. Singly-connected Directed Acyclic Graphs (DAG). A DAG is singly-connected when there exists only one undirected path between any two nodes in the graph. An undirected path in the DAG is to ignore the direction of the edges in the DAG.

In fact the "pushed inward" inner sums of equation 3.12 can be seen as a message being passed from node A to nodes B and C. Variable elimination seeks the conditional probablity of a single variable given its obervations, BP computes the conditional probability all the conditional distributions of the unobserved variables given the observed variables. Given a factor graph, the messages can be computed as:

$$m_{x\to f}(x) = \prod_{h\in\partial x\setminus f} m_{h\to x}(x) \tag{A.1}$$

$$m_{f\to x}(x) = \sum_{\mathbf{X}\setminus x} f(\mathbf{X}) \prod_{y\in\partial f\setminus x} m_{y\to f}(y) \tag{A.2}$$

$$P(x) \propto \prod_{h\in\partial x} m_{h\to x}(x) \tag{A.3}$$

Where ∂x and ∂f refers to the neighbours of variable node x and function node f respectively. The algorithm works by first initializing the observed variables to the correct value appropriately. It then selects a root node and messages a propagated down the leaves of the tree and back up to the root node again. The heart of this algorithm is that the messages are cached in the edges of the graph and the messages can be reused in the computation of the marginal probability of every node.

A.2 Junction Tree Algorithm

The junction tree algorithm (JLO90) is applicable when a graph is not singly-connected. The main idea is to group variables appropriately such that a tree is obtained from this reduction and the message passing algorithm can then be used.

In general, the junction tree algorithm works in three stages:

1. *Moralisation* is the first step involved in converting a DAG into an undirected graph. This is done by introducing edges between every pair of variables with the same child. The directed edges are then replaced by undirected ones.

2. *Triangulation* is then performed by adding edges such that there are loops in the undirected graph with lengths 4 or more.

3. *Construct a maximal spanning tree* from the cliques in the graph.

4. *Inference* on the graph using belief propagation algorithms.

The point of triangulating the graph is to ensure that there exists a tree which has the *running intersection property*.

Definition A.2.1. (Running intesection property). A clique tree posses the running intersection property if for every pair of cliques V and W, all cliques on the path between V and W contains $V \cap W$.

This property assures that the variables are consistent and not represented in disjoint parts of the tree. The complexity of this algorithm rises with the size of the largest clique in the graph.

A.3 Exepectation Propagation

Expectation Propagation (EP) (Min01b) (Min01a) is another method for approximate probabilistic inference under the class of variational methods. Given an *iid* data set $D = x_1, \ldots, x_N$, a model with parameters θ is assumed to have factorized posterior:

$$P(\theta|D) = P(\theta)\prod_{i=1}^{N} P(x_i|\theta) = \prod_{i=0}^{N} f_i(\theta) \tag{A.4}$$

Where $f_0(\theta) = P(\theta)$ and $f_i(\theta) = P(x_i|\theta)$. This product of factors can be approximated by a product of simpler terms:

$$Q(\theta) = \prod_{i=0}^{N} \tilde{f}_i(\theta) \tag{A.5}$$

The true and approximate distributions can be minimized by searching for the $Q(\theta)$ which minimizes the KL divergence between the two distributions:

$$\min_{Q(\theta)} KL\left(\prod_{i=0}^{N} f_i(\theta) \| \prod_{i=0}^{N} \tilde{f}_i(\theta)\right) \tag{A.6}$$

However, the KL divergence of equation A.6 is intractable in general as the KL divergence involves averaging with respect to the true posterior distribution. A simple alternative will be to take the KL divergence between each individual factor, $KL(f_i(\theta)\|\tilde{f}_i(\theta))$. However, it might be a poor approximation as it does not take the product of the factors into account. Hence EP proposes a slight modification to the naive approach:

$$\min_{\tilde{f}_i(\theta)} KL\left(f_i(\theta)\prod_{i\neq j}^{N} \tilde{f}_j(\theta) \| \tilde{f}_i(\theta)\prod_{i\neq j}^{N} \tilde{f}_j(\theta)\right) \tag{A.7}$$

The KL divergence in EP is different from that of Variational Bayes. Given the approximation distribution Q, EP minimizes the KL divergence $KL(Q\|P)$ while Variational Bayes minimizes the KL divergence $KL(P\|Q)$. Additionally, Variational bayes method performs the KL divergence with respect to the true distribution whereras EP averages with respect to the approximate distribution and is often tractable. A simple EP algorithm iterates the updates of the approximation factors $\tilde{f}_i(\theta)$:

Algorithm 6: Simple Expectation Propagation Algorithm

repeat
  for $i = 0 \ldots N$ **do**
  `/* Deletion Of Appropriate Factor */`
  $Q_{\setminus i}(\theta) = \frac{Q(\theta)}{\tilde{f}_i(\theta)} = \prod_{j\neq i} \tilde{f}_j(\theta)$;
  `/* Update factor */`
  $f_i^{\tilde{new}}(\theta) = \arg\min_{f(\theta)} KL(f_i(\theta)Q_{\setminus i}(\theta)\|f(\theta)Q_{\setminus i}(\theta))$;
  `/* Store New Factor */`
  $Q(\theta) = f_i^{\tilde{new}}(\theta)Q_{\setminus i}(\theta)$;
until *Until Convergence* ;

Appendix B

Gaussian Process

B.1 Mercer's Theorem

Theorem B.1.1. ***(Mercer's Theorem)***. *Let $(\mathcal{X}, \mu)$ be a finite measure space and suppose k is a continuous positive semi-definite kernel on a compact set $\mathcal{X}$ and the integral operator $T_k : L_2(\mathcal{X}, \mu) \mapsto L_2(\mathcal{X}, \mu)$ defined by*

$$(T_k f)(.) = \int_{\mathcal{X}} k(., x) f(x) dx \tag{B.1}$$

is positive semi-definite (see definition 4.18). Then there is an orthonormal basis $\{\phi_i\}_{i=1}^{\infty} \in L_2(\mathcal{X}, \mu)$ consisting of eigenfunctions of T_k such that its eigenvectors $\{\lambda_i\}_{i=1}^{\infty}$ are non-negative and absolutely summable. A PSD kernel function k has the representation:

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i(x') \tag{B.2}$$

The convergence of equation B.2 requires that $\sum_i \lambda_i < \infty$, thus $\lambda_i \to 0$ as $i \to \infty$. The convergence is also absolute and uniform:

$$\lim_{n \to \infty} \sup_{x, x'} |k(x, x') - \sum_{i=1}^{n} \lambda_i \phi_i(x) \phi_i(x')| = 0 \tag{B.3}$$

□

The spectral decomposition of covariance functions from Mercer's Theorem (theorem B.1.1) leads to a *Karhunen-Loeve expansion*. Given a zero-mean GP with covariance function k, i.e. $Z \sim \mathcal{GP}(\mathbf{0}, k)$:

$$Z_n(x) = \sum_{i=0}^{n} z_i \phi_i(x) \tag{B.4}$$

where $z_i \sim \mathcal{G}(0, \lambda_i)$ and the different z_i are mutually independent. $\{\phi_i\}$, $\{\lambda_i\}$ are the eigenvectors and eigenvalues repsectively from the spectral decomposition of $k(x, x')$. $Z_n(x)$ converges to $Z(x)$ in quadratic mean:

$$\lim_{n\to\infty} \int [Z_n(x) - Z(x)]^2 \, d\mu(x) = 0 \tag{B.5}$$

The space of GP samples is a Hilbert space. Eigenvalues are the projection of $Z(x)$ to its eigenfunction:

$$z_i = \int Z(x)\phi_i(x) d\mu(x) \tag{B.6}$$

B.2 Duality With Reproducing Kernel Hilbert Spaces

From Mercer's theorem and the GP formulation with a covariance function k, it seems to suggest a link between GPs and vector spaces. In fact there is a duality between *Reproducing Kernel Hilbert Spaces (RKHS)* and GPs with covariance function $k(x, x')$. A *RKHS* is an inner product space with the following properties:

Definition B.2.1. (Reproducing Kernel Hilbert Space). A *Reproducing Kernel Hilbert Space (RKHS)* $\mathcal{H}_k$ of real functions f defined on an index set $\mathcal{X}$ is an inner product space that is complete and separable with respect to the norm defined by the inner product, $< \cdot, \cdot >_{\mathcal{H}_k}$ with the following properties:

1. There exists a function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ where $k(x, \cdot) \in \mathcal{H}\ \forall x \in \mathcal{X}_k$.
2. k is a *represemter of evaluation*: $< k(x, \cdot), f(\cdot) >= f(x)$
3. k has the *reproducing property*: $< k(x, \cdot), k(x', \cdot) >= k(x, x')$ □

The kernel as the *representer of evaluation* is analogous to the delta function in L_2 space. A L_2 space with the dot product defined by $< f, g >_{L_2} = \int f(x)g(x)dx$ and contains non smooth-functions. In L_2, the delta function is the representer of evaluation, i.e. $f(x) = \int f(x')\delta(x - x')dx'$. The equivalent for kernels is stated in definition B.2.1. There are commonly two different ways of constructing a RKHS:

Reproducing Kernel Map Construction The RKHS is a vector space of functions represented by linear combinations of its kernel:

$$\mathcal{H}_k = \left\{ f(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i) \right\} \tag{B.7}$$

where $n \in \mathbb{N}$, $x_i \in \mathcal{X}$ and $\alpha_i \in \mathbb{R}$. The inner product between two functions $f, g \in \mathcal{H}_k$ is give by:

$$\begin{aligned} < f, g > &= \left\langle \sum_{i=1}^{N_1} \alpha_i k(x_i, \cdot), \sum_{j=1}^{N_2} \beta_j k(x_j, \cdot) \right\rangle \\ &= \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \alpha_i \beta_j < k(x_i, \cdot), k(x_j, \cdot) > \\ &= \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \alpha_i \beta_j k(x_i, x_j) \end{aligned} \tag{B.8}$$

From equation B.8, we can easily see that the intuition behind RKHS. The squared norm $\|f\|^2_{\mathcal{H}_k}$ is a generalization of the quadratic form $f^T K f$.

Construction via Mercer's Theorem From Mercer's theorem (theorem B.1.1), the RKHS can alternatively be constructed as a linear combination of its eigenfunctions:

$$\mathcal{H}_k = \left\{ f(x) = \sum_{i=1}^{\infty} f_i \phi_i(x) \right\} \tag{B.9}$$

Given a RKHS $\mathcal{H}_k$ with eigenvectors $\{\phi_i(x)\}_{i=1}^{\infty}$ and its corresponding eigenvalues $\{\lambda_i\}_{i=1}^{\infty}$. Given two functions within this space, $f(x) = \sum_{i=1}^{\infty} f_i \phi_i(x)$ and $g(x) = \sum_{i=1}^{\infty} g_i \phi_i(x)$, the inner product is defined as:

$$< f, g >_{\mathcal{H}_k} = \sum_{i=1}^{\infty} \frac{f_i g_i}{\lambda_i} \tag{B.10}$$

The norm of $f(x)$ in $\mathcal{H}_k$, $\|f\|_{\mathcal{H}_k}$ is then given by $\|f\|^2_{\mathcal{H}_k} =< f, f >_{\mathcal{H}_k} = \sum_{i=1}^{\infty} f_i^2/\lambda_i$. Hence for $\|f\|_{\mathcal{H}_k}$ to be finite, the sequence of coefficients $\{f_i\}$ must decay quickly. This also means that vectors in RKHS are relatively smooth. With the norm of the RKHS defined, the kernel as the representer of evaluation can be easily shown:

$$< f(\cdot), k(x, \cdot) > = \left\langle \sum_i f_i \phi_i(\cdot), \sum_i \lambda_i \phi_i(x) \phi_i(\cdot) \right\rangle \tag{B.11}$$

$$= \sum_i \frac{f_i \lambda_i \phi_i(x)}{\lambda_i} \tag{B.12}$$

$$= f(x) \tag{B.13}$$

One might think that a zero-mean GP with covariance function k is equivalent to a distribution in RKHS with reproducing kernel k, $\mathcal{H}_k$. However, this has shown not to be true (Wah90). In fact, all samples functions from the zero-mean GP are not in H_k with probability 1. Given $f \sim \mathcal{GP}(\mathbf{0}, k)$, the expectation of $\|f\|_{\mathcal{H}_\|}$ is:

$$E[\|f\|^2_{\mathcal{H}_\mathcal{K}}] = E\left[\sum_{i=1}^{K} \frac{f_i^2}{\lambda_i}\right] = K \tag{B.14}$$

From the RKHS norm in equation B.14 will thus be ∞ as $K \to \infty$ and hence a sample function from a GP is not in general in RKHS unless there is only a finite number of non-zero eigenvalues in the spectral decomposition of covariance funtion k.

Intuitively, $\mathcal{H}_k$ contains smooth functions unlike the space L_2. The difference is also further exemplified by looking at the semantic difference between the inner products between the two spaces. A RKHS norm is a measure of the "roughness" of a function whereras the L_2 norm measures the "expected square distance" from the zero function. The relationship between GPs and RKHS is that there exists an isomorphic mapping $f(x) \mapsto k(x, \cdot)$ where $f(x)$ is replaced by $k(x, \cdot)$:

$$< f(x), f(x') >_{\mathcal{GP}} = E[f(x)\ f(x')] =< k(x, \cdot), k(x', \cdot) >= k(x, x') \tag{B.15}$$

Most of the time, the space of GP samples can be regarded as a RKHS with reproducing kernel k. This is due to the fact that most practial applications involves only a finite number of input variables $x \in \mathcal{X}$ conditioned upon or for prediction purposes. One important point is that although sample functions from GPs are not in $\mathcal{H}_k$, the posterior mean after observing data will be in $\mathcal{H}_k$ due to smoothing properties in averaging.

B.2.1 Classification

The problem of classification is similar to that of regression. However, the main difference is in its output space. Regression has a continuous output space whereras classification has a discrete output space (or labels).

GP classification models has similarities with linear models in classification. A large class of linear models can be obtained from *generalised linaer models (GLMs)* (MN89). GLMs provides the framework to construct a class of learning algorithms (regression or classification) with relatively low learning complexity while being able to accomodate different distributions of data.. GLMs makes the following assumptions:

1. GLMs builds on the exponential family of probability distributions to model the the value of output variables $y \in \mathcal{Y}$ as a function of its inputs $x \in \mathcal{X}$. An exponential family of probability distributions has the canonical representation:

$$P(y; \eta) = b(y) \exp(\eta^T T(y) - a(\eta)) \tag{B.16}$$

Where η is the *natural parameter*; $T(y)$ its *sufficient statistic* and $a(\eta)$ its *log partition function* which ensures that $\int P(y; \eta) dy = 1$.

2. The goal given $x \in \mathcal{X}$ is to compute the sufficient statistics $T(y)$. By the definition of sufficient statistics, given $T(y)$, the conditional distribution of y can be fully represented independently of its underlying parameters of the distribution. With $T(y)$, the expectation $E[y|x]$ can be chosen as the learned hypothesis.

3. The natural parameter η is linearly related to its inputs, i.e. $\eta = \theta^T x$ where θ is a vector of linear weighting factors.

Consider the simple case of binary classification where the labels are $\mathcal{Y} = \{1, 0\}$. The probability distribution over labels can thus be represented as a Bernoulli distribution with parameter ϕ:

$$\begin{aligned} P(y;\phi) &= \phi^y (1-\phi)^{1-y} \\ &= \exp\left(\left(\log\frac{\phi}{1-\phi}\right) y + \log(1-\phi)\right) \end{aligned} \tag{B.17}$$

From equation B.17, we can see that $\eta = \log(\phi/(1-\phi))$ and inverting this we get $\phi = 1/(1+e^{-\eta})$ which is the sigmoid function, $\sigma(\eta)$ (see figure B.1. The hypothesis for the binary classification is then:

$$\begin{aligned} E[y|x,\theta] &= \phi \\ &= 1/(1+e^{-\theta^T x}) && \text{(B.18)} \\ &= \sigma(\theta^T x) && \text{(B.19)} \end{aligned}$$

In standard GLM methods, learning is performing by maximizing the likelihood of the parameters θ using gradient based optimization. GP classification methods puts a GP prior in place of the linear relation $\theta^T x$ in equation B.19. Thus the prior on the classificator with a GP prior over $f(x)$ is given by:

$$P(y = 1|x) = \sigma(f(x)) \tag{B.20}$$

Inference in GP classification is primarily in two stages:

1. The distribution over latent variables $f(x)$ has to be computed from training data consisting of a set of training inputs x and outputs (class labels) y:

$$P(f_*|x, y, x_*) = \int P(f_*|x, x_*, f)P(f|x, y)df \tag{B.21}$$

   where x_* are the testing inputs and f_* the testing outputs given by the latent function f.

2. The latent function f_* is then "squashed" through the sigmoid function like in GLM classification methods. The main difference is where the latent functions f_* are integrated out to give the probabilistic prediction on the classification:

$$P(y_* = 1|x, y, x_*) = \int \sigma(f_*)P(f_*|x, y, x_*)df_* \tag{B.22}$$

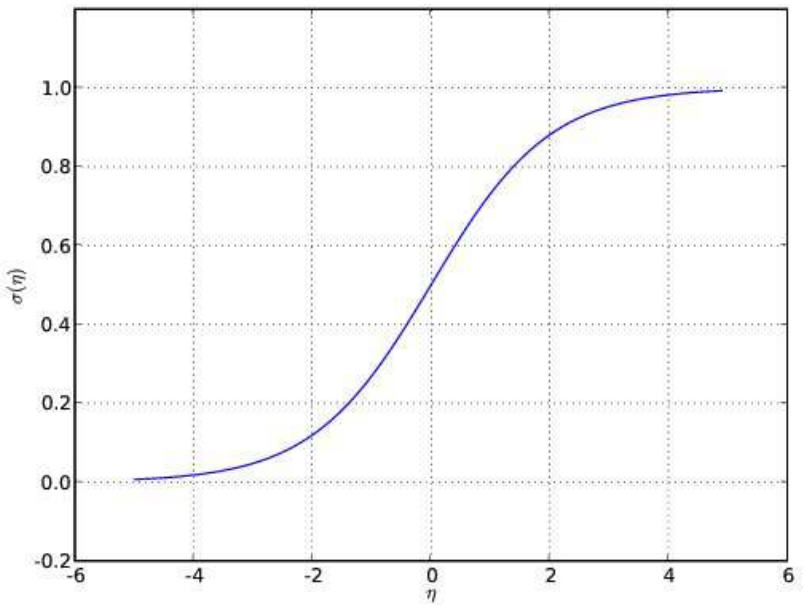


Figure B.1: Plot of sigmoid function

where y_* is the random variable indicating the predicted class given its corresponding input x_* during classification.

In regression (sect. 4.3.3), the predictions are relatively easy to compute as Gaussians are analytic. However, in classification, the equations B.21 and B.22 are analyticaly intractable in general. It is therefore common to seek analytic approximations of these intergrals or compute it approximatively via Monte Carlo sampling.

The simplest approximation is to approximate $P(f|x, y)$ with a Gaussian via Laplace approximation (WB98a). Laplace approximation uses the mode of the distribution as its mean and the Hessian as the inverse of the covariance matrix of a Gaussian distribution. More sophisticated methods such as the expectation propagation (see sect 3.6.2) can be used. Other approximation methods such as MCMC methods can be found in (GM97) (JH99) (GM00) (See00) (Nea98).

The case of binary classification can be extended to handle multi-class classification. Each training data can be associated with every class. In multi-class classification with C labels, the liklihood of all data for each label $c \in C$ are given as:

$$P(Y|f^c) = \prod_{i=1}^{N} P(y_i|f^c) \tag{B.23}$$

where f^c is the latent GP function for class c. The likelihood that a data belongs to a

certain class c amonst all classes in C can be computed using the softmax function:

$$P(y_i = c|f_i^C) = \frac{exp(f_i^C)}{\sum_{c'} exp(f_i^{c'})} \tag{B.24}$$

The softmax function (eqn. B.24) comes from the GLM for mult-class classification. This formulation induces a vector latent function values $f = (f_1^1, \ldots, f_N^1, f_1^2, \ldots, f_N^2, \ldots f_1^C, \ldots, f_N^C)$ given N training data and C classification labels. f is GP distributed with a covariance function K^c for each class. It is common to assume a block-diagonal structure in the covariance matrix where each GP for the classes are mutually independent.

Bibliography

[Adm04] National Highway Traffic Safety Administration. Vehicle-based countermeasures for signal and stop sign violation. Technical Report DOT HS 809 716, NHTSA, U.S. DOT, 2004.

[AMGC02] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. 2002.

[BB08] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. NIPS Foundation (http://books.nips.cc), 2008.

[BBCT05] M. Bennewitz, Wolfram Burgard, G. Cielniak, and S. Thrun. Learning motion patterns of people for compliant robot motion. *International Journal of Robotics Research*, 24(1), 2005.

[BBSP06] Dietmar Bauer, Norbert Brandle, Stefan Seer, and Roman Pflugfelder. Finding highly frequented paths in video sequences. *International Conference on Pattern Recognition*, 1:387–391, 2006.

[Bes74] Julian Besag. Spatial interaction and the statistical analysis of lattice systems. 1974.

[BGV92] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, New York, NY, USA, 1992. ACM Press.

[Bis06] C. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[BK00] Matthew Brand and Vera Kettnaker. Discovery and segmentation of activities in video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):844–851, 2000.

[BK01] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. In *In Energy*

Minimization Methods in Computer Vision and Pattern Recognition, pages 359–374, 2001.

[BOP97] M. Brand, N. Oliver, and A. Pentland. Coupled hidden markov models for complex action recognition. pages 994–999, 1997.

[Bot04] Léon Bottou. Stochastic learning. In Olivier Bousquet and Ulrike von Luxburg, editors, *Advanced Lectures on Machine Learning*, Lecture Notes in Artificial Intelligence, LNAI 3176, pages 146–168. Springer Verlag, Berlin, 2004.

[BPPF06] R. Benenson, S. Petti, M. Parent, and Th. Fraichard. Integrating perception and planning for autonomous navigation of urban vehicles. In *IEEE IROS*, 2006.

[Bur07] Julien Burlet. *Suivi Multi-Objets Adaptatif : Application à la Classification de Comportements de Mobiles*. PhD thesis, Institut National Polytechnique de Grenoble, France, December 2007.

[Buz04] Dan Buzan. Extraction and clusterincg of motion trajectories in video. In *Journal of Artificial Intelligence Research*, pages 521–524, 2004.

[CB01] A. Corduneanu and C. M. Bishop. Variational bayesian model selection for mixture distributions. In *Proceedings Eighth International Conference on Artificial Intelligence and Statistics*, pages 27–34, 2001.

[CCP07] S. Calderara, R. Cucchiara, and A. Prati. Detection of abnormal behaviors using a mixture of von mises distributions. *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, pages 141–146, Sept. 2007.

[Cox61] Richard Threlkeld Cox. *The Algebra of Probable Inference*. The Johns Hopkins Press, Baltimore, 1961.

[Cre93] N.A.C. Cressie. *Statistics for Spatial Data*. Wiley, New York, 1993.

[CT00] N. Cristiani and Shawe J. Taylor. *An introduction to support vector machines*. Cambridge University Press, Cambridge, UK, 2000.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[EDD+95] Matthias Eck, Tony DeRose, Tom Duchamp, Hugues Hoppe, Michael Lounsbery, and Werner Stuetzle. Multiresolution analysis of arbitrary meshes. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 173–182, New York, NY, USA, 1995. ACM.

[FA03] T. Fraichard and H. Asama. Inevitable collision states. a step towards safer robots? In *IEEE IROS*, volume 1, pages 388–393, October 2003.

[FBCM04] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nystrom method. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(2):214–225, Feb. 2004.

[FJ07] K. Fuerstenberg and J.Chen. New european approach for intersection safety - results of the ec project intersafe. In *Proc. International Forum on Advanced Microsystems for Automotive Application*, 2007.

[Fri95] Bernd Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, 1995.

[FT02] A. Faul and M. Tipping. Analysis of sparse bayesian learning, 2002.

[FTSL08] Chiara Fulgenzi, Christopher Tay, Anne Spalanzani, and Christian Laugier. Probabilistic navigation in dynamic environment using Rapidly-exploring Random Trees and Gaussian Processes. In *IEEE/RSJ 2008 International Conference on Intelligent RObots and Systems*, Nice France, 2008.

[Ful09] C. Fulgenzi. *Autonomous navigation in dynamic uncertain environment using probabilistic models of perception and collision risk prediction.* PhD thesis, Institut National Polytechnique de Grenoble, 2009.

[GG84] S. Geman and D. Geman. Stochastic relaxation, gibbs distribution and the bayesian restoration of images. *IEEE Transaction in Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[Gib97] Mark Gibbs. *Bayesian Gaussian Processes for Regression and Classification.* PhD thesis, Department of Physics, University of Cambridge, 1997.

[GJH01] Aphrodite Galata, Neil Johnson, and David Hogg. Learning variable length markov models of behaviour, 2001.

[GJP95] Federico Girosi, Michael Jones, and Tomaso Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–269, 1995.

[GM97] Mark Gibbs and David J. C. Mackay. Efficient implementation of gaussian processes. Technical report, 1997.

[GM00] Mark N. Gibbs and David J. C. Mackay. Variational gaussian process classifiers. *IEEE Transactions on Neural Networks*, 11:1458–1464, 2000.

[Gre95] P. Green. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82:711–732, 1995.

[GY04] J. Geng and J. Yang. Autobib: Automatic extraction of bibliographic information on the web. In *IDEAS*, pages 193–204, 2004.

[Has70] W.K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[HBLC08] A. Hervieu, P. Bouthemy, and J.-P. Le Cadre. A statistical video content recognition method using invariant features on object trajectories. *IEEE Trans. on Circuits and Systems for Video Technology (Special Issue on "Event Analysis in Videos" (accepted for publication)*, 2008.

[HBN00] Somboon Hongeng, Francois Brémond, and Ramakant Nevatia. Representation and optimal recognition of human activities. In *In Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2000*, pages 818–825, 2000.

[Hor93] K. Hornik. Some new results on neural network approximation. *Neural Netw.*, 6(9):1069–1072, 1993.

[HS52] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, Dec 1952.

[HXF+06] Weiming Hu, Xuejuan Xiao, Zhouyu Fu, Dan Xie, Tieniu Tan, and Steve Maybank. A system for learning statistical motion patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(9):1450–1464, 2006.

[HXT04] Weiming Hu, Dan Xie, and Tieniu Tan. A hierarchical self-organizing approach for learning the patterns of motion trajectories. *Neural Networks, IEEE Transactions on*, 15(1):135–144, Jan. 2004.

[IB98] M. Isard and A. Blake. Condensation – conditional density propagation for visual tracking. *International Journal Of Computer Vision*, 29:5–28, 1998.

[IG07] T. Izo and W.E.L. Grimson. Unsupervised modeling of object tracks for fast anomaly detection. *Image Processing, 2007. ICIP 2007. IEEE International Conference on*, 4:IV –529–IV –532, 16 2007-Oct. 19 2007.

[Jay03] E. T. Jaynes. *Probability Theory : The Logic of Science*. G. Larry Bretthorst, 2003.

[JH96] Neil Johnson and David Hogg. Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14:583–592, 1996.

[JH99] Tommi S. Jaakkola and David Haussler. Probabilistic kernel regression models. In *In Proceedings of the 1999 Conference on AI and Statistics*. Morgan Kaufmann, 1999.

[JJS04] I.N. Junejo, O. Javed, and M. Shah. Multi feature path modeling for video surveillance. *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, 2:716–719 Vol.2, Aug. 2004.

[JLO90] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quaterly*, 4:269–282, 1990.

[JR85] B.H. Juang and L.R. Rabiner. *AT&T Technical Journal*, 64(2):391–408, Feb 1985.

[JU97] Simon J. Julier and Jeffrey K. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *In Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, pages 182–193, 1997.

[Kal60] R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME - Journal of Basic Engineering*, (82 (Series D)):35–45, 1960.

[KmG01] Esther B. Koller-meier and Luc Van Gool. Gool, modeling and recognition of human actions using a stochastic approached. In *2 nd European Workshop on Advanced Video-Based Surveillance Systems*, pages 17–28, 2001.

[LEE76] David N. LEE. *A theory of visual control of braking based on information about time-to-collision.* Perception, Edinburgh, 1976.

[LFH+03] Lin Liao, D. Fox, J. Hightower, H. Kautz, and D. Schulz. Voronoi tracking: location estimation using sparse and noisy sensor data. *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, 1:723–728 vol.1, Oct. 2003.

[LK99] S.M. LaValle and Jr. Kuffner, J.J. Randomized kinodynamic planning. volume 1, pages 473–479 vol.1, 1999.

[LPFK07] Lin Liao, Donald J. Patterson, Dieter Fox, and Henry Kautz. Learning and inferring transportation routines. *Artif. Intell.*, 171(5-6):311–331, 2007.

[LPRMt02] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillo t. Least squares conformal maps for automatic texture atlas generation. In ACM, editor, *ACM SIGGRAPH conference proceedings*, Jul 2002.

[MA00] R. McEliece and S. M. Aji. The generalized distributive law. *EEE Trans. Inform. Theory*, 46:325–343, 2000.

[Mac98] D. J. C. MacKay. Introduction to Gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning*, NATO ASI Series, pages 133–166. Kluwer Academic Press, 1998.

[ME02] Dimitrios Makris and Tim Ellis. Spatial and probabilistic modelling of pedestrian behaviour. In *British Machine Vision Conference 2002, vol.2*, pages 557–566, 2002.

[Min01a] Thomas Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the 17th Annual Conference on Uncertainty in Artificial Intelligence (UAI-01)*, pages 362–36, San Francisco, CA, 2001. Morgan Kaufmann.

[Min01b] Thomas Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, MIT Media Lab, 2001.

[MN89] P. McCullagh and J. A. Nelder, editors. *Generalized Linear Models*. Chapman & Hall, 2nd edition, 1989.

[Nea96] R.M. Neal. *Bayesian Learning for Neural Networks*. NY: Springer Verlag, 1996.

[Nea98] R. M. Neal. Regression and classification using gaussian process priors, 1998.

[Nea99] R. Neal. Suppressing random walks in Markov chain Monte Carlo using ordered overrelaxation. pages 205–228. MIT Press, 1999.

[NPVB05] Nam T. Nguyen, Dinh Q. Phung, Svetha Venkatesh, and Hung Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden markov models. In *CVPR 05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR05) - Volume 2*, pages 955–960, Washington, DC, USA, 2005. IEEE Computer Society.

[OHG02] N. Oliver, E. Horvitz, and A. Garg. Layered representations for human activity recognition. pages 3–8, 2002.

[Pap91] A. Papoulis. *Probability, Random Variables, and Stochastic Processes*. 3rd edition, 1991.

[Pea88] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[PF05] S. Petti and Thierry Fraichard. Safe motion planning in dynamic environments. In *IEEE IROS*, 2005.

[PF07] C. Piciarelli and G.L. Foresti. Anomalous trajectory detection using support vector machines. *Advanced Video and Signal Based Surveillance, 2007. AVSS 2007. IEEE Conference on*, pages 153–158, Sept. 2007.

[PHK+05] C. Pradalier, J. Hermosillo, C. Koike, C. Braillon, P. Bessière, and C. Laugier. The cycab: a car-like robot navigating autonomously and safely among pedestrians. *Robotics and Autonomous Systems*, 50(1):51–68, 2005.

[PJL+00] J. Pierowicz, E. Jocoy, M. Lloyd, A. Bittner, and B. Pirson. Intersection collision avoidance using its countermeasures. Technical Report DOT HS 809 171, NHTSA, U.S. DOT, 2000.

[PJP93] Ulrich Pinkall, Strasse Des Juni, and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, 2:15–36, 1993.

[PS04] Christopher J. Paciorek and Mark J. Schervish. Nonstationary covariance functions for gaussian process regression. In *In Proc. of the Conf. on Neural Information Processing Systems (NIPS*. MIT Press, 2004.

[Rab89] Lawrence Rabiner. A tutorial on hmm and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, February 1989.

[Rip81] B. Ripley. *Spatial Statistics*. Wiley, New York, 1981.

[RRRS] A. Reynolds, G. Richards, and V. Rayward-Smith. The application of k-medoids and pam to the clustering of rules. *Lecture Notes In Computer Science*, 3177:173–178.

[RW05] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[RW06] Carl Edward Rasmussen and Christopher K.I. Williams. *Gaussian processes for machine learning*. The MIT Press, 2006.

[See00] M. Seeger. Bayesian model selection for support vector machines, gaussian processes and other kernel classifiers. *Neural Information Processing Systems 12*, pages 603–609, 2000.

[SEG00] Chris Stauffer, W. Eric, and L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:747–757, 2000.

[SF08] Rowland Sillito and Robert Fisher. Semi-supervised learning for anomalous trajectory detection. In *Proc. British Machine Vision Conference (BMVC)*, pages 1035–1044, Sep 2008.

[SHT+07] N. Suzuki, K. Hirasawa, K. Tanaka, Y. Kobayashi, Y. Sato, and Y. Fujino. Learning motion patterns and anomaly detection by human trajectory analysis. *Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on*, pages 498–503, Oct. 2007.

[SS01] Bernhard Schlkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. The MIT Press, December 2001.

[TDV07] Nils Appenrodt Trung-Dung Vu, Olivier Aycard. Online localization and mapping with moving object tracking in dynamic outdoor environments. In *IEEE Intelligent Vehicles Symposium*, Istanbul, 2007.

[Tik63] A. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet. Math. Dokl.*, 5:1035–1038, 1963.

[Tip01] Michael E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

[TJBB06] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei. Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[TMY+08] Christopher Tay, Kamel Mekhnacha, Manuel Yguel, Christophe Coue, Cédric Pradalier, Christian Laugier, Thierry Fraichard, and Pierre Bessiere. The Bayesian Occupation Filter. In P. Bessière, C. Laugier, and R. Siegwart, editors, *Probabilistic Reasoning and Decision Making in Sensory-Motor Systems Springer Tracts in Advanced Robotics Series*, volume 46 of *Springer Tracts in Advanced Robotics Series*. Springer, 2008.

[VG07] Dizan Alejandro Vasquez Govea. *Incremental Learning for Motion Prediction of Pedestrians and Vehicles*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble (Fr), February 2007.

[VGF04] Dizan Alejandro Vasquez Govea and Thierry Fraichard. Motion prediction for moving objects: a statistical approach. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 3931–3936, New Orleans, LA (US), April 2004.

[VGF07] Dizan Alejandro Vasquez Govea and Thierry Fraichard. Intentional motion on-line learning and prediction. *Machine Vision and Applications*, 2007.

[VW99] F. Vivarelli and C. Williams. Discovering hidden features with gaussian process regression. 1999.

[WA90] D. Wang and M.A. Arbib. Complex temporal sequence learning based on short-term memory. *Proceedings of the IEEE*, 78(9):1536–1543, Sep 1990.

[Wah90] G. Wahba. *Spline Models for Observational Data*. SIAM, Philadelphia, 1990.

[WB98a] Christopher K. I. Williams and David Barber. Bayesian classification with gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1342–1351, 1998.

[WB98b] A.D. Wilson and A.F. Bobick. Recognition and interpretation of parametric gesture. pages 329–336, Jan 1998.

[Whi84] P. Whittle. *Prediction and Regulation by Linear Least-square Methods*, volume 2. Basil Blackwell Publisher, Oxford, 1984.

[Wie64] Norbert Wiener. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. The MIT Press, 1964.

[WM01] Eric A. Wan and Rudolph Van Der Merwe. Chapter 7: The unscented kalman filter. In *Kalman Filtering and Neural Networks*, pages 221–280. Wiley, 2001.

[WMNG08] X.G. Wang, K.T. Ma, G.W. Ng, and W.E.L. Grimson. Trajectory analysis and semantic region modeling using a nonparametric bayesian model. pages 1–8, 2008.

[WN08] S. Worrall and E. Nebot. A probabilistic method for detecting impending vehicle interactions. *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1787–1791, May 2008.