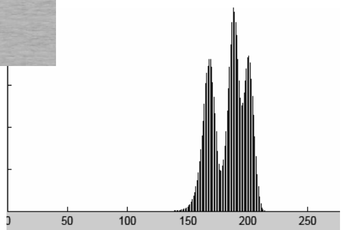
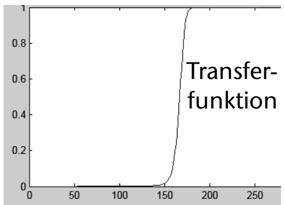
Problem der Histogram-Equalization

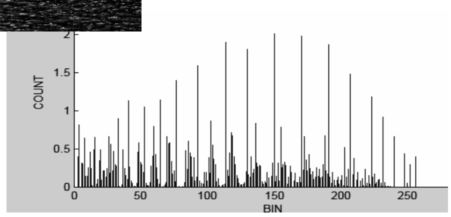
- Problem: ein sehr schmales Histogramm im Eingabebild



- Ergebnis: unerwünschter Kontrast

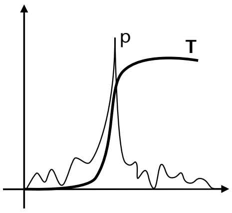


Resultierendes Histogramm



Tone Reproduction nach Ward et al. [1997]

- Problem der Histogram-Equalization:
  - Sehr steile Abschnitte der Transferfunktion T können sichtbares Rauschen hervorrufen
- Idee: beschränke die Steigung von T
- Algo:



1. Bestimme das Histogramm h
   - Erinnerung: $h \approx p = T'$
2. Clampe zu große Bins auf einen Wert $\alpha \cdot \frac{N}{B}$, wobei $\alpha \approx 0.5 \ldots 1.5$, N = Anzahl Pixel, B = Anzahl Bins
3. Setze $N' = \sum_{i=0}^{L-1} h(x_i)$
4. Führe damit Equalization durch und wiederhole ein paar Mal.

Exkurs: Das Weber-Fechner-Gesetz [~1850]

- Experimenteller Befund:
  - Die "*just noticeable difference*" (JND) eines Stimulus (z.B. Gewicht) hängt von der Stärke des Stimulus ab ("differentielle Wahrnehmbarkeitsschwelle")
  - Das Verhältnis von JND zu Stärke des Stimulus ist eine (stimulusabhängige) Konstante.
- Mathematische Formulierung:
  - Sei S der Betrag des Stimulus, und ΔS die JND
  - Weber's Gesetz sagt:

$$\frac{\Delta S}{S} = \text{const}$$

- Das Weber-Fechner-Gesetz:
  Sei E die Stärke des wahrgenommenen Sinneseindrucks von S ("gefühltes Gewicht").
  Dann gilt für ΔE = die JND von E:

$$\Delta E = k\frac{\Delta S}{S} \quad \Rightarrow \quad \frac{dE}{dS} = k\frac{1}{S}$$

- Integrieren liefert:

$$E = k \cdot \ln S + c$$

  - Dabei ist c eine Konstante, die den minimaler Stimulus S_0 beschreibt, bei dem gerade noch ein Sinneseindruck $E \approx 0$ entsteht (Schwellenreiz):

$$c = -k \cdot \ln S_0$$

- Zusammen:

$$E = k \cdot \ln \frac{S}{S_0}$$

Exkurs²: die Stevens'sche Potenzfunktion

- Eine andere plausible Annahme scheint (m.M.n.) folgende:

$$\frac{\Delta E}{E} = k\frac{\Delta S}{S}$$

- Umformen liefert:

$$S \cdot \Delta E = k \cdot E \cdot \Delta S \quad \Rightarrow \quad S \cdot \mathrm{d}E = k \cdot E \cdot \mathrm{d}S \quad \Rightarrow$$

$$\frac{1}{E}\mathrm{d}E - k\frac{1}{S}\mathrm{d}S = 0 \quad \Rightarrow \quad \ln E - k \ln S = c \quad \Rightarrow$$

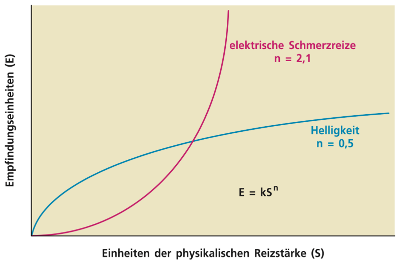
$$\ln \frac{E}{S^k} = c \quad \Rightarrow \quad \frac{E}{S^k} = e^c = c' \quad \Rightarrow$$

- Ergibt schlussendlich Stevens' Power Law:

$$E = cS^k$$

wobei E = Empfindungsstärke ("gefühltes Gewicht"), S = Stimulus (physikalische Größe), c und k = Konstanten, die vom Sinnesorgan abhängen.

- Für viele Reize ist $k < 1$ (Helligkeit ≈ 0.5, Lautstärke ≈ 0.6)
- Für manche Reize ist $k > 1$ (Temperatur ≈ 1-1.6, elektrischer Schock ≈ 2-3)



Bemerkungen zu den Gesetzen

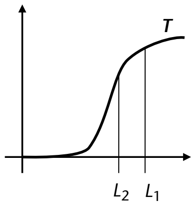
- Das Weber-Fechner-Gesetz beschreibt (anscheinend) besser die Wahrnehmung von Stimuli im mittleren Bereich, das Stevens-Power-Law besser im unteren und oberen Bereich.
- Die Forschung zu beiden Gesetzen ist noch voll im Gang
- Es gibt erste Anzeichen, daß neuronale Netze bzw. zelluläre Automaten auch dieses Verhalten zeigen, wenn man Sinneswahrnehmung (Erregung + Transport) damit simuliert!

Anwendung im Tone Mapping

- Annahme: zwei benachbarte Pixel im Originalbild haben gerade einen Intensitätsunterschied von der JND, also

$$\Delta L = L_1 - L_2 = J(L_1)$$

(oBdA ist $L_1 > L_2$)



- Gesucht ist eine Transferfunktion T, so daß diese Bedingung eine Invariante ist, also

$$T(L_1) - T(L_2) \leq J(T(L_1))$$

- Umformen:

$$p(L_1) = T'(L_1) \approx \frac{T(L_1) - T(L_2)}{L_1 - L_2} \leq \frac{J(T(L_1))}{L_1 - L_2} = \frac{J(T(L_1))}{J(L_1)}$$

- Algorithmus:
  1. Berechne das Histogramm
  2. Berechne das kumulative Histogramm, d.h., T
  3. Clampe alle Bins, so daß

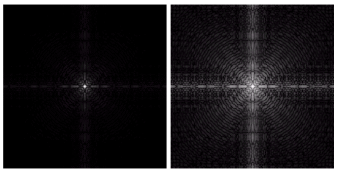
$$h(i) \leq \frac{J(T(L_i))}{J(L_i)}$$

     wobei L_i der Intensitätslevel von Bin i ist
  4. Wiederhole ein paar Mal

- Nebenbemerkung: das Weber-Fechner-Gesetz ist auch der Grund, warum die Histogram-Equalization bzw. das Tone-Mapping sehr oft im sog. "Log-Space" durchgeführt wird

Luminanz-Bild → Log. → Brightness-Bild → Histogram-Equalization → Brightness-Bild → Exp. → Luminanz-Bild

Fourier-Spektrum eines Bildes vor und nach dem Log.



Beispiel

Weitergehende Ideen

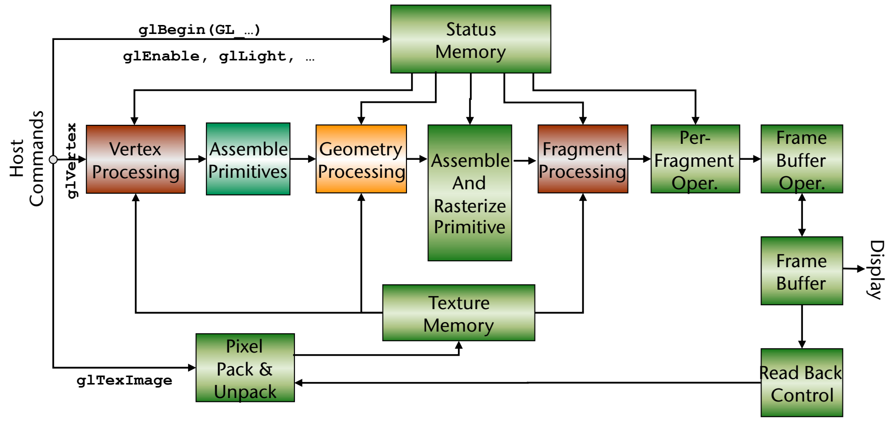
- Problem: diese Methode verhindert $\Delta L > J(L)$ auch zwischen Pixeln, die nicht benachbart sind
  - Idee: mappe jedes Pixel individuell nur unter Berücksichtigung der Nachbar-Pixel
    - →Echter lokaler Tone-Mapping-Operator (TMO)
  - Führt leider wieder zu anderen Problemen (z.B. sog. "Halos")
- Weitere Beschränkungen des Human Visual Systems (HVS):
  - Blendung (glare): starke Lichtquellen in der Peripherie reduzieren Kontrastempfindlichkeit des Auges
  - Skotopisches / mesopisches Sehen: bei niedriger Luminanz nimmt die Farbempfindlichkeit stark ab
  - Ebenso nimmt räumliches Auflösungsvermögen ab
  - → Könnte man im TMO ausnutzen

Erzeugen eines Histogramms auf der GPU

- Gegeben: Grauwertbild (= Textur)
- Ziel: Histogramm als 1D-Textur
  - Jedes Texel = ein Bin
- Problem: "Verteilen" auf Bins
  - Ziel-Adresse eines Fragment-Shaders ist ja fest
- Erste Idee:
  - Pro Pixel im Originalbild einen Punkt (GL_POINT) "rendern";
  - im Vertex-Shader das entsprechende Bin ausrechnen (statt Transf. mit MVP-Matrix);
  - die "Koordinate" dieses Bins als Koordinate des Punktes an den Fragment-Shader weiterreichen
- Problem:
  - Hohes Datenübertragungsvolumen CPU → GPU
  - Z.B.: 1024^2x2x4 Bytes = 8 MB zusätzlich zum 1024^2-Bild

Enter ... the Geometry Shader [Nov 2006]

- Eine dritte Shader-Art:



- Doku: http://www.opengl.org/registry/specs/NV/geometry_shader4.txt

Allgemeine Funktionsweise des Geometry Shaders

- Applikation generiert irgendwelche Primitive (Points, Lines, Triangle-Fans, etc.)
- Vertex-Shader transformiert diese (immer ein Vertex auf einmal)
- Geometry-Shader bekommt von der Assembly-Stufe Primitive
  - Nur GL_POINTS, GL_LINES, GL_TRIANGLES (+ 2 weitere)
- Geometry-Shader gibt neue Primitive aus
  - Nur GL_POINT, GL_LINE_STRIP, GL_TRIANGLE_STRIP
  - Muß nichts mit der eingebenen Geometrie zu tun haben
  - Anzahl kann (fast) beliebig sein, ist unabhängig von der Anzahl der eingebenen Primitive!
- Typ der Input-/Output-Geometrie muß vorab festgelegt werden
- Zugriff auf OpenGL-State und Texturen wie üblich

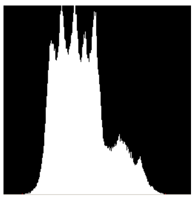
- Output des Vertex-Shaders:
  gl_Position →
  gl_Normal →
  gl_TexCoord →
  …

Input des Geometry-Shaders:
gl_PositionIn[] →
gl_NormalIn[] →
gl_TexCoordIn[][] →

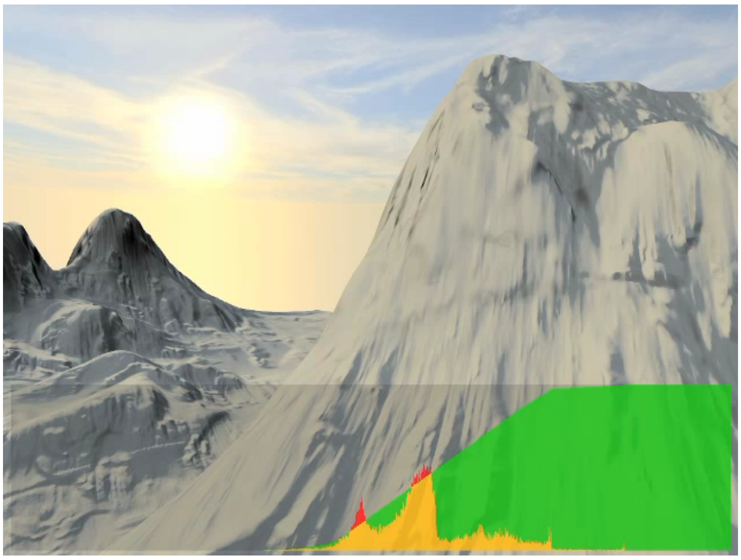
Output des Geometry-Shaders:
gl_Position
gl_Normal
gl_TexCoord[]

Erzeugung von Histogrammen mittels Geometry-Shader

- Ein Quad in der Applikation rendern
- Vertex-Shader ist (fast) leer
- Der Geometry-Shader …
  - läuft durch das Bild,
  - erzeugt für jedes Pixel ein Point-Primitiv mit der x-Koordinate = Bin , y=0
- Fragment-Shader …
  - nimmt die Points,
  - gibt Farbe (1,0,0,0) aus,
  - an der Position (x,0)
- Fragment-Operation …
  - ist auf Blending eingestellt mit **glBlendFunc(GL_ONE,GL_ONE)** = Akkumulation (aktuelle Karten können das auch mit FP-FBOs)



Video



Thorsten Scheuermann, Justin Hensley; 2007.
Graphics Product Group, Advanced Micro Devices Inc.

Berechnung der Transferfunktion auf der GPU

- Erinnerung: *parallel prefix sums* bzw. *summed area tables*

High-Dynamic Range Imaging in der Photographie

- Waren sogar zuerst da   [Charles Wyckoff, 1930-40]
- Inzwischen alles in Photoshop & Co. integriert

Beispiele