

Wochenbericht KW 17

Christoph Gnip

22.4.-28.4.2013

Anwesenheit

Mo	Di	Mi	Do	Fr	Sa	So
Bo	Bo	Bo	Ge	Ge	x	x

1. Allgemeines

Arbeiten an der Masterarbeit aufgenommen.

1.1. Organisation der Arbeit

Titel der Arbeit: [DE]	Entwicklung eines Systems zur Entfernungsabschätzung für Phasen basiertes UHF RFID Tracking durch Verwendung evolutionärer Berechnungsverfahren
Titel der Arbeit: [EN]	Development of a Distance Estimation System for Phase-Based UHF RFID Tracking by Utilizing Methodes of Evolutionary Computation
Interne Projektbezeichnung:	PRPS-Evo
Zeitraum:	22.April-2.September (20 Wochen)

Am Donnerstag, den 18.4., fand ein Treffen mit Herrn Prof. Dr. Frank Bärmann statt. Ihm wurde die Masterthesis vorgestellt und er erklärt sich als Erstbetreuer für die Arbeit einverstanden.

Die Anmeldung erfolgt am 2.5. Der Termin verspätet sich aufgrund der Öffnungszeiten des Prüfungsamts.

2. Projektfortschritt

In dieser Woche haben die Arbeiten an der Masterarbeit begonnen. Der Rahmen der Arbeit (Muss-, Soll- und Wunschkriterien) wurde detaillierter im Pflichtenheft abgesteckt. Das fertige Pflichtenheft wird bis Anfang KW 20 ausgearbeitet.

Weiterhin wurden in dieser Woche die in diesem Projekt verwendeten Entwicklungsumgebung(en) aufgesetzt und bedarfsgerecht installiert. Die Entwicklungsumgebungen, die im Rahmen dieser Arbeit verwendet werden, werden im Pflichtenheft ausführlicher vorgestellt und diskutiert.

Folgende Umgebungen wurden in dieser Woche aufgesetzt:

2.1. Projektverwaltung

Für die Verwaltung dieser Arbeit wird das Versionsverwaltungssystem Git verwendet. Es wird mit GitHub [git13] webbasierter Hosting-Dienst verwendet um die Datensicherung zu gewährleisten. Da GitHub im Rahmen einer Abschlussarbeit verwendet wird, erlaubt es der Betreiber das Repository den öffentlich Zugang zu verweigern. Somit geht das mit der Geheimhaltung konform. Git wird bereits in anderen Projekten der Amedo GmbH eingesetzt.

2.2. Entwicklung

- Installation der virtuellen Maschine, die als Produktivumgebung in diesem Projekt verwendet wird, als Betriebssystem wird Ubuntu in der Version 12.04 verwendet

- Installation aller zur Kompilierung notwendigen Software (u.A. GCC)
- Übersetzung und Installation der Shark-Library auf der Linux-Maschine

2.3. Dokumentation

- Für die Erstellung der Dokumentation und der Thesis selbst, wird \LaTeX verwendet.

2.4. Shark-Library

1. Bereits in dieser Woche ist es gelungen die Beispiel-Programme der Shark-Library zu erstellen.
2. Einarbeitung in den CMA-ES Algorithmus. Der Algorithmus wurde erfolgreich in Matlab umgesetzt und erste Versuche (Benchmarks) wurden mit verschiedenen Populationsgrößen durchgeführt. Die Ergebnisse des Benchmarks sind hier im Einzelnen nicht vorgestellt, da sie nur der Einarbeitung dienen. Ein Auszug aus dem Quellcode ist im Anhang A gezeigt.
3. Bereits in dieser Woche ist es gelungen die Beispiel-Programme der Shark-Library zu erstellen.

2.5. Recherche

Am Donnerstag wurde im Rahmen des Aufenthalts an der Westf. Hochschule Recherchearbeiten durchgeführt. Dabei lag der Schwerpunkt auf der Stand der Technik zum den Themen RFID-Tracking und Evolutionäre Verfahren, insb. die Kombination aus beiden Themenkomplexen. Ein interessantes Paper [SJ12] der gefundenen Literatur beschreibt, wie man mittels Phasen Messung und Kalman Filtern [Wik13] und einem Glättungsfilter. Die präsentierte Methode war in ihrem Setup ähnlich dem von der Amedo GmbH verwendeten Aufbau.

3. Probleme

Die Einrichtung der \LaTeX Umgebung dauerte länger als erwartet. Es war ein Tag Arbeit vorgesehen, aber es wurden zwei benötigt. Unter Windows gab es Stabilitätsprobleme mit dem verwendeten Editor Kile und dem KBibTeX-Tool. Die beobachteten Instabilitäten traten nicht in der Ubuntu-Umgebung auf. Ein entsprechender Bug-Report wurde eingereicht.

Anhänge

A. Matlab CMA-ES Code

Listing 1: CMA-ES Matlab Code; Entnommen aus [Han11] und für die ersten, einfachen Tests verwendet

```

1 function xmin=purecmaes    % (mu/mu_w, lambda)-CMA-ES
2
3 % ----- Initialization -----
4 % User defined input parameters (need to be edited)
5 strfitnessfct = 'frozenbrock'; % name of objective/fitness function
6 N = 20; % number of objective variables/problem dimension
7 xmean = rand(N,1); % objective variables initial point
8 sigma = 0.5; % coordinate wise standard deviation (step size)
9 stopfitness = 1e-10; % stop if fitness < stopfitness (minimization)
10 stopeval = 1e3*N^2; % stop after stopeval number of function evaluations
11
12 % Strategy parameter setting: Selection
13 lambda = 4+floor(3*log(N)); % population size, offspring number
14 mu = lambda/2; % number of parents/points for recombination
15 weights = log(mu+1/2)-log(1:mu)'; % muXone array for weighted recombination
16 mu = floor(mu);
17 weights = weights/sum(weights); % normalize recombination weights array
18 mueff=sum(weights)^2/sum(weights.^2); % variance-effectiveness of sum w_i x_i
19
20 % Strategy parameter setting: Adaptation
21 cc = (4+mueff/N) / (N+4 + 2*mueff/N); % time constant for cumulation for C
22 cs = (mueff+2) / (N+mueff+5); % t-const for cumulation for sigma control
23 c1 = 2 / ((N+1.3)^2+mueff); % learning rate for rank-one update of C
24 cmu = min(1-c1, 2 * (mueff-2+1/mueff) / ((N+2)^2+mueff)); % and for rank-mu update
25 damp = 1 + 2*max(0, sqrt((mueff-1)/(N+1))-1) + cs; % damping for sigma
26 % usually close to 1
27 % Initialize dynamic (internal) strategy parameters and constants
28 pc = zeros(N,1); ps = zeros(N,1); % evolution paths for C and sigma
29 B = eye(N,N); % B defines the coordinate system
30 D = ones(N,1); % diagonal D defines the scaling

```

```

31 C = B * diag(D.^2) * B';           % covariance matrix C
32 invsqrtC = B * diag(D.^-1) * B';   % C^-1/2
33 eigeneval = 0;                     % track update of B and D
34 chiN=N^0.5*(1-1/(4*N)+1/(21*N^2)); % expectation of
35                                     % ||N(0,I)|| == norm(randn(N,1))
36
37 % ----- Generation Loop -----
38 counteval = 0; % the next 40 lines contain the 20 lines of interesting code
39 while counteval < stopeval
40
41     % Generate and evaluate lambda offspring
42     for k=1:lambda,
43         arx(:,k) = xmean + sigma * B * (D .* randn(N,1)); % m + sig * Normal(0,C)
44         arfitness(k) = feval(strfitnessfct, arx(:,k)); % objective function call
45         counteval = counteval+1;
46     end
47
48     % Sort by fitness and compute weighted mean into xmean
49     [arfitness, arindex] = sort(arfitness); % minimization
50     xold = xmean;
51     xmean = arx(:,arindex(1:mu))*weights; % recombination, new mean value
52
53     % Cumulation: Update evolution paths
54     ps = (1-cs)*ps ...
55         + sqrt(cs*(2-cs)*mueff) * invsqrtC * (xmean-xold) / sigma;
56     hsig = norm(ps)/sqrt(1-(1-cs)^(2*counteval/lambda))/chiN < 1.4 + 2/(N+1);
57     pc = (1-cc)*pc ...
58         + hsig * sqrt(cc*(2-cc)*mueff) * (xmean-xold) / sigma;
59
60     % Adapt covariance matrix C
61     artmp = (1/sigma) * (arx(:,arindex(1:mu))-repmat(xold,1,mu));
62     C = (1-c1-cmu) * C ... % regard old matrix
63         + c1 * (pc*pc' ... % plus rank one update

```

```

64         + (1-hsig) * cc*(2-cc) * C) ... % minor correction if hsig==0
65         + cmu * artmp * diag(weights) * artmp'; % plus rank mu update
66
67     % Adapt step size sigma
68     sigma = sigma * exp((cs/damps)*(norm(ps)/chiN - 1));
69
70     % Decomposition of C into B*diag(D.^2)*B' (diagonalization)
71     if counteval - eigeneval > lambda/(c1+cmu)/N/10 % to achieve O(N^2)
72         eigeneval = counteval;
73         C = triu(C) + triu(C,1)'; % enforce symmetry
74         [B,D] = eig(C); % eigen decomposition, B==normalized eigenvectors
75         D = sqrt(diag(D)); % D is a vector of standard deviations now
76         invsqrtC = B * diag(D.^-1) * B';
77     end
78
79     % Break, if fitness is good enough or condition exceeds 1e14,
80     % better termination methods are advisable
81     if arfitness(1) <= stopfitness || max(D) > 1e7 * min(D)
82         break;
83     end
84
85 end % while, end generation loop
86
87 xmin = arx(:, arindex(1)); % Return best point of last iteration.
88 % Notice that xmean is expected to be even
89 % better.
90
91 % -----
92 function f=frosenbrock(x)
93     if size(x,1) < 2 error('dimension_must_be_greater_one'); end
94     f = 100*sum((x(1:end-1).^2 - x(2:end)).^2) + sum((x(1:end-1)-1).^2);

```

B. Tabellen

Populationsgröße	Ausführungszeit	Funktion
x	x	x
y	y	y

Tabelle 1: Performancevergleich für unterschiedliche Populationsgrößen

Literatur

- [git13] github.com. <https://github.com/>, 2013.
- [Han11] Nikolaus Hansen. Juni 2011.
- [SJ12] Ville V. Viikari Miika Huusko Simo Särkkä and Kaarle Jaakkola. Phase-Based UHF RFID Tracking With Nonlinear Kalman Filtering and Smoothing. feb 2012.
- [Wik13] Wikipedia. Kalman-Filter — Wikipedia, The Free Encyclopedia, 2013. [Online; zuletzt editiert am 4-April-2013].