# **Mitschrift NVS**

## **24. Oktober 2018**

## Bildbearbeitung

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| r | g | b | t |

Bmp 🡪 verlustfrei kompirmierbar

Jpg, png 🡪 verlustbehaftet komprimierbar //aufwändig Fra…transformation + Frequenzauswahl + Lauflängenkompression 🡪 Anzeige wieder rqbt pro Pixel also 4 2 dim Felder

R[1920] [1080]

r[0] [0] r[0] [1] r[0] [2] …

r[1] [0] r[1] [1] r[1] [2] …

r[2] [0] r[2] [1] r[2] [2] …

Wenn man ein Pixel filtern möchte, müssen alle Nachbarn in der Nähe des Bildes betrachtet werden am besten symmetrisch. Sonst Phasenverschiebung. Objekte ändern ihren Platz.

Je mehr Nachbarn desto größer der Rechenaufwand v2

* Median: noch schlimmer, weil beim Median sortiert wird

z..B: 3x3 Median d.h. für ein Pixel wird der Median aus allen Nachbarn links, recht, oben und unten und diagonalen gebildet 🡪 9 Werte werden sortiert. Bei FullHD mit 3x3 = 1920\*1080 \* 4 Sortieralgorithmen

**Mittwoch, 24. Oktober 2018**

Filter

File

Image Original

Image edited

Statusbar

Bitmapimage 🡪 bytearray 🡪 r[x, y]; g[x, y]; b[x, y] 🡪 rf[x, y]; gf[x, y]; bf[x, y]🡪bytearray (writePixels)🡪bitmapimage

(stride[bytes/line], CopyPixels)

**October 1, 2018**

# Datenaustausch

OS Linux 🡨🡪Linux nfs

Arbeitet mit IP und UID (linux befehl usermode einstellen) 🡪 nur wenn es zentral verwaltet wird und es gibt nur einen root

/etc/exports

CSMA/CR – Resolution

Header:

Höchste T1: 1111

Zweite T2: 1110

…

Beim Senden zuerst Header

T1 + T2 gleichzeitig

T1 5v

🡪in Summe 5v 🡪 Carrier Sense: T1 und T2 sehen die 5V 🡪 T2 hat 0V gelegt und sieht 5V und hört auf

T2 0V

Typisch für CAN-Bus im Anto

Höchste Priorität Bremsen, Airbag, …

Bitstuffing: serielle Datenübertragungen 🡪 eine Leitung

Muster aus n0 und 1 und alle Teilnehmer müssen die Taktlänge kennen da sie sonst das Muster falsch interpretieren.

* Zeitmessung oder Synchronisation über Flanken

Bei der Startflanke beginnen alle mit ihrer Zeitmessung

Bitstuffting

Viele 0 hintereinander 🡪 zusätzlich 1

**24. Oktober 2018**

# CRC

Bitfolge wird zum Empfänger gesendet

Bitfolge dividiert durch Polynom (z.B. CPS 10101) (mit Xor statt Subtraktion (schneller))

Rest an Empfänger senden

Empfänger subtrahiert Rest von Bitmuster

Dividiert durch das gleiche Polynom

* Gutes Polynom; viele Bits falsch angekommen damit Rechnung trotzdem auf 0 ausgeht