Wochenbericht KW 17

22.4. - 28.4.2013

Projektwoche: 1

Erstellt durch
Christoph Gnip

Fachbereich Elektrotechnik und angewandte Naturwissenschaften
Westfälische Hochschule

Mai 2013

1 Allgemeines

In dieser Woche hat die Bearbeitung der Masterarbeit begonnen. Dieses Dokument ist der erste Wochenbericht. Es werden jede Woche ein in Umfang an den jeweiligen Projektfortschritt angepasster Bericht erstellt.

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-9.September (20 Wochen)

Am Donnerstag, den 18.4., fand ein Treffen mit Herrn Prof. Dr. Frank Bärmann statt. Ihm wurde das Thema und der grobe Ablauf der Masterthesis vorgestellt. Er erklärt sich als Erstbetreuer für die Arbeit einverstanden. Die Anmeldung erfolgt am 2.5. Der Vorgang verspätete 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. Auch ein grober Projektplan wird im Zuge des Pflichtenhefts erstellt.

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.

Im Folgenden werden die Arbeiten dieser Woche Kategorisch beschrieben.

2.1 Projektverwaltung

Für die Verwaltung dieser Arbeit wird das Versionsverwaltungssystem Git verwendet. Es wird mit GitHub [1] webbasierter Hosting-Dienst verwendet um die Datensicherung zu gewährleisten. Da GitHub im Rahmen einer Abschlussarbeit verwendet wird, erlaubt es das Repository den öffentlich Zugang zu verweigern. Somit geht das mit der Geheimhaltung konform. Git wird bereits in anderen Projekten der Amedo GmbH eingesetzt. Durch seine Flexibilität wird Git für alle Arbeiten und Dokumente dieser Arbeit verwendet, nicht nur für die entwickelten Programme.

2.2 Entwicklung

- Installation der virtuellen Maschine, die als Produktivumgebung in diesem Projekt verwendet wird, als Betriebssystem wird Ubuntu in der Version 12.04 (LTS) verwendet.
- Installation aller zur Kompilierung notwendigen Software (u.A. GCC).
- Übersetzung und Installation der Shark-Library (Siehe 2.4) in der Linux Umgebung.

2.3 Dokumentation

- Anlegen einer geeigneten Verzeichnisstruktur auf dem Server; Diese wird analog in dem Git-Repository verwendet.
- Für die Erstellung der Dokumentation und der Thesis selbst, wird LaTeX verwendet.

2.4 Shark-Library

Bei der Shark-Library [2] handelt es sich um eine C++-Bibliothek in der unterschiedliche Methoden der linearen- und nicht-linearen Optimierung implementiert sind. Insbesondere ist auch der in dieser Arbeit zur Anwendung kommen Algorithmus CMA-ES (Covariance Matrix Adaption - Evolutionary Strategy) implementiert und kann so ggf. gegen andere Verfahren verglichen werden. Das Verfahren wird, sobald es umfassend verstanden ist, in einem Bericht vorgestellt.

Des Weiteren:

1. Einarbeitung in den CMA-ES Algorithmus; In Shark stehen verschiedene Tutorials zur Verfügung; Diese wurden zum aktuellen Zeitpunkt jedoch nicht vollständig verstanden.
2. Bereits in dieser Woche ist es gelungen die Beispiel-Programme der Shark-Library zu erstellen.
3. Ein erstes eigenes C++ Programm wurde erstellt; Es soll später mit dem Matlab-Code (Anhang A) vergleichen werden.

Außerdem: Der Algorithmus wurde in einem Matlab-Skript umgesetzt und erste Versuche (Benchmarks) wurden in dieser Umgebung mit verschiedenen Populationsgrößen durchgeführt. Da dieser Schritt der Einarbeitung dient, werden die Ergebnisse des Benchmarks noch nicht vorgestellt. Das Matlab-Skript ist im Anhang A gezeigt, es ist im wesentlichen Kopiert und wurde in anderen Varianten für Tests verwendet.

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. Literatur die eine Kombination aus beiden Themenkomplexen enthält. Ein dem Paper [3] ist beschrieben, wie man mittels Phasen Messung, Kalman Filtern [4] und der Verwendung eines Glättungsfilter eine gute Positionsgenauigkeit erreicht. Die präsentierte Methode war in ihrem Setup ähnlich dem von der Amedo GmbH verwendetem.

3 Probleme

Die Einrichtung der LaTeX Umgebung dauerte länger als erwartet. Es war ein Tag Arbeit vorgesehen, jedoch 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. Die Erstellung des ersten Wochenberichts dauerte länger und dieser konnte erst in KW 18 fertiggestellt werden.

Anhänge

A Matlab CMA-ES Code

Listing 1: CMA-ES Matlab Code; Entnommen aus [5] und für die ersten, einfachen Tests verwendet

```
function xmin=purecmaes % (mu/mu_w, lambda)-CMA-ES

  % -------------------- Initialization --------------------------------
  % User defined input parameters (need to be edited)
  strfitnessfct = 'frosenbrock'; % name of objective/fitness function
  N = 20;              % number of objective variables/problem dimension
  xmean = rand(N,1);  % objective variables initial point
  sigma = 0.5;         % coordinate wise standard deviation (step size)
  stopfitness = 1e-10; % stop if fitness < stopfitness (minimization)
  stopeval = 1e3*N^2; % stop after stopeval number of function evaluations

  % Strategy parameter setting: Selection
  lambda = 4+floor(3*log(N)); % population size, offspring number
  mu = lambda/2;              % number of parents/points for recombination
  weights = log(mu+1/2)-log(1:mu)'; % muXone array for weighted recombination
  mu = floor(mu);
  weights = weights/sum(weights); % normalize recombination weights array
  mueff=sum(weights)^2/sum(weights.^2); % variance-effectiveness of sum w_i x_i

  % Strategy parameter setting: Adaptation
  cc = (4+mueff/N) / (N+4 + 2*mueff/N); % time constant for cumulation for C
  cs = (mueff+2) / (N+mueff+5); % t-const for cumulation for sigma control
  c1 = 2 / ((N+1.3)^2+mueff); % learning rate for rank-one update of C
  cmu = min(1-c1, 2 * (mueff-2+1/mueff) / ((N+2)^2+mueff)); % and for rank-mu update
  damps = 1 + 2*max(0, sqrt((mueff-1)/(N+1))-1) + cs; % damping for sigma
                                                      % usually close to 1
  % Initialize dynamic (internal) strategy parameters and constants
  pc = zeros(N,1); ps = zeros(N,1); % evolution paths for C and sigma
  B = eye(N,N);                     % B defines the coordinate system
  D = ones(N,1);                    % diagonal D defines the scaling
  C = B * diag(D.^2) * B';          % covariance matrix C
  invsqrtC = B * diag(D.^-1) * B'; % C^-1/2
```

```
eigeneval = 0;                  % track update of B and D
chiN=N^0.5*(1-1/(4*N)+1/(21*N^2)); % expectation of
                                %   ||N(0,I)|| == norm(randn(N,1))

% -------------------- Generation Loop --------------------------------
counteval = 0; % the next 40 lines contain the 20 lines of interesting code
while counteval < stopeval

    % Generate and evaluate lambda offspring
    for k=1:lambda,
        arx(:,k) = xmean + sigma * B * (D .* randn(N,1)); % m + sig * Normal(0,C)
        arfitness(k) = feval(strfitnessfct, arx(:,k)); % objective function call
        counteval = counteval+1;
    end

    % Sort by fitness and compute weighted mean into xmean
    [arfitness, arindex] = sort(arfitness); % minimization
    xold = xmean;
    xmean = arx(:,arindex(1:mu))*weights; % recombination, new mean value

    % Cumulation: Update evolution paths
    ps = (1-cs)*ps ...
         + sqrt(cs*(2-cs)*mueff) * invsqrtC * (xmean-xold) / sigma;
    hsig = norm(ps)/sqrt(1-(1-cs)^(2*counteval/lambda))/chiN < 1.4 + 2/(N+1);
    pc = (1-cc)*pc ...
         + hsig * sqrt(cc*(2-cc)*mueff) * (xmean-xold) / sigma;

    % Adapt covariance matrix C
    artmp = (1/sigma) * (arx(:,arindex(1:mu))-repmat(xold,1,mu));
    C = (1-c1-cmu) * C ...              % regard old matrix
        + c1 * (pc*pc' ...              % plus rank one update
                + (1-hsig) * cc*(2-cc) * C) ... % minor correction if hsig==0
        + cmu * artmp * diag(weights) * artmp'; % plus rank mu update

    % 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);
```

Literatur

[1] github.com, "GitHub — Repository Host." `https:/github.com`, 2013. [Online, zuletzt geprüft am 30.4.2013].

[2] C. Igel, V. Heidrich-Meisner, and T. Glasmachers, "Shark," *Journal of Machine Learning Research*, vol. 9, pp. 993–996, 2008.

[3] Simo Särkkä, K. Jaakkola, and V. V. V. M. Huusko, "Phase-Based UHF RFID Tracking With Nonlinear Kalman Filtering and Smoothing," February 2012.

[4] Wikipedia, "Kalman-Filter — Wikipedia, The Free Encyclopedia." `http://de.wikipedia.org/w/index.php?title=Kalman-Filter\&oldid=116893284`, 2013. [Online; zuletzt editiert am 4-April-2013].

[5] N. Hansen, *The CMA Evolution Strategy*. Juni 2011.