

**Berufsakademie Sachsen**  
Studiengang Informatik  
Klausur-Teil, März 2024 / Fragen:  
(Dozent: Dieter Herrmann, Dresden)  
Fragen: 6; Punkte (von 72): 70+;  
Hilfsmittel: PC, Mitschrift, TaschenRechner,  
außer Funknetze / Zusammenarbeit, ohne Internet.

Dresden, 14.03.2024

**Name, Vorname:**  
-----  
SGr.: -----

**Computer-Grafik+Animation CS21- (Informatik), März 2024 (90 Min)**  
Comp-Grafik, Animate Color, Motion, Path, Transform, Effects, 3D-2D, Timing

**Frage 1)**

Erläutern Sie kurz die folgenden sechs Begriffe sowie deren Funktionsweise oder Verfahren.  
(12 Punkte)

**Computer Vision :**

-  
-

**Fraktals, IFS :**

-  
-

**Generative AI :**

-  
-

**Point Cloud :**

-  
-

**SDR, HDR :**

-  
-

**W3.org, WoT :**

-  
-

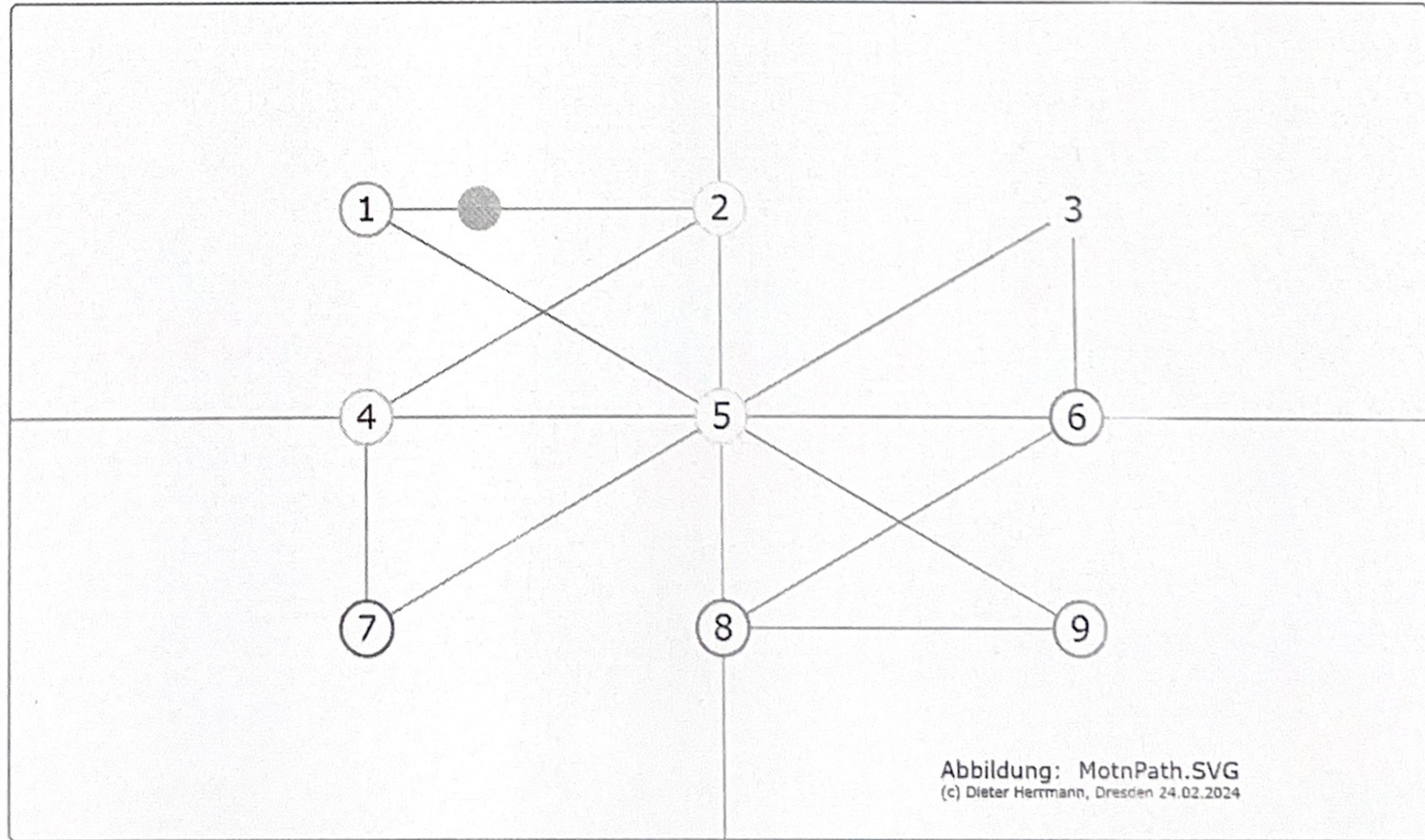
## Frage 2)

Ein Motion Path soll auf einem UHD Bildschirm 3840x2160 Pixel @ 16:9 einen Pattern Lock oder Muster Wisch-Pfad als Zick-Zack Kurve (SFC - Space Filling Curve) in einem 3x3 Ziffernblock visualisieren. Teilt man die X-Achse in 1/4, 2/4 und 3/4 sowie auch die Y-Achse in 1/4, 2/4 und 3/4, ergeben sich die x-y Koordinaten für die Ziffern 1 bis 9 von links oben nach rechts unten auf dem UHD Bildschirm. Tragen Sie in `polyline id="WischPfad"` `points=` die 9 x-y Koordinaten für die Zick-Zack Curve 1-2-4-7-3-6-8-9-1 ein. Beachten Sie dabei, dass von 7-3 der Punkt 5 übersprungen wird und dass am Ende die Kurve von 9-1 wieder zum Ursprung geschlossen wird. Übertragen Sie dann die 9 Punkte in `animateTransform id="Pattern_Path" values=`. Da sich die Translation jeweils auf `cx="0" cy="0"` bezieht, können die absoluten x-y Koordinaten direkt übernommen werden. (12 Punkte)

```
<svg id="id="PatternLock_3x3" width="100%" height="100%">
    viewBox=" -20 -20 3880 2200 " . . .
<title>Pattern Lock 3x3 - Animate Motion Path in UHD</title>
<rect id="canvas" x="0" y="0" width="3840" height="2160" . . . />
<line id="horiz_x" x1="0" y1="1080" x2="3840" y2="1080" . . . />
<line id="verti_y" x1="1920" y1="0" x2="1920" y2="2160" . . . />

<!-- Pattern 3x3 Number Block _ 1 2 3 _ 4 5 6 _ 7 8 9 --&gt;
&lt;polyline id="WischPfad" points="_____,_____,_____,_____,_____,_____,_____,_____,_____" /&gt;

&lt;circle id="Flying_Spot" cx="0" cy="0" r="60" &gt;
    &lt;!-- Animate Path : Zick-Zack Curve _ 1-2-4-7-3-6-8-9-1 --&gt;
    &lt;animateTransform id="Pattern_Path" attributeName="transform"
        type="translate" begin="0.5s" dur="12.5s"
        repeatDur="indefinite" values="_____,_____,_____,_____,_____,_____,_____,_____,_____
            ; _____,_____,_____,_____,_____,_____,_____,_____,_____
            ; _____,_____,_____,_____,_____,_____,_____,_____,_____
            ; _____,_____,_____,_____,_____,_____,_____,_____,_____
            " /&gt;
&lt;/circle&gt; &lt;/svg&gt;</pre>
```



### Frage 3)

Bei Bildstabilisatoren, Objekt-Suche und Verfolgung, Gesichts-Erkennung oder Personen-Identifikation, in QR-Codes, VR/AR, AI Media bis zum Predictive Modeling wird Pattern-Matching (Block-Matching) eingesetzt.

In der MPEG Video-Kompression mit Motion-Compensation MC oder MPEG-7 zur visuellen Beschreibung (Visual Description) von Merkmalen wie Farben, Texturen, Kanten, Formen, Bewegungen und dergleichen wird ebenso Block-Matching genutzt. Ein Pixel-Raster von  $4 \times 4$  Bildpunkten mit Objekt [1 1 1] bewegt sich relativ vom I-Frame 1 zu P-Frame 2. Im Block-Matching der zentralen  $[2 \times 2] = 4$  Pixel sind jeweils die 3 Einzel-Differenzen  $di=f(Frame\ 2 - MC\ Frame\ 1)$  in allen 9 Richtungen um die 9 Fadenkreuze einzutragen.

Aus den jeweils 3 Einzel-Differenzen ist unten die Summe der Differenz-Quadrat zu bilden. Abschließend sind die Abweichungen in den 9 Richtungen nach einem Ranking zu sortieren von 1 für Minimal bis 9 für Maximal. Eventuelle Gleichrangigkeit wird entsprechend Abarbeitung von links oben nach rechts unten ausgeschlossen. Die minimale Abweichung 1 entspricht damit dem gefundenen und wahrscheinlichsten Bewegungs-Vektor. (12 Punkte)

I-Frame 1	Motion Frame 2	Motion Estimation - $d[2 \times 2]$ $di=f(Frame\ 2 - MC\ Frame\ 1)$
+---+---+---+---+	+---+---+---+---+	-   2 -   -
2   3   4   5	2   3   5   6	2   3
+---+---+---+---+	+---+---+---+---+	- +--- - +--- - +---
3   4   1   6	3   4   5   7	-   -   -
+---+---+---+---+	+---+---+---+---+	- +--- - +--- - +---
4   1   1   7	1   2   6   7	
+---+---+---+---+	+---+---+---+---+	- +--- - +--- - +---
5   6   7   8	3   2   7   8	-   -   -
+---+---+---+---+	+---+---+---+---+	- +--- - +--- - +---

Block-Summe (Diff) <sup>2</sup>	Ranking Min-Max 1...9
+---+---+---+	+---+---+---+
17	
+---+---+---+	+---+---+---+
+---+---+---+	+---+---+---+
+---+---+---+	+---+---+---+

**Frage 4)**

In visuellen Systemen (PC, Grafik, Foto, Video, HDTV, UHD, Cinema, Internet, Web, Print) spielen Farb-Kodierungen von Text, Grafik, JPEG, MPEG, Pixeln, Point Clouds oder Voxeln eine entscheidende Rolle. Entsprechend unserem Trichromatischen Farbsehen und Monochromatischen Helligkeitssehen werden je nach Anwendung unterschiedliche, häufig 3D Farbraum-Modelle eingesetzt. Beispiele sind sRGB und RGB Color Cube für PC, TV und Internet, HSL Doppelkegel für Web und Internet mit CSS Styles nach W3.org, oder YCbCr in JPEG und MPEG sowie entsprechend ITU.int.

Tragen Sie in den beiden unteren Tabellen für SDR – Standard Dynamic Range mit je 8 bit und dezimal von 0 bis 255 die fehlenden Farb-Kodierungen ein. (12 Punkte)

**Tabelle Chrominanz, Farbigkeit, mit  $Y(RGB) = 0.3 R + 0.6 G + 0.1 B$** 

#	Name	rgb(r,g,b)	Color	Grad	hsl(h,s,l)	Y(1.00)	Y(255)
0	Red	255, 0, 0	R	0°	0, 100%, 50%	0.30	77
128	Orange	255, 128, 0	O	°	, 100%, 50%	0.	
255	Yellow	255, 255, 0	Y	°	, 100%, 50%	0.	
510	Green	0, 255, 0	G	°	, 100%, 50%	0.	
638	Emarald	0, 255, 128	E	°	, 100%, 50%	0.	
765	Cyan	0, 255, 255	C	°	, 100%, 50%	0.	
1020	Blue	0, 0, 255	B	°	, 100%, 50%	0.	
1275	Magenta	255, 0, 255	M	°	, 100%, 50%	0.	
1530	Red	255, 0, 0	R	360°	0, 100%, 50%	0.30	77

**Tabelle Luminanz, Helligkeit, mit  $Y(RGB) = 0.3 R + 0.6 G + 0.1 B$** 

#	Name	rgb(r,g,b)	Gray	Grad	hsl(h,s,l)	Y(1.00)	Y(255)
255	White	255, 255, 255	W	0°	0, 0%, 100%	1.00	255
170	Light	,	L	0°	, %, %	0.	
128	Gray	,	G	0°	, %, %	0.	
85	Dark	,	D	0°	, %, %	0.	
0	Black	0, 0, 0	K	0°	0, 0%, 0%	0.00	0

### Frage 5)

Ein Panorama Bild mit einem Blickfeld von 180 Grad horizontal und 90 Grad vertikal hat eine höhere Winkelauflösung als das menschliche Sehvermögen und mit 14592 x 7296 px im Seitenverhältnis @ 2 : 1 ein Pixel-Volumen von rund 107 Mega-Pixeln.

Die Vorschau soll auf einem HDTV Bildschirm mit 1920x1080px vollständig eingepasst, unverzerrt und zentriert in `id="Underscan"` erfolgen. Berechnen Sie dazu die `transform="matrix( )"` und tragen Sie die Parameter auf 6 Stellen genau entsprechend ein. Die Detailansicht soll auf einem UHD 8K Bildschirm mit 7680x4320px Bildschirm füllend, unverzerrt und ab horizontal 16 Grad (also mit negativem Einzug) in `id="Overscan"` erfolgen. Berechnen Sie dazu die `transform="matrix( )"` und tragen Sie die Parameter auf 6 Stellen genau entsprechend ein. (12 Punkte)

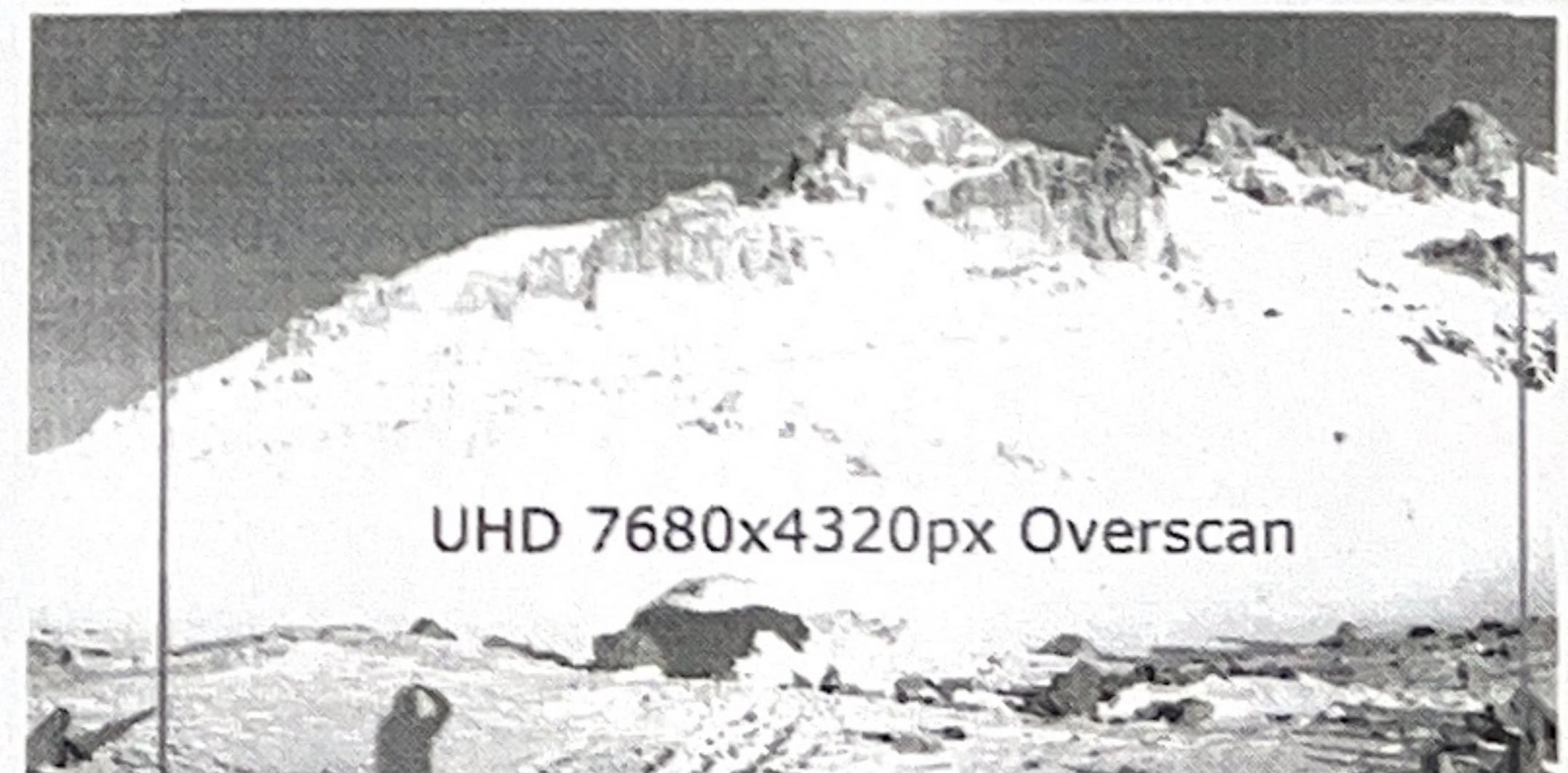
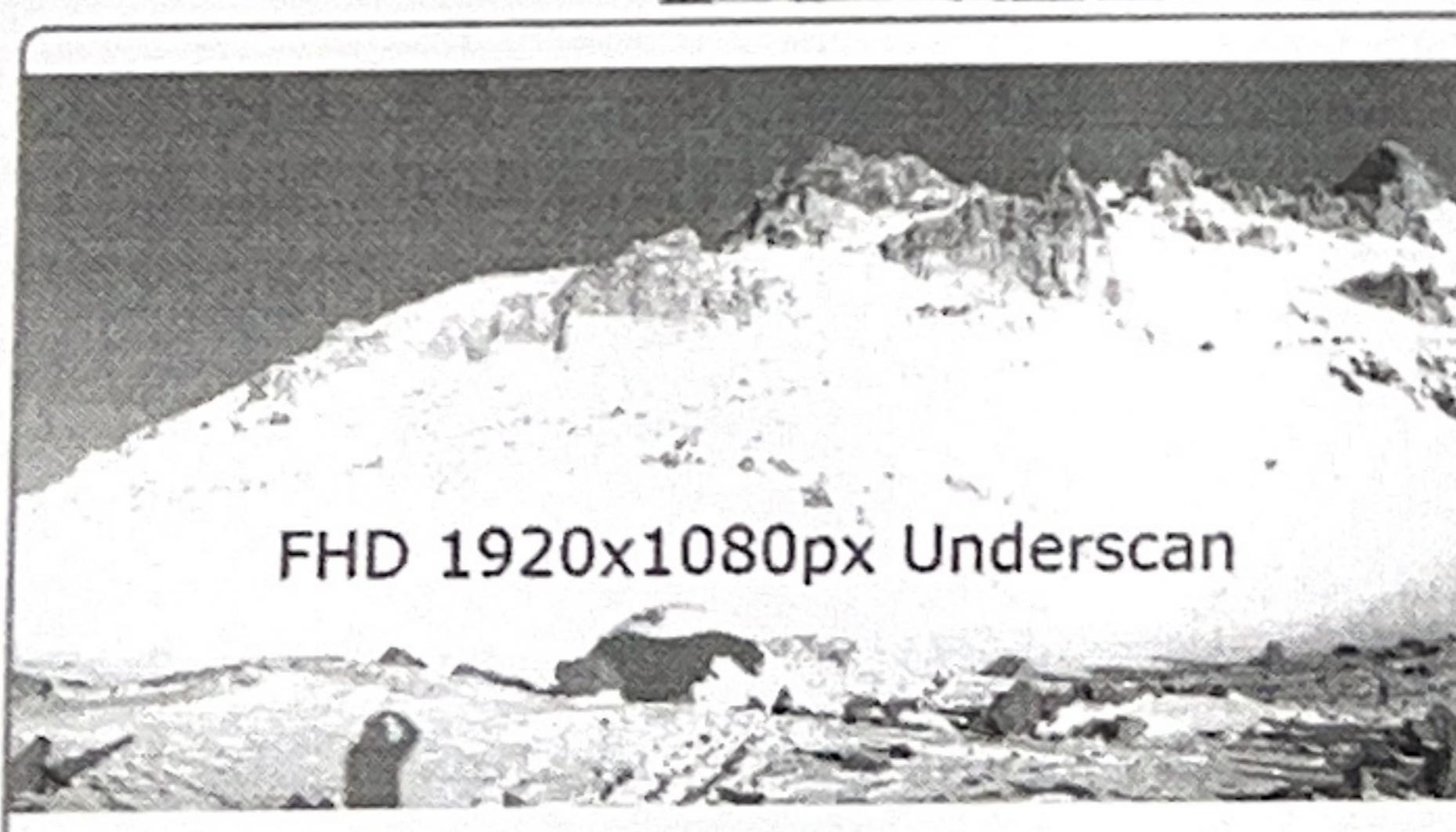
```
<svg id="Panorama" width="100%" height="100%"  
      viewBox="-20 -20 3880 2200" . . . >  
  
<defs>  
  <image id="Pano_Input_107MPix" x="0" y="0" width="14592" height="7296"  
    xlink:href="Panorama Aconcagua 6962m _180x90 Grad at 2 zu 1.png" />  
</defs>    <use id="Panorama_180x90_Grad_107MPix" . . . />  
  <text id="Text_Panorama" . . . > . . . </text>  
  
<g id="Underscan" > <use id="107MPix_in_FHD2K_Underscan" x="0" y="0"  
      xlink:href="#Pano_Input_107MPix"  
  
  transform="matrix(_____)"/>  
  
<rect id="FHD2K_1920x1080px" x="0" y="0" width="1920" height="1080"  
  rx="30" ry="30" /> </g> <text id="Text_Underscan" . . . > . . . </text>  
  
<g id="Overscan" > <use id="107MPix_in_UHD8K_Overscan" x="0" y="0"  
  xlink:href="#Pano_Input_107MPix"  
  
  transform="matrix(_____)"/>  
  
<rect id="UHD8K_7680x4320px" x="0" y="0" width="7680" height="4320"  
  rx="120" ry="120" /> </g> <text id="Text_Overscan" . . . > . . . </text>  
  . . . </svg>
```



Panorama 14592x7296px Original



Pano180d.SVG  
D. Herrmann, 01.03.2024



### Frage 6)

Alle visuellen Systeme (CRT, TFT, LCD, OLED, DSC, DVC, TV, Video, Print, UHD, Beamer, LED Displays, VR/AR) für den Menschen sind an sein Sehsystem (HVS – Human Visual System) angepasst. Die Grenzauflösung wird häufig mit 60 Pixeln pro Grad im Blickfeld oder Sehfeld (FoV – Field of View) dimensioniert. Die Funktion  $\tan(\Phi \text{ in } {}^\circ)$  ergibt sich aus dem Verhältnis Pixel-Größe oder Pixel Abstand (Pixel Pitch) und dem optimalen Betrachtungsabstand D (Viewing Distance). Die vertikalen Pixel ergeben die Zeilenzahl pro Bildhöhe H in cm. Die erforderliche Zeilenzahl für eine bestimmte Bildhöhe H [cm] und den vorgesehenen Betrachtungsabstand D [cm] lässt sich nach Umstellung der tan Funktion mit folgender Beziehung berechnen :  $Z = f(H, D) = 3437,75 * H / D$ .

Im Diagramm unten ist auf der X-Achse D von 0 cm bis 500 cm (5 m) gegeben. Auf der Y-Achsen ist die Zeilenzahl von 0 bis 5500 pro Bildhöhe H angegeben, beispielsweise 1080p für HDTV, 2160p für UHD 4K oder 4320p für UHD 8K. Die Bildhöhe H ist je nach Einsatz unterschiedlich, für PC Display oft unter 50 cm, Interaktiven Tafeln sind bis 150 cm hoch, und im Public Viewing, bei Info-Displays oder im Conferencing wird Life Size oder Raumhöhe bis 300 cm oder sogar mehr erreicht. In der Wertetabelle sind im Kopf 10 Abstände D gegeben, in der ersten Spalte ist der Parameter H mit 3 Werten gegeben. Tragen Sie die optimalen Zeilen entsprechend obiger Formel ein. Skizzieren Sie mit den Stützstellen die 3 Hyperbeln mit Beschriftung (H300, H150 und H50). (12 Punkte)

