Dataset Cleaning Instructions

Overview

This document provides step-by-step instructions for cleaning the basketball shooting form dataset using the smart filter. The dataset contains **19,494 images** that need to be filtered to meet the exact user requirements.

Prerequisites

- ✅ Python 3.8+ installed
- ✅ MediaPipe installed (`pip install mediapipe`)
- ✅ OpenCV installed (`pip install opencv-python`)
- ✅ Training data located at `/home/ubuntu/basketball_app/training_data/`
- ✅ Sufficient disk space for quarantine directory (~10GB recommended)

Quick Start

Option 1: Dry Run (Recommended First)

Test the filter without moving any files:

```
cd /home/ubuntu/basketball_app
python clean_dataset.py --dry-run
```

This will:
- Analyze all images
- Generate statistics
- Create report
- **NOT move any files**

Option 2: Full Cleaning (Run as Background Job)

Due to the large dataset (19,494 images), this will take **several hours**. Run as a background job:

```
cd /home/ubuntu/basketball_app
nohup python clean_dataset.py > dataset_cleaning.log 2>&1 &
```

Monitor progress:

```
# Check if still running
ps aux | grep clean_dataset.py

# Monitor log in real-time
tail -f dataset_cleaning.log

# Check progress from log
grep "Progress:" dataset_cleaning.log | tail -1
```

Understanding the Smart Filter

What Makes an Image “Good”?

The smart filter accepts images that meet **ALL** of these criteria:

1. ✅ **Single player as main subject** (head to toe visible)
2. ✅ **Shooting motion detected** (arms raised, elbow angle > 90°)
3. ✅ **Full body visible** (head, shoulders, hips, knees, feet)
4. ✅ **NOT dribbling** (ball above waist, arm not extended downward)
5. ✅ **NOT layup** (vertical alignment, not running)
6. ✅ **Player is main object** (largest + reasonably centered)

Key Feature: Background People Are OK!

IMPORTANT: The smart filter allows other people in the frame AS LONG AS one player is clearly the main shooting subject. This is a critical difference from previous filters.

User’s Specification:

> “with no other players or distractions, **except in cases where other players are present in the scene**; in such cases, center the focus on the designated player for analysis”

Expected Results

Acceptance Rate

Based on test runs, expect:

- **Total Images:** 19,494
- **Expected Accepted:** 1,000 - 2,500 (5-15%)
- **Expected Rejected:** 17,000 - 18,500 (85-95%)

Common Rejection Reasons

Reason	Expected %	Description
Dribbling Motion	40-50%	Ball below waist, arm extended downward
Partial Body	20-30%	Missing head, feet, or other body parts
Multiple Shooters	10-20%	Unclear which player is the focus
No People Detected	5-10%	MediaPipe couldn’t detect any poses
Processing Errors	5-10%	Image load failures, corrupt files

Directory Structure After Cleaning

```
basketball_app/
├── training_data/                     # KEPT images (accepted)
│   ├── image1.jpg
│   ├── image2.jpg
│   └── ...
│
├── training_data_quarantine/          # REJECTED images (by reason)
│   ├── no_people_detected/
│   ├── partial_body/
│   ├── dribbling_motion/
│   ├── not_shooting/
│   ├── arm_position_unclear/
│   ├── processing_error/
│   └── failed_to_load/
│
└── dataset_cleaning_reports/          # Detailed reports
    ├── cleaning_report_20251213_HHMMSS.json
    ├── cleaning_report_20251213_HHMMSS.md
    └── ...
```

Detailed Command Options

Basic Usage

```
python clean_dataset.py [OPTIONS]
```

Options

Option	Description	Default
`--dry-run`	Analyze without moving files	False
`--training-dir PATH`	Training data directory	`/home/ubuntu/basketball_app/training_data`
`--quarantine-dir PATH`	Quarantine directory	`/home/ubuntu/basketball_app/training_data_quarantine`
`--report-dir PATH`	Report directory	`/home/ubuntu/basketball_app/dataset_cleaning_reports`

Examples

Dry run:

```
python clean_dataset.py --dry-run
```

Custom directories:

```
python clean_dataset.py \
  --training-dir /path/to/images \
  --quarantine-dir /path/to/quarantine \
  --report-dir /path/to/reports
```

Background job with custom paths:

```
nohup python clean_dataset.py \
  --training-dir /custom/path/training_data \
  --quarantine-dir /custom/path/quarantine \
  > cleaning.log 2>&1 &
```

Monitoring Progress

Real-Time Monitoring

```
# Watch progress
tail -f dataset_cleaning.log

# Check acceptance rate
grep "ACCEPTED\|REJECTED" dataset_cleaning.log | tail -20

# Count processed images
grep "Progress:" dataset_cleaning.log | tail -1
```

Check Status

```
# Is the process still running?
ps aux | grep clean_dataset.py

# How many images in quarantine?
find training_data_quarantine -type f | wc -l

# How many images remaining in training_data?
find training_data -type f | wc -l
```

Estimated Time

Based on MediaPipe performance:

- **Processing Speed:** ~2-3 images/second (varies by image size)
- **Total Images:** 19,494
- **Estimated Time: 2-3 hours** (with model_complexity=1)

After Cleaning Complete

1. Review the Report

```
# View latest report
cat DATASET_CLEANING_REPORT.md

# Or open in browser
firefox /home/ubuntu/basketball_app/DATASET_CLEANING_REPORT.md
```

2. Verify Statistics

Check the report for:

- ✅ Acceptance rate (should be 5-15%)
- ✅ Rejection reasons breakdown
- ✅ Example images in each category
- ✅ Processing errors (should be < 5%)

3. Spot-Check Accepted Images

```
# View random accepted images
cd training_data
ls *.jpg | shuf -n 10 | xargs -I {} python ../smart_shooting_form_filter.py {}
```

4. Review Quarantine

```
# Check quarantine categories
ls -lh training_data_quarantine/

# View images in each category
cd training_data_quarantine/dribbling_motion
ls *.jpg | head -5
```

5. Manually Review Edge Cases

If acceptance rate is too low (< 5%) or too high (> 20%), manually review:

- **Too Low:** Check if filter is too strict
- Review `arm_position_unclear/` category
- Check `partial_body/` for false rejections
- **Too High:** Check if filter is too lenient
- Spot-check random accepted images
- Verify no dribbling/layup images accepted

Troubleshooting

Issue 1: Process Killed/Out of Memory

Symptom: Process terminates unexpectedly

Solution: Reduce MediaPipe model complexity

```
# Edit smart_shooting_form_filter.py
# Change: model_complexity=1 to model_complexity=0
```

Issue 2: Very Slow Processing

Symptom: < 1 image/second

Solution:
1. Use model_complexity=0 for faster processing
2. Process in batches instead of all at once
3. Use a machine with GPU support

Issue 3: Too Many Rejections (> 95%)

Symptom: Very few images accepted

Possible Causes:
1. Training data contains mostly dribbling/layup images
2. Filter is too strict
3. Images are low quality/cropped

Solution:
1. Review a sample of rejected images
2. If false rejections, adjust filter thresholds in `smart_shooting_form_filter.py`:
`python`
```
   self.SHOOTING_ELBOW_ANGLE_MIN = 80  # Lower threshold
   self.MIN_VISIBILITY = 0.4            # Lower visibility requirement
```

Issue 4: Too Few Rejections (< 50%)

Symptom: Most images accepted

Possible Causes:
1. Dataset already contains mostly good images
2. Filter is too lenient

Solution:
1. Spot-check random accepted images
2. If false acceptances, increase strictness:
`python`
```
   self.SHOOTING_ELBOW_ANGLE_MIN = 100  # Higher threshold
   self.MIN_VISIBILITY = 0.6             # Higher visibility requirement
```

Configuration Tuning

Filter Thresholds

Edit `smart_shooting_form_filter.py`:

```
# Line ~60-70
self.SHOOTING_ELBOW_ANGLE_MIN = 90  # Arm raised (shooting)
self.DRIBBLING_ELBOW_ANGLE_MAX = 80  # Arm lowered (dribbling)
self.MIN_VISIBILITY = 0.5            # Landmark visibility
self.MIN_BOX_AREA = 0.10             # Subject size in frame
self.CENTER_TOLERANCE = 0.3          # How centered subject should be
```

MediaPipe Settings

```
# Line ~50-55
model_complexity=1,             # 0=fastest, 1=balanced, 2=most accurate
min_detection_confidence=0.5,   # Lower = more detections
min_tracking_confidence=0.5     # Lower = more tracking
```

Recovery and Rollback

If Cleaning Goes Wrong

Restore from Quarantine:

```
# Move all quarantined images back to training_data
cd training_data_quarantine
find . -name "*.jpg" -exec mv {} ../training_data/ \;
```

Start Over:

```
# Delete quarantine
rm -rf training_data_quarantine

# Run dry-run first
python clean_dataset.py --dry-run
```

Best Practices

1. Always Do Dry Run First

```
python clean_dataset.py --dry-run
```

Review the report before committing to actual file moves.

2. Keep Backups

```
# Backup training_data before cleaning
cp -r training_data training_data_backup
```

3. Run in Stages

For very large datasets, process in batches:

```
# Process first 5000 images
python clean_dataset.py --training-dir training_data_batch1

# Review results

# Process next batch
python clean_dataset.py --training-dir training_data_batch2
```

4. Monitor Disk Space

```
# Check available space
df -h /home/ubuntu/basketball_app

# Space needed: ~2x training_data size
```

5. Document Your Settings

Keep notes on:
- Filter thresholds used
- Acceptance rate achieved
- Any manual adjustments made

Integration with Basketball App

Update App Configuration

After cleaning, update app config files:

```
# In config file
TRAINING_DATA_DIR = "/home/ubuntu/basketball_app/training_data"
IMAGE_COUNT = 2500  # Update with actual accepted count
LAST_CLEANED = "2025-12-13"
```

Verify App Compatibility

```
# Test app with cleaned dataset
cd basketball_app
python test_dataset_loading.py
```

Additional Resources

- **Image Requirements:** See `IMAGE_REQUIREMENTS.md`
- **Visual Guide:** Open `basketball_test_results/IMAGE_REQUIREMENTS_VISUAL_GUIDE.html`
- **Test Results:** Open `basketball_test_results/TEST_RESULTS_VIEWER.html`
- **Filter Code:** Review `smart_shooting_form_filter.py`
- **Config:** Check `config_image_requirements.py`

Support

For issues or questions:

1. Review `IMAGE_REQUIREMENTS.md` for exact specifications
2. Check `DATASET_CLEANING_REPORT.md` for statistics
3. Manually review quarantined images
4. Adjust filter thresholds if needed
5. Re-run with adjusted settings

Checklist: Before Running Full Clean

- [ ] Reviewed image requirements (`IMAGE_REQUIREMENTS.md`)
- [ ] Ran dry-run and reviewed report
- [ ] Backed up training_data directory
- [ ] Confirmed sufficient disk space (10GB+)
- [ ] Tested filter on sample images
- [ ] Set realistic expectations (5-15% acceptance rate)
- [ ] Prepared to run as background job (2-3 hours)
- [ ] Know how to monitor progress (`tail -f dataset_cleaning.log`)

Checklist: After Cleaning Complete

- [ ] Reviewed cleaning report (`DATASET_CLEANING_REPORT.md`)
- [ ] Verified acceptance rate (5-15%)
- [ ] Spot-checked 50-100 accepted images
- [ ] Reviewed rejection reasons breakdown
- [ ] Manually inspected edge cases in quarantine
- [ ] Updated app configuration with new image count
- [ ] Tested app with cleaned dataset
- [ ] Committed changes to version control
- [ ] Documented any threshold adjustments made

Last Updated: December 13, 2025
Version: 1.0
Filter: SmartShootingFormFilter v1.0