# DETECTING SNAP POINTS IN EGOCENTRIC VIDEO WITH A WEB PHOTO PRIOR

Marc Carné Herrera

# Introduction

- Wearable cameras -> egocentric vision, first person vision.
- Capturing every 30 seconds = 1.000-1.400 images/day

*“What happens when that user’s camera is always on, worn at eye-level, and has the potential to capture everything he sees throughout the day?”*

- Camera follows approximately wearer’s activity and gaze.

# Any problem?

- Many problems!!
  - Non-informative frames or uninteresting content
  - Blurred frames
  - Poorly composed
  - Very similar frames and a lot of them (static activities)
  - Accidental images (images captured when the user isn’t wear the camera).
  - Retrieval paradigm

# Snap point concept

- Egocentric image → non-intentional taken, objectives frames.
- Snap point → frame that has been taken with intentionally.

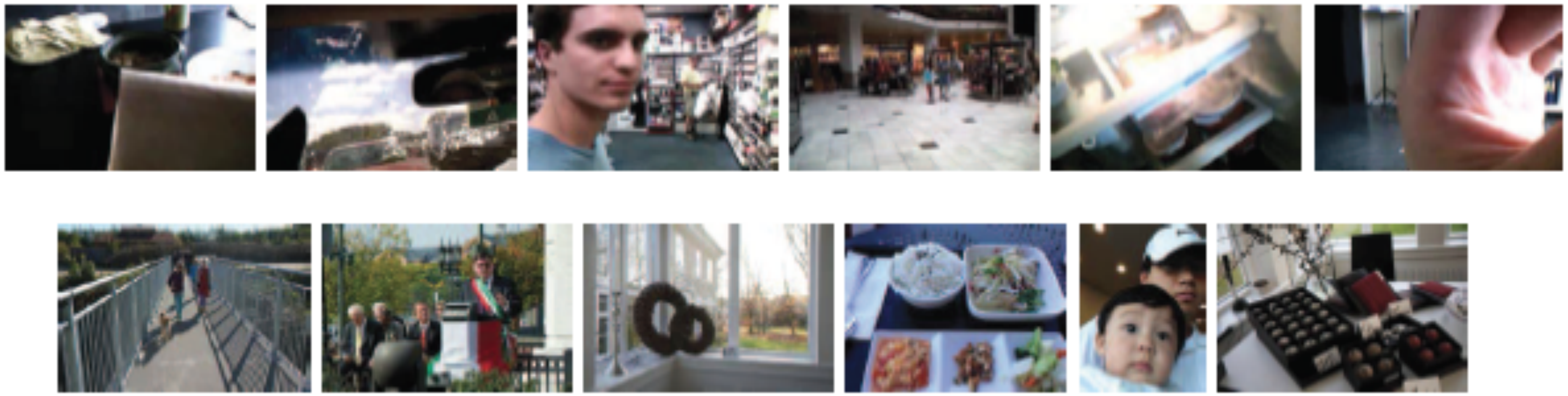


Fig. 1. Can you tell which row of photos came from an egocentric camera?

# Egocentric videos value

- Contains a wide variety of scene types, activities and actors.

Fig. 3. Example images from the SUN dataset [42].

# Objective

- We want to find well-composed images, snap points, egocentric images (non-intentionally taken) that look like intentionally ones.
- The difference between egocentric vs. normal images suggest that it's possible to learn generic properties of a well-composed images.

# Snap point detector

1. Large domain invariant and generalization across many subjects.
2. Optimal snap point is likely to differ in subtle ways from its less-good temporal neighbors.
   - Simuilar in content buy different in snap point quality.
3. Manual annotation needed, users may specify which frames look like intentional

# Approach

- Generative model that detects snap point from egocentric video without human annotation.
- Snap points example:
  - Social networks → people tend to upload images that vary vastly and are completly intentionaly taken.
- Egocentric images ≠ normal images
  - Adaptation needed, due to mismatch between visual features.
- Two applications:
  - Object detection → actual approaches accuracy drops because models trained with human-taken photos not generalize well.
  - Keyframe selection from egocentric video.

# Approach

- Goal → detect snap points
- Camera-user → trigger = human
- Steps:
  1. Learn how snap points look like.
  2. Estimate a domain-invariant feature connecting Web and egocentric images.
  3. Given a novel egocentric video frame, predict the snap point score.
- Also, explore applications of the approach.

# Snap points features

- Web photos → extract image descriptors that capture cues for composition and intention.
- Features of web images:
  - Human-taken
  - Variety of contexts
  - Element of intention
- Dataset → SUN Database
  - 899 categories
  - 70K WordNet terms
  - Can be well-matched with wearable camera data

# Image descriptors

- Descriptors that capture intentional composition:
  - Motion → non-snap points occurs when camera wearer is moving quickly.
    - Descriptor: motion blur.
  - Composition → spatial regularity aligned to the image's axis
      - Horizon in an outdoor photo
      - Buildings in the street
      - Tables in a restaurant
    - Line aligment desciptors
  - Feature combination → reduce dimensionallity with PCA + standardize each dimension + concatenate reduced vectors.

# Adapting from Web to Egocentric

- Mismatch between statistics of this domains.
  - Web = high resolution and high quality.
  - Egocentric = low resolution, low quality lenses…


- Domain-invariant feature connecting both spaces:
  - First, create a common space obtained with PCA processing.
  - Then, intermediate subspaces that transforms gradually from Web subspace to egocentric subspace.

$$K_{GFK}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{z}_i^{\infty}, \boldsymbol{z}_j^{\infty} \rangle = \int_0^1 (\phi(t)^T \boldsymbol{x}_i)^T (\phi(t)^T \boldsymbol{x}_j) dt,$$

- $x_i$ → Web image descriptor
- $x_j$ → Egocentric frame descriptor
- Compute the projection of $x_i$ on superspace ø(t):
  - 't' belong to 0-1 interval.
    - 0 = subspace closer to Web prior.
    - 1 = subspace similar to egocentric frames.
- Kernel → computes similarity between Web image to Egocentric image.

# Predicting snap points

- Web prior + image features + similarity measure.
- Simple data-driven approach.
- Estimate likelihood of the novel egocentric frame and Web prior images → K-NN.
- Algorithm:
  - W → set of Web images features.
  - $K_{GFK}$ → similarity
  - Find the ‘k’ highest values of kernel and sum = $S(x^e)$
  - The $S(x^e)$ higher means that the egocentric image is a snap point

# Advantages

- Label-free → all training examples are positive, Web image.

# Applications

- Object detection:
  - Detectors trained on one dataset tends to generalize poorly to another.
  - Use the predicted snap points for detection.
- Keyframe selection:
  - To create keyframes summaries of egocentris video.
  - Simple selection strategy.
  - Algorithm:
    - Identify clusters (same images, scene, physical location).
    - For each event select the frame most confidently scored as a snap point.

# Dataset and GT for snap points

- UT Egocentric → four videos of 3-5 hour each, captured with head-mounted camera, people doing daily life activities.
- Mobile robot dataset → newly collected by wheely robot for this project.

# Dataset features

- This datasets are incidentally captured video from always-on, dynamic cameras and uscripted activity.
- There are other datasets that are not useful due to their focus on a controlled environment and limited activity.

# “Magic camera” scenario MTurk

- “Suppose you are creating a **visual dairy** out of photos. You have a **portable camera** that you carry all day long, in order to **capture everyday moments** of your life… Unfortunately, your magic camera can also trigger itself from time to time to take **random pictures**, evem while you are holding the camera. At the end of the day, all pictures, both the ones you took **intentionally** and the ones **accidentally** taken by the camera, are **mixed together.**”

- <u>**Task**</u>: distinguish the pictures that you toom intentionally from the resto of pictures that were accidentally taken by your camera.

# Mturk task

- For each photo users have to choose a category:
  a) Very confident intentional
  b) Somewhat confident intentional
  c) Somewhat confident accidental
  d) Very confident accidental

- **Each image** → labeled by 5 users (avoid ambiguity and subjective).

# Results

- No existing method that perform snap points, we have to use baselines:
  - Saliency → CRF-based saliency to score an image.
    - People tend to compose images with a salient object in the center.

- Blurriness → blur estimates (no-reference perceptual blur metric) to score and image.
  - Intentionally taken images tend to lack motion blur.

- People likelihood → person detector to score an image by how lokely is to contain persons.
  - People tend to take images with their families = meaningful moments.

- Discriminative SVM → RBF kernel SVM trained with snap/non-snap samples.
  - Requieres more training effort than our appoach.

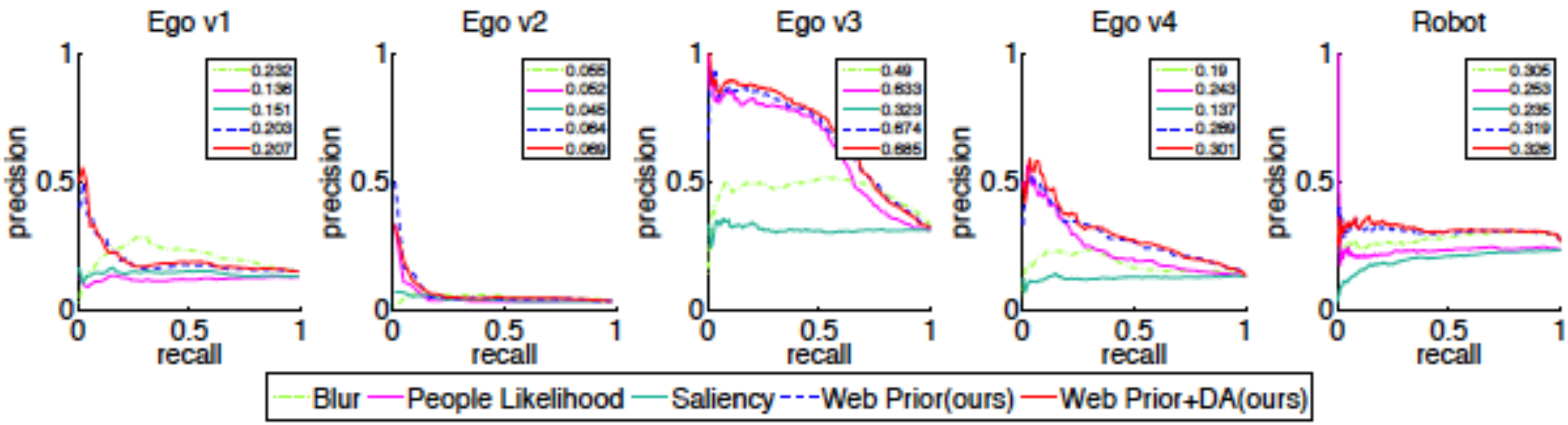
# Snap points accuracy

- How accurately the method predict snap points.
- Compare → precision-recall curves for this method and the 3 state-of-the-art techniques.

- Proposed approach → always outperforms all methods
  - Ego v1 → blur is similar
  - Ego v2 → mAP low for all methods.
  - Ego v3 → likelihood have high perform due to nice portraits.
  - Ego v4 → x2 nearest competing baseline (blur).

- SUN Web prior → less close-up object-centris photos.

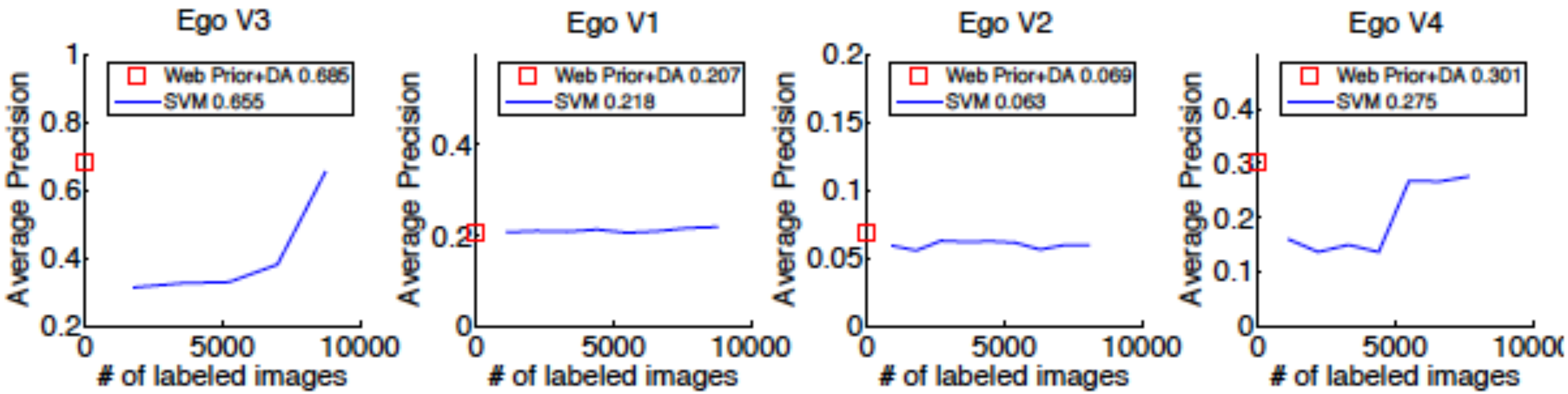
- Any baseline requiere labels (except for discriminative SVM classifier).




**Despite learning without labels, this approach outperform SVM.**

| Methods | Ego v1 | | Ego v2 | | Ego v3 | | Ego v4 | | Robot | |
|---|---|---|---|---|---|---|---|---|---|---|
| rank coefficient | $\rho$ | $\tau$ | $\rho$ | $\tau$ | $\rho$ | $\tau$ | $\rho$ | $\tau$ | $\rho$ | $\tau$ |
| Blurriness | **0.347** | **0.249** | 0.136 | 0.094 | 0.479 | 0.334 | 0.2342 | 0.162 | 0.508 | 0.352 |
| People Likelihood | 0.002 | 0 | -0.015 | -0.011 | 0.409 | 0.289 | 0.190 | 0.131 | 0.198 | 0.134 |
| Saliency | 0.027 | 0.019 | 0.008 | 0.005 | 0.016 | 0.011 | -0.021 | -0.014 | -0.086 | -0.058 |
| Web Prior (Ours) | 0.321 | 0.223 | 0.144 | 0.100 | **0.504** | **0.355** | **0.452** | 0.317 | 0.530 | 0.373 |
| Web Prior+DA (Ours) | 0.343 | 0.239 | **0.179** | **0.124** | 0.501 | 0.353 | **0.452** | **0.318** | **0.537** | **0.379** |

# Supervised vs. unsupervised

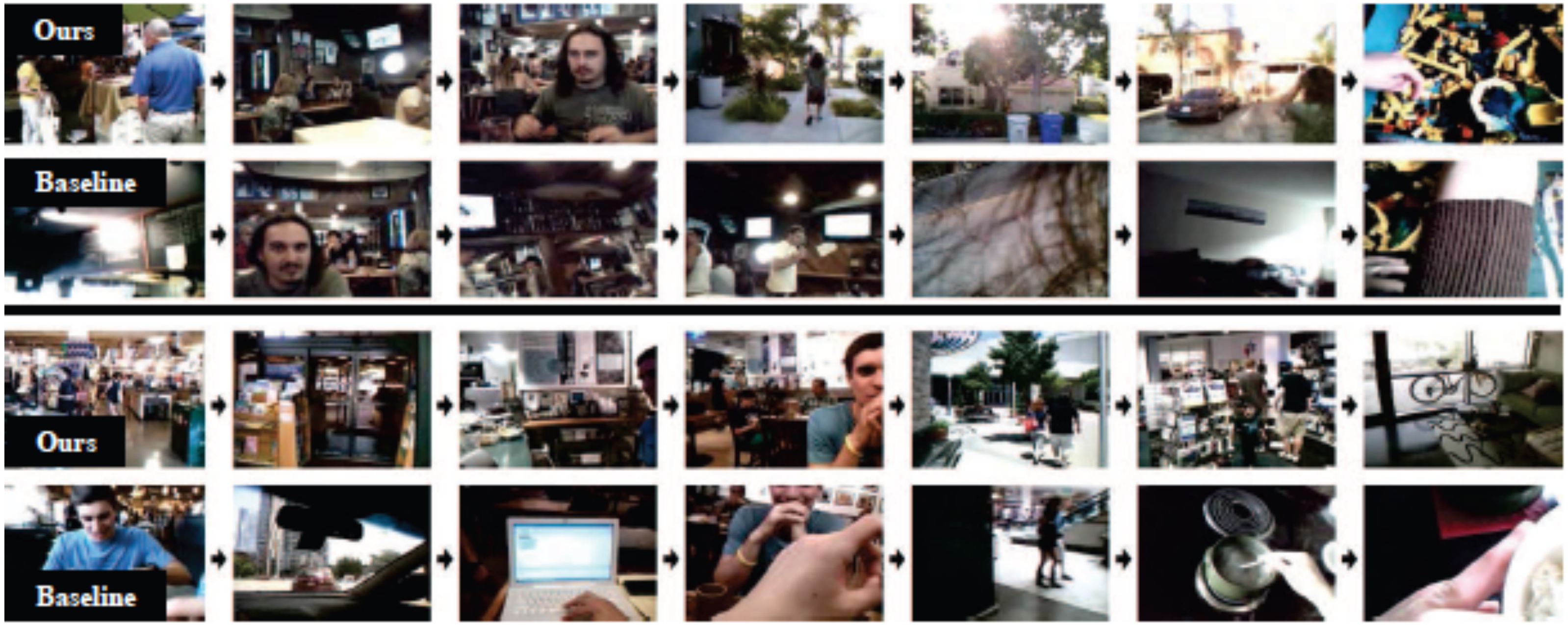


# Object detector APP

- A system trained with human-taken only fit for 10% or 15% egocentric images.
- GT for egocentric → DrawMe
  - 1000 labeled bounding boxes for people
  - 200 labeled bounding boxes for car

# Keyframe selection APP

- Appealing way to peruse long egocentric videos.
- Keyframe → gist of what was seen.
- Compare:
  ① Cluster + most kernel value → contains well-construct images
  ② Cluster + random selection



# Conclusions

① Long egocentric videos
② Automated method for intelligently filtering the data of great interest
③ Tranfer existing visual recognition methods to the ego domain
④ Approach → visual information + no manual labels
⑤ May be run online for egocentric videos → **preprocess**
⑥ Bottleneck → feature extraction