Parallel Tracking

Henry Spang
Ethan Peters
Contents

• Introduction
• HAAR Cascades
  • Viola Jones
• Descriptors
  • FREAK Descriptor
• Parallel Tracking
  • GPU Detection
• Conclusions
• Questions
Introduction

• Tracking is a “embarrassingly parallel” problem
• Involves examination of many individual points/pixels that have no dependencies on each other
• Done repeatedly throughout the sequence
• Parallelization allows for tracking to be implemented in real time scenarios
  • This allows for a new range of human interactions with computer environments
Haar Cascades

- Haar features are rectangular “template” images that are meant to be a computationally efficient alternative to the Haar wavelet transform.
- A Haar feature is effectively a gradient map of the intensity of an image, and they are constructed to represent certain classes of image constructs.
- Take, for example, the two rectangles shown to the right:
- This feature will match a portion of an image that has a dark area below a brighter area, such as an edge.
Viola-Jones

• Paul Viola and Michael Jones, in 2001, applied Haar features to the task of detecting a face in an image.
• They used 2, 4 and 6 rectangle features, shown to the right:
• These features match key areas of the face, such as the eyes, nose and mouth.
Viola-Jones Cont.

- A given Haar feature typically has an accuracy rate between 55% and 65%, which is to be expected, they’re kind of coarse.

- There are two common ways to beat this (both are typically used):
  - Use a cascade of Haar features: as long as each feature has a low false negative rate, its false positive rate can be defeated by statistics.
    \[
    \lim_{k \to \infty} p^k = 0 \text{ if } p < 1
    \]
  - Use a machine learning tool to “teach” the detector how to appropriately weight the rectangles in a Haar feature to make the detector more robust.
Descriptors

• Unlike Haar Cascades, descriptors describe a region instead of detecting what is in it
• By providing generic information about a given image, they allow for matching between frames
• Based on the algorithm chosen, various forms of robustness are acquired
  • Scale invariance, rotation invariance, illumination invariance
Fast Retinal Keypoint (FREAK)

- Very modern descriptor, based off the human eye’s methodology of vision
- The retinal density of an eye is coarse towards the outside, but very dense towards the inside
- This is approximated by utilizing a search area that is a series of circular areas of varying density (low density outside, high density inside)
  - Each of these search areas is smoothed with a Gaussian kernel that also varies based on the areas size
FREAK Cont.

- The actual descriptor is taken to be a series of intensity comparisons between the search areas, yielding a binary string.
- Utilizing all available keypoints, and all available regions, this would yield a descriptor set that is much too large.
- To compensate for this, the descriptors are sorted first based on their variance, and then based on their mean.
  - A mean of .5 is desired as a binary string cannot have a more even distribution than that.
- The keypoints are clustered into 4 groups, 128 bits from each one is taken, yielding a descriptor of length 512 bits.
Haar Parallelization

• The Viola-Jones Cascade Classifier (and other face detection systems like it) exhibits extreme amounts of data parallelism:
  • For each cascade of Haar features to be performed on a single location (pixel) in the image, all of the Haar features can be evaluated simultaneously
  • All cascades can be performed (at each pixel) in parallel
  • To account for variations in scale, the entire image may need to be processed several times (with Haar features of several different sizes), and each of those iterations can be done entirely in parallel
Haar Parallelization
Overhead

• There is, of course, overhead associated with parallelizing the Viola-Jones face detector:
  • The detector relies on forming an integral image. This still must be done in sequential time, but it must also be communicated to every processing node, and is of size \( n \)
  • Since the work of evaluating Haar rectangles, features and cascades is being distributed, the result must be communicated to some head node that must take on the responsibility of deciding the final result of a given feature, and later of a given cascade
Haar Parallelization Implementation (Pipeline)

Integration → Haar Rectangle 0 → Haar Feature 0 → Decision

Sequential Time (1 proc): $O(n)$

Communication: $O(nP1)$

Parallel (P1 processors): $O(L/P1) = O(mk/P1)$

Communication: $O(P2/P1) \approx O(1)$

Parallel (P2 processors): $O(M/P2) = O(mk/P2)$

Communication: $O(P2)$

$L = ML_1$, $M = mk$

$L_1$ = number of rectangles per feature (constant)

$m$ = number of Haar features

$k$ = number of scales to evaluate
FREAK Parallelization

- FREAK has the benefit of being parallelizable at each of its individual operations
- When constructing the set of $n$ descriptors each of the keypoints can have its descriptor calculated separately from the others
- In addition to this, the calculation of parameters (mean and variance) can also be easily divided amongst the processors
- If this is all done together it can allow for a single communication between the slaves and master processor, which would then handle the final sorting
Examining each section individually the time complexity can be seen as:

\[ O(n) + O(n) + O(\log(n)) \]

\[ O(n) \]

Because each stage can be evenly divided by \( p \) processors, the complexity can be taken down to:

\[ O\left(\frac{n}{p}\right) \]

Finally, utilizing the processors to combine the calculating and parameterizing of the descriptors the complexity is:
Parallel Tracking

- Facial Detection and Image tracking can be used for face tracking in a few ways:
  - The following algorithm is dead-simple, but effective:
    1. Read a single frame out of a video stream
    2. Perform Viola-Jones face detection
    3. Highlight the results of the detection
    4. Show the result to the user
    5. Go back to step 1
  - A more sophisticated algorithm is much preferred:
    1. Read a single frame out of a video stream
    2. Perform Viola-Jones face detection
    3. Use a descriptor (such as FREAK) to get a compact description of the face
    4. Read a single frame out of a video stream
    5. Pull sub-images out of the image from areas around the previously detected face
    6. Get descriptions of each of those images and compare them to the known face
    7. Highlight the closest match and remember where it is
    8. Go back to step 4
Parallel Tracking Cont.

• Using the twin powers of cleverness and math, an even more effective algorithm is developed:
  1. Read a single frame out of a video stream
  2. Perform Viola-Jones face detection
  3. Use a descriptor (such as FREAK) to get a compact description of the face
  4. Repeat steps 1-3 several times to build a dictionary of face images
  5. Read a single frame out of a video stream
  6. Pull sub-images out of the image from areas around the previously detected face
  7. Get descriptions of each of those images and compare them to the known face
  8. If there is a sufficiently close match
     • Highlight it and remember where it is
     • Otherwise enact steps 1-3, then go back to step 5
  9. Go back to step 5
Applications to Parallel Tracking

- Tracking requires a high frame rate to maintain appropriate motion
- Parallelization allows for the high frame rate required for a large number of algorithms
- The speedup given to Haar detectors allows for a detection to be performed every frame
- Increasing the speed of FREAK allows for descriptors to be made and matched for many regions of an image every frame
Questions?

It's QUESTION TIME!!