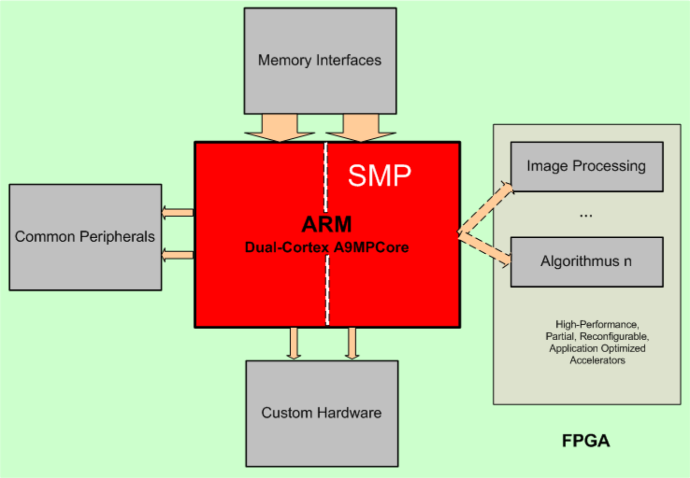
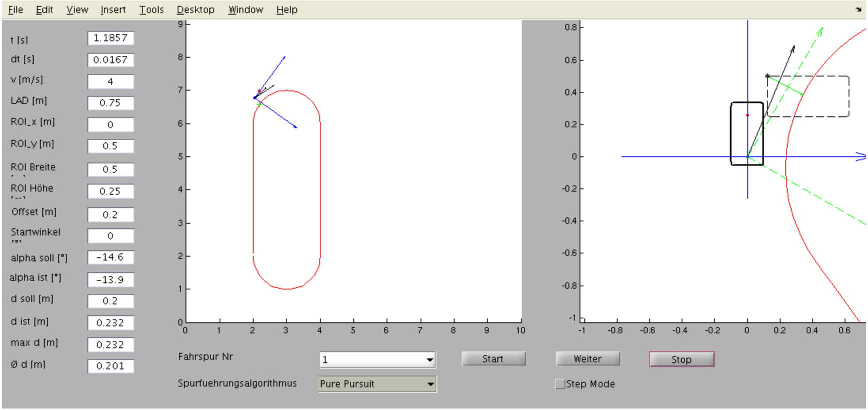
Parallele Programmierung von Eingebetten SMP-Plattformen am Beispiel von Spurführungs-Algorithmen

Erik Andresen

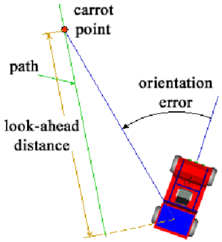
4. Mai 2011

- Mathematische Algorithmen zur Synthese in AccelDSP
- Anwendung: Fahrzeug
- Kalman-Filter zur Erkennung einer Fahrspur
- AW2: Fahrspur erkannt. Fragestellung:
  - Fahrspur ist durch eine Bildverarbeitungspipeline erkannt
  - Wie fahre ich?
  - Basierend auf den Arbeiten von Schneider 2011 (WiP), Nikolov.
- Software vs Hardware
- Läuft auf Laptop und FPGA.
- Parallelisierung auf Mikroprozessor?

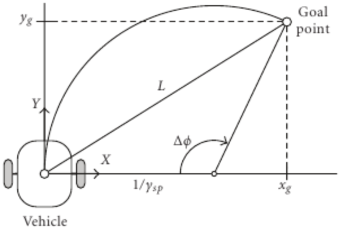

Memory Interfaces
SMP
ARM
Dual-Cortex A9MPCore
Common Peripherals
Image Processing
...
Algorithmus n
High-Performance,
Partial, Reconfigurable,
Application Optimized
Accelerators
Custom Hardware
FPGA


File Edit View Insert Tools Desktop Window Help
t [s] 1.1857
dt [s] 0.0167
v [m/s] 4
LAD [m] 0.75
ROI_x [m] 0
ROI_y [m] 0.5
ROI Breite 0.5
ROI Höhe 0.25
Offset [m] 0.2
Startwinkel 0
alpha soll [°] -14.6
alpha ist [°] -13.9
d soll [m] 0.2
d ist [m] 0.232
max d [m] 0.232
Ø d [m] 0.201
Fahrspur Nr 1
Spurfuehrungsalgorithmus Pure Pursuit
Start
Weiter
Stop
Step Mode

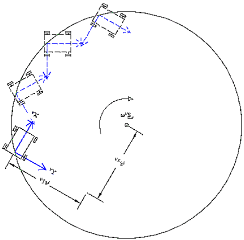
- Fragestellung: Wie schnell darf ich fahren und wie muss ich lenken um einen Weg zu folgen
- Pfad besteht aus einzelnen Punkten: Goal point
- Look-ahead distance



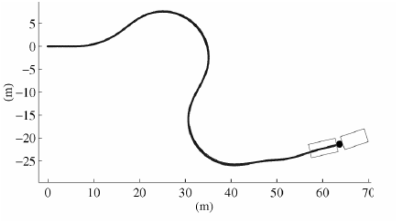
- $atan(x/y)$
- Praxis wenig genutzt, gut zum Vergleich.
- PID-Regler
- Einfach zu implementieren
- Schneidet Ecken
- Oszillierendes Verhalten



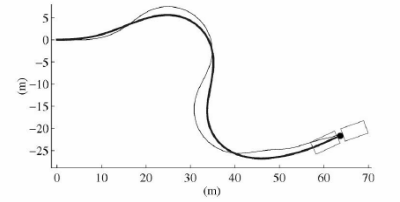
- Aus Militär: Rakete
- „Verfolgung des Punktes“, Kurve zum Ziel
- Pythagoras
- In der Praxis häufig genutzt.
- Varianten: Feedforward, Adaptive
- CMU, DARPA, Universidad de Málaga, HAW, ...



- Schraubentheorie: Bewegung von Objekten aufeinander abbilden.
- Mit korrekter Ausrichtung und Lenkwinkel am Ziel ankommen.
- Bei optimaler Strecke ist das Ergebnis vergleichbar mit Pure Pursuit.
- University of Florida



(a) Follow-the-Past



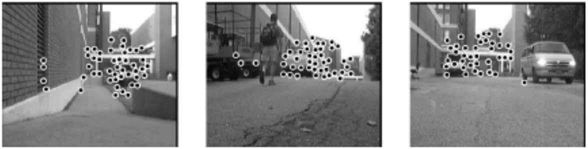
(b) Pure Pursuit

- Für Einsatz bei Wald Maschinen
- Folgt aufgenommenem Pfad
- In Entwicklung seit 2005
- Umea University, Schweden

- Datenbasiert
- Reinforcement learning
- Optimales Fahren als Eingabe
- 2007, Stanford, Osnabrück
- HAW: Nikolov

Rule1	if (*input1 is Very close*) and (*input 2 is Nomatterwhat*) and (*input3 is Poserr*) then (*output 1 is veryvery slow*) (*output2 is zero*)
Rule2	if (*input1 is close*) and (*input 2 is Nomatterwhat*) and ((*input3 is Poserr*) then (*output 1 is very slow*) (*output2 is zero*)
Rule3	if (*input1 is Medium*) and (*input 2 is Nomatterwhat*) and (*input3 is Poserr*) then (*output 1 is slow*) (*output2 is zero*)
Rule4	if (input1 is Far) and (*input 2 is Nomatterwhat*) and (*input3 is Poserr*) then (*output 1 is Fast*) (*output2 is zero*)
Rule5	if (*input1 is Very far*) and (*input 2 is Nomatterwhat*) and ((*input3 is Poserr*) then (*output 1 is Very fast*) (*output2 is zero*)
Rule6	if (*input1 is Nomatterwhat*) and (*input 2 is Big negative*) and (*input3 is Poserr*) then (*output 1 is veryvery slow*) (*output2 is Big negative*)
Rule7	if (*input1 is Nomatterwhat*) and (*input 2 is Negative*) and (*input3 is Poserr*) then (*output 1 is veryvery slow*) (*output2 is Negative*)
Rule8	if (*input1 is Nomatterwhat*) and (*input 2 is Straight*) and ((*input3 is Poserr*) then (*output 1 is veryvery slow*) (*output2 is zero*)
Rule9	if (*input1 is Nomatterwhat*) and (*input 2 is Positive*) and ((*input3 is Poserr*) then (*output 1 is veryvery slow*) (*output2 is Positive*)
Rule10	if (*input1 is Nomatterwhat*) and (*input 2 is Big positive*) and ((*input3 is Poserr*) then (*output 1 is veryvery slow*) (*output2 Big positive*)
Rule11	if (*input1 is Nomatterwhat*) and (*input 2 is Nomatterwhat*) and (*input3 is Negerr*) then (*output 1 is Zero*) (*output2 Zero*)

- 2010, Quebec University
- 2010, University of Jinan, China

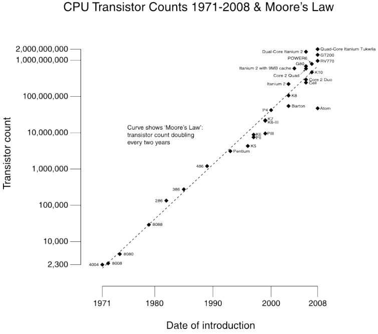


- Vergleich von Bildern
- 2009
- Clemson University, USA

- z.B. Pure Pursuit & PID-Regler
- Pure Pursuit Oszilliert mehr, ist aber näher am Pfad als PID
- Durchschnitt von beiden.
- HAW: 2/3 PID, 1/3 Pure Pursuit

- Andocken Unterwasser: Beginn mit Pure Pursuit

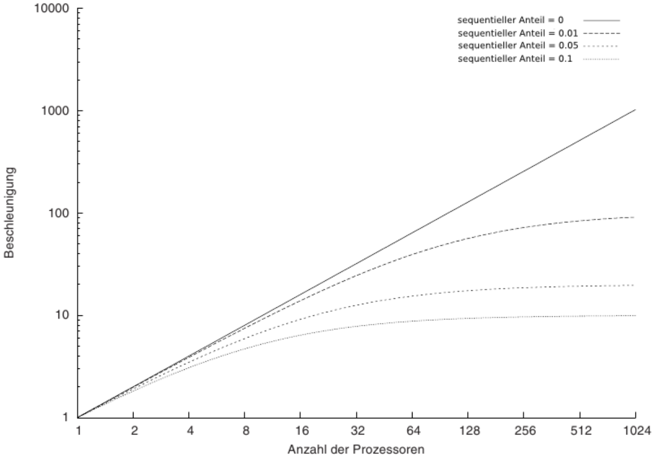
Moore'sches Gesetz

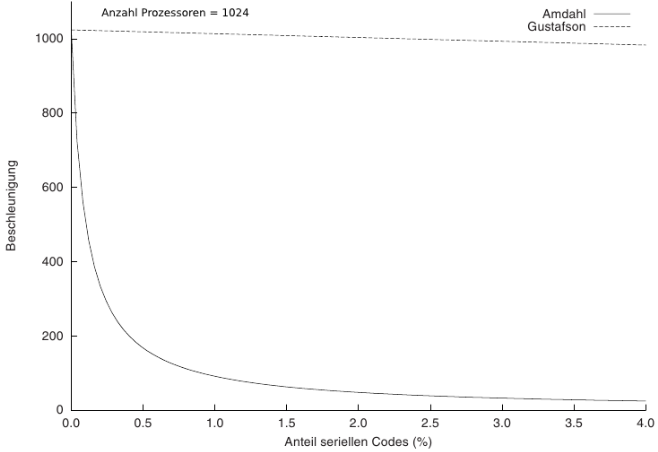


- Moore'sche Gesetz
- Embedded Prozessor vs FPGA

Amdahl'sche & Gustafson'sche Gesetz

- S Beschleunigung
- n Anzahl Prozessoren
- p parallele Programmcode
- Amdahl'sche Gesetz: $S = \frac{1}{1-p+\frac{p}{n}}$
- Gustafson'sche Gesetz: $S = (1-p) + n * p$
- Unsere Anwendung?


sequentieller Anteil = 0
sequentieller Anteil = 0.01
sequentieller Anteil = 0.05
sequentieller Anteil = 0.1
Beschleunigung
10000
1000
100
10
1
1
2
4
8
16
32
64
128
256
512
1024
Anzahl der Prozessoren


Anzahl Prozessoren = 1024
Amdahl
Gustafson
Beschleunigung
1000
800
600
400
200
0
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Anteil seriellen Codes (%)

Zu Lösen ist..

- Ziel: Embedded SMP
- Gemeinsam genutze Ressourcen und Synchronisation
- Gegenseitiger Ausschluss
- Erzeuger-Verbraucher Problem
- Bibliotheken
- z.B. Mutual Exclusion, University of Northern British Columbia (Kanada), 2011

- Portables, paralleles Programmiermodell. Hoher Abstraktionsgrad.
- Parallelisierung durch minimale Änderungen am sequentiellen Quellcode
- Unterstützung in diversen Compilern.
- http://www.openmp.org, http://www.compunity.org/
- fork, join, Barriere, geteilter Speicher
- Anzahl CPUs

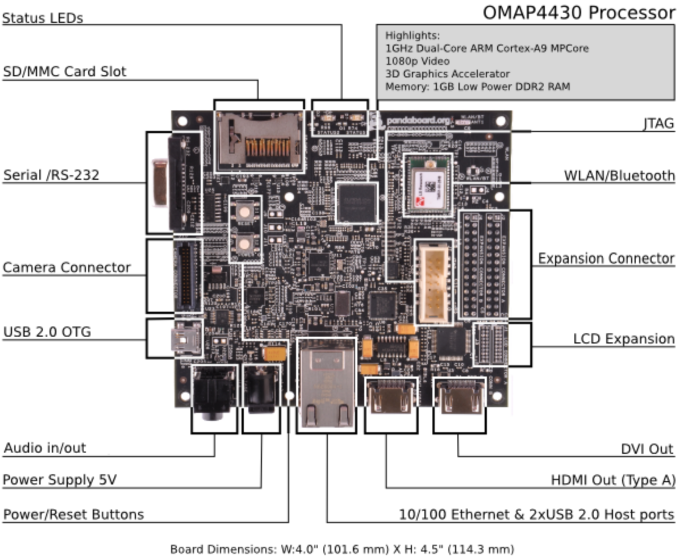
```
# pragma omp parallel for
for(i=0; i<len; i++) {
        something_complicated(i);
}
```

- Vorgeschlagene Erweiterung für OpenMP
- 2009, Barcelona Supercomputing Center
- Idee
- Vergleichbar OpenCL, Cuda, Brook+ als Beschleuniger
- Heterogenität: Grafikkarte, FPGA, Cell
- Initialisierungszeit
- Hardware Bibliothek
- 2010, National University of Defense Technology, China
- 2010, Median Filter, Madurai Kamaraj University, Indien
- 2010, Overhead, China

Message Passing Interface

- Nachrichtenaustausch für verteilte Computer
- SMP..Cluster
- Synchronisiert
- MPI_Send(), MPI_Recv()
- Image Processing, University of China, Peking, 2010
- Particle swarm, Indian Institute of Technology, Indien, 2010
- OpenMP & MPI: z.B. Cloud: Southwest University for Nationalities, China

Pandaboard: SMP auf Dual-core ARM Cortex-A9



1. Spurführung: Pure Pursuit & PID
2. SMP Betriebssystem
3. SMP Anwendung

- R. Craig Coulter, „Implementation of the Pure Pursuit Path Tracking Algorithm“, tech. report CMU-RI-TR-92-01, Robotics Institute, Carnegie Mellon University, January, 1992
- Alonzo Kelly, „A Feedforward Control Approach to the Local Navigation Problem for Autonomous Vehicles“, tech. report CMU-RI-TR-94-17, Robotics Institute, Carnegie Mellon University, May, 1994
- Jeffrey S. Wit: „Vector Pursuit Path Tracking for autonomous ground vehicles“, dissertation, University of Florida, 2000.
- Robert W. Hogg and Arturo L. Rankin and Stergios I. Roumeliotis and Michael C. Mchenry and Daniel M. Helmick and Charles F. Bergh and Larry Matthies: „Algorithms and Sensors for Small Robot Path Following“, IEEE International Conference on Robotics and Automation, Washington D.C, 2002.
- Martin Lundgren: „Path Tracking for a Miniature Robot“, Umea University Sweden, 2003.
- F. Georgsson, T.Hellström, T.Johansson, K.Prorok, O.Ringdahl, U.Sandström: „Development of an Autonomous Path Tracking Forest Machine“, Umea University Sweden, 2005.
- Thomas Hellström, Ola Ringdahl: „Follow the Past: a path-tracking algorithm for autonomous vehicles“, Umea University Sweden, 2006.
- Thomas Hellström, Thomas Johansson, Ola Ringdahl: „Development of an Autonomous Forest Machine for Path Tracking“, Umea University Sweden, 2006.
- Jesus Morales, Jorge L. Martinez, Maria A. Martinez, Anthony Mandow: „Pure-Pursuit Reactive Path Tracking for Nonholonomic Mobile Robots with a 2D Laser Scanner“, Universidad de Malaga, Spain, 2009
- Ivo Nikolov: „Verfahren zur Fahrbahnverfolgung eines autonomen Fahrzeugs mittels Pure Pursuit und Follow-the-carrot“, HAW Hamburg, Bachelorarbeit, 2009.
- Ivo Nikolov: „NFQ zur Optimierung eines Lenkungsreglers“, HAW Hamburg, Ausarbeitung, 2010.
- Christian Schneider: „Ein SoC-basiertes Fahrspurführungssystem“, HAW Hamburg, Bachelorarbeit, 2011.
- Zhichao Chen, Stanley T. Birchfield: „Qualitative Vision-Based Path Following“, IEEE, 2009.
- Hasan Mehrjerdi, Maarouf Saad, Jawhar Ghommam: „Hierarchical Fuzzy Cooperative Control and Path Following for a Team of Mobile Robots“, IEEE, 2010.

- Clay Breshears: „The Art of Concurrency“, O'Reilly, 2009, ISBN: 978-0-596-52153-0
- Maurice Herlihy, Nir Shavit: „The Art of Multiprocessor Programming“, Morgan Kaufmann, 2008, ISBN: 978-0-12-370591-4
- Simon Hoffmann, Rainer Lienhart: „OpenMP Eine Einführung in die parallele Programmierung mit C/C++“, Springer, 2008, ISBN: 978-3-540-73122-1
- Daniel Cabrera, Xavier Martorell, Georgi Gaydadjiev, Eduard Ayguade, Daniel Jimenez-Gonzalez: „OpenMP extensions for FPGA Accelerators“, Barcelona Supercomputing Center, 2009.
- Ying Peng, Fang Wang: „Cloud computing model based on MPI and OpenMP“, College of Computer Science and TechnoloSouthwest University for Nationalities, Chengdu, China, 2010.
- Tao Tang, Yisong Lin, Xiaoguang Ren: „Mapping OpenMP Concepts to the Stream Programming Model“, National Laboratory for Parallel and Distributed Processing National University of Defense Technology, Changsha, China, 2010.
- Alex A. Aravind: „Yet Another Simple Solution for the Concurrent Programming Control Problem“, IEEE, 2011.
- Jinliang Wan, Jinbao Song, Long Ye, Qin Zhang: „A Parallel Framework for Texture Substitution Coding“, Information Engineering School Communication University of China, Beijing, China, 2010.
- Kusum Deep, Sunita Sharma, Millie Pant: „Modified Parallel Particle Swarm Optimization for Global Optimization Using Message Passing Interface“, India, 2010.
- Dr.E.Ramaraj, Prof.A.Senthil rajan: „Median filter using open multiprocessing in agriculture“, Kamaraj University, India, 2010.
- Wan-Qing LI, Tang Ying, Yunfa LI, Zhang Wei, Jangcong Feng, Zhang Jilin: „Analysis of Parallel Computing Environment Overhead of OpenMP For Loop with Multi-core Processors“, China, 2010.