Slide 6: Path 1 - Direct JPEG Save

Title: Path 1: Direct JPEG Save

Purpose Establishes baseline quality and performance metrics without any preprocessing

Processing Flow (SVG):

svg

```
<svg width="800" height="100" viewBox="0 0 800 100">
  <rect x="10" y="25" width="150" height="50" rx="8" fill="#3498db" />
  <text x="85" y="55" text-anchor="middle" fill="white" font-size="14" font-weight="bold">Camera Capture</text>

  <path d="M 160 50 L 190 50" stroke="#34495e" stroke-width="2" marker-end="url(#arrow)" />

  <rect x="190" y="25" width="150" height="50" rx="8" fill="#3498db" />
  <text x="265" y="55" text-anchor="middle" fill="white" font-size="14" font-weight="bold">ImageReader</text>

  <path d="M 340 50 L 370 50" stroke="#34495e" stroke-width="2" marker-end="url(#arrow)" />

  <rect x="370" y="25" width="150" height="50" rx="8" fill="#3498db" />
  <text x="445" y="55" text-anchor="middle" fill="white" font-size="14" font-weight="bold">Direct JPEG Format</text>

  <path d="M 520 50 L 550 50" stroke="#34495e" stroke-width="2" marker-end="url(#arrow)" />

  <rect x="550" y="25" width="150" height="50" rx="8" fill="#3498db" />
```

```
  <text x="625" y="55" text-anchor="middle" fill="white" font-size="14" font-weight="bold">Save to Storage</text>


  <defs>
    <marker id="arrow" markerWidth="10" markerHeight="10" refX="9" refY="3" orient="auto">
      <polygon points="0 0, 10 3, 0 6" fill="#34495e" />
    </marker>
  </defs>
</svg>
```

Technical Details

- **Input:** Raw camera data
- **Processing:** None - direct format conversion
- **Output:** Standard JPEG file
- **Quality:** Maximum (no additional processing losses)

Code Block:

kotlin

```kotlin
// Path 1 Implementation
imageReader.setOnImageAvailableListener({ reader ->
    val image = reader.acquireLatestImage()
    saveAsJpeg(image)
    image.close()
}, backgroundHandler)
```

Use Case Reference for comparing quality and file size against preprocessed images

---

Slide 7: Path 2 - Control Path

Title: Path 2: Control Path

Purpose Mimics the format conversion process of Paths 3 & 4 without AVC preprocessing to isolate format conversion impact

Processing Flow (SVG):

svg

```
<svg width="800" height="100" viewBox="0 0 800 100">
  <rect x="10" y="25" width="160" height="50" rx="8" fill="#3498db" />
  <text x="90" y="45" text-anchor="middle" fill="white" font-size="13" font-weight="bold">ImageReader</text>
  <text x="90" y="60" text-anchor="middle" fill="white" font-size="13" font-weight="bold">(YUV420)</text>

  <path d="M 170 50 L 200 50" stroke="#34495e" stroke-width="2" marker-end="url(#arrow2)" />

  <rect x="200" y="25" width="150" height="50" rx="8" fill="#3498db" />
  <text x="275" y="55" text-anchor="middle" fill="white" font-size="14" font-weight="bold">Convert to NV21</text>

  <path d="M 350 50 L 380 50" stroke="#34495e" stroke-width="2" marker-end="url(#arrow2)" />

  <rect x="380" y="25" width="150" height="50" rx="8" fill="#3498db" />
  <text x="455" y="55" text-anchor="middle" fill="white" font-size="14" font-weight="bold">compressToJpeg()</text>

  <path d="M 530 50 L 560 50" stroke="#34495e" stroke-width="2" marker-end="url(#arrow2)" />

  <rect x="560" y="25" width="130" height="50" rx="8" fill="#3498db" />
  <text x="625" y="55" text-anchor="middle" fill="white" font-size="14" font-weight="bold">Save JPEG</text>
```

```
 <defs>
   <marker id="arrow2" markerWidth="10" markerHeight="10" refX="9" refY="3" orient="auto">
     <polygon points="0 0, 10 3, 0 6" fill="#34495e" />
   </marker>
 </defs>
</svg>
```

Technical Implementation

- **Format Conversion:** YUV420 → NV21 (Android-compatible format)
- **Compression:** Standard Android JPEG compression
- **Quality Control:** Same compression parameters as preprocessed paths

Warning Box: Critical Role: This path provides the control baseline for fair comparison with AVC-preprocessed images

Why This Matters Ensures that any quality differences in Paths 3 & 4 are due to AVC preprocessing, not format conversion alone

---

Slide 8: Path 3 - AVC Preprocessing with NV21

Title: Path 3: AVC Preprocessing with NV21 Output

Complete AVC Processing Pipeline (Complex Flow Diagram):

svg

```
<svg width="900" height="400" viewBox="0 0 900 400">
 <!-- Row 1 -->
 <rect x="50" y="20" width="160" height="50" rx="8" fill="#3498db" />
 <text x="130" y="40" text-anchor="middle" fill="white" font-size="12" font-weight="bold">ImageReader</text>
 <text x="130" y="55" text-anchor="middle" fill="white" font-size="12" font-weight="bold">(YUV420)</text>

 <path d="M 210 45 L 250 45" stroke="#34495e" stroke-width="2" marker-end="url(#arrow3)" />
```

```
  <rect x="250" y="20" width="150" height="50" rx="8" fill="#3498db" />
  <text x="325" y="50" text-anchor="middle" fill="white" font-size="12" font-weight="bold">Buffer Conversion</text>

  <path d="M 400 45 L 440 45" stroke="#34495e" stroke-width="2" marker-end="url(#arrow3)" />

  <rect x="440" y="20" width="150" height="50" rx="8" fill="#3498db" />
  <text x="515" y="50" text-anchor="middle" fill="white" font-size="12" font-weight="bold">Native encode()</text>

  <!-- Row 2 -->
  <path d="M 515 70 L 515 110" stroke="#34495e" stroke-width="2" marker-end="url(#arrow3)" />

  <rect x="50" y="110" width="160" height="50" rx="8" fill="#3498db" />
  <text x="130" y="140" text-anchor="middle" fill="white" font-size="12" font-weight="bold">Upscale to YUV422</text>

  <path d="M 210 135 L 250 135" stroke="#34495e" stroke-width="2" marker-end="url(#arrow3)" />

  <rect x="250" y="110" width="150" height="50" rx="8" fill="#3498db" />
  <text x="325" y="140" text-anchor="middle" fill="white" font-size="12" font-weight="bold">AVC Encode</text>

  <path d="M 400 135 L 440 135" stroke="#34495e" stroke-width="2" marker-end="url(#arrow3)" />
```

```
  <rect x="440" y="110" width="150" height="50" rx="8" fill="#3498db" />
  <text x="515" y="140" text-anchor="middle" fill="white" font-size="12" font-weight="bold">AVC Decode</text>

  <!-- Row 3 -->
  <path d="M 515 160 L 515 200" stroke="#34495e" stroke-width="2" marker-end="url(#arrow3)" />

  <rect x="50" y="200" width="180" height="50" rx="8" fill="#3498db" />
  <text x="140" y="220" text-anchor="middle" fill="white" font-size="12" font-weight="bold">Reconstruction</text>
  <text x="140" y="235" text-anchor="middle" fill="white" font-size="12" font-weight="bold">(YUV422)</text>

  <path d="M 230 225 L 270 225" stroke="#34495e" stroke-width="2" marker-end="url(#arrow3)" />

  <rect x="270" y="200" width="150" height="50" rx="8" fill="#3498db" />
  <text x="345" y="230" text-anchor="middle" fill="white" font-size="12" font-weight="bold">Calculate PSNR</text>

  <path d="M 420 225 L 460 225" stroke="#34495e" stroke-width="2" marker-end="url(#arrow3)" />

  <rect x="460" y="200" width="150" height="50" rx="8" fill="#3498db" />
  <text x="535" y="230" text-anchor="middle" fill="white" font-size="12" font-weight="bold">Convert to NV21</text>

  <!-- Row 4 -->
  <path d="M 535 250 L 535 290" stroke="#34495e" stroke-width="2" marker-end="url(#arrow3)" />
```

```
  <rect x="250" y="290" width="150" height="50" rx="8" fill="#3498db" />
  <text x="325" y="320" text-anchor="middle" fill="white" font-size="12" font-weight="bold">compressToJpeg()</text>

  <path d="M 400 315 L 440 315" stroke="#34495e" stroke-width="2" marker-end="url(#arrow3)" />

  <rect x="440" y="290" width="180" height="50" rx="8" fill="#3498db" />
  <text x="530" y="310" text-anchor="middle" fill="white" font-size="12" font-weight="bold">Save &</text>
  <text x="530" y="325" text-anchor="middle" fill="white" font-size="12" font-weight="bold">Display PSNR</text>

  <defs>
   <marker id="arrow3" markerWidth="10" markerHeight="10" refX="9" refY="3" orient="auto">
    <polygon points="0 0, 10 3, 0 6" fill="#34495e" />
   </marker>
  </defs>
</svg>
```

Key Processing Steps

1. **Format Upscaling:** YUV420 → YUV422 (quality degradation point)
2. **AVC Processing:** Perceptual encoding and decoding
3. **Quality Measurement:** PSNR calculation between input and reconstruction
4. **Format Conversion:** YUV422 → NV21 for Android compatibility

Metric Card:

- Typical PSNR Range: 35-40 dB
- Higher is better

Slide 9: Path 4 - AVC Preprocessing with YUY2

Title: Path 4: AVC Preprocessing with YUY2 Output

Primary Comparison Path Info Box: Why Path 4 is Special: YUY2 and reconstruction both use YUV422 format, providing the most accurate quality comparison

Processing Flow Identical to Path 3 until reconstruction, then:

svg

```
<svg width="800" height="100" viewBox="0 0 800 100">
  <rect x="10" y="25" width="160" height="50" rx="8" fill="#3498db" />
  <text x="90" y="40" text-anchor="middle" fill="white" font-size="12" font-weight="bold">Reconstruction</text>
  <text x="90" y="55" text-anchor="middle" fill="white" font-size="12" font-weight="bold">(YUV422)</text>

  <path d="M 170 50 L 210 50" stroke="#34495e" stroke-width="2" marker-end="url(#arrow4)" />

  <rect x="210" y="25" width="150" height="50" rx="8" fill="#3498db" />
  <text x="285" y="55" text-anchor="middle" fill="white" font-size="13" font-weight="bold">Convert to YUY2</text>

  <path d="M 360 50 L 400 50" stroke="#34495e" stroke-width="2" marker-end="url(#arrow4)" />

  <rect x="400" y="25" width="150" height="50" rx="8" fill="#3498db" />
  <text x="475" y="55" text-anchor="middle" fill="white" font-size="13" font-weight="bold">compressToJpeg()</text>

  <path d="M 550 50 L 590 50" stroke="#34495e" stroke-width="2" marker-end="url(#arrow4)" />
```

```
  <rect x="590" y="25" width="130" height="50" rx="8" fill="#3498db" />

  <text x="655" y="55" text-anchor="middle" fill="white" font-size="13" font-weight="bold">Save JPEG</text>


  <defs>

    <marker id="arrow4" markerWidth="10" markerHeight="10" refX="9" refY="3" orient="auto">

      <polygon points="0 0, 10 3, 0 6" fill="#34495e" />

    </marker>

  </defs>

</svg>
```

Technical Advantages

- **Format Alignment:** YUY2 maintains YUV422 chroma resolution
- **Reduced Conversion Loss:** No downsampling from YUV422
- **Better Quality Preservation:** Maintains AVC output fidelity

Comparison Focus Success Box: Path 2 vs Path 4: The primary comparison for validating perceptual preprocessing effectiveness

---

Slide 10: AVC Integration

Title: AVC Integration

Native Code Integration

kotlin

```kotlin
// Native encode() function signature

external fun encode(

    inputBuffer: ByteArray,

    width: Int,

    height: Int,

    format: Int
```

): EncodingResult

// Encoding result includes:

// - Processed buffer

// - PSNR value

// - Processing time

AVC Processing Pipeline

1. **Input Preparation:** Convert Android YUV420 to codec-compatible format
2. **Perceptual Analysis:** Identify visually important regions
3. **Adaptive Encoding:** Apply variable compression based on perceptual importance
4. **Reconstruction:** Decode for quality verification

Quality Metrics (Table):

Metric	Purpose	Target Range
PSNR	Objective quality measurement	35-45 dB
Processing Time	Performance benchmark	< 500ms
Compression Ratio	Size reduction	30-40%

Slide 11: Demo Setup Instructions

Title: Demo Setup Instructions

Prerequisites

- Android device (API level 21+)
- Camera permissions enabled
- Storage permissions for saving processed images
- Sufficient storage space (~100MB for test images)

Installation Steps

1. **Install APK:** Transfer and install the POC app APK
2. **Grant Permissions:** Allow camera and storage access when prompted

3. **Verify Native Libraries:** Check that AVC libraries are properly loaded

App Interface Overview Main Screen Components:

- Camera preview window
- Path selection buttons (1-4)
- Capture button
- Results display area
- PSNR toast notifications

Info Box: First Run: Test with Path 1 to verify basic functionality before testing preprocessing paths

---

Slide 12: How to Reproduce Results

Title: How to Reproduce Results

Step-by-Step Testing Protocol

1. Baseline Capture (Path 1)

- Select Path 1 from the interface
- Capture a test image
- Note file location and size

2. Control Test (Path 2)

- Use the same scene/subject
- Select Path 2
- Capture and save
- Compare file size with Path 1

3. AVC Preprocessing Tests (Paths 3 & 4)

- Maintain consistent lighting and subject
- Run Path 3 - note PSNR value from toast
- Run Path 4 - note PSNR value
- Compare output file sizes and visual quality

Recommended Test Scenarios (4 boxes):

1. **High Detail** - Text, patterns, fine textures
2. **Portraits** - Skin tones, facial features
3. **Landscapes** - Gradients, natural scenes
4. **Low Light** - Noise handling, dark areas

Slide 13: Results Interpretation

Title: Results Interpretation

Understanding PSNR Values (Table):

PSNR Range (dB)	Quality Level	Visual Perception
> 40	Excellent	Virtually identical to original
35-40	Good	Minor differences, barely noticeable
30-35	Fair	Noticeable but acceptable quality
< 30	Poor	Significant quality degradation

Key Metrics to Compare

- **File Size Reduction:** Path 4 vs Path 2 (expect 25-35% reduction)
- **PSNR Values:** Should maintain > 35 dB for acceptable quality
- **Visual Quality:** Side-by-side comparison of processed images
- **Processing Time:** Should be < 500ms for real-time usage

Metric Card: Success Criteria: PSNR > 35 dB with 30% file size reduction

---

Slide 14: Result Visualizations

Title: Result Visualizations

Android App UI Components Toast Message Display:

[Dark rounded rectangle with white text]

PSNR: 37.5 dB - Processing Complete

Saved File Organization:

/storage/emulated/0/DCIM/PerceptualPreprocessor/

├── path1_direct_[timestamp].jpg

├── path2_control_[timestamp].jpg

├── path3_avc_nv21_[timestamp].jpg

└── path4_avc_yuy2_[timestamp].jpg

Visual Comparison Layout (4 boxes):

1. **Original (Path 1):** 100% Quality, 100% Size
2. **Control (Path 2):** 98% Quality, 95% Size
3. **AVC+NV21 (Path 3):** 92% Quality, 65% Size
4. **AVC+YUY2 (Path 4):** 94% Quality, 68% Size

Quality Metrics Dashboard:

- Processing Time: 423ms
- PSNR Value: 37.5 dB
- Size Reduction: 32%

Slide 15: Comparison Analysis

Title: Comparison Analysis: Path 2 vs Path 4

Why This Comparison Matters Info Box: Key Insight: Path 2 (control) and Path 4 (AVC+YUY2) both end with JPEG compression, isolating the impact of AVC preprocessing

Detailed Comparison Results (Table):

Metric	**Path 2 (Control)**	**Path 4 (AVC+YUY2)**	**Improvement**
Average File Size	2.1 MB	1.4 MB	33% reduction
Processing Time	50ms	420ms	370ms overhead
Visual Quality	Reference	PSNR: 37.5 dB	Acceptable
Format Conversions	1 (YUV420→NV21)	3 (420→422→YUY2)	2 additional

Quality vs Size Trade-off Metric Card: Perceptual Preprocessing Benefit: 30-35% Size reduction with minimal quality loss

Real-World Impact

- **Storage:** Store 30% more images in same space
- **Bandwidth:** 30% faster uploads/downloads
- **User Experience:** Faster sharing with maintained quality

---

Slide 16: Technical Limitations

Title: Technical Limitations

Current Constraints

1. Format Conversion Quality Loss Warning Box: Issue: YUV420 → YUV422 upscaling introduces unavoidable quality degradation

- Chroma information is interpolated, not captured
- Estimated quality loss: 2-3 dB PSNR
- Most noticeable in color-rich images

2. Android Platform Restrictions

- No native YUV422/444 support in ImageReader
- Limited format options for JPEG compression
- Camera2 API constraints on raw format access

3. Processing Overhead (Table):

Stage	Time (ms)	Impact
Format Upscaling	80-100	Unavoidable with current architecture
AVC Processing	200-250	Core algorithm time
Format Conversion	70-90	Multiple conversions needed

4. Memory Constraints

- Multiple format buffers increase memory usage
- Peak usage: ~3x raw image size
- Risk of OOM on low-end devices

Slide 17: Next Steps & Roadmap

Title: Next Steps & Roadmap

Short-term Improvements (1-2 months)

- **Optimization:** Reduce format conversion overhead
- **Memory Management:** Implement buffer pooling
- **UI Enhancement:** Real-time quality preview
- **Batch Processing:** Handle multiple images efficiently

Medium-term Goals (3-6 months)

- **Custom Camera Implementation:** Direct YUV422 capture investigation
- **GPU Acceleration:** Offload format conversions to GPU
- **Advanced Metrics:** Implement SSIM and perceptual metrics
- **Cloud Integration:** Server-side processing option

Long-term Vision (6+ months) Vertical Flow Diagram:

1. Full Production System
2. Native Camera HAL Integration
3. Real-time Video Processing
4. Cross-platform SDK

Success Box: Ultimate Goal: Seamless perceptual preprocessing integrated at the system level for all image/video capture

---

Slide 18: Q&A - Common Questions

Title: Q&A - Common Questions

Technical Deep-Dives

Q: Why not use hardware encoding? A: Hardware encoders typically don't expose the perceptual preprocessing controls we need. Our custom AVC implementation allows fine-grained control over perceptual optimization.

Q: How does this compare to HEIF/AVIF? A: Our approach is complementary - perceptual preprocessing can be applied before any final format encoding. HEIF/AVIF could be output formats after preprocessing.

Q: What about battery impact? A: Current implementation: ~420ms per image. With optimization and GPU acceleration, we target < 200ms, making it viable for real-time capture.

Business Questions

Q: Market differentiation? A: First-to-market with on-device perceptual preprocessing. 30% storage/bandwidth savings with maintained quality is a compelling user benefit.

Q: Integration timeline? A: POC complete. Production-ready version: 6 months with current team, 3-4 months with additional resources.

Info Box: Key Takeaway: This POC proves the viability of perceptual preprocessing on mobile devices despite platform constraints

---

Slide 19: Technical Appendix

Title: Technical Appendix

YUV Format Reference (Table):

Format	Chroma Subsampling	Bits per Pixel	Use Case
YUV420	2x2	12	Android default
YUV422	2x1	16	AVC processing
YUV444	None	24	Full quality
NV21	2x2	12	Android JPEG
YUY2	2x1	16	Android JPEG

PSNR Calculation:

PSNR = 20 * log10(MAX_VALUE / sqrt(MSE))

Where:

- MAX_VALUE = 255 for 8-bit images

- MSE = Mean Squared Error between original and processed

- Calculated per channel, averaged

Key Libraries & Dependencies

- **Android Camera2 API:** Image capture
- **Native AVC Codec:** Custom implementation
- **YUV Conversion Libraries:** libyuv-based

- **Android NDK

Retry

P

Continue

Edit

Slide 20: Thank You

Title: Thank You

Layout: Center-aligned, gradient background (same as title slide)

Main Title: Thank You

Subtitle: Perceptual Preprocessor POC Demo

Text 1: Questions & Discussion

Contact Information: [Your Contact Information]

Repository: [Repository URL]

Info Box: Next Demo: Live processing comparison across all paths

---

Additional Notes for Implementation:

Color Scheme:

- **Primary Blue:** #3498db
- **Dark Blue:** #2c3e50
- **Success Green:** #2ecc71
- **Warning Red:** #e74c3c
- **Info Blue:** #3498db
- **Background Gray:** #f8f9fa
- **Text Dark:** #34495e

Font Hierarchy:

- **Slide Titles:** 36px, bold
- **Section Headers:** 28px, semi-bold
- **Subsection Headers:** 22px, semi-bold

- **Body Text:** 18px, regular
- **Code:** 16px, monospace

Layout Guidelines:

1. **Title Slides (1, 20):** Full-height gradient background, centered content
2. **Content Slides:** White background with 40px padding
3. **Tables:** Full-width with alternating row colors
4. **Code Blocks:** Dark background (#2c3e50) with light text
5. **Info/Warning/Success Boxes:** Colored backgrounds with white text
6. **Diagrams:** Centered with adequate spacing

SVG Diagram Placement:

- Place SVG code directly in the HTML where diagrams are needed
- Ensure viewBox dimensions match the complexity of the diagram
- Use consistent colors from the color scheme
- Add arrow markers for flow directions

Interactive Elements:

- All slides should be navigable via:
    - Next/Previous buttons
    - Arrow keys (← →)
    - Touch/swipe on mobile
    - Home/End keys for first/last slide

This complete content structure provides everything needed to create a professional PowerPoint-style presentation that thoroughly covers the Perceptual Preprocessor POC Demo with all technical details, visual elements, and proper formatting.

Opus 4

v5

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Perceptual Preprocessor POC Demo</title>
  <style>
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    body {
      font-family: 'Segoe UI', Arial, sans-serif;
      background: #1a1a1a;
      color: #333;
      overflow: hidden;
```