CellVision: Microscopy Auto-Analyst with AI Narration

Buildathon Project Guide

🔬 Problem Statement

The Challenge

Cell biology researchers spend **5-10 hours per experiment** manually analyzing microscopy images - counting cells, measuring morphology, identifying phenotypes, and writing detailed figure legends for publications. This bottleneck slows research progress and introduces human error and inconsistency.

The Impact

- PhD students waste **30% of their time** on repetitive image analysis
- Manual cell counting varies **20-40% between observers**
- Writing publication-quality figure legends takes **30-60 minutes per image**
- Small labs without image analysis expertise cannot compete with well-funded institutions

Our Solution

CellVision democratizes expert-level microscopy analysis by combining computer vision with natural language AI. Upload any cell image and receive instant segmentation, quantitative metrics, and publication-ready figure legends generated by GPT-4 Vision - transforming a 2-hour manual task into a 30-second automated analysis.

🚀 Technical Roadmap (5 Hours)

Hour 0-1: Environment Setup & Model Testing

Installation Requirements

```bash
# Core installations
pip install cellpose numpy opencv-python streamlit
pip install openai pillow matplotlib scikit-image
```

Initial Model Test

```python
```

```python
# Download and test CellPose
from cellpose import models
import numpy as np
from skimage import io

# Initialize model (use CPU for reliability)
model = models.Cellpose(gpu=False, model_type='cyto2')

# Test on sample image
test_img = io.imread('sample_cells.png')
masks, flows, styles, diams = model.eval(test_img)
print(f"Detected {len(np.unique(masks))-1} cells")  # Verify it works
```

✅ **Deliverable**: Working CellPose installation with successful test segmentation

---

Hour 1-2: Core Analysis Pipeline

Main Analysis Function

```
python
```

```python
from skimage.measure import regionprops_table
import numpy as np

def analyze_microscopy_image(image_path):
    """
    Complete microscopy image analysis pipeline.

    Args:
        image_path: Path to microscopy image

    Returns:
        masks: Segmentation masks for each cell
        metrics: Dictionary of quantitative measurements
    """
    # 1. Load and preprocess
    img = io.imread(image_path)

    # 2. Cell segmentation with CellPose
    masks, _, _, diams = model.eval(img, diameter=30)

    # 3. Extract quantitative metrics
    cell_count = len(np.unique(masks)) - 1

    # 4. Morphology analysis
    props = regionprops_table(masks, img,
        properties=['area', 'perimeter', 'eccentricity', 'solidity'])

    # 5. Calculate statistics
    metrics = {
        'total_cells': cell_count,
        'avg_area': np.mean(props['area']) if len(props['area']) > 0 else 0,
        'avg_circularity': np.mean(4*np.pi*props['area']/props['perimeter']**2) if len(props['area']) > 0 else 0,
        'density': cell_count / (img.shape[0] * img.shape[1]),
        'size_variation': np.std(props['area']) / np.mean(props['area']) if len(props['area']) > 0 else 0
    }

    return masks, metrics
```

✅ **Deliverable**: Function that inputs image → outputs segmentation masks + metrics dictionary

---

Hour 2-3: GPT-4 Vision Integration

AI Narrative Generation

python

```python
import base64
from openai import OpenAI
import matplotlib.pyplot as plt
from skimage import io

def generate_analysis_narrative(image_path, masks, metrics, api_key):
    """
    Generate publication-quality figure legend using GPT-4 Vision.

    Args:
        image_path: Path to original image
        masks: Segmentation masks from CellPose
        metrics: Quantitative metrics dictionary
        api_key: Azure OpenAI API key

    Returns:
        str: Publication-ready figure legend
    """
    # 1. Create visualization
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))
    ax1.imshow(io.imread(image_path))
    ax1.set_title('Original')
    ax1.axis('off')
    ax2.imshow(masks, cmap='tab20')
    ax2.set_title(f'Segmented: {metrics["total_cells"]} cells')
    ax2.axis('off')
    plt.savefig('analysis.png', dpi=100, bbox_inches='tight')
    plt.close()

    # 2. Encode image for GPT-4 Vision
    with open('analysis.png', 'rb') as f:
        img_base64 = base64.b64encode(f.read()).decode('utf-8')

    # 3. Call GPT-4 Vision with metrics context
    client = OpenAI(api_key=api_key)

    prompt = f"""You are an expert cell biologist. Analyze this microscopy image.

    Quantitative metrics detected:
    - Cell count: {metrics['total_cells']}
    - Average cell area: {metrics['avg_area']:.1f} pixels²
    - Cell density: {metrics['density']:.4f} cells/pixel²
    - Size variation coefficient: {metrics['size_variation']:.2f}
```

```
    - Average circularity: {metrics['avg_circularity']:.2f}

    Generate a publication-quality figure legend describing:
    1. Cell morphology and distribution patterns
    2. Notable features or abnormalities
    3. Quantitative summary with the provided statistics

    Format as: "Figure X: [Comprehensive scientific description]"
    Keep it concise but scientifically precise.
    """

    response = client.chat.completions.create(
        model="gpt-4-vision-preview",
        messages=[{
            "role": "user",
            "content": [
                {"type": "text", "text": prompt},
                {"type": "image_url", "image_url": {
                    "url": f"data:image/png;base64,{img_base64}"
                }}
            ]
        }],
        max_tokens=500
    )

    return response.choices[0].message.content
```

✅ **Deliverable**: Function that generates natural language analysis from image + metrics

---

Hour 3-4: Streamlit Interface

Complete Web Application

```python
```

```python
import streamlit as st
import matplotlib.pyplot as plt
from datetime import datetime
import os

def main():
    # Page config
    st.set_page_config(
        page_title="CellVision",
        page_icon="🔬",
        layout="wide"
    )

    # Header
    st.title("🔬 CellVision: AI-Powered Microscopy Analysis")
    st.markdown("Transform microscopy images into quantitative insights and publication-ready descriptions in seconds.")

    # Sidebar for API key
    with st.sidebar:
        st.header("⚙️ Configuration")
        api_key = st.text_input("OpenAI API Key", type="password")
        st.markdown("---")
        st.markdown("### 📊 Sample Images")
        if st.button("Load Cancer Cell Example"):
            # Load pre-saved example
            pass

    # Main content
    col1, col2 = st.columns([1, 1])

    with col1:
        st.subheader("📤 Upload Image")
        uploaded_file = st.file_uploader(
            "Choose a microscopy image",
            type=['png', 'jpg', 'jpeg', 'tif', 'tiff'],
            help="Supported formats: PNG, JPEG, TIFF"
        )

        if uploaded_file:
            # Save uploaded file
            temp_path = f"temp_{datetime.now().timestamp()}.png"
            with open(temp_path, "wb") as f:
                f.write(uploaded_file.getbuffer())
```

```python
        # Display original
        st.image(temp_path, caption="Original Image", use_column_width=True)

    with col2:
        st.subheader("🔍 Analysis Results")

        if uploaded_file and st.button("🚀 Analyze Image", type="primary"):
            if not api_key:
                st.error("Please enter your OpenAI API key in the sidebar.")
            else:
                # Progress indicator
                progress_bar = st.progress(0)
                status_text = st.empty()

                # Step 1: Segmentation
                status_text.text("Segmenting cells...")
                progress_bar.progress(33)
                masks, metrics = analyze_microscopy_image(temp_path)

                # Display segmentation
                fig, ax = plt.subplots(figsize=(8, 8))
                ax.imshow(masks, cmap='tab20')
                ax.set_title(f"Detected {metrics['total_cells']} cells")
                ax.axis('off')
                st.pyplot(fig)
                plt.close()

                # Step 2: Metrics
                status_text.text("Calculating metrics...")
                progress_bar.progress(66)

                # Display metrics in columns
                metric_cols = st.columns(4)
                metric_cols[0].metric("Cell Count", metrics['total_cells'])
                metric_cols[1].metric("Avg Area", f"{metrics['avg_area']:.0f} px²")
                metric_cols[2].metric("Density", f"{metrics['density']:.4f}")
                metric_cols[3].metric("Circularity", f"{metrics['avg_circularity']:.2f}")

                # Step 3: AI Analysis
                status_text.text("Generating AI analysis...")
                progress_bar.progress(90)

                narrative = generate_analysis_narrative(temp_path, masks, metrics, api_key)
```

```
        # Complete
        progress_bar.progress(100)
        status_text.text("Analysis complete!")

        # Display narrative
        st.markdown("### 📝 Publication-Ready Figure Legend")
        st.info(narrative)

        # Download button
        st.download_button(
            label="📥 Download Analysis Report",
            data=narrative,
            file_name=f"figure_legend_{datetime.now().strftime('%Y%m%d_%H%M%S')}.txt",
            mime="text/plain"
        )

        # Cleanup
        os.remove(temp_path)

if __name__ == "__main__":
    main()
```

✅ **Deliverable**: Complete working web interface with upload → analyze → download flow

---

Hour 4-5: Testing, Polish & Demo Prep

Testing Checklist

- [ ] Test with 5 different cell types from LIVECell dataset
- [ ] Verify cell counts within 10% of manual annotation
- [ ] Ensure GPT-4 narratives are scientifically accurate
- [ ] Test edge cases: dense clusters, low contrast, artifacts
- [ ] Verify error handling for corrupted images
- [ ] Test with different image sizes (512x512 to 2048x2048)

Polish Tasks

```
python
```

```python
# Add these enhancements if time permits:

def add_batch_processing():
    """Allow multiple image upload and analysis"""
    uploaded_files = st.file_uploader("Choose images", accept_multiple_files=True)
    if uploaded_files:
        for file in uploaded_files:
            # Process each file
            pass

def export_pdf_report():
    """Generate comprehensive PDF report with all results"""
    from reportlab.lib.pagesizes import letter
    from reportlab.pdfgen import canvas
    # Create PDF with images, metrics, and narrative
    pass

def add_comparison_mode():
    """Side-by-side comparison of two conditions"""
    col1, col2 = st.columns(2)
    # Allow comparison of treated vs untreated cells
    pass
```

Demo Data Preparation

1. **Download test images**:
   - Healthy epithelial cells
   - Cancer cells (HeLa)
   - Drug-treated cells showing apoptosis
2. **Create demo narrative**:
   - "Watch CellVision identify apoptotic cells invisible to the untrained eye"
   - Show 3-step progression: healthy → cancer → treated
3. **Backup plan**:
   - Save screenshots of each step
   - Pre-generate example outputs
   - Have PDF report ready to show

👥 Team Task Assignments

Person 1 - Data Curator

- [ ] Download 20 diverse cell images from LIVECell dataset
- [ ] Organize by cell type (A172, BT474, BV2, Huh7, MCF7, etc.)
- [ ] Create metadata spreadsheet with image properties
- [ ] Select 3 best images for demo

Person 2 - Validator

- [ ] Manually count cells in 5 test images
- [ ] Create ground truth spreadsheet
- [ ] Calculate accuracy metrics
- [ ] Document edge cases where algorithm struggles

Person 3 - UX Designer

- [ ] Create Figma mockup of interface
- [ ] Design CellVision logo
- [ ] Choose color scheme (suggest: teal/purple for biotech feel)
- [ ] Create slide deck template (5 slides max)

Person 4 - Demo Narrator

- [ ] Write 90-second pitch script
- [ ] Prepare problem → solution → impact storyline
- [ ] Create slide deck content:
  - Slide 1: Problem (30 sec)
  - Slide 2: Solution (30 sec)
  - Slide 3: Live Demo (60 sec)
  - Slide 4: Impact & Next Steps (20 sec)
- [ ] Practice transitions between speakers

📊 Success Metrics

Technical Goals

- ✅ Process any microscopy image in **<30 seconds**
- ✅ Cell count accuracy **within 90%** of manual annotation
- ✅ Generate coherent, scientific figure legends
- ✅ Support images from **512x512 to 2048x2048** pixels

- ✅ Handle **PNG, JPEG, and TIFF** formats

Demo Goals

- ✅ Live demo works flawlessly with 3 different cell types
- ✅ Clear value proposition: **2 hours → 30 seconds**
- ✅ Show quantitative improvement in consistency
- ✅ Generate "wow" moment with AI-generated narrative

🎯 Winning Pitch Framework

Opening Hook (15 seconds)

"Every day, 50,000 biology PhD students spend 2 hours counting cells by hand, clicking on dots one by one. Meanwhile, critical drug discoveries are delayed because small labs can't afford $100K image analysis software."

Problem Validation (15 seconds)

"Manual cell counting varies 40% between researchers. One image takes 30 minutes to analyze properly. Publication-quality figure legends take another 30 minutes to write."

Solution Demo (60 seconds)

1. **Upload** cancer cell image (5 sec)
2. **Show** instant segmentation with 234 cells detected (10 sec)
3. **Display** quantitative metrics dashboard (10 sec)
4. **Reveal** AI-generated figure legend (20 sec)
5. **Compare** to manual analysis time/accuracy (15 sec)

Impact Statement (20 seconds)

"CellVision democratizes advanced microscopy analysis. Any researcher with a phone camera microscope can now produce Nature-quality image analysis. We transform a 2-hour expert task into 30 seconds of automated precision."

Ask & Next Steps (10 seconds)

"We're seeking clinical research partners to validate CellVision across 10 cell types and apply for NIH SBIR funding to build the definitive microscopy analysis platform."

🚨 Common Pitfalls to Avoid

1. **Don't oversell accuracy** - Be honest about 90% accuracy vs 100%

2. **Don't ignore edge cases** - Acknowledge when cells are too dense/overlapping
3. **Don't skip validation** - Always compare to ground truth
4. **Don't forget citations** - Credit CellPose and LIVECell dataset
5. **Don't overcomplicate** - Simple, working demo beats complex, buggy features

---

📚 Resources & References

- **CellPose Documentation**: https://cellpose.readthedocs.io/
- **LIVECell Dataset**: https://sartorius-research.github.io/LIVECell/
- **OpenAI Vision API**: https://platform.openai.com/docs/guides/vision
- **Streamlit Docs**: https://docs.streamlit.io/
- **Sample Microscopy Images**: https://bbbc.broadinstitute.org/

🏆 Why This Wins

1. **Visual Impact**: Transformation is immediately visible and impressive
2. **Real Problem**: Every biology lab faces this challenge daily
3. **Technical Innovation**: First to combine segmentation + GPT-4 Vision narration
4. **Feasible in 5 Hours**: Core functionality proven in 2 hours, polish in remaining 3
5. **Democratizes Science**: Levels playing field between large and small labs
6. **Clear Metrics**: 2 hours → 30 seconds, 40% variation → 10% variation

Remember: Judges want innovation and out-of-the-box thinking. CellVision delivers both while solving a real, painful problem that every biologist understands.