Clustering Media Items Stemming from Multiple Social Networks

THOMAS STEINER[1], RUBEN VERBORGH[2], JOAQUIM GABARRO[1], ERIK MANNENS[2] AND RIK VAN DE WALLE[2]

[1] *Universitat Politècnica de Catalunya – Department LSI, Spain*
[2] *Ghent University – iMinds – Multimedia Lab, Belgium*
Email: tsteiner@lsi.upc.edu

We have created and evaluated an algorithm capable of deduplicating and clustering exact- and near-duplicate media items that get published and shared on multiple social networks in the context of events. This algorithm works in an entirely *ad-hoc* manner, without any pre-calculation. When people attend events, they more and more share event-related media items publicly on social networks to let their social network contacts relive and witness the attended events. In the past, we have worked on methods to accumulate such public user-generated multimedia content in order to summarize events visually, for example, in the form of media galleries or slideshows. In this paper, first, we introduce social-network-specific reasons and challenges that cause near-duplicate media items. Second, we detail an algorithm for the task of deduplicating and clustering exact- and near-duplicate media items stemming from multiple social networks. Finally, we evaluate the algorithm's strengths and weaknesses and show ways to address the weaknesses efficiently.

Keywords: clustering, deduplication, social networks, media items, event summarization, media galleries, slideshows

Received 14 February 2013; revised 23 July 2013

1. INTRODUCTION

1.1. Motivation and Previous Work

Mobile devices like smartphones, tablets, or digital cameras together with social networks enable people to create, share, and consume enormous amounts of media items like photos and videos. Mobile devices are omnipresent at all sorts of events, where—given a stable network connection—part of the event-related media items are published on social networks, both as the event happens and afterwards, once a stable network connection has been re-established.

In the past, we have developed and evaluated an application and related methods for media item enrichment [1, 2, 3, 4] to provide a scalable and near-realtime solution that realizes event summarization and media item compilation in form of media galleries. For any event with given event *title(s)*, (potentially vague) event *location(s)*, and (arbitrarily fine-grained) event *date(s)*, with our approach, we first extract binary media item data from social networks or media item hosting platforms. Second, we deduplicate exact and near-duplicate media items to then cluster similar media items for the ultimate goal of generating media galleries or slideshows. In this paper, we focus on the media item clustering and deduplication task in the context of multiple social networks, which comes with its very specific challenges that we will detail and motivate in Section 2.

1.2. Definitions

Camera Shot Boundary Detection In video production and filmmaking, a shot is a series of frames that runs for an uninterrupted period of time. Shots are always filmed with a single camera and can be of any duration. Camera shot boundary detection is a field of research of video processing.

Social Media Item We define a social media item as either a photo (image) or video that was *publicly* shared or published on at least one social network. In the following, we will use the shorter term media item rather than the full term.

Exact and Near-Duplicates for Photos We define two media items of type photo as *exact-duplicates*, if their pixel contents are exactly the same. We define two media items of type photo as *near-duplicates*, if their pixel contents differ no more than a given threshold after resampling.

Exact and Near-Duplicates for Videos We define two media items of type video as *exact-duplicates*, if their pixel contents are frame by frame exactly the same. We define two media items of type video as *near-duplicates*, if their pixel contents per frame differ no more than a given threshold. In practice, we lower this condition and instead of every frame consider only frames at camera shot boundaries [5], which accelerates the process without reducing the precision noticeably. We make *no* requirements on the audio, *i.e.*, two videos in two different languages that fulfill the pixel contents equality

condition are considered exact-duplicates. Note that we also do not consider video subsegments near-duplicates, so a media fragment [6] is considered distinct.

Special Case of Photo Contained in a Video We define the special case of *a photo being contained in a video*, if the pixel contents of a photo differ no more than a given threshold from the pixel contents of any one of the frames of a video. In practice, we lower this condition and instead of every frame consider only frames at camera shot boundaries.

1.3. Paper Structure

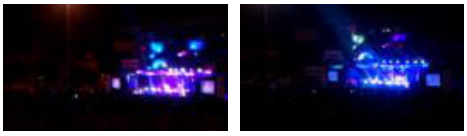
Section 2 motivates the chosen approach in the context of how users interact with, modify, and consume media items in social networks. Section 3 details our clustering algorithm's design goals, its high- and low-level matching conditions, and its implementation. In Section 4, we introduce the events used for our experiments to then evaluate the clustering algorithm and discuss the algorithm's strengths and weaknesses. In Section 5, we outline how the algorithm can be used for video clustering. Related work in the field of image and video clustering is covered in Section 6. The paper ends with an outlook on future work and a conclusion in Section 7.

2. PROBLEM STATEMENT

Our work is situated in the broader context of summarizing events based on social network data. In order to get an overview of a given event based on a *potentially huge* set of event-related media items, this set of media items needs to be *pruned* to exclusively contain highly relevant media items that are as representative for the event as possible. Rather than showing the viewer all media items, clusters of similar media items need to be formed. Within each cluster, the most representative media item has to be recognized as such, according to well-defined criteria. Undesired duplicate or near-duplicate content in the context of social networks arises in a number of situations that we will illustrate in the following. We highlight that the stated problem is distinct from the general image and video clustering problem, where features such as Scale-Invariant Feature Transform (SIFT) [7] excel. All shown examples of media items below were actually shared on social networks and were clustered correctly as near-duplicates by our clustering algorithm, which we will detail in Section 3.

Different Viewing Angle When two people attend the same event and create media items at roughly the same time and covering the same scene, their media items will be similar and—the capturing devices' qualities aside—only differ in the viewing angle. Figure 1 shows a concrete example.

Logo, Watermark, Lower Third, or Caption Insertion Oftentimes, organizations or individuals insert logos, watermarks, lower thirds, or captions into media items to highlight their origin, to convey related information, or to claim ownership of a media item. An example of caption, logo, and lower third insertion can be seen in Figure 2.



(a) Viewing angle 1 (b) Viewing angle 2

FIGURE 1: Slightly different viewing angles of a concert stage

(a) Original (b) Caption (c) Lower third

FIGURE 2: Caption, logo, and lower third insertion for a speaker

(a) Original (b) Cropped

FIGURE 3: Original and cropped version of a media item

Cropping Cropping refers to the removal of the outer parts of a media item to improve framing, accentuate subject matter, or to (lossily) change the aspect ratio. Cropping either happens manually via an image editing application, or, more often, by the social networks themselves to obtain a square aspect ratio that better fits the timeline view of users, as can be seen in the example in Figure 3.

Different Keyframes We have shown an approach to camera shot boundary detection in [5]. Different frames stemming from the same camera shot can occur on social networks, when preview heuristics attempt to auto-select a well-identifying poster frame from a video with different approaches, typically resulting in different frames for different social networks. Figure 4 shows an example of this phenomenon.

Aspect Ratio Changes with Squeezing or Stretching Aspect ratio changes can either happen combined with cropping (and thus losing parts of the media item), and/or combined with squeezing or stretching (and thus deforming the media item). Figure 5 shows an example where a media item gets stretched.

Photo Filters With the raising popularity of Instagram with its 90 million monthly active users,[3] photo filters that, *e.g.*, emulate retro Polaroid™ or tilt-shift effects, are a considerable reason for near-duplicate media content on social networks. Figure 6 shows a typical example.

[3] `http://instagram.com/press/`, accessed 04/30/2013

(a) Frame 1 (b) Frame 2

FIGURE 4: Two different frames from the same camera shot

(a) Original (b) Stretched

FIGURE 5: Original and stretched version of a media item

(a) Original (b) Photo filter

FIGURE 6: Original photo and version with an applied filter

3. MEDIA ITEM CLUSTERING ALGORITHM

3.1. Algorithm Design Goals

In the previous section, we have outlined reasons and sources for duplicate and near-duplicate content. In this section, we describe an algorithm tailored to deduplicating and clustering exact-duplicate and near-duplicate media items. Design goals for the algorithm include the capability to detect exact-duplicate and near-duplicate media items in a timely, entirely *ad-hoc* manner, without any pre-calculation. In general—and especially for big events—event coverage on social networks is very broad, *i.e.*, there exist more media items than one could consume in a reasonable time. In consequence, it is tolerable for the algorithm to cluster media items aggressively, rather than leaving too many media items unclustered. The algorithm has a twofold approach to clustering: *low-level* analysis, by looking at tile-wise pixel data combined with *high-level* analysis, by detecting faces in media items. In the following, we describe the face detection component of our media item clustering algorithm.

3.2. Face Detection

Face detection is a computer vision technology that determines the regions of faces in media items. Rotation-invariant face detection aims to detect faces with arbitrary rotation angles and is crucial as the first step in automatic face detection for general applications, as face images on social media are seldom upright and frontal. Face detection is a subclass of the broader class of object detection. The Viola–Jones object detection framework proposed in 2001 by Paul Viola and Michael Jones [8, 9] provides competitive object detection rates in realtime and was motivated primarily by the problem of face detection. In the future, this can also enable more advanced features like face image retrieval [10] or face recognition [11]. We use an algorithm that further improves Viola–Jones, based on work by Huang *et al.* [12] and Abramson *et al.* [13], in a JavaScript implementation made available by Liu [14]. This algorithm is fast enough to be applied to hundreds of media items in well less than a second overall processing time on a standard laptop (mid-2010 MacBook Pro).

3.3. Algorithm Description

Our near-duplicate media item clustering algorithm belongs to the family of tile-wise histogram-based clustering algorithms. As an additional semantic feature, the algorithm considers detected faces as described above. For two media items to be clustered, the following conditions have to be fulfilled.

1. Out of m tiles of a media item with n tiles ($m \leq n$), at most *tiles_threshold* tiles may differ not more than *similarity_threshold* from their counterpart tiles.
2. The numbers f_1 and f_2 of detected faces in both media items have to be the same. We note that we do not *recognize* faces, but only *detect* them.

The simplified algorithm pseudocode can be seen in Listing 1. In the actual implementation, some speed improvements, like, for example, looking up already calculated distances[4] have been applied; these were omitted in the listing for legibility reasons. We calculate the histograms and distances only once initially. The clusters are then recalculated dynamically whenever either *tiles_threshold* or *similarity_threshold* change. The given values of $rows = cols = 10$ and $tiles\_threshold = 67 = \lceil rows \cdot cols \cdot 2/3 \rceil$ and $similarity\_threshold = 10$ were determined empirically on a large corpus of event-related media items and are known to deliver solid results. The corpus has been made available publicly, see Subsection 4.2 for the details.

3.4. Algorithm Debug View

In order to illustrate the way the algorithm clusters media items, Figure 7 shows a debug view of the algorithm for two clustered media items related to the *Grammy Awards Nominations 2013* event. The red border around the media item indicates at least one detected face. Independent from the actual media item's aspect ratio, the tile-wise comparison always happens based on a potentially squeezed square aspect ratio version. The two slightly different media items (caption insertion, lighting change) were clustered, because out of the $10 \cdot 10 = 100$ tiles, 85 of the minimum required *tiles_threshold* of 67 tiles differed not more than the *similarity_threshold* of 10 per tile. In both media items,

[4] `distances[outerItem][innerItem] = distances[innerItem][outerItem]`

```
Input: mediaItems, a list of media items
Output: clusters, a list of clustered media items

init:


# Algorithm settings
ROWS = 10
COLS = 10
TILES_THRESHOLD = ceil(ROWS * COLS * 2/3)
SIMILARITY_THRESHOLD = 10


# Calculates tile-wise histograms
histograms = {}
faces = {}
for item in mediaItems
  faces[item] = getFaces(item)

  histograms[item] = {}
  for tile in item
    histograms[item][tile] = getHistogram(tile)
  end for
end for


# Calculates tile-wise distances
distances = {}
for outerItem in mediaItems
  distances[outerItem] = {}
  for innerItem in mediaItems
    distances[outerItem][innerItem] = {}
    for tile in histograms[outerItem]
      distances[outerItem][innerItem][tile] =
          abs(histograms[outerItem][tile] -
              histograms[innerItem][tile])
    end for
  end for
end for


# Calculates clusters
clusters = {}
for outerItem in mediaItems
  clusters[outerItem] = []
  for innerItem in mediaItems
    if outerItem == innerItem then continue

    similarTiles = 0
    distance = distances[outerItem][innerItem]
    for tile in distance
      if distance[tile] <= SIMILARITY_THRESHOLD then
        similarTiles++
      end if
    end for

    # Check condition 1 (tiles)
    if similarTiles >= TILES_THRESHOLD then
      # Check condition 2 (faces)
      if faces[outerItem] == faces[innerItem] then
        clusters[outerItem].push(innerItem)
      end if
    end if
  end for
end for

return clusters
```

Listing 1: Simplified pseudocode of the exact- and near-duplicate media item deduplication and clustering algorithm

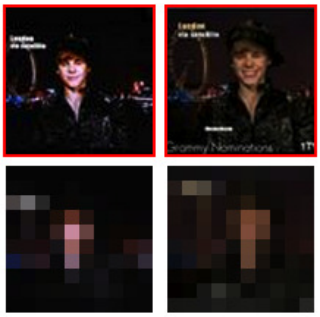


FIGURE 7: Algorithm debug view for two clustered media items related to the *Grammy Awards Nominations 2013* event (the red border around the media items indicates at least one detected face)

exactly 1 face was detected. A screenshot of the complete media item clustering application (with a different event) is available online at `http://twitpic.com/c02qfs/full` (accessed 04/30/2013).

3.5. Selection of a Cluster's Visual Representative

In the previous subsections, we have introduced an algorithm that clusters similar enough media items so that they can be treated as just one media item. In this subsection, we detail an algorithm for deciding on a particular cluster's visual representative, *i.e.*, the one media item that will be used from thereon to represent the whole cluster visually. Naturally, through the way the clustering algorithm works, the contained media items are already visually similar. The algorithm's objective is thus to decide on the one media item with the highest quality. A straight-forward media item quality criterion in the specific context of mobile devices that are used in practice to create media items shared on social networks is camera *resolution*. For example, the original iPhone had a 2.0 Megapixel camera, whereas the latest iPhone 5 has an 8.0 Megapixel camera. In consequence, in a cluster that contains media items created with an original iPhone and an iPhone 5, one certainly will prefer media items created by the latter. A special case exists when a cluster contains videos *and* photos, *i.e.*, a photo in the cluster is contained in a video in the cluster. In this case, we prefer the moving video over the still photo. The algorithm ensures this by artificially assigning a pseudo resolution of infinity to videos, which guarantees them to outpace any photos in the cluster. In practice, we found that the corner case of more than one video being contained in a cluster, where we would have to decide on the video with the highest quality, barely occurs, which justifies the chosen approach. Yet we give an outlook on how video deduplication and clustering could work with our approach in Section 5. Listing 2 shows the cluster visual representative selection algorithm that selects the media item with the highest Megapixel resolution as the cluster's visual representative. In the future, more elaborate heuristics that, *e.g.*, consider compression rate or histogram richness, may be applied.

```
Input: cluster, a cluster of media items
Output: representative, the visual representative

init:

maxPixels = 0
representative = null
for item in cluster

  # Ensure that videos will always be preferred
  if item.type == 'video' then
    item.width = item.height = INFINITY
  end if

  resolution = item.width * item.height
  if resolution >= maxPixels then
    maxPixels = resolution
    representative = item
  end if
end for

return representative
```

Listing 2: Pseudocode of the cluster visual representative selection algorithm that finds the highest quality media item of a cluster

4. EVALUATION

4.1. Experiments

We have evaluated the media item clustering algorithm with two events from recent history with high social network coverage that we will briefly describe in the following.

Grammy Awards Nominations 2013 The Grammy Award[5] is an award by the National Academy of Recording Arts and Sciences of the United States to recognize outstanding achievement in the music industry. The annual ceremony features performances by prominent artists, and some of the awards are presented in a widely viewed televised ceremony. On December 5, 2012, the nominees for the 55th Annual Grammy Awards were announced at an event broadcasted live by CBS titled *Grammy Nominations Concert Live*, during which Taylor Swift and LL Cool J revealed the nominees in several key categories. CBS suggested the hashtag `#GRAMMYNoms` to be used for the event.

Victoria's Secret Fashion Show 2012 The *Victoria's Secret Fashion Show*[6] is an annual event sponsored by Victoria's Secret, a brand of lingerie and sleepwear. The show features some of the world's leading fashion models and is used by the brand to promote and market its goods in a high-profile setting. The show is a lavish event with elaborate costumed lingerie and varying music by leading entertainers that attracts hundreds of celebrities and entertainers. CBS suggested the hashtag `#VSFashionShow` to be used for the event.

[5] `http://en.wikipedia.org/wiki/2013_Grammy_Awards`, accessed 04/30/2013

[6] `http://en.wikipedia.org/wiki/Victoria's_Secret_Fashion_Show`, accessed 04/30/2013

4.2. Discussion

We have collected and made available publicly[7] datasets for both events with 379 media items for the *Victoria's Secret Fashion Show 2012* event and 949 media items for the *Grammy Awards Nominations 2013* event. These media items were collected using the media item extraction framework described in [1, 2, 3, 4] using a mix of hashtag searches with the official event hashtags combined with full-text searches for event titles and variations thereof. Due to the short-lived nature of communication on social networks, the returned results of the media item extraction process itself are not reproducible, however, the focus of this paper is on media item deduplication and clustering. The clustering parameters for the algorithm were set as follows.

1. *rows* = *cols* = 10
2. *tiles_threshold* = 67
3. *similarity_threshold* = 10

In the following, we discuss the clustering and deduplication results. Figure 8 and Figure 9 show the top clusters for the *Victoria's Secret Fashion Show 2012* and the *Grammy Awards Nominations 2013* events respectively. For the prior, we have left face detection enabled, for the latter, we disabled it. This resulted in bigger, however, less precise clusters. Given the dataset sizes (379 media items for *Victoria's Secret Fashion Show 2012* and 949 media items

[7] `https://www.dropbox.com/sh/2llvjaut32juwrx/7eGLodfP_2`, accessed 04/30/2013

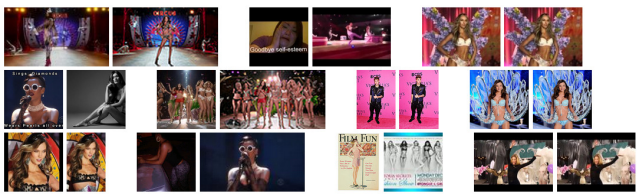


FIGURE 8: Top clusters for the *Victoria's Secret Fashion Show 2012* event (face detection *enabled*)

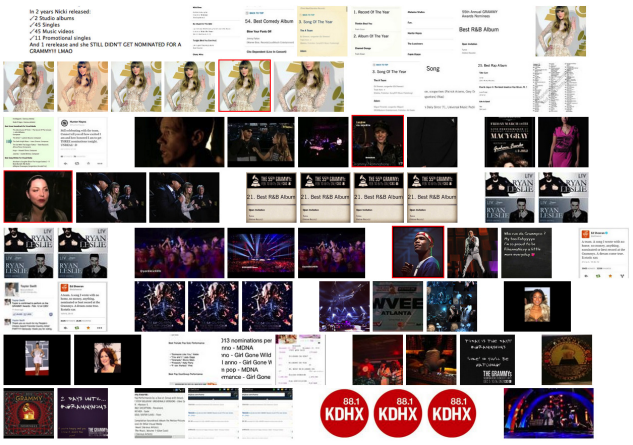


FIGURE 9: Top clusters ordered by cluster size for the *Grammy Awards Nominations 2013* event (face detection *disabled*; some clusters span multiple rows)

for *Grammy Awards Nominations 2013*), this is fully aligned with our algorithm's design goals that state that aggressive clustering is acceptable. We will pick some representative examples from both events and have a closer look at the clustering algorithm's strengths and weaknesses.

4.2.1. Algorithm Strengths

Figure 10 shows a brunette fashion model in a pink robe and a heart-shaped pink spotlight as central elements of both media items. Even though the model is captured from different angles and at different times, the media items are successfully clustered due to the very identifying colors and the high tile similarity. Figure 11 shows two different views of a stage taken at slightly different times. The left media item covers a detail of the scene, whereas the right media item covers the entire stage. Due to the high tile-wise similarity of the relevant tiles of the scene detail, the media items are successfully clustered. Figure 12 shows two views of a stage under different lighting conditions. Due to the tile color tolerances and the high tile-wise similarity, the media items are successfully clustered.

FIGURE 10: High tile-wise similarity of a dominating color

FIGURE 11: Cropped view of a stage scene

FIGURE 12: View of an entire stage and detail of a stage under different lighting conditions

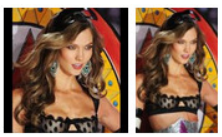


FIGURE 13: Zoomed view of a model with black bars left and right

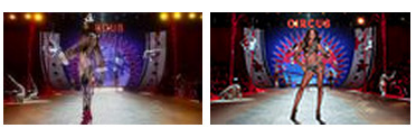


FIGURE 14: Two views of the same stage with different person

Figure 13 shows two media items of the same model where the left media item is a zoomed version of the right media item with added black bars so that the resulting media item has a square aspect ratio. Despite the differences, the media items are successfully clustered. Figure 14 shows two views of the same stage, however, with a different person. Due to the dominating tile-wise similarity of the stage tiles, the media items are correctly clustered.

4.2.2. Algorithm Weaknesses

Figure 15 shows five media items with pure white as the dominating color and a pure black font, where users had taken screenshots of the Grammy results from Web pages. The algorithm in its previously described form clusters such media items. This may or may not be desired. Likewise at the other end of the color spectrum, Figure 16 shows two media items of a woman with pure black as the dominating color, one time with, and the other time without added black bars to fit a letterbox aspect ratio. We note that in its previously described form, the algorithm does *not* cluster such media items (unless a really small number *tiles_threshold* of required similar tiles is selected).

In the majority of cases, though, clustering such media items *is* desired. Our response to both issues is to ignore a certain part of the color spectrum in the algorithm's similarity measure. In the concrete case, ignoring pure white and pure black correctly fixed the clustering in our chosen example events in all but one cases, without negatively impacting previously correctly formed clusters.

FIGURE 15: Pure white as dominating color stemming from screenshots of Web pages

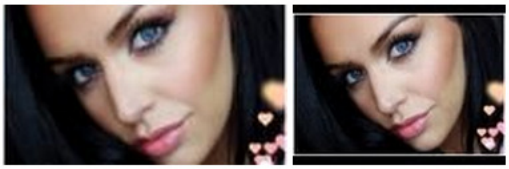


FIGURE 16: Black bars added to fit a 16:9 image in a 4:3 letterbox (the white border is part of the original media item)

FIGURE 17: Entirely different thumbnail-size media items with similar tile histograms, faces were not detected due to the small size

Finally, Figure 17 shows two entirely different media items that were incorrectly clustered as the tile histograms were similar enough under the chosen similarity threshold. The explanation for this is twofold. First, the original source media items were very small thumbnail-like images, which hindered face recognition (there is actually an *unequal* number of faces in each image). Second, the way the algorithm works, which is best understood by looking at Figure 7, causes the tiles of very tiny media items like the ones in question to blur.

4.2.3. *The Impact of Face Detection*

In our algorithm, face detection is an optional, per default *enabled* feature that improves the precision of the clustering at a slight cost of recall. Face detection can fail, *e.g.*, with too small media items or with only partially visible faces. In such cases the face detection matching condition can be *disabled*. However, detected faces in a cluster can still be used, *e.g.*, for the selection of the cluster's visual representative.

4.2.4. *The Impact of Parameters*

We have experienced in our experiments that there is no single perfect combination of algorithm parameters, so the only way to address this issue (besides ignoring too small media items, which in practice may be the easiest and best solution) is to make the parameters flexible. Additionally, the perceived Quality of Experience (QoE) [15] will differ from user to user. In the future, an *ad hoc* Machine-Learning-based approach for learning parameter settings may prove useful. In our graphical user interface we have created sliders that let the user interactively preview clustering changes. As noted before, a screenshot of the application is available online at the URL `http://twitpic.com/c02qfs/full`.

5. VIDEO CLUSTERING

In Section 3, we have introduced an algorithm for media item clustering. In this section, we will outline the conceptual framework that combines this algorithm with the previously introduced video shot boundary detection algorithm from [5]. Its goal is to, on the one hand, directly cluster videos on a shot boundary frame basis and, on the other hand, detect whether a given photo is contained in a video.

5.1. Photo-contained-in-Video Workflow

In a first step, we detect shot boundaries as described before in [5] for a given video. To illustrate this, Figure 18 shows an excerpt of detected shot boundaries for a video related to the *Victoria's Secret Fashion Show 2012* event. The first frame of each camera shot's film stripe is selected as the particular shot's representative frame. To detect whether a given photo stemming from social networks is contained in the video in question, the set of extracted shot representative frames is compared with social network photos, examples can be seen in Figure 8. We note, however, that especially for longer videos (about 4 minutes and more) this approach does not scale due to the sheer number of shot boundaries in common videos shared on social networks, which causes the process to consume too much time in practice. As a work-around, however, at the expense of exactness, the (typically few) poster still frames that are commonly returned by video hosting platform APIs like the one of YouTube[8] can be used, rather than extracting shot boundaries manually.

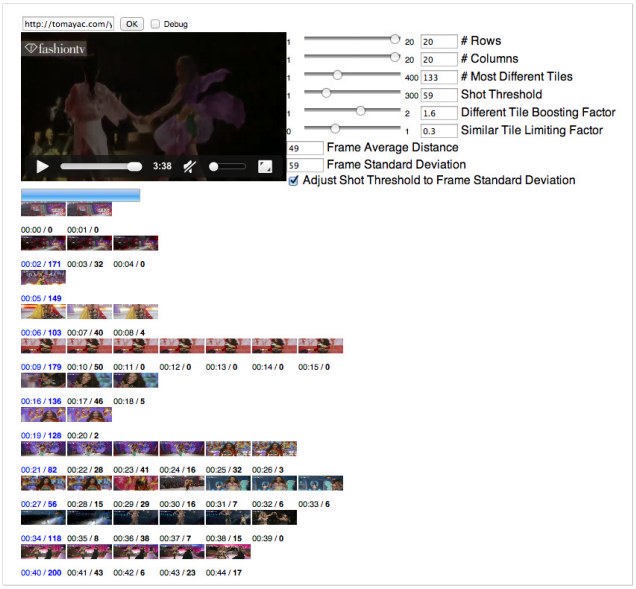


FIGURE 18: Excerpt of detected camera shot boundaries in a *Victoria's Secret Fashion Show 2012* event video

5.2. Video-contained-in-Video Workflow

To detect whether a given video A is a subsegment of another video B, we propose a similar approach as outlined in the previous subsection, with the sole difference that we need to compare all detected shot boundary representative frames of video A with the ones from video B. We recall that by our definition subsegment videos are *not* considered near-duplicates. Notwithstanding the above, in the following, we sketch the overall idea. Naturally, this approach is even less scalable with regards to system response time. The practicable work-around is, as before, to limit oneself to poster frames delivered by video hosting platform APIs. Our experiments with other events besides the ones described in Subsection 4.1 have shown that this approach works very well for common social network user behavior. For example, the 3:19 minutes long video of Mark Zuckerberg explaining the design and engineering challenges behind Facebook's recently announced Graph Search product was initially published on Facebook,[9] however, people republished the same video multiple times on YouTube. As the YouTube-generated poster frames of each republished video were similar—and even if the other video metadata like title and description were different—the described work-around

[8] `https://developers.google.com/youtube/2.0/reference#youtube_data_api_tag_media:thumbnail`, accessed 04/30/2013

[9] `https://www.facebook.com/about/graphsearch`, accessed 04/30/2013

approach was able to effectively deduplicate and cluster the videos in question. We recall that our algorithm is explicitly tailored to such observed social network sharing behavior.

6. RELATED WORK

Image Deduplication and Clustering Work on ordinal measures that serve as a general tool for image matching was performed by Bhat *et al.* in [16]. Chum *et al.* have proposed a near-duplicate image detection method using min-Hash and term frequency-inverse document frequency (tf-idf) weighting [17]. They use a visual vocabulary of vector quantized local feature descriptors based on Scale-Invariant Feature Transform (SIFT) [7]. Gao *et al.* [18] have proposed an image clustering method in the context of Web image clustering, which clusters images based on the consistent fusion of the information contained in both low-level features and surrounding texts. Also in the context of Web pages, Cai *et al.* [19] have proposed a hierarchical clustering method using visual, textual, and link analysis. Goldberger *et al.* [20] have combined discrete and continuous image models based on a mixture of Gaussian densities with a generalized version of the information bottleneck principle for unsupervised hierarchical image set clustering. Chen *et al.* [21] have introduced an image retrieval approach, which tackles the semantic gap problem by learning similarities of images of the same semantics.

Video Deduplication and Clustering More specialized methods for video deduplication exist, for example [22, 23] by Min *et al.* who, given the observation that transformations tend to preserve the semantic information conveyed by the video content, propose an approach for identifying near-duplicate videos by making use of both low-level visual features and high-level semantic features detected using trained classifiers. In [24], Oliveira *et al.* report on four large-scale online surveys wherein they have confirmed that humans perceive videos as near-duplicates based on both non-semantic features like different photo or audio quality, but also based on semantic features like different videos of similar content. A survey of video deduplication methods has been conducted by Lian *et al.* in [25]. In [26], Guil *et al.* have proposed a method for detecting copies of a query video in a videos database that groups frames with similar visual content while maintaining their temporal order. In [27], Okamoto *et al.* have proposed an approach that is based on fixed length video stream segments. By generating spatio-temporal images, they employ co-occurrence matrices to express features in the time dimension explicitly. Yi *et al.* have proposed motion histograms in [28], where the motion content of a video at pixel level is represented as a Pixel Change Ratio Map (PCRM), which captures the motion intensity, spatial location, and size of moving objects in a video sequence.

Image and Video Deduplication and Clustering A method for both photos and videos has been proposed by Yang *et al.* [29]. The authors describe a system for detecting duplicate images and videos in a large collection of multimedia data that uses local difference patterns as the unified feature to describe both images and videos. It has been demonstrated that the proposed method is robust against common image-processing tasks used to produce duplicates.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we have treated the topic of media item deduplication and clustering in different ways. We have first defined the meaning of *exact* and *near-duplicate* for both photos and videos, including the special case of a photo being contained in a video. We have then motivated common reasons for duplicate or near-duplicate content in the context of social networks. Afterwards, we have introduced an algorithm for media item deduplication and clustering with low-level and semantic features, including an algorithm for deciding on a cluster's visual representative. We have evaluated the algorithm with two recent events that had broad social media coverage. The thereby generated datasets were made available publicly and we invite researchers working on related topics to use the data to compare the obtained results. Finally, we have outlined how the algorithm can be used for video clustering.

Future work will address the encountered algorithm weaknesses, especially when dealing with small-sized media items, albeit efficient improvements of the algorithm have already been implemented in form of ignoring a part of the color spectrum. Further, we will investigate different categories of media items, *e.g.*, parody[10] or so-called memes.[11] Downloading big video files over the Internet remains an issue that needs to be dealt with before any of the outlined processing chain can begin. In the long-term, we want to explore speed and scalability improvement opportunities for the presented on-the-fly video deduplication and clustering approaches as well as "photo contained in video" detection algorithms, which—given the *ad hoc* nature of social networks—requires a combination of fast camera-shot detection algorithms tailored to streaming video and parallel processing of image and video media items.

Media item deduplication and clustering of both exact and near-duplicate media items is a fundamental step in dealing with huge amounts of social media and media overload in general. Highly popular media items not only tend to receive many social interactions on the social network they were initially published on, but also tend to get shared on other social networks. Derivates of popular media items as motivated in Section 2 further add noise to the social media landscape. Concluding, with our media item deduplication and clustering algorithm, we have contributed effective and efficient tools to deal with social media overload and to identify the few needles in the social network haystack.

[10] https://twitter.com/_Happy_Gilmore/statuses/276196671430463488, accessed 04/30/2013

[11] https://twitter.com/RetweetabIe/status/276175119620116480, accessed 04/30/2013

ACKNOWLEDGEMENTS

This work was partially supported by the European Commission under Grant No. 248296 FP7 I-SEARCH project, The described research activities were funded by Ghent University, the Institute for the Promotion of Innovation by Science and Technology in Flanders, the Fund for Scientific Research Flanders, and the European Union.

REFERENCES

[1] Khrouf, H., Atemezing, G., Rizzo, G., Troncy, R., and Steiner, T. (2012) Aggregating Social Media for Enhancing Conference Experience. *International AAAI Conference on Weblogs and Social Media*. AAAI Publications.

[2] Rizzo, G., Steiner, T., Troncy, R., Verborgh, R., Redondo García, J. L., and Van de Walle, R. (2012) What fresh media are you looking for?: retrieving media items from multiple social networks. *Proceedings of the 2012 International Workshop on Socially-Aware Multimedia*, New York, NY, USA SAM '12, pp. 15–20. ACM.

[3] Steiner, T., Verborgh, R., Valles, J., and de Walle, R. (2011) Adding meaning to facebook microposts via a mash-up api and tracking its data provenance. *Next Generation Web Services Practices (NWeSP), 2011 7th International Conference on*, pp. 342–345.

[4] Steiner, T., Verborgh, R., Gabarró Vallés, J., and Van de Walle, R. (2013) Adding meaning to social network microposts via multiple named entity disambiguation APIs and tracking their data provenance. *International Journal of Computer Information Systems and Industrial Management*, **5**, 69–78.

[5] Steiner, T., Verborgh, R., Gabarró Vallés, J., Hausenblas, M., Troncy, R., and Van de Walle, R. (2012) Enabling on-the-fly video shot detection on YouTube. *Proceedings of the 21st International Conference on World Wide Web*, April. ACM.

[6] Troncy, R., Mannens, E., Pfeiffer, S., Van Deursen, D., Hausenblas, M., Jägenstedt, P., Jansen, J., Lafon, Y., Parker, C., and Steiner, T. (2012) Media Fragments URI 1.0 (basic). W3C Recommendation. W3C.

[7] Lowe, D. G. (1999) Object recognition from local scale-invariant features. *Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2*, Washington, DC, USA ICCV '99, pp. 1150–. IEEE Computer Society.

[8] Viola, P. and Jones, M. (2001) Rapid object detection using a boosted cascade of simple features. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, pp. I–511–I–518 vol.1.

[9] Viola, P. and Jones, M. J. (2004) Robust real-time face detection. *Int. J. Comput. Vision*, **57**, 137–154.

[10] Morikawa, C. and Aizawa, K. (2012) Iconic Visual Queries for Face Image Retrieval. *Journal of Convergence*, **3**, 39–46.

[11] Satone, M. and Kharate, D. G. (2012) Face Recognition Based on PCA on Wavelet Subband of Average-Half-Face. *Journal of Information Processing Systems*, **8**, 483–494.

[12] Huang, C., Ai, H., Li, Y., and Lao, S. (2007) High-performance rotation invariant multiview face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **29**, 671–686.

[13] Abramson, Y., Steux, B., and Ghorayeb, H. (2007) Yet even faster (yef) real-time object detection. *Int. J. Intell. Syst. Technol. Appl.*, **2**, 102–112.

[14] Liu, L. (2012). JavaScript Face Detection Explained. `http://liuliu.me/eyes/javascript-face-detection-explained/`. Accessed 02/14/2013.

[15] Bhattacharya, A., Wu, W., and Yang, Z. (2011) Quality of experience evaluation of voice communication systems using affect-based approach. , **?**, 929–932.

[16] Bhat, D. N. and Nayar, S. K. (1998) Ordinal measures for image correspondence. *IEEE Trans. Pattern Anal. Mach. Intell.*, **20**, 415–423.

[17] Chum, O., Philbin, J., and Zisserman, A. (2008) Near duplicate image detection: min-hash and tf-idf weighting. In Everingham, M., Needham, C. J., and Fraile, R. (eds.), *BMVC*, pp. 1–10. British Machine Vision Association.

[18] Gao, B., Liu, T.-Y., Qin, T., Zheng, X., Cheng, Q.-S., and Ma, W.-Y. (2005) Web image clustering by consistent utilization of visual features and surrounding texts. *Proceedings of the 13th Annual ACM International Conference on Multimedia*, New York, NY, USA MULTIMEDIA '05, pp. 112–121. ACM.

[19] Cai, D., He, X., Li, Z., Ma, W.-Y., and Wen, J.-R. (2004) Hierarchical clustering of WWW image search results using visual, textual and link information. *Proceedings of the 12th Annual ACM International Conference on Multimedia*, New York, NY, USA MULTIMEDIA '04, pp. 952–959. ACM.

[20] Goldberger, J., Gordon, S., and Greenspan, H. (2006) Unsupervised image-set clustering using an information theoretic framework. *Image Processing, IEEE Transactions on*, **15**, 449–458.

[21] Chen, Y., Wang, J. Z., and Krovetz, R. (2003) Content-based image retrieval by clustering. *Proceedings of the 5th ACM SIGMM International Workshop on Multimedia Information Retrieval*, New York, NY, USA MIR '03, pp. 193–200. ACM.

[22] Min, H.-s., Choi, J. Y., De Neve, W., and Ro, Y. M. (2011) Bimodal fusion of low-level visual features and high-level semantic features for near-duplicate video clip detection. *Image Commun.*, **26**, 612–627.

[23] Wu, X., Ngo, C.-W., Hauptmann, A. G., and Tan, H.-K. (2009) Real-time near-duplicate elimination for web video search with content and context. *Trans. Multi.*, **11**, 196–207.

[24] Oliveira, R. D., Cherubini, M., and Oliver, N. (2010) Looking at near-duplicate videos from a human-centric perspective. *ACM Trans. Multimedia Comput. Commun. Appl.*, **6**, 15:1–15:22.

[25] Lian, S., Nikolaidis, N., and Sencar, H. (2010) Content-based video copy detection – a survey. In Sencar, H., Velastin, S., Nikolaidis, N., and Lian, S. (eds.), *Intelligent Multimedia Analysis for Security Applications*, Studies in Computational Intelligence, **282**, pp. 253–273. Springer Berlin Heidelberg.

[26] Guil, N., González-Linares, J., Cózar, J., and Zapata, E. (2007) A clustering technique for video copy detection. In Martí, J., Benedí, J., Mendonça, A., and Serrat, J. (eds.), *Pattern Recognition and Image Analysis*, Lecture Notes in Computer Science, **4477**, pp. 451–458. Springer Berlin Heidelberg.

[27] Okamoto, H., Yasugi, Y., Babaguchi, N., and Kitahashi, T. (2002) Video clustering using spatio-temporal image with fixed length. *Multimedia and Expo, 2002. ICME '02. Proceedings. 2002 IEEE International Conference on*, pp. 53–56 vol.1.

[28] Yi, H., Rajan, D., and Chia, L.-T. (2005) A new motion histogram to index motion content in video segments. *Pattern Recognition Letters*, **26**, 1221–1231.

[29] Yang, X., Zhu, Q., and Cheng, K.-T. (2009) Near-duplicate detection for images and videos. *Proceedings of the First ACM workshop on Large-scale multimedia retrieval and mining*, New York, NY, USA LS-MMRM '09, pp. 73–80. ACM.