Fine Grain Quality Management

Jacques Combaz Jean-Claude Fernandez Mohamad Jaber
Joseph Sifakis Loïc Strus

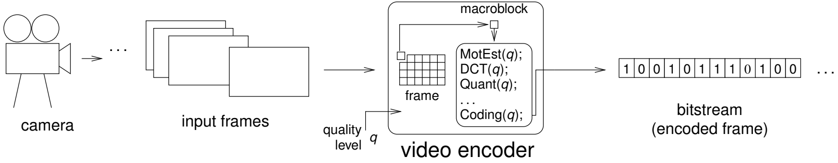
Verimag Lab.
Université Joseph Fourier
Grenoble, France

DCS seminar, 10 June 2008, Col de Porte

1. Motivations
   - Video Encoder
   - Problem & Approaches
2. Fine Grain QoS Control
   - Model
   - Approach
   - Summary
3. Stochastic Approach
   - Model & Problem
   - Stochastic Mixed Quality Management Policy
   - Performance Analysis
   - Conclusion
4. Learning
   - Problem
   - Neural networks
   - Experimental results

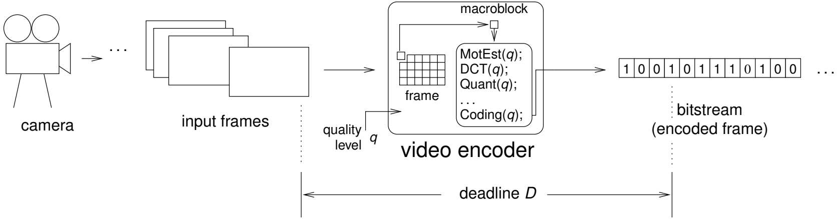
Video Encoder

- collaboration with **STMicroelectronics** (**GaloGiC** project)
- **embedded video encoder**



Video Encoder

- collaboration with **STMicroelectronics** (**GaloGiC** project)
- **embedded video encoder**



  - real-time constraints (e.g. $D = 1/30$ s $= 33$ ms)
  - limited resources
  - intensive computing
  - uncertainty on execution times
- we propose an online adaptation of the **quality level** parameters q

Problem & Approaches

Problem (embedded video encoder)

- meet the deadlines
  - input buffer minimization
  - avoid frame skipping
- optimal use of resources
  - QoS maximization

Problem & Approaches

Problem (embedded video encoder)

- meet the deadlines
  - input buffer minimization
  - avoid frame skipping
- optimal use of resources
  - QoS maximization

Existing Approaches

- hard real-time (critical system engineering)
  - worst-case analysis
  - meet the deadlines
- soft real-time (best-effort engineering)
  - average-case analysis $\rightarrow$ no guarantee
  - efficient use of resources

Problem & Approaches

Problem (embedded video encoder)

- meet the deadlines
  - input buffer minimization
  - avoid frame skipping
- optimal use of resources
  - QoS maximization

Our Approach

- bridging the gap by using **adaptive techniques**
- adapting software behaviour by setting **quality level parameters**
- optimal use of computing resources with **real-time guarantees**

Model

Application Software

$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \dots \xrightarrow{a_n} s_n$$

s_i: control point
a_i: *actions* parameterized by integer quality levels $q_i \in Q = [q_{min}, q_{max}]$

Model

Application Software

$$s_0 \xrightarrow{a_1} s_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} s_n$$

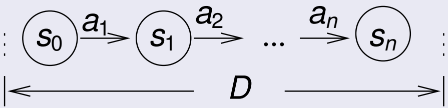
s_i: control point
a_i: *actions* parameterized by integer quality levels $q_i \in Q = [q_{min}, q_{max}]$

Estimates of Execution Times (increasing with quality)

$C^{av}(a_i, q)$: *average* execution time of a_i at the quality level q
$C^{wc}(a_i, q)$: *worst-case* execution time of a_i at the quality level q

Model

Application Software



s_i: control point
a_i: *actions* parameterized by integer quality levels $q_i \in Q = [q_{min}, q_{max}]$

Estimates of Execution Times (increasing with quality)

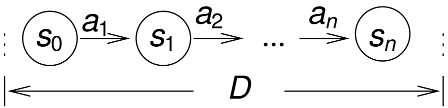
$C^{av}(a_i, q)$: *average* execution time of a_i at the quality level q
$C^{wc}(a_i, q)$: *worst-case* execution time of a_i at the quality level q

Real-Time Constraints

D: *deadline* for the completion of the actions

Quality Management Problem

Application software:



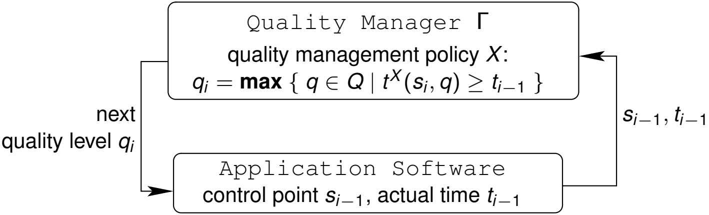
The Problem

find quality level parameters q_i such that:

- **safety** — deadlines are met
- **optimality** — maximization of the quality levels
- **smoothness** of the quality levels

→ **online** computation of the quality levels q_i

Quality Manager Design



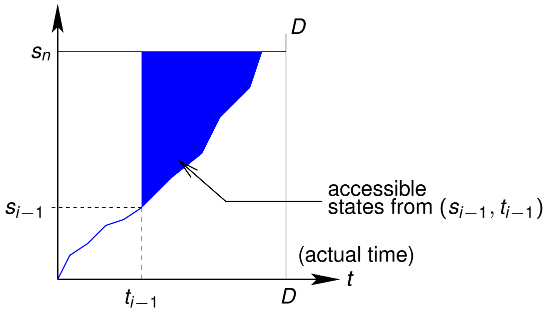
- remaining execution time (estimate) for quality q: $C^X(a_i..a_n, q)$
- *feasibility* criterion: $D \geq C^X(a_i..a_n, q) + t_{i-1}$
- we define $t^X(s_{i-1}, q) = D - C^X(a_i..a_n, q)$

Quality Management Policy X at state (s_{i-1}, t_{i-1})

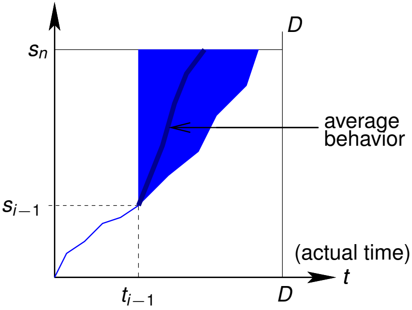
choosing the best quality level which is *feasible*, i.e.:

$$q_i := \mathbf{max} \; \{ \; q \mid t^X(s_{i-1}, q) \geq t_{i-1} \; \}$$

Mixed Quality Management Policy [EMSOFT'05,RTS]

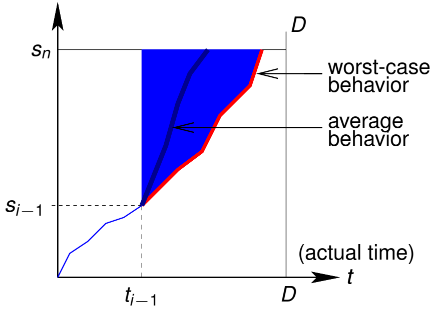


Mixed Quality Management Policy [EMSOFT'05,RTS]



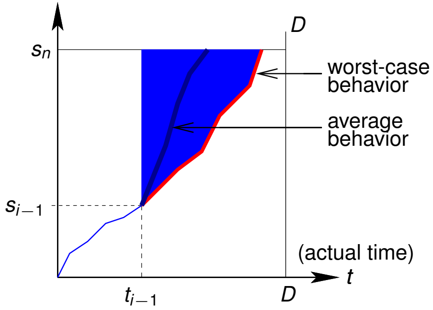
- average behavior: $C^{av}(a_i..a_n, q) = C^{av}(a_i, q) + \ldots + C^{av}(a_n, q)$

Mixed Quality Management Policy [EMSOFT'05,RTS]



- average behavior: $C^{av}(a_i..a_n, q) = C^{av}(a_i, q) + \ldots + C^{av}(a_n, q)$
- worst-case behavior (under control): $C^{sf}(a_i..a_n, q) = C^{wc}(a_i, q) + C^{wc}(a_{i+1}, q_{min}) + \ldots + C^{wc}(a_n, q_{min})$

Mixed Quality Management Policy [EMSOFT'05,RTS]



- average behavior: $C^{av}(a_i..a_n, q) = C^{av}(a_i, q) + \ldots + C^{av}(a_n, q)$
- worst-case behavior (under control):
  $C^{sf}(a_i..a_n, q) = C^{wc}(a_i, q) + C^{wc}(a_{i+1}, q_{min}) + \ldots + C^{wc}(a_n, q_{min})$

$$\boxed{C^{mx} = C^{av} + \delta^{max}}, \text{ where}$$

$$\delta^{max}(a_i..a_n, q) = \textbf{max}\ \{\ C^{sf}(a_j..a_n, q) - C^{av}(a_j..a_n, q) \mid j \geq i\ \}$$

Summary

- **(+)** fine grain quality management — mixed policy:
  - improves predictability
  - reduces the impact of the worst-case (measured by δ^{max})
  - safety, optimality, smoothness

Summary

- **(+)** fine grain quality management — mixed policy:
  - improves predictability
  - reduces the impact of the worst-case (measured by δ^{max})
  - safety, optimality, smoothness

- **(-)** limitations — worst-case estimates:
  - complex HW platforms — computation of worst-case estimates
  - non-flexibility — only 100% guarantee
  - low predictability and low time budget utilization for:

$$\delta^{max}(\overbrace{a_1 a_2 \ldots\ldots a_j}^{\text{controllable}} \overbrace{a_{j+1} a_{j+2} \ldots\ldots a_n}^{\text{uncontrollable}}, q) \text{ can be sizable}$$

Summary

- **(+)** fine grain quality management — mixed policy:
  - improves predictability
  - reduces the impact of the worst-case (measured by δ^{max})
  - safety, optimality, smoothness
- **(-)** limitations — worst-case estimates:
  - complex HW platforms — computation of worst-case estimates
  - non-flexibility — only 100% guarantee
  - low predictability and low time budget utilization for:

$$\delta^{max}(\overbrace{a_1 a_2 \ldots\ldots a_j}^{\text{controllable}} \overbrace{a_{j+1} a_{j+2} \ldots\ldots a_n}^{\text{uncontrollable}}, q) \text{ can be sizable}$$

→ we need **soft real-time** methodologies that provide **guarantees & predictability**

Model & Problem

Estimates of the Execution Times

the integer probability distribution function d_i^q:

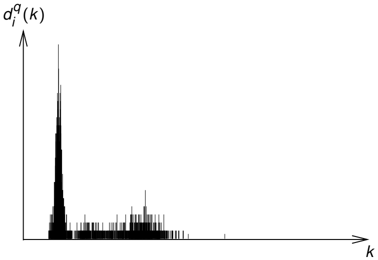
- $d_i^q : \mathbb{N} \to [0, 1]$
- $d_i^q(k) =$ P[" execution time of a_i at quality q is k"]
- $\sum_{k=0}^{+\infty} d_i^q(k) = 1$
- independent execution times

Problem

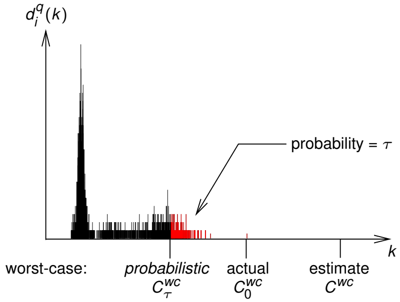
find a Quality Manager Γ s.t.:

- **safety** — the deadline miss ratio is $\leq$ a target ratio $\mu \in [0, 1]$
- **optimality** — maximization of the quality levels
- **smoothness** of the quality levels

Stochastic Mixed Quality Management Policy



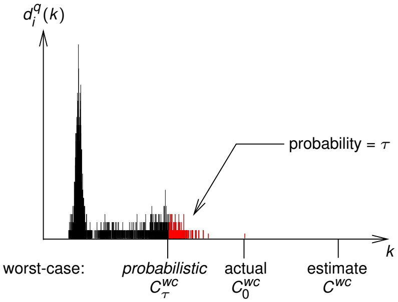
Stochastic Mixed Quality Management Policy



- *probabilistic* worst-case execution time $C_\tau^{wc}(a_i, q)$ s.t.:

$$C_\tau^{wc}(a_i, q) = \mathbf{min} \; \{ \; l \mid \sum_{k=l}^{+\infty} d_i^q(k) \leq \tau \}$$

Stochastic Mixed Quality Management Policy



- *probabilistic* worst-case execution time $C_\tau^{wc}(a_i, q)$ s.t.:

$$C_\tau^{wc}(a_i, q) = \textbf{min} \; \{ \; l \mid \sum_{k=l}^{+\infty} d_i^q(k) \leq \tau \}$$

- average execution time (mean value): $C^{av}(a_i, q) = \sum_{k=0}^{+\infty} k d_i^q(k)$

Performance Analysis

- d_i: probability distribution for actual time t_i (completion of $a_1..a_i$)
- recursive computation of d_i:

$$d_i(k) = \sum_{l=0}^{+\infty} d_{i-1}(l) d_i^{q_i}(k-l), \text{ where } q_i = \Gamma(s_{i-1}, l)$$

Performance Analysis

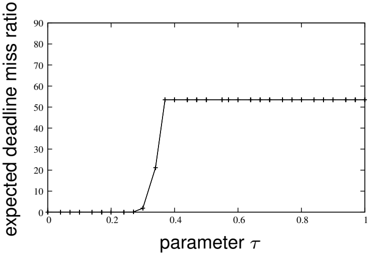
- d_i: probability distribution for actual time t_i (completion of $a_1..a_i$)
- recursive computation of d_i:

$$d_i(k) = \sum_{l=0}^{+\infty} d_{i-1}(l) d_i^{q_i}(k-l), \text{ where } q_i = \Gamma(s_{i-1}, l)$$

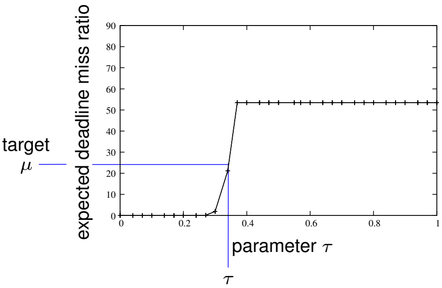
Performance of the Quality Manager Γ

- expected deadline miss ratio: $\mathrm{P}[t_n > D] = \sum_{k>D} d_n(k)$
- expected time budget utilization: $\mathrm{E}(t_n) = \sum_{k=0}^{+\infty} k d_n(k)$

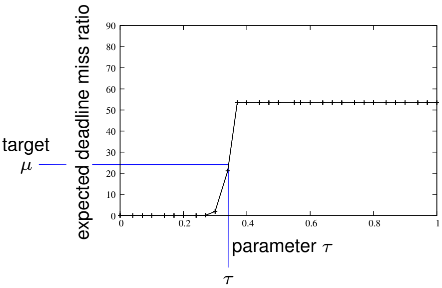
Performance Analysis

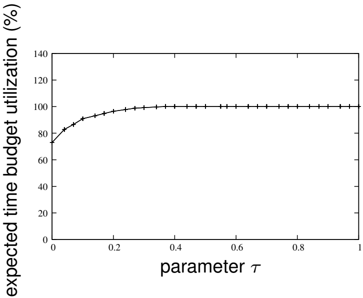


Performance Analysis

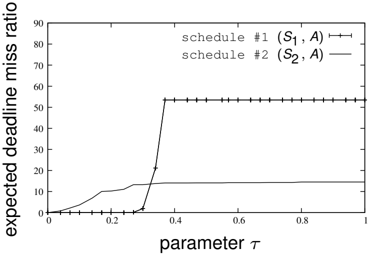


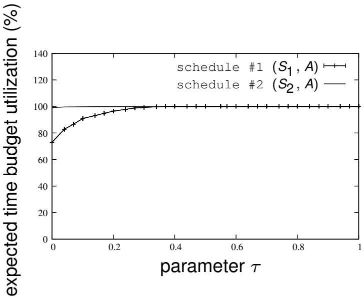
Performance Analysis



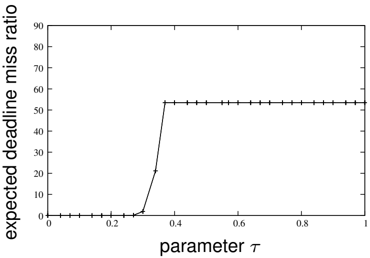


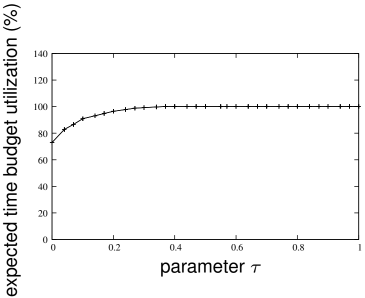
Performance Analysis

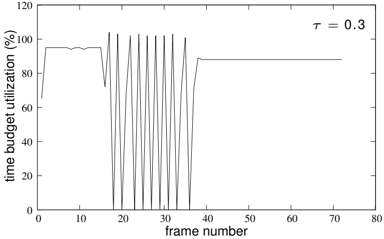


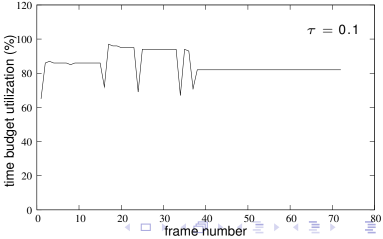


Performance Analysis









Conclusion

- stochastic mixed quality management policy:
  - soft real-time guarantees
  - achieving a tradeoff time budget utilization / deadline miss ratio
  - parameterized worst-case scenario influence:

parameter τ:	0	1
deadlines:	hard	soft

  - code reuse: one source for different target HWs / QoS requirements

- perspectives:
  - depencies between execution times
  - multiple tasks, multiple processors, operating systems

Problem

Using worst-case execution time is problematic in some cases !

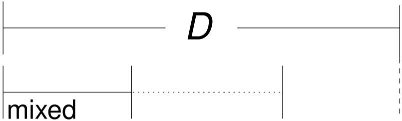
$$\overbrace{a_1, a_2, \ldots, a_j}^{\textit{controllable}} \; \overbrace{a_{j+1}, \ldots, a_n}^{\textit{uncotrollable}}$$

$$\vdash\!\!-\!\!-\!\!-\!\!-\; D \;-\!\!-\!\!-\!\!-\!\!\dashv$$

Problem

Using worst-case execution time is problematic in some cases !

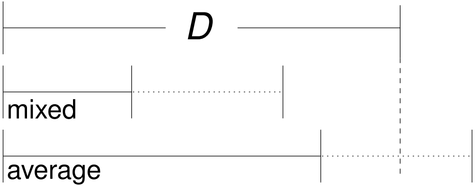
$$\overbrace{a_1, a_2, \ldots, a_j}^{\textit{controllable}} \; \overbrace{a_{j+1}, \ldots, a_n}^{\textit{uncotrollable}}$$



Problem

Using worst-case execution time is problematic in some cases !

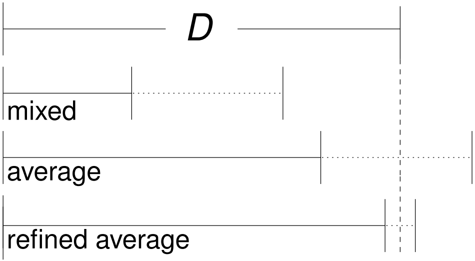
$$\overbrace{a_1, a_2, \ldots, a_j}^{\textit{controllable}}\ \overbrace{a_{j+1}, \ldots, a_n}^{\textit{uncotrollable}}$$



Problem

Using worst-case execution time is problematic in some cases !

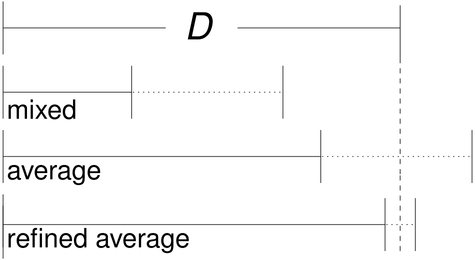
$$\overbrace{a_1, a_2, \ldots, a_j}^{\textit{controllable}}\ \overbrace{a_{j+1}, \ldots, a_n}^{\textit{uncotrollable}}$$



Problem

Using worst-case execution time is problematic in some cases !

$$\overbrace{a_1, a_2, \ldots, a_j}^{\textit{controllable}} \; \overbrace{a_{j+1}, \ldots, a_n}^{\textit{uncotrollable}}$$



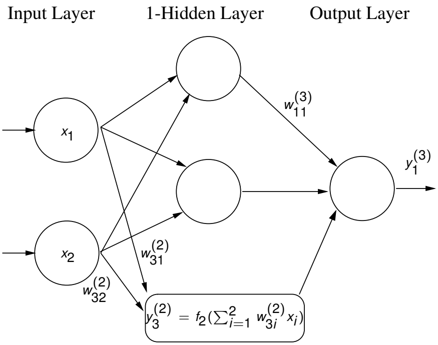
Refine average execution times

- Better predict the execution time of action using its input
- Using neural netwroks to predict the execution time

Neural Network

A neural network is defined by:

- $L = \{ L_1, L_2, \ldots, L_n \}$
- $w_{ij}^{(l)}$ is the weight
- $\theta_i^{(l)}$ is a scalar bias
- N_l is the number of neurons
- f_l is the activation function
- $y_i^{(l)} = \begin{cases} f_l(\sum_{j=1}^{N_{l-1}} y_j^{(l-1)} w_{ij}^{(l-1)} + \theta_i^{(l)}) & \text{if } l \neq 1 \\ x_i \text{ (input of neuron } i) & \text{if } l = 1. \end{cases}$



Architecture and learning algorithm

Architecture

- the number of layers (n) one hidden layer approximate every continuous function
- the number of neurons in each layer (N_l) number of input * 2
- the activation functions (f_l) *identity* : $x \mapsto x$ and *sigmoid* : $x \mapsto \frac{1}{1+e^{-\beta x}}$

Learning algorithm

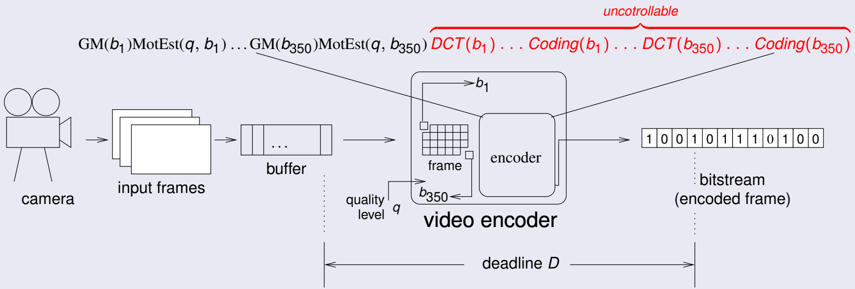
(X, Y) where $X = (x_1, \ldots, x_{N_1})$ input and $Y = (y_1, \ldots, y_{N_n})$ desired output.

$$\text{Minimize} \quad E(W) = 1/2 \sum_{j=1}^{N_n} (y_j - y_j^{(n)})^2.$$

1. Initialize the network weights W.
2. Select a new sample (X, Y).
3. Update weights of the output and hidden layers using specific rules, that is, $W \leftarrow W + \Delta W$.
4. Go to 3 if the error $E(W)$ is above a tolerance value.
5. Go to 2 if other samples must be learnt.

Experimental framework

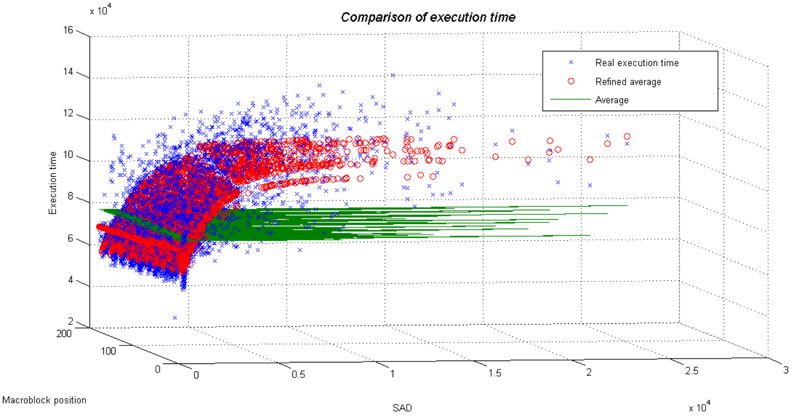
MPEG4 video encoder



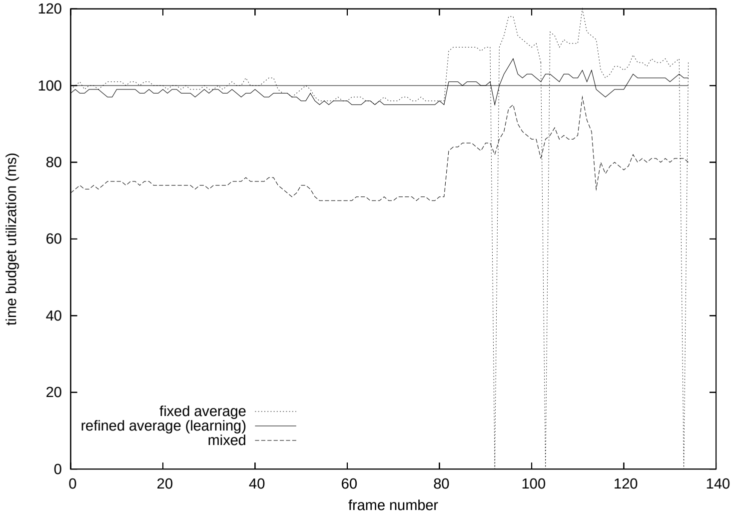
Use neural networks for computing refined average of uncotrollable actions. We consider $X = (x_1, x_2)$:

- x_1 is the SAD value (sum of absolute difference)
- x_2 is the position of the macroblock

Results


Comparison of execution time
x 10^4
16
14
12
10
8
6
4
2
Execution time
Real execution time
Refined average
Average
200
100
0
0
0.5
1
1.5
2
2.5
3
Macroblock position
SAD
x 10^4

Results



120
100
80
60
40
20
0
time budget utilization (ms)
0
20
40
60
80
100
120
140
frame number
fixed average
refined average (learning)
mixed

Perspective

BIP