

DÉVELOPPEMENT D'UN TRAIN VIRTUEL DE CYCABS AVEC SCILAB/SCICOS/SYNDEX

QUENTIN QUADRAT

Stage de fin d'étude,
1er Janvier au 1er Juillet 2007

—

Supervisé par :

- Yves Sorel, Directeur de Recherche à l'INRIA,
- Patrice Bodu, Ingénieur expert à l'INRIA.



Page 1 sur 37

Précéd. Suivant



1. SOMMAIRE

- Présentation de la voiture électrique CyCab.
- Méthodologie AAA.
- Logiciels Scilab, Scicos, SynDEEx.
- Application de conduite manuelle.
- Application de conduite automatique.
- Conclusion.

2. INRIA, AOSTE, IMARA

2.1. INRIA.

- Institut National de Recherche en Informatique et en Automatique (créé en 1967)
- Il est placé sous la double tutelle du ministre chargé de la Recherche et de l'Industrie.

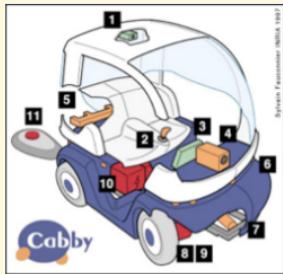
2.2. L'équipe AOSTE.

- Analysis Optimisation of Systems with real-Time and Embedded constraints,
- Méthodologie AAA (Adéquation Algorithme Architecture),
- *SynDEX* : logiciel de CAO pour l'aide à l'implantation de systèmes distribués temps réel embarqués.

2.3. L'équipe IMARA.

- Domaine de la "Route Automatisée".
- Contrôle-commande du véhicule, outils de programmation temps réel distribués, traitement du signal.

3. CYCAB



3.0.1. Architecture.

- 1 PC embarqué, Linux temps réel (RTAI).
- 2 cartes à cœur PowerPC,
- Communication par protocole CAN.

3.0.2. Capteurs, Actionneurs.

- 4 roues motrices, 2 vérins de direction.
- 6 contrôleurs / amplificateurs moteurs.
- 6 encodeurs incrémentaux / absous.
- 1 joystick de direction, 1 caméra FireWire.

3.0.3. *Type de conduite.*

- Logiciel applicatif de conduite *manuelle* (joystick).
- Logiciel applicatif de conduite *automatique* (bas coût : caméra).

3.0.4. *But du stage.*

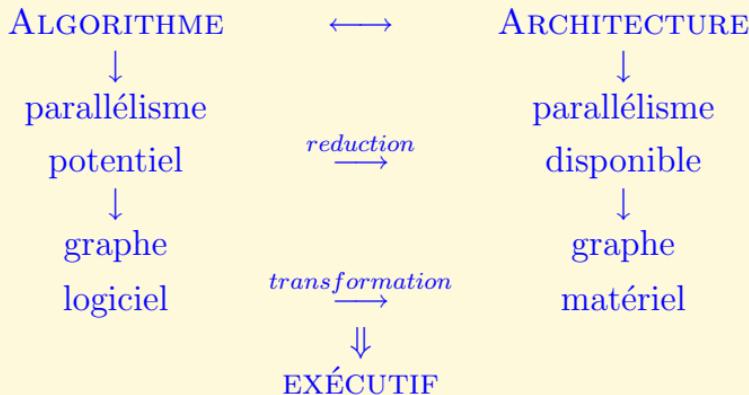
- *Remplacer* le joystick par une caméra pour faire du *suivi automatique* de CyCab.
- Mettre en place un régulateur pour *maintenir une distance fixe* entre les 2 CyCab.
- *Simuler* l'algorithme de suivi avec Scicos et de distribuer l'algorithme avec SynDEX.

4. PROBLÉMATIQUE DES SYSTÈMES TEMPS RÉEL EMBARQUÉS

- Une opération doit pouvoir être exécutée *avant* un temps prévu à l'avance.
- Il faut une *puissance* de calcul embarqué *suffisante* souvent à faible coût.
- Cette puissance de calcul est obtenue avec *plusieurs processeurs* embarqués.
- Il faut donc répartir les tâches sur les différents processeurs embarqués et les faire communiquer (logiciel SynDEx permet de le faire automatiquement).

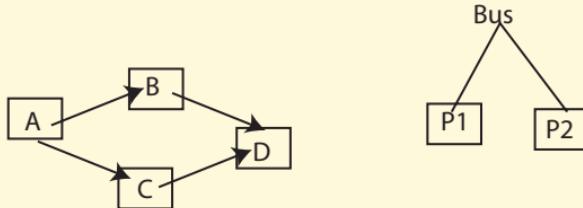
5. MÉTHODOLOGIE AAA

ADÉQUATION



- Algorithme : graphe d'ordonnancement des tâches.
- Architecture : graphe des ressources et des moyens de communication.
- Exécutifs : affectation temporelle des tâches aux ressources.
- Adéquation : générer l'exécutif.

6. EXEMPLE DE DISTRIBUTION



- 4 opérations : A, B, C, D.
- 2 processeurs : P₁, P₂.
- 1 bus de communication.
- Durée des opérations : *1 unité* de temps.
- Durée d'une communication : *1 unité* de temps.

D s'exécute *après* B et C qui s'exécutent *après* A.

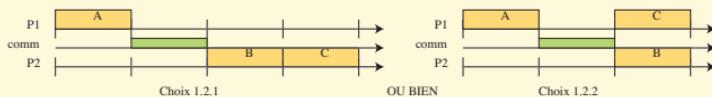
Distribution de l'opération A :



Distribution de l'opération B :



Distribution de l'opération C :



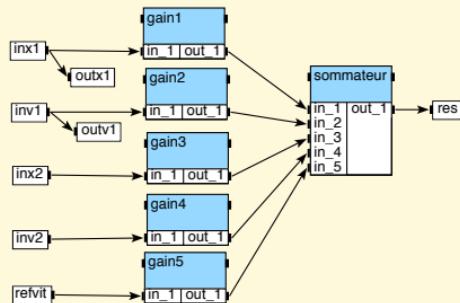
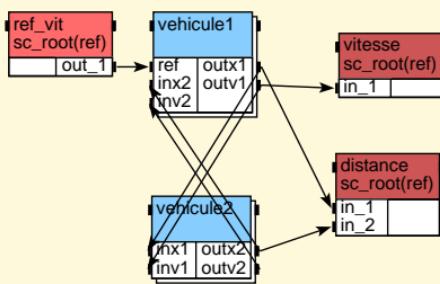
Distribution de l'opération D : etc ...

7. SYNDEX

- Met en oeuvre la méthodologie AAA.
- Utilise des entrées graphiques pour représenter l'algorithme et l'architecture.
- Génère les exécutifs distribués temps réel, sans interblocage et statiques écrits en macro-code m4.
- Visualise la prédition des performances temps réel pour le dimensionnement de l'architecture.
- Utilise des *heuristiques* pour trouver une bonne solution de distribution (glouton, génétique, ...) car *problème NP-complet*.
- Utilise des *bibliothèques* de noyaux d'exécutifs génériques pour : Linux, RTAI, MPC 555, bus CAN, TCP/IP, ...

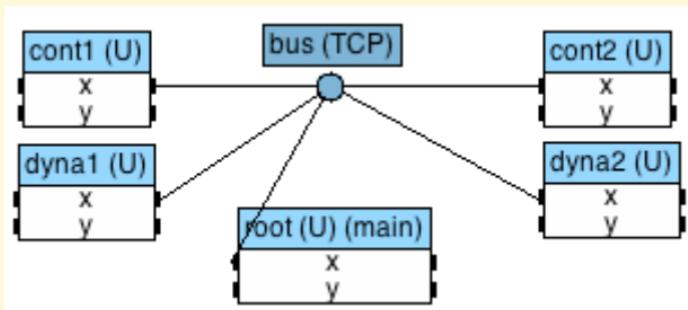
8. EXEMPLE DE GRAPHE D'ALGORITHME SYNDEX

- Un graphe hiérarchique définit l'algorithme.
- Les *noeuds* sont les tâches (*opérations*) à réaliser et les *arcs* définissent les *relations de précédence* .



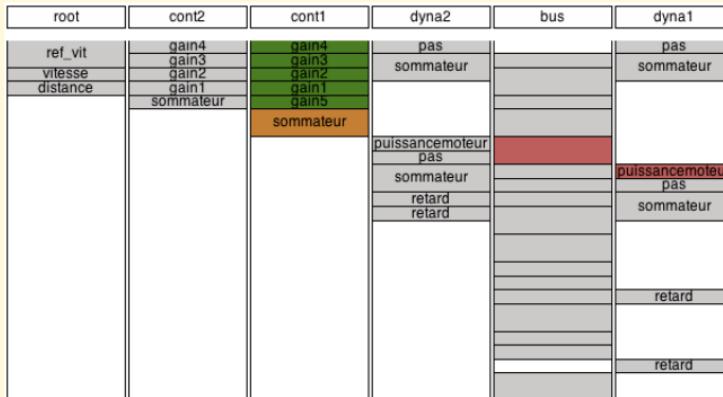
9. EXEMPLE DE GRAPHE D'ARCHITECTURE SYNDEX

- Un graphe non hiérarchique définit l'architecture.
- Les *noeuds* sont les *ressources* (soit des opérateurs soit des médias de communication).
- Les *arcs* les *communications* entre les ressources.



10. DIAGRAMME TEMPOREL

La *synchronisation* des exécutions des tâches peut être visualisée sous forme d'un *diagramme temporel* :



- Colonnes : ressources disponibles.
- Contenu des colonnes : agenda des tâches à exécuter.
- Epaisseur des opérations : durée des opérations.

11. EXÉCUTIF

```

include(syndex.m4x)dnl
processor_(195,root) ABCD,
SynDEX v5.1c (c)INRIA 2000,
Thu Mar 16 14:07:12 2000
}
semaphores_{
  B_d_empty_can, B_d_full_can,
  A_c_empty_can, A_c_full_can,
}
alloc_(int,A_b)
alloc_(int,A_c)
alloc_(int,B_d)

main_
spawn_thread_(can)
sensor()
loop_
  Suc0_(A_c_empty_can)
  sensor(A_b, A_c)
  Pre1_(A_c_full_can)
  Suc0_(B_d_empty_can)
  compute(A_b, B_d)
  Pre1_(B_d_full_can)
endloop_
sensor()
wait_endthread_(can)
endmain_
endprocessor_

```



```

include(syndex.m4x)dml
processor_(195,0) ABCD,
SynDEX v5.1c (c)INRIA 2000,
Thu Mar 16 14:07:11 2000
}
semaphores_{
  B_d_empty_can, B_d_full_can,
  A_c_empty_can, A_c_full_can,
}
alloc_(int,B_d)
alloc_(int,A_c)
alloc_(int,C_d)

main_
spawn_thread_(can)
pre1_(A_c_empty_can)
actuator()
pre1_(B_d_empty_can)
loop_
  Suc1_(A_c_full_can)
  send_(A_c, 555,root,p)
  Pre0_(A_c_empty_can)
  Suc1_(B_d_full_can)
  send_(B_d, 555,root,p)
  Pre0_(B_d_empty_can)
endloop_
actuator()
recv_(A_c, 555,root,p)
pre0_(A_c_full_can)
Suc1_(B_d_empty_can)
recv_(B_d, 555,root,p)
pre0_(B_d_full_can)
Suc0_(B_d_full_can)
actuator(B_d, C_d)
Pre1_(B_d_empty_can)
endloop_
actuator()
wait_endthread_(can)
endmain_
endprocessor_

```

Les opérations et les synchronisations sont spécifiées en macro-code m4.

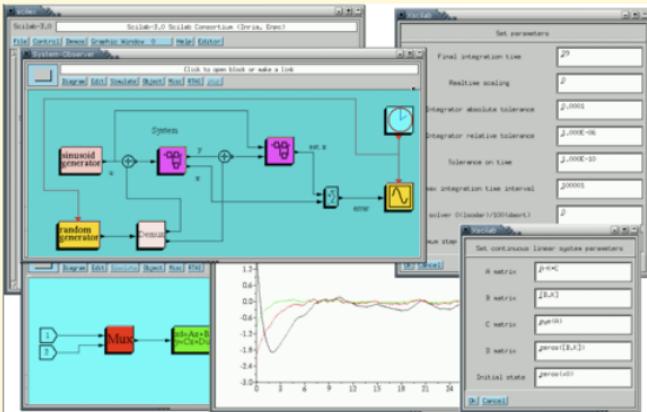
12. EXÉCUTION

- Compiler les executifs.
- Flasher les différents programmes sur l'architecture.
- Lancer l'exécutif distribué.



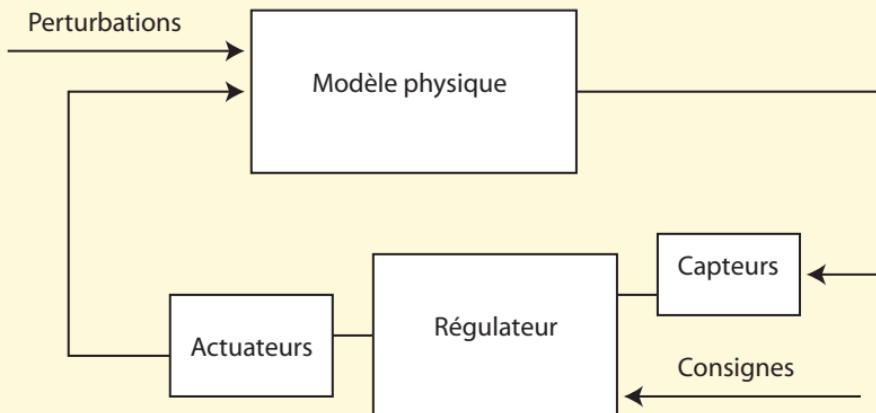
13. SCILAB/Scicos

- Le logiciel standard utilisé pour faire du calcul numérique en ligne et hors ligne et générer le code embarqué est : Maltlab/Simulink.
- L'analogue libre développé à l'INRIA est Scilab/Scicos.

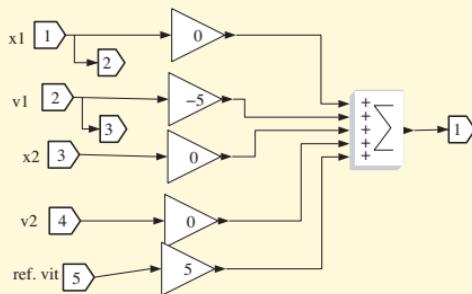


14. SCILAB/Scicos/SYNDEx

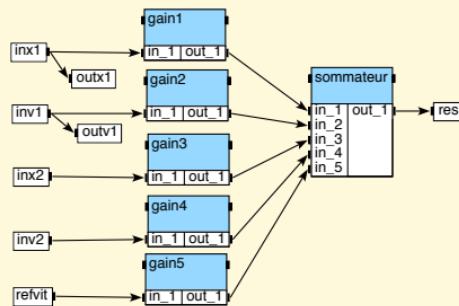
- *Simulation/validation* du régulateur, capteurs, actionneurs et du modèle physique dans Scilab/Scicos.



- Traduction de la partie à embarquer en algorithme SynDEX (*régulateur, capteurs, actionneurs*).



Régulateur sous Scicos.



Régulateur sous SynDEX.

15. CONTEXTE DU STAGE

15.1. Ce qui existait avant le stage.

- Applicatif SynDEx de conduite *manuelle* de CyCab.
- *Ebauche* d'applicatif SynDEx de conduite *automatique* de CyCab.

15.2. Problèmes à résoudre.

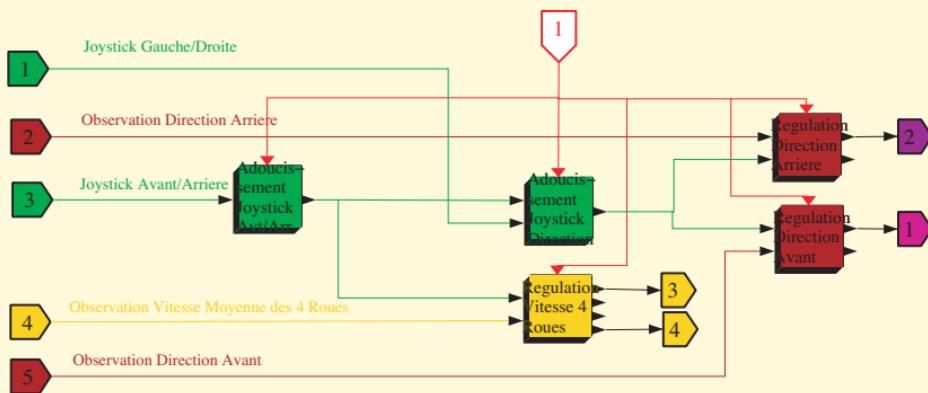
- *Pas* de possibilité *de simulation* dans SynDEx.
- Les tâches des algorithmes SynDEx pour la conduite manuelle du CyCab *écrites en assembleur MPC555*, donc difficile à comprendre.
- Les *plages* des différents signaux n'étaient *pas spécifiés*, donc difficile de régler les gains des régulateurs.

16. TRAVAIL EFFECTUÉ

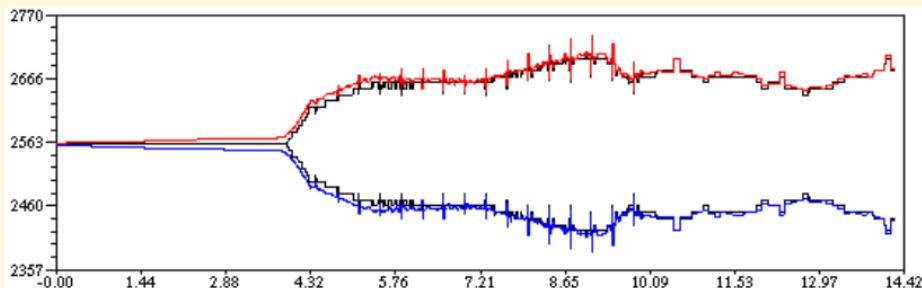
- *Traduction* de l'application de conduite manuelle SynDEx vers une application Scicos.
- *Traitement d'images* pour la détection de la distance entre 2 CyCabs.
- Réalisation du *régulateur de la distance* entre 2 CyCabs.
- *Mise à jour du matériel/logiciel embarqué* (achat de cartes et de PC, installation de Linux et RTAI).
- *NOMBREUSES pannes électroniques* des noeuds MPC 555 à détecter et à faire réparer.

17. TRADUCTION DE L'APPLICATION DE CONDUITE MANUELLE SYNDEX VERS UNE APPLICATION SCICOS

- Programme permettant d'*observer les signaux* réels et de les sauvegarder.
- *Traduction* du code asm MPC555 en blocs Scicos.
- *Identification* du modèle physique du CyCab.



17.1. Problème rencontré. Les signaux simulés ne *correspondaient* pas aux signaux réels à cause des problèmes d'arrondis (MPC555 32 bit, Scicos: flottant).

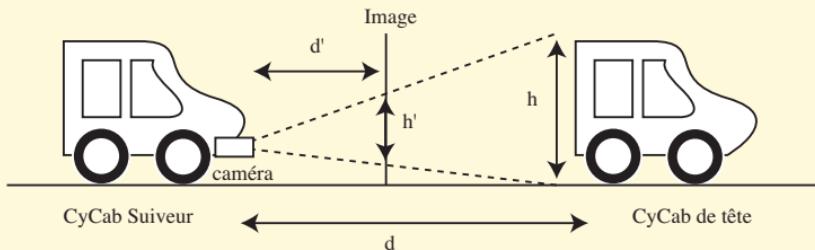


Qualité de la simulation après résolution des *problèmes d'arrondis* (noir : signaux observés, rouge et bleu : signaux simulés).

18. RÉGULATION DE LA DISTANCE

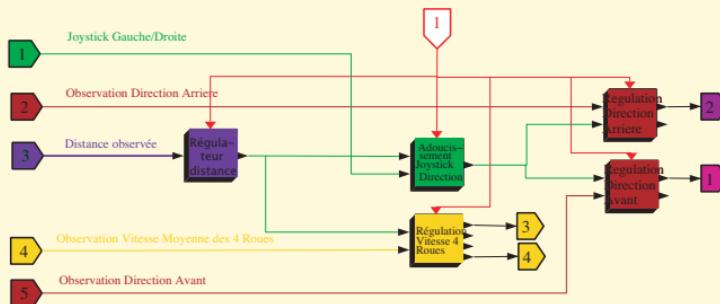
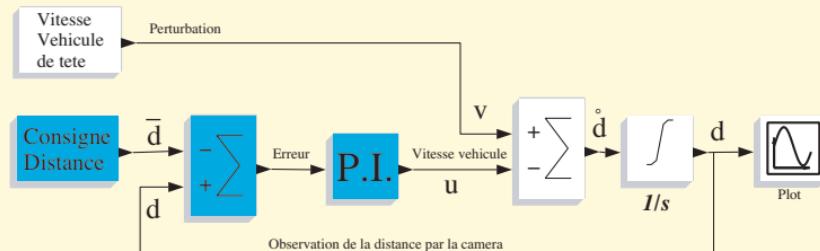
18.1. Principe du platooning.

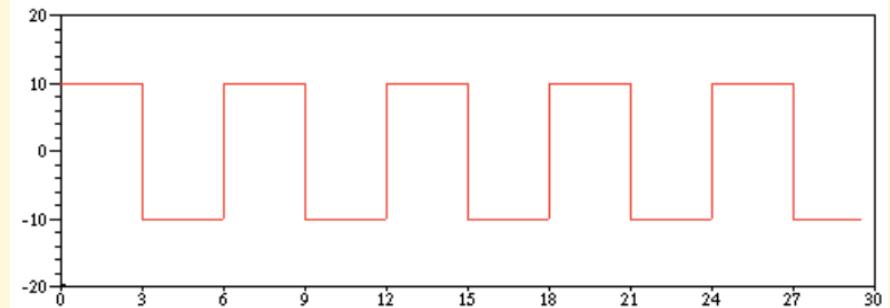
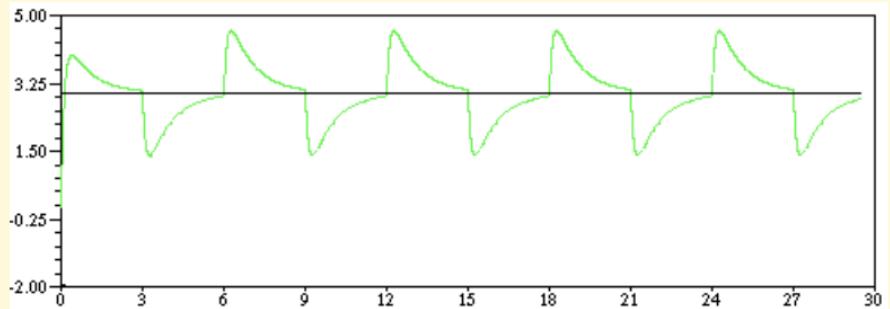
- Un Cycab *Suiveur* suit un autre *Leader* avec une consigne de distance \bar{d} à respecter.



- La *vitesse v* du Leader *varie arbitrairement*. Elle est inconnue. C'est une *perturbation à rejeter*.

18.2. Bloc diagramme du système régulé avec Scicos. Les blocs bleus du régulateur sont à embarquer grâce à SynDEX.





En rouge : la vitesse du Leader. En vert : la distance entre les 2 CyCab.

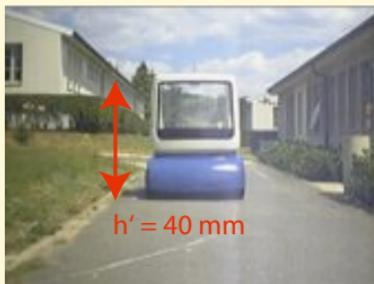
19. ESTIMATION DE LA DISTANCE ENTRE 2 CYCAB

19.1. Comment obtenir la distance d ?

- Pas simple de calculer directement d (projection, ...)
- Idée utiliser la *hauteur h* du CyCab *comme distance*.



d est petit, h est grand.



d est grand, h est petit.

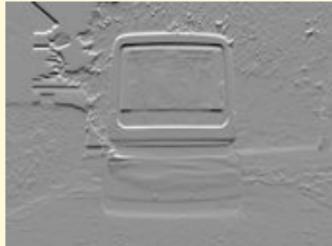
19.2. Comment obtenir la hauteur h ?

Par traitement de l'image : – contours (filtre de Sobel), – lignes horizontales (histogramme), – prédiction des positions des raies (Kalman).

20. DÉTECTION DE CONTOURS



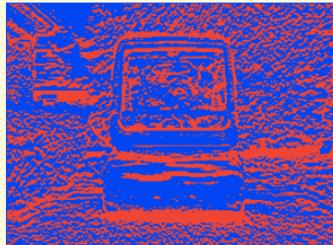
20.1. Filtre de Sobel. Visualisation du *gradient vertical* des couleurs pour faire apparaître les *contours horizontaux*.



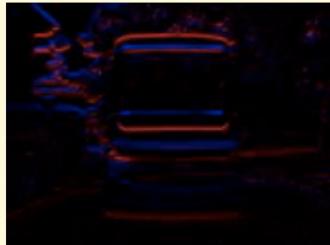
Gradient signé



Valeur absolue

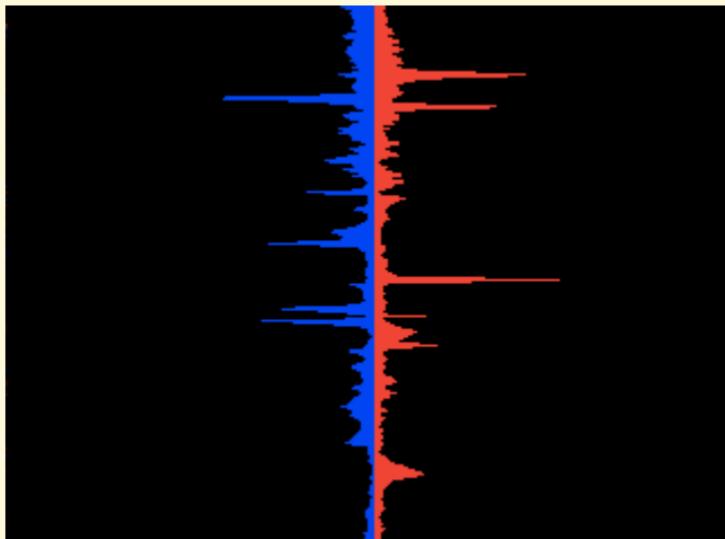


Signe du gradient

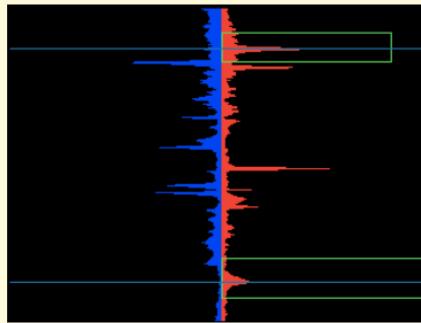
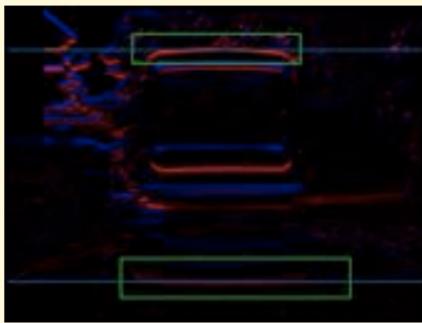


Gradient couleur

20.2. Histogramme. Histogramme des pixels rouges et bleus rencontrés dans les ignes horizontales.



- Les grandes raies correspondent *en principe* aux contours du CyCab.
- L'environnement *peut aussi* créer des raies.
- Pour *filtrer l'environnement* on considère uniquement l'histogramme des pixels à l'*intérieur des cadres*.
- Pour *suivre les cadres*, on prédit leur centre grâce à un filtre de *Kalman*.

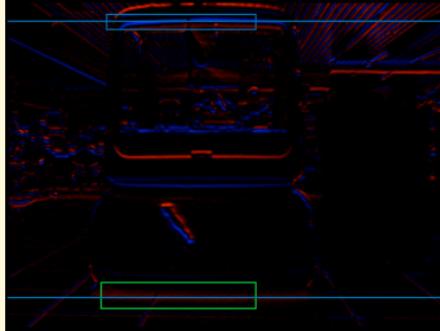
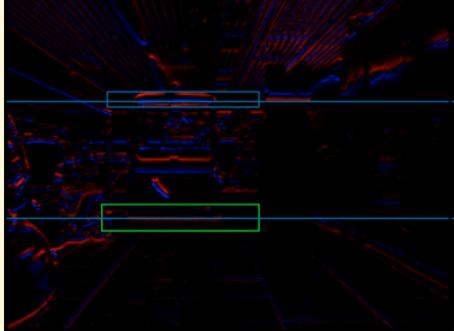


20.3. Filtre de Kalman. Modèle du déplacement de la raie dans l'image. La vitesse v de *déplacement* vertical des raies est *supposée localement constante* ($\dot{v} = 0$) et donc, leurs positions x vérifient $\dot{x} = v$, soit en temps discret le modèle :

$$\begin{cases} v_{n+1} = v_n, \\ x_{n+1} = x_n + hv_n, \\ y_n = x_n + w_n. \end{cases}$$

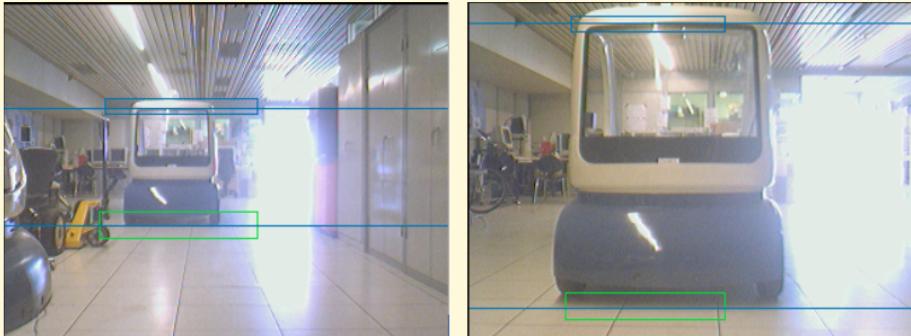
On peut calculer l'observateur optimal donné par le filtre Kalman en résolvant une équation de Riccati dans Scilab.

20.4. Résultat expérimental. Lors d'un déplacement du Cy-Cab, les raies correspondant au bord du CyCab sont suivies correctement.



- La couleur rouge ou bleue permet de mieux détecter la raie à suivre.
- La différence de hauteur observée entre deux raies donne l'observation de distance cherchée.

20.5. Résultat expérimental.



De la même façon, le suivi de raies verticales permettrait la poursuite 2D.

21. TYPE DE TÂCHES SOUS LINUX TEMPS RÉEL

On distingue 4 types de tâches selon les contraintes temporelles et d'espace (noyau/utilisateur) dans lequel elles s'exécutent :

	non TR	TR
Noyau	module	module RTAI
Utilisateur	processus	processus LXRT

Avantages/inconvénients :

- Modules : pas de protection mémoire, pas d'appels au fonctions de programmation utiles (open, write, printf, ...).
- Processus : protection mémoire, appels au fonctions de programmation utiles (open, write, printf, ...).

22. APPLICATION AU CYCAB

- Modules RTAI du CyCab : timer 10 ms pour les noeuds MPC555 (choix historique douteux).
- Processus LXRT : gestion de la caméra FireWire, traitement de l'image, IHM (choix intéressant car il permet à la fois de respecter les contraintes temporelles et donne accès aux fonctions de programmation Linux).
- A cause du manque de puissance de calcul, les contraintes temporelles ne pouvaient être satisfaites avec le matériel existant. Il a fallu acheter un PC plus puissant et installer un Linux temps réel RTAI.

23. CONCLUSION

- Responsabilité d'une application temps réel complète (traitement d'images, régulation, électronique, OS).
- Héritage d'un travail existant sans que les auteurs des travaux précédents ne soient accessibles (RobotSoft, stagiaires AOSTE).
- J'ai pu mener à bien la partie traitement de l'image et la partie régulation en simulation.
- Des pannes matérielles ont retardées la mise en oeuvre du régulateur (en cours).

24. FIN

Questions ?