TV Position Detection Fix

Problem Summary

Issue: TVs were being created with labels (TV 01, TV 02, etc.) but positions were NOT being set correctly based on where TVs appear in uploaded layout images.

Root Cause:
- The system was using hardcoded position logic based on text descriptions
- No actual AI vision analysis was being performed on uploaded images
- The `extractPositionFromWall()` function generated positions based on wall descriptions, not actual image coordinates
- Images were uploaded and stored but never analyzed by AI vision APIs

Solution

New AI Vision Integration

Created a new vision analysis API (`/api/ai/vision-analyze-layout`) that:

1. **Uses Real AI Vision**: Integrates OpenAI GPT-4 Vision or Anthropic Claude Vision to analyze actual layout images
2. **Detects TV Positions**: Identifies TV markers/numbers in the image and calculates accurate x/y coordinates
3. **Returns Precise Coordinates**: Provides percentage-based positions (0-100% from top-left) for each TV
4. **Fallback Support**: Gracefully falls back to grid positioning if no API keys are configured

Changes Made

1. New Vision Analysis API

File: `src/app/api/ai/vision-analyze-layout/route.ts`

- Analyzes uploaded layout images using AI vision
- Detects TV positions with accurate x/y coordinates
- Supports both OpenAI GPT-4 Vision and Anthropic Claude Vision
- Returns structured data with TV numbers, positions, and confidence scores

2. Updated Layout Analysis API

File: `src/app/api/ai/analyze-layout/route.ts`

- Now calls the vision API when an image URL is provided
- Converts vision detections to the existing TVLocation format
- Falls back to description parsing if vision analysis fails
- Added `determineWallFromPosition()` helper function

3. Dependencies Added

- `openai` - OpenAI API client for GPT-4 Vision
- `@anthropic-ai/sdk` - Anthropic API client for Claude Vision

Configuration

API Keys Setup

To enable real AI vision analysis, configure API keys in `.env`:

```
# Option 1: OpenAI GPT-4 Vision (Recommended)
OPENAI_API_KEY="sk-proj-..."

# Option 2: Anthropic Claude Vision (Alternative)
ANTHROPIC_API_KEY="sk-ant-..."

# You only need ONE of the above - the system will try OpenAI first, then Anthropic
```

Getting API Keys

OpenAI API Key

1. Go to https://platform.openai.com/api-keys
2. Create a new API key
3. Copy the key (starts with `sk-proj-...`)
4. Add to `.env` as `OPENAI_API_KEY`
5. Ensure you have GPT-4 Vision access (may require paid account)

Anthropic API Key

1. Go to https://console.anthropic.com/settings/keys
2. Create a new API key
3. Copy the key (starts with `sk-ant-...`)
4. Add to `.env` as `ANTHROPIC_API_KEY`
5. Claude 3.5 Sonnet has vision capabilities

Without API Keys (Fallback Mode)

If no API keys are configured, the system will:
- Use a grid-based fallback positioning system
- Create 25 TVs in a 5x5 grid layout
- Set confidence to 50% to indicate fallback mode
- Display a warning that AI vision is not configured

How It Works

With AI Vision (API Keys Configured)

1. **Upload**: User uploads layout image (PNG, JPG, or PDF)
2. **Vision Analysis**: Image is sent to OpenAI/Anthropic vision API
3. **Detection**: AI detects TV markers/numbers and their positions
4. **Coordinate Calculation**: AI calculates x/y percentages for each TV
5. **Mapping**: System maps detected TVs to Wolfpack outputs
6. **Display**: TVs are positioned accurately on the layout

Vision API Prompt

The AI is instructed to:
- Look for numbered markers, TV icons, screen symbols, or labeled positions

- Calculate positions as percentages from top-left corner (x: 0-100%, y: 0-100%)
- Identify TV numbers from labels like “TV 1”, “1”, “Marker 1”, etc.
- Provide confidence scores (90-100 for clear, 70-89 for partial, <70 for uncertain)
- Return structured JSON with all detections

Example Vision Response

```
{
  "totalTVs": 25,
  "imageWidth": 1920,
  "imageHeight": 1080,
  "detections": [
    {
      "number": 1,
      "label": "TV 1",
      "position": {
        "x": 15.5,
        "y": 22.3
      },
      "confidence": 95,
      "description": "Located on left wall, upper section"
    },
    {
      "number": 2,
      "label": "TV 2",
      "position": {
        "x": 15.2,
        "y": 38.7
      },
      "confidence": 92,
      "description": "Located on left wall, middle section"
    }
    // ... 23 more TVs
  ]
}
```

Testing

Test with User’s Layout

The user’s layout image is at: `/home/ubuntu/Uploads/Graystone Layout.png`

To test the fix:

1. **Configure API Keys** (see Configuration section above)
2. **Start the Development Server**:
   `bash`
   `cd /home/ubuntu/github_repos/Sports-Bar-TV-Controller`
   `npm run dev`
3. **Upload the Layout**:
   - Navigate to the Layout Configuration page
   - Upload the Graystone Layout.png image
   - The system will automatically trigger vision analysis

4. **Verify Results**:
   - Check that 25 TVs are detected
   - Verify positions match the actual layout image
   - Confirm TVs are placed where they appear in the image

Manual API Test

You can test the vision API directly:

```
curl -X POST http://localhost:3000/api/ai/vision-analyze-layout \
  -H "Content-Type: application/json" \
  -d '{
    "imageUrl": "/uploads/layouts/your-layout.png"
  }'
```

Deployment

Prerequisites

- Node.js 18+ installed
- npm or yarn package manager
- OpenAI or Anthropic API key (for vision analysis)

Deployment Steps

1. **Pull the Latest Changes**:
   `bash`
   `cd ~/Sports-Bar-TV-Controller`
   `git pull origin main`

2. **Install Dependencies**:
   `bash`
   `npm install`

3. **Configure API Keys**:
   ```bash
   # Edit .env file
   nano .env

# Add your API key:
OPENAI_API_KEY=”sk-proj-your-key-here”
# OR
ANTHROPIC_API_KEY=”sk-ant-your-key-here”
```

1. **Build the Application**:
   `bash`
   `npm run build`

2. **Restart the Server**:
   ```bash
   # If using PM2:
   pm2 restart sports-bar-tv-controller
   ```

```
# If using systemd:
sudo systemctl restart sports-bar-tv-controller

# Or start manually:
npm start
```

1. **Verify the Fix**:
   - Upload a layout image
   - Check that TVs are positioned correctly
   - Verify all 25 TVs are created with accurate positions

Backward Compatibility

The fix maintains full backward compatibility:

- **With API Keys**: Uses new AI vision analysis for accurate positioning
- **Without API Keys**: Falls back to grid-based positioning (similar to old behavior)
- **Existing Layouts**: Continue to work without changes
- **API Interface**: No breaking changes to existing API endpoints

Performance Considerations

API Costs

- **OpenAI GPT-4 Vision**: ~$0.01-0.03 per image analysis
- **Anthropic Claude Vision**: ~$0.01-0.02 per image analysis
- Analysis is only performed once per layout upload

Response Times

- **Vision Analysis**: 3-8 seconds (depends on image size and API)
- **Fallback Mode**: <100ms (instant grid generation)
- **Caching**: Consider caching results for frequently used layouts

Rate Limits

- **OpenAI**: 500 requests/day (Tier 1), 10,000/day (Tier 2+)
- **Anthropic**: 1,000 requests/day (free tier), higher for paid
- System automatically falls back if rate limits are hit

Troubleshooting

Issue: “No AI vision API keys configured”

Solution: Add `OPENAI_API_KEY` or `ANTHROPIC_API_KEY` to `.env` file

Issue: “Vision API failed, falling back to description parsing”

Possible Causes:
- Invalid API key
- Rate limit exceeded
- Network connectivity issues
- Image format not supported

Solution: Check logs for specific error, verify API key, check rate limits

Issue: TVs still not positioned correctly

Possible Causes:
- Image quality too low
- TV markers not clearly visible
- Numbers/labels not readable

Solution:
- Use higher resolution images (300 DPI recommended)
- Ensure TV markers are clearly visible
- Use clear numbering/labeling

Issue: Only some TVs detected

Possible Causes:
- Some markers obscured or unclear
- Inconsistent labeling
- Image cropping

Solution:
- Verify all TV markers are visible in the image
- Use consistent numbering (1, 2, 3... or TV 1, TV 2, TV 3...)
- Ensure full layout is captured in the image

Future Enhancements

Potential improvements for future versions:

1. **Image Preprocessing**: Enhance image quality before analysis
2. **Manual Position Adjustment**: Allow users to fine-tune AI-detected positions
3. **Batch Analysis**: Analyze multiple layouts simultaneously
4. **Position Validation**: Verify positions don't overlap
5. **Custom Detection Models**: Train custom models for specific layout types
6. **Caching**: Cache vision analysis results to reduce API costs
7. **Alternative Vision APIs**: Support Google Vision, Azure Computer Vision

Related Issues

- **PR #145**: Fixed issue where only 12 outputs were created instead of 25
- **Current PR**: Fixes TV position detection to use actual image coordinates

Support

For issues or questions:
1. Check the troubleshooting section above
2. Review server logs for detailed error messages
3. Verify API key configuration
4. Test with the fallback mode first
5. Contact the development team with specific error details