Scientific Research Publishing

Journal of Computer and Communications, 2017, 5, 19-42
http://www.scirp.org/journal/jcc
ISSN Online: 2327-5227
ISSN Print: 2327-5219

Real-Time Range of Motion Measurement of Physical Therapeutic Exercises

Raghu Raj Prasanna Kumar[1], Suresh Muknahallipatna[2], John McInroy[2], Mark McKenna[3], Lori Franc[3]

[1]National Center for Atmospheric Research, Boulder, CO, USA
[2]Dept. of Electrical Engineering, University of Wyoming, Laramie, WY, USA
[3]Premier Bone & Joint Centers, Laramie, WY, USA
Email: sureshm@uwyo.edu

How to cite this paper: Kumar, R.R.P., Muknahallipatna, S., McInroy, J., McKenna, M. and Franc, L. (2017) Real-Time Range of Motion Measurement of Physical Therapeutic Exercises. *Journal of Computer and Communications*, **5**, 19-42.
https://doi.org/10.4236/jcc.2017.59003

Received: June 1, 2017
Accepted: June 30, 2017
Published: July 3, 2017

Abstract

Physical therapeutic exercise (PTE) is the planned process of performing bodily movements, postures, or physical activities to provide a patient with the ability to remediate or prevent impairments at a minimum. The efficacy of the PTE involves measuring accurately the range of motion (ROM) of joint functions and parameters that indicate the onset of fatigue, jerky motion, and muscle/joint resistance to the PTE. A physical therapist (PT) typically determines the efficacy of a PTE by measuring joint angles in clinical diagnosis to assess the ROM using the simple device Goniometer since motion capture systems are generally expensive, difficult to use, and currently not suited for real-time operations. The joint angle measurement using Goniometer suffers from low accuracy, low reliability and subjective. Furthermore, a patient when performing PTE by themselves at remote locations like their home or community centers cannot use a Goniometer to determine the efficacy. In this study, we present the approach of using an inexpensive, simple human motion capture system (HMCS) consisting of a single camera and a graphical processing unit (GPU) to perform the efficacy of the PTE in real-time. The approach involves the use of general purpose graphic processing unit (GPGPU) computer vision technique to track and record human motion and relate the tracked human motion to the prescribed physical therapy regimen in real-time. We have developed a tracking algorithm derived from the Klein's algorithm known as the Modified Klein's algorithm (MKA) capable of tracking human body parts while the original Klein's algorithm was only capable of tracking objects with sharp edges. The MKA algorithm is further modified for parallel execution on a GPU to operate in real-time. Using the GPU, we are able to track multiple markers in a high definition (HD) frame of the HD video in 1.77 msecs achieving near real-time capability of ROM measurements. Furthermore, the error in the ROM measurements in comparison to

Goniometer measurements is in the range of −2.4˚ to +1.4˚, which is well within the joint measurement prescribed standards. The suitability of the HMCS for elbow PTE is also presented.

Keywords

Computer vision, Therapeutic Exercise, GPU, Parallel Programming

1. Introduction

A typical therapy session constitutes a patient performing PTE consisting of a series of complicated bodily motions in a pattern at certain intensity level to achieve a beneficial range of motion. Current therapy sessions are performed by a patient under the supervision of a physical therapist in a clinical setting. The PT at all times is in the immediate proximity of the patient assisting the patient in performing an exercise or set of exercises in a proper way. Furthermore, the PT ensures the full compliance of the patient in meeting the prescribed motions and number of repetitions through verbal feedback. This individual attention by a PT to a patient during a therapy session is a viable approach if an adequate number of PTs are available.

However, with the ever-increasing demand for physical therapy services by the aging population coupled with the shortage of highly skilled PTs, has made individual patient attention approach by immediate proximity a non-viable solution. Furthermore, in many geographical areas in the US, patients have to travel a long distance to avail the services of a PT. The lack of a sufficient number of PTs, long distance travel and expenses force many of the patients to skip entirely or inadequately perform physical therapy sessions. An Ad-hoc approach currently used to address the above issues is expecting the patients to perform the exercises remotely on their own using diagrams of various exercises made available by the PT. The ad-hoc approach leads to a deleterious effect on the efficacy of the physical therapy. Efficacy of physical therapy is directly correlated to the patient's ability and willingness to comply with the assigned work. Furthermore, it is very difficult to measure the outcomes of PTE, due to lack of evidence-based medicine associated with physical therapy. Physical therapy remains a largely manual process with less objective data generated than most other disciplines. Currently, there is no common, cost-effective way to monitor patients' compliance and outcome. Hence, to address the above issues, there is a need for an alternate approach that would allow a patient to perform PTE remotely but still under the supervision of a PT.

We propose an alternate physical therapy strategy of using a human motion capture system (HMCS) based system to assist the patient in performing repeated physical therapy sessions with real-time feedback and supervision by a PT. In this paper, we present the ongoing work of the human motion capture system based on the use of general purpose graphic processing unit (GPGPU)

computer vision technique to track and record human motion.

Generally, physical therapy actions are correlated with the geometric motion of specific body parts. The body parts may include, for example, a limb, elbow, foot, wrist, torso, neck and head. The motion is measured by calculating the position of a body part of interest in relation to another body part [1], which may or not be stationery. The motion is not measured by determining the position in relation to physical surroundings. Typically, a PT uses Goniometer to measure body angles to determine the ROM of a specific body part. The ROM is measured and recorded before starting the physical therapy sessions. The measurement may be performed again at the end of each therapy session or after few sessions to capture the efficacy of treatment. Since, the measurements are performed manually using the Goniometer, measurements during the PTE is not possible. For example, there is no objective data recorded which would help the PT to determine the intensity of the prescribed PTE.

In our research related to the HMCS, we envision a patient performing PTE can have ROM, velocity, acceleration and other parameters measured accurately in real-time, using a single off-the-shelf high-definition camera in a cost effective way. The PT now receiving the measured parameters in real-time can provide feedback to the patient and supervise the PTE.

This paper is organized as follows: In Section 2, we present the existing human motion capture approaches. Section 3 presents an overview of the proposed HMCS. In Section 4, the computer vision based tracking and the Klein's tracking algorithm is discussed. In Section 5, the modified Klein's algorithm suited for tracking the motion of human body parts is presented. Section 6 presents the parallelization of the modified Klein's algorithm for GPGPU computing. In Section 7, the implementation of MKA on GPU is presented. The MKA performance and the application of HMCS to determine the efficacy of elbow PTE is presented in Section 8. Section 9 presents conclusions and future work.

2. Related Work

Muller, *et al.*, in [2] have used two inertial measurement units (IMU) to perform online measurements of the flexion/extension and the pronation/supination angle of the elbow. They use kinematic constraints of the elbow to estimate the two dominant axes of reference and thereby eliminate the need for alignment of the IMUs. To achieve real-time capability a window of measurements is chosen with the window shifted by each new measurement. This approach, however, requires a patient to attach the IMUs at the correct position on the elbow and in the same position repeatedly. Furthermore, the computational time required to estimate the two dominant axes of reference is 20 secs making this approach not suitable for real-time application.

Chang, *et al.*, have demonstrated the use of the Microsoft Kinect to track the motion of a human body [3] in real-time. They use the Kinect to track the twenty joint positions inherently tracked by the Kinect SDK. They provide feedback to the patient about the number of repetitions a particular geometric motion was

performed and change in the ROM. However, this approach requires the use of the special camera the Microsoft Kinect, which is typically now owned by elderly patients. The measurements performed using the Kinect is based on the SDK database of joint positions of a large number of healthy human beings. Using the joint positions of healthy human beings as reference position introduces errors when the joint positions of a patient performing PTE are measured.

Zhao proposes an assessment method using kinematic rules [4] for each rehabilitation exercise to provide specific feedbacks regarding the quality of exercise. The Microsoft Kinect is used to measure joint angles, and a fuzzy inference engine is used to assess the overall quality of the execution of a PTE. The fuzzy inference engine ability to assess the quality of the PTE performed is dependent on a set of kinematic rules specifying the requirements for each PTE fully and measures the joint angles accurately using the Kinect camera.

Two Microsoft Kinect cameras are used by Chen, *et al.*, to construct 3D joint positions [5] and then they construct center of voxels upon the joint positions using the mean shift algorithm. The joint moveable angles and the motion data are calculated from the center of voxels frame by frame. They compare their ROM measurements with VICON motion capture system. This approach of measuring the ROM of a patient first requires the patient to perform the PTE in front of a VICON motion capture system to record the skeleton data of the patient body as Golden Standard or ground truth. The two Kinect cameras have to be positioned at a precise distance from the patient. Due to the reconstruction of 3D joint positions and center of voxels in each frame makes their approach computationally intensive, and not suitable for real-time application.

Liu, *et al.*, have proposed a method to track human motion using multiple low-cost RGBD cameras [6]. They use multiple RGBD cameras to address the missing depth values, and noise issues with low-cost Microsoft Kinect cameras. They also address the problem of body occlusions with a single camera by using multiple cameras. Using the color and depth images and point cloud acquired in each view, they extract an initial pose. The initial pose is dynamically updated by combining a filtering approach with a Markov model to estimate the new poses and thereby measure the ROM of the entire human body. However, this approach is heavily dependent on correctly aligning the point clouds from three depth cameras. If the alignment is not performed accurately, the tracking will fail. The approach requiring pose determination is computationally intensive making it unsuitable for real-time application.

Penelle, *et al.*, have used multiple Microsoft Kinect cameras to track human motion by fitting a model to the observed data, which are the depth images from the Kinect cameras [7]. They have been able to achieve near real-time execution speed using a GPU. However, the tracking accuracy is dependent on having a perfect 3D model of the patient, which requires pose estimation using multiple cameras.

All of the discussed approaches for tracking human motion require a) the use of specialized camera or cameras, b) multiple cameras c) active sensors/markers

attached to parts of the human body or d) involve complex manual adjustments to get the right output. The patient is expected to position the multiple cameras at precise locations to construct a 3D model of the patient or estimate pose of the various parts of the human body. Furthermore, the majority of the approaches are computationally intensive. Especially with HD cameras, the time needed to process images is so high that the approaches are unsuitable for real-time use. Our proposed approach differs significantly from the previous work by using a single camera without the need of pose estimation or 3D model of the patient. In our approach, we process high definition resolution monocular images with passive optical markers on body parts of interest to measure individual ROM. The camera can be of any RGB type without any depth measuring capability. Furthermore, our approach has a parallel implementation on GPUs, which facilitates real-time processing.

3. Overview of Proposed HMCS

The primary function of our HMCS is to measure the joint position or angles when a patient is performing a particular PTE. As an example, in Figure 1, a typical PTE a patient performs after an elbow injury is shown.

The patient performing the elbow PTE is expected to perform the flexion and extension motions of the arm maintaining the shoulder at a stationary position. If a patient allows the shoulder to move along with the flexion and extension motions, the elbow muscles and the joint will be subjected to forces that are not prescribed in the PTE and will result in poor efficacy. A high efficacy of the elbow PTE is achieved when the patient performs a smooth flexion and extension motions according to the prescribed joint angles and repetitions without experiencing fatigue. This necessitates measuring the joint angle position in reference to the shoulder horizontal axis during the PTE to determine the number of repetitions, smooth motion, ROM and fatigue free motion. In Figure 2, a patient performing the elbow PTE with passive optical markers (blue circular markers)

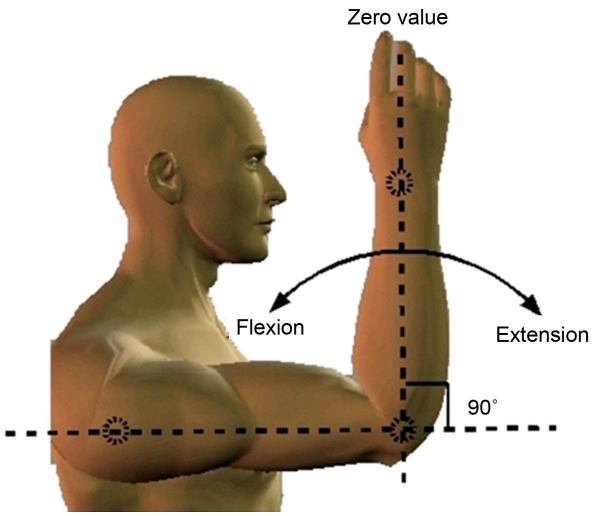

Figure 1. Elbow physical therapeutic exercise.

is shown. The joint angle position is measured by tracking the passive markers in each frame of the PTE video in real-time. By computing the position differences between markers on the arm and the reference marker on the shoulder the efficacy of the PTE can be determined.

The HMCS structure overview is shown in Figure 3.

As shown in Figure 3, an off the shelf camera is used to capture the video of a patient performing PTE with passive optical markers attached to the body part performing PTE as shown in Figure 2. The video can be transmitted in real-time to the therapy clinic or to a laptop/PC with GPU for computing the ROM. The flow diagram in Figure 4 depicts the computational steps required to process the video and measure the joint angles.

In Figure 4, the loop represents the modified Klein's tracking algorithm, which is executed in parallel for each marker on a GPU. The first input to the algorithm is the initial position of the markers, which is located approximately at the center of a marker determined visually by a mouse click. The second and the third inputs are the size of a marker in pixels and the number of markers. The output of the algorithm is the centroid of the markers in each video frame. The Klein's and MKA for tracking human body are discussed in detail in the following section.

4. Klein's Tracking Algorithm

The Klein's tracking algorithm [8] focuses on tracking augmented reality workspaces. The algorithm uses the corners of different objects in a workspace as the

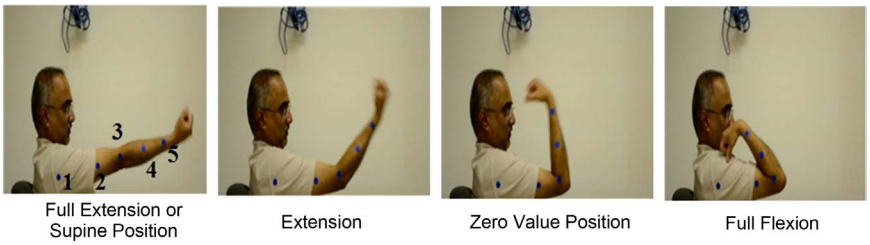

Figure 2. Elbow PTE with passive optical markers.

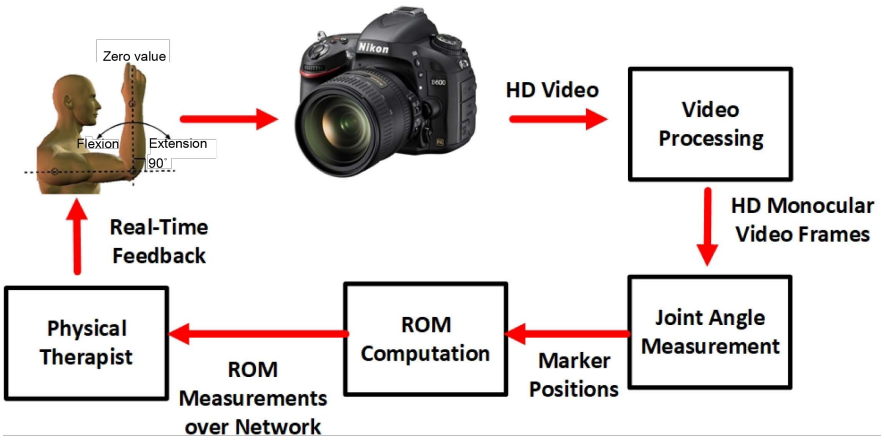

Figure 3. HMCS structure overview for real-time application.

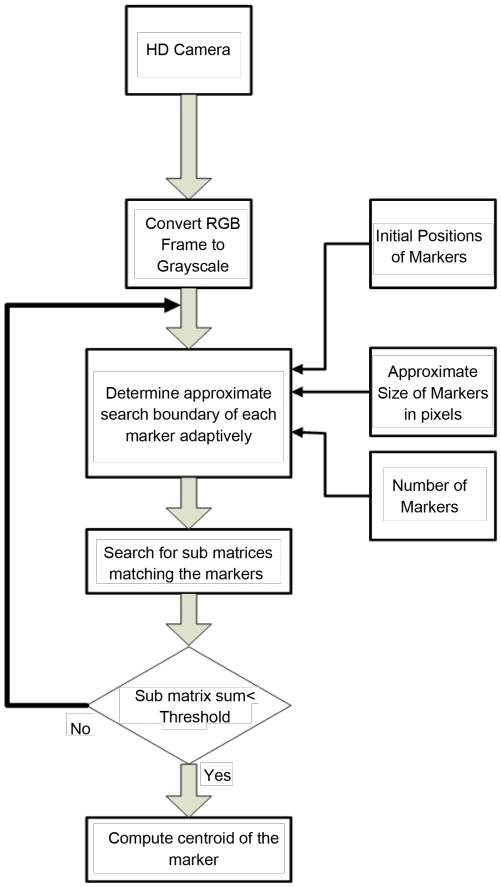

Figure 4. Passive optical marker tracking.

tracking points or objects. In order to track, a relative motion between the camera and the workspace is required. As the camera moves relative to the workspace, the tracking algorithm segments the tracking features and tracks them from one image to another using the following steps:

a. Transformation Stage 1 (S1)—Resilience: Workspaces can have different illuminations. To provide resilience to lighting changes, mean intensity value is subtracted from individual image pixels, *i.e.*, a mean of the pixel's surrounding pixel values within a pre-determined boundary is obtained and subtracted from the pixel's value. This is carried out for every pixel on the image frame.

b. Segmentation: Features from Accelerated Segment Test (FAST) algorithm are used to detect corners in the image. FAST algorithm processes an image and provides a set of points for every edge on the image. Each set of points, for the workspace, provides one corner of an object. Hence each set is termed as a tracking feature or marker.

c. Transformation Stage 2 (S2)—Object based transformation: An affine warp is performed on the marker. An affine warp is a differential computation that makes the marker resilient to rotations. This helps in tracking the marker even if the relative rotation between the camera and the marker is significant. A search template of the marker is generated using the warped conversion.

The image and the marker template are down-sampled by several levels, reducing the dimension of the image and the template. One level of sub-sampling involves reducing the image or marker template to half its original size, *i.e.*, replacing one pixel with four adjacent pixels. This helps in reducing the overhead of the search.

d. Definition: The tracking is performed by evaluating the sum of the squared differences (SSD) between the image and the marker template at all locations within a given search region of an image. A search region is a fixed boundary around the occurrence of the marker in the previous image. The boundary is pre-determined for each workspace. The location within the search region having the smallest SSD value is taken as the location of the marker in the new image.

A flow diagram representing the different phases of the Klein's algorithm is shown in **Figure 5**. The algorithm begins at the S1 part of the transformation for resilience, followed by segmentation. The segmented image is then subjected to the S2 part of the transformation to obtain the position of the marker on the new image. This is repeated for each image in the sequence.

The algorithm begins at the S1 part of the transformation for resilience, followed by segmentation. The segmented image is then subjected to the S2 part of the transformation to obtain the position of the marker on the new image. This is repeated for each image in the sequence.

5. Modified Klein's Algorithm

Klein's algorithm has been modified by its author Klein as well as other researchers as seen in [9]-[16]. These modifications focus on a) tracking an object with lesser or different parameters object parameters [9] [10], b) using low per devices [11], which may involve cloud computing for robots [12] c) using different camera configurations/error estimation techniques [13], d) improving

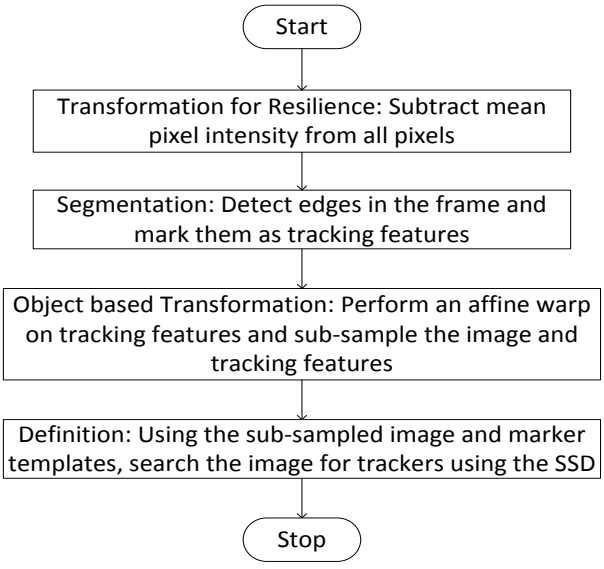

Figure 5. Klein's algorithm tracking phases.

tracking capability [14], or for limited capability cameras [15] [16]. However, this section discusses a modification to Klein's tracking algorithm focusing on better parallelization, and hence improved performance, of the algorithm. This new algorithm, called as MKA, is designed and developed to track human body parts. The FAST algorithm used in Klein, detects only sharp edges. Since not all human body parts have well defined edges, FAST is replaced in the MKA. Subsequently, changes in transformation and segmentation have been introduced to compliment the replacement of the FAST algorithm.

5.1. Transformation

The first part of the transformation in Klein's algorithm provides resilience. The approach here is to make the tracking more resilient not just to lighting changes, but also to the background of the image. An overview of the transformation phase is shown in Figure 6.

Consider a camera, capable of taking $m \times n$ resolution video at f_s images per second, which is used to capture the video to track objects. The video provides RGB (Red Green Blue) image sequences, as shown in top part of Figure 6, at f_s RGB images every second. The RGB color model is an additive color model in which red, green, and blue light are added together in various ways to

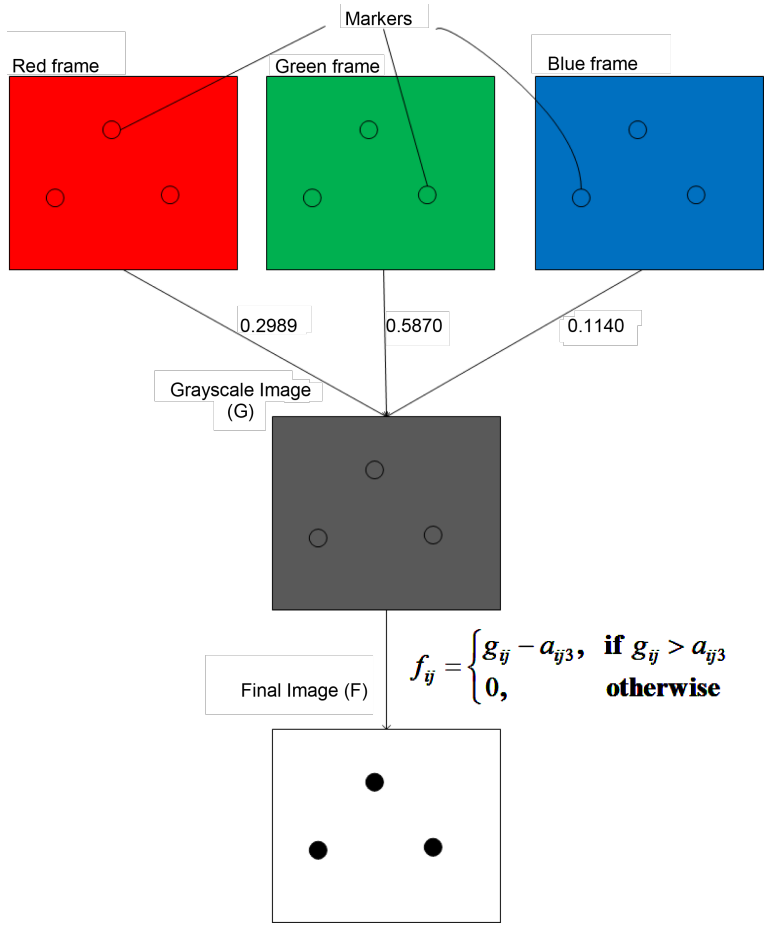

Figure 6. An overview of transformation phase of tracking algorithm.

reproduce a broad array of colors, meaning each RGB frame provides 3 sub-frames of $m \times n$ pixels, where the sub-frames belong to the red, green and blue color models respectively. Let each RGB image be given by A_k, where

$$A_k \in R^{m \times n \times 3},$$

k is the time instance,

$$a_{ijh} \in A_k,$$

$$i = 1, 2, \cdots, m-1, m,$$

$$j = 1, 2, \cdots, n-1, n \text{ and}$$

$h = 1, 2, 3$ —1 represents red frame, 2 represents green frame and 3 represents blue frame.

The element $a_{ijk} \in A_k$, contain the pixel values (intensity) of the whole image including the markers. The markers are assumed to be uniformly colored objects. Since the human body lacks uniform coloring, small uniformly colored stickers are assumed to be present on the relevant body part. This is considered as a tracking feature. This aids in segmenting an image more efficiently, reducing noise in the images. Considering blue stickers as markers, the blue frame would have higher numerical pixel values for the marker locations than red or green frames ($a_{ij3} > a_{ij1}, a_{ij2}$). Using this constraint, the transformation is formulated as shown in the middle part of **Figure 6**, using:

1. A_k is converted into a gray scale image $G_k \in R^{m \times n}$ using the following standard conversion:

$$g_{ij} = 0.2989 a_{ij1} + 0.5870 a_{ij2} + 0.114 a_{ij3} \quad (1)$$

where, $g_{ij} \in G_k$, $a_{ijk} \in A_k$. This reduces the blue content in G_k.

2. The final step of the transformation is to obtain the output image $F_k \in R^{m \times n}$, as shown in the bottom part of **Figure 6**, using:

$$f_{ij} = \begin{cases} g_{ij} - a_{ij3}, & \text{if } g_{ij} > a_{ij3} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

where, $f_{ij} \in F_k$. The output image F_k renders the markers as perfect black spots having zero values. The two-step transformation not only provides resilience to lighting changes by lowering and subtracting the intensity but also helps in identifying the markers more effectively by differentiating them from the rest of the background.

5.2. Segmentation

The segmentation step is based on the search regions in the original Klein's algorithm. The boundary of the search region is used to segment the image F_k. The boundary is determined based on the relative motion between the camera and the markers, the dimension of the marker in the image and f_s. Let the velocity of the marker motion relative to the camera be v. Let the dimension of the l^{th} marker on the image be $p^l \times q^l$. Let the maximum possible displacement of any given marker between two consecutive images be given by v_x and

v_y along the horizontal and vertical axes respectively. The v_x and v_y satisfy the relationship:

$$v_x \propto \frac{v, p^l}{f_s}, \quad v_y \propto \frac{v, q^l}{f_s} \tag{3}$$

Using (v_x, v_y), and the known marker location-centroid of the tracking feature on the $(k-1)^{th}$ image given by (x, y), the search for the marker in the k^{th} image can be limited within the boundary $(x \pm v_x, y \pm v_y)$. The initial values of v_x, v_y are estimated using the v, f_s, p^l and q^l values provided. If this estimation fails to track the markers on any frame, then v_x, v_y are incremented from the initial estimation to widen the boundaries of search region. The new v_x, v_y values are stored for successive frames. Hence the image F_k will be segmented as shown in Figure 7.

In Figure 7, it can be observed that the dimension of the search region on an image is given by $(2v_x+1)(2v_y+1)$, where $(2v_x+1) > p^l, (2v_y+1) > q^l$. This implies that there are

$$4\left(v_x + 1 - \frac{p^l}{2}\right)\left(v_y + 1 - \frac{q^l}{2}\right) \tag{4}$$

number of $p^l \times q^l$ dimension marker sized regions within the search region. Each of the $p^l \times q^l$ dimension marker sized region, called as match blocks, is possibly the actual marker position, *i.e.*, every match block is a potential marker. Hence all match blocks are compared with the marker templates. The technique to compare these match blocks forms the definition part of the algorithm.

5.3. Definition

In order to find a marker, a search template (s) is used. The squared summation of the difference (SSD) between each element of the (s) template and the corresponding element in the match block let b represent a match block, then the SSD is given by:

$$\text{SSD} = \sum_{i=0}^{p^l}\sum_{j=0}^{q^l}\left(s_{ij} - b_{ij}\right)^2 \tag{5}$$

In the MKA's transformation phase, all s_{ij} are reduced to zeros, therefore, the SSD for MKA can be computed using:

$$\text{SSD} = \sum_{i=0}^{p^l}\sum_{j=0}^{q^l}\left(b_{ij}\right)^2 \tag{6}$$

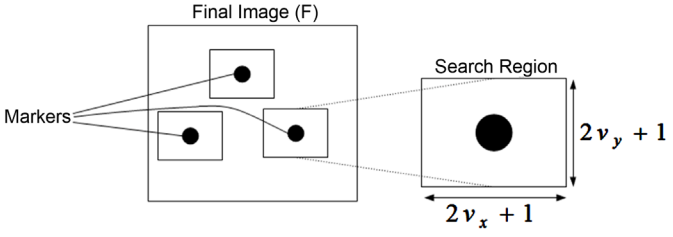

Figure 7. Segmentation phase of the algorithm.

This SSD can be implemented in a number of different ways. A technique adopted in [8] is described in Figure 8. The SSD for the first match block position within the search region begins at the first possible match block location, *i.e.*, first row and first column, as shown in the first row of the search in Figure 8. The search continues by performing SSD for subsequent match block locations, which occur in the same row but in subsequent columns. This will continue till the end of the row is reached, and then the search switches to next row, as seen in the second row of the search in Figure 8. This is continued row-wise till the end of search region is reached as shown in the bottom part of Figure 8 Since $4\left(v_x+1-\frac{p^l}{2}\right)\left(v_y+1-\frac{q^l}{2}\right)$ match blocks are present, equal number of SSD computation for match blocks is performed. Hence for a given marker region, this generates multiple SSD values. The results are interpreted as follows:

$$\mathrm{SSD}\begin{cases}=0, & \text{possible marker location}\\ <T, & \text{possible marker location for high illumination scenarios}\\ >T, & \text{away from marker location}\end{cases} \tag{7}$$

The threshold T is based on the illumination and background of the image. It's determined on a per scenario basis by trial and error. If the SSD provides one match, then the centroid of the marker location is taken to be the centroid of the object. If multiple marker positions provide a SSD value satisfying the threshold then the average of all the locations is obtained to get the centroid of the object. Once the centroid of one marker is obtained, the definition part of the algorithm moves on to the next marker until centroids of all markers in an image are obtained. Once all markers in an image are processed, the tracking algorithm moves on to the next image in the sequence.

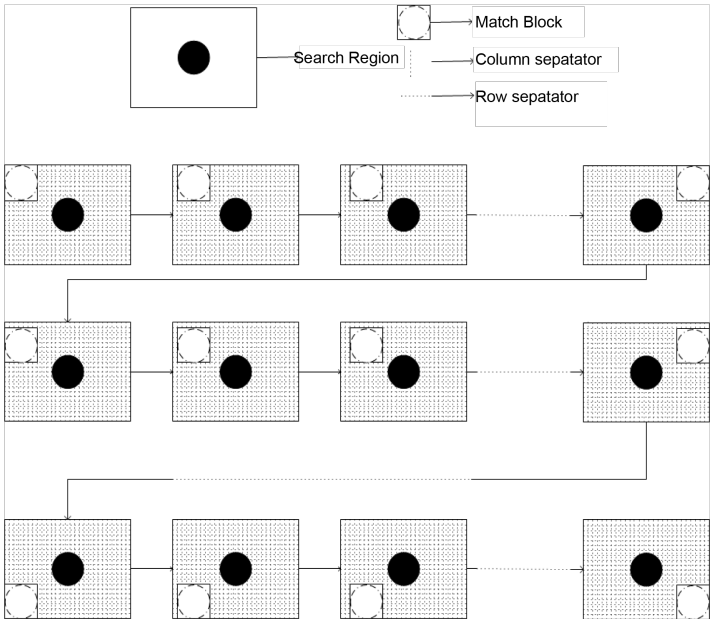

Figure 8. The brute force technique to compute SSD for all match blocks within search boundary.

6. Parallelization of Modified Klein's Algorithm

The MKA needs to be redesigned for a parallel implementation. The re-design is performed in two phases:

a. Computationally Intensive Region Identification: This phase is to identify the bottleneck of parallelism in the code.
b. Algorithm Modification: Once the region is identified, the algorithm is modified to achieve maximum parallelism.

6.1. Computationally Intensive Region Identification

The SSD operation was given in Equation 6. Consider a search region described in Figure 8 represented as shown below:

$$\begin{bmatrix} f_{11} & \cdots & f_{1(2v_y+1)} \\ \vdots & \ddots & \vdots \\ f_{(2v_x+1)1} & \cdots & f_{(2v_x+1)(2v_y+1)} \end{bmatrix}$$

The search region contains multiple match blocks as observed in Figure 8. SSD is performed on each of the match blocks within the search region as seen in Figure 8. Since the dimension of a match block is $p^l \times q^l$, using the brute force technique to perform SSD on a match block requires $p^l q^l$ multiplications and $\left(p^l + q^l - 2\right)$ additions, leading to $\left(p^l q^l + p^l + q^l - 2\right)$ number of operations. Using the number of possible match blocks in a search region given in Equation (4), total number of operations to perform SSD within a search region is given by:

$$4\left(v_x + 1 - \frac{p^l}{2}\right)\left(v_y + 1 - \frac{q^l}{2}\right)\left(p^l q^l + p^l + q^l - 2\right) \tag{8}$$

This leads to $O\left(v_x v_y p^l q^l\right)$ complexity for one search region. Hence this is identified as the bottleneck of the algorithm due to the large number of operations.

6.2. Algorithm Modification

The reduce and re-use guidelines are merged as the following techniques not only reduce the number of operations but also aid in re-use of elements.

The SSD of match blocks have repeated operations. For example, consider the first two SSDs, SSD^1 and SSD^2 as shown in Figure 8. They are given by:

1. $SSD^1 = \sum_{i=0}^{p^l} \sum_{j=0}^{q^l} \left(f_{ij}\right)^2$
2. $SSD^2 = \sum_{i=0}^{p^l} \sum_{j=1}^{q^l+1} \left(f_{ij}\right)^2$

f_{12} and f_{22} are squared and added twice in the first two summations. These repeated computations increase with an increase in the size of the search region and number of markers.

A five-step modification is introduced that reduces the number of operations for SSD. In the first step, all the elements of the matrix are squared. In steps 2

and 3, row-wise summation of search region is computed such that the summation is limited to the marker size. Steps 4 and 5 are similar to steps 2 and 3 but are executed column-wise to obtain summation of the search region limiting it to marker size by columns. These steps are computed as follows:

Step 1: Squaring assuming the search region mentioned above, each element is squared as shown below:

$$\begin{bmatrix} f_{11}^2 & \cdots & f_{1(2v_y+1)}^2 \\ \vdots & \ddots & \vdots \\ f_{(2v_x+1)1}^2 & \cdots & f_{(2v_x+1)(2v_y+1)}^2 \end{bmatrix}$$

A total of $(2v_x+1)(2v_y+1)$ multiplications are required to complete this step.

Step 2: Prefix row Summation: For every row, perform a prefix sum as shown below:

$$\begin{bmatrix} f_{11}^2 & f_{11}^2+f_{12}^2 & \cdots & f_{11}^2+f_{12}^2+\cdots+f_{1(2v_y+1)}^2 \\ \vdots & \ddots & \ddots & \vdots \\ f_{(2v_x+1)1}^2 & f_{(2v_x+1)1}^2+f_{(2v_x+1)2}^2 & \cdots & f_{11}^2+f_{12}^2+\cdots+f_{(2v_x+1)(2v_y+1)}^2 \end{bmatrix}$$

If each element is computed independently, then there are $(2v_x+1)(2v_y+1)v_y$ additions. However, for a given row, if dependency is assumed between column summations, *i.e.*, the third column does not compute till the second column is computed, then for a given row there are $2v_y$ additions. Hence overall there are $4v_xv_y$ additions.

Step 3: Marker dimension row subtraction: For every row, based on the q^l, consecutive elements starting from $(q^l+1)^{th}$ column of each row are subtracted from each column (say r^{th} column) with $(r-q^l)^{th}$ column of every row, as shown below:

$$\begin{bmatrix} f_{11}^2 & \cdots & f_{12}^2+\cdots+f_{1(q^l+1)}^2 & \cdots & f_{1(2v_y+1-q^l)}^2+\cdots+f_{1(2v_y+1)}^2 \\ \vdots & & \ddots & & \vdots \\ f_{(2v_x+1)1}^2 & \cdots & f_{12}^2+\cdots+f_{(2v_x+1)(q^l+1)}^2 & \cdots & f_{(2v_x+1)(2v_y+1-q^l)}^2+\cdots+f_{(2v_x+1)(2v_y+1)}^2 \end{bmatrix}$$

This requires a total of $(2v_x+1)(2v_y+1-q^l)$ subtractions.

Step 4: Prefix column summation: Starting from the $(q^l)^{th}$ column, perform a column wise prefix sum as shown below:

$$\begin{bmatrix} f_{11}^2 & \cdots & f_{1(2v_y+1-q^l)}^2+\cdots+f_{1(2v_y+1)}^2 \\ f_{11}^2+f_{21}^2 & \ddots & \left(f_{1(2v_y+1-q^l)}^2+\cdots+f_{1(2v_y+1)}^2\right)+\left(f_{2(2v_y+1-q^l)}^2+\cdots+f_{2(2v_y+1)}^2\right) \\ \vdots & & \vdots \\ f_{11}^2+\cdots+f_{(2v_x+1)1}^2 & \cdots & \left(f_{1(2v_y+1-q^l)}^2+\cdots+f_{1(2v_y+1)}^2\right)+\cdots+\left(f_{(2v_x+1)(2v_y+1-q^l)}^2+\cdots+f_{(2v_x+1)(2v_y+1)}^2\right) \end{bmatrix}$$

similar to step 2, this step requires a total of $(2v_x)(2v_y+2-q^l)$ additions.

Step 5: Object dimension column subtraction: Similar to the row-wise submatrix summation, start from the $\left(p^l+1\right)^{th}$ row and $\left(q^l\right)^{th}$ column and perform the column wise summation as shown below:

$$\begin{bmatrix} f_{11}^2 & \cdots & f_{1\left(2v_y+1-q^l\right)}^2+\cdots+f_{1\left(2v_y+1\right)}^2 \\ \vdots & \ddots & \vdots \\ f_{21}^2+\cdots+f_{\left(p^l+1\right)1}^2 & \cdots & \left(f_{2\left(2v_y+1-q^l\right)}^2+\cdots+f_{1\left(2v_y+1\right)}^2\right)+\cdots+\left(f_{\left(p^l+1\right)\left(2v_y+1-q^l\right)}^2+\ldots+f_{\left(p^l+1\right)\left(2v_y+1\right)}^2\right) \\ \vdots & \ddots & \vdots \\ f_{\left(2v_x+1-p^l\right)1}^2+\cdots+f_{\left(2v_x+1\right)1}^2 & \cdots & \left(f_{\left(2v_x+1-p^l\right)\left(2v_y+1-q^l\right)}^2+\cdots+f_{\left(2v_x+1-p^l\right)\left(2v_y+1\right)}^2\right)+\cdots+\left(f_{\left(2v_x+1\right)\left(2v_y+1-q^l\right)}^2+\cdots+f_{\left(2v_x+1\right)\left(2v_y+1\right)}^2\right) \end{bmatrix}$$

This step requires a total of $\left(2v_x+1-p^l\right)\left(2v_y+1-q^l\right)$ additions. The summary of the computations required is given below in **Table 1**.

Since $\left(10v_x+3\right)>p^l,\left(2v_y+1\right)>q^l$, overall SSD per search region is reduced to a computational complexity of $O\left(v_xv_y\right)$ which is significantly lower than Equation 8 calculated using the brute force technique shown in **Figure 8**.

7. GPU Implementation

General-Purpose Computing on GPU (GPGPU) is a new parallelization domain to accelerate scientific and engineering computations in vision algorithms [17]. NVIDIA's Compute Unified Device Architecture (CUDA) is the earliest and most widely adopted programming model for GPGPU [18]. Hence, for the implementation of parallelized MKA, we have used NVIDIA GPU with CUDA programming model.

The GPUs, at a high level, are a collection of Streaming Multiprocessors (SMs), with each SM having the ability to concurrently execute large number of threads that are grouped in blocks, with each SM and thread having access to dedicated shared and register memory. Hence, the implementation of parallel algorithms has lot parameters such as number of threads, blocks, shared memory size, etc. to be determined.

In our previous work [19], we have proposed and implemented parallel version of the Scale Invariant Feature Transform (SIFT) algorithm on GPUs. In [19], we describe the architecture of GPUs (section 4) and a framework on how to calculate the parameters (section 5) to obtain a near optimal implementation

Table 1. Summary of the computations required for all match blocks' SSD.

Step number	Computations
1	$\left(2v_x+1\right)\left(2v_y+1\right)$
2	$4v_xv_y$
3	$\left(2v_x+1\right)\left(2v_y+1-q^l\right)$
4	$\left(2v_x\right)\left(2v_y+2-q^l\right)$
5	$\left(2v_x+1-p^l\right)\left(2v_y+1-q^l\right)$
Total	$\left(10v_x+3\right)\left(2v_y+1\right)-\left(6v_xq^l+2v_yp^l+2q^l+p^l-p^lq^l\right)$

of the parallel algorithm. We have adopted the same framework in this paper to calculate the amount of shared and register memory and to determine the number of blocks and threads to be used for execution of the algorithm.

The implementation for MKA's SSD operation can be written such that it uses different amount of register and shared memory. For example, the elements of the search region can be completely put on shared memory or a row or column of the search dimension can be moved to register memory. As such the blocks and threads will vary based on the implementation. While the memory access latency on shared memory is higher than register memory, the execution time for code using shared memory or shared and register memory combination are comparable. For this paper, the code was written to minimize the number of registers and maximize the shared memory, since this resulted in a cleaner and single source code for different search region dimensions without compromising much of performance. Using the minimized registers code and using NVIDIA profiler to determine the total amount of shared and register memory, the number of threads per block and the number of blocks was determined using Equations 28 and 30 respectively in [19] for each search region. In general, the number of threads per block was found to be 256 for search dimensions having less than 2000 pixels and 512 for search dimensions having more than 2000 pixels. The number of blocks varied based on the search region dimension.

8. Results and Analysis

In this section, we present and analyze the effectiveness of the HMCS in tracking the motion of a human body part which performing PTE. First, to determine the suitability of the HMCS for real-time applications we compare the performance of sequential Klein's algorithm with MKA. Second, we compare the performance of the parallel implementation of Klein's algorithm and MKA, parallelized using standard libraries. Lastly, we look at the performance of the parallel MKA, parallelized using our framework.

For analysis, the programs are executed on Intel Xeon CPU, having two E5620 processors operating at 2.40 GHz and running a 64 bit Windows 7 Pro Operating System. The sequential programs are executed on a 64-bit Matlab R2014b software. Matlab's original core has been developed from LINPACK and EISPACK [20]. LINPACK and EISPACK have proven to be computationally effective ways to solve linear algebra problems [21]. Hence we use this software to obtain the reference time for sequential programs. For parallel program execution, the CPU is equipped with a NVIDIA Tesla C2075 card. The card is equipped with 448 cores and 6 GB of memory for general computations. Though K20x and K40m cards seem to be a good option for GPU parallelization, we limit ourselves to a low-cost GPU such as C2075.

For developing parallel Klein's algorithm and MKA code using standard libraries, a combination of cuLA and cuBLAS library package was used. cuLA is a CUDA based accelerated Linear Algebra (LA) package provided by NVIDIA for GPUs. Similarly, cuBLAS library package is another accelerated Basic Linear Al-

gebra Subprograms (BLAS) library provided by NVIDIA for GPUs. Lastly, for profiling NVIDIA's Visual Profiler, NVVP, compatible with CUDA 6.0 and Microsoft Visual 2015 was used to profile the code.

We use execution time as a measure of performance. Low execution time is considered to be better. Each implementation of an algorithm for a given search dimension is considered as one simulation and the execution time is collected. Each simulation is executed one thousand times and AET is computed. The standard deviation of AET for all simulations was observed to be under 6.5%. Each simulation has been verified by manually inspecting the tracked positions on the image. Data, for verification of algorithms, is taken from videos under indoor computer lab conditions using 60fps at 1080p resolution. For the tracking algorithm, the marker size is generally small and the search region varies from image to image. For the simulations, $p^l = q^l = 8$ and $v_x = v_y = \{32, 64, 128, 256, 512\}$ is considered.

8.1. Performance of Sequential Execution

Figure 9 shows the speed-up of sequential execution of MKA over Klein's algorithm for tracking a single marker. The speed-up is obtained by dividing Klein's algorithm AET with MKA's AET. As expected, the MKA outperforms Klein's algorithm by a significant margin. This is expected since MKA has significantly lower computations when compared to Klein's algorithm. Since, Klein's algorithm has $O\left(v_x v_y p^l q^l\right)$, and MKA has $O\left(v_x v_y\right)$, and $p^l = q^l = 8$, we could expect at least 64x performance improvement. The speedup observed is in close agreement with expectation, especially with higher search dimensions where speedup saturates at 132x.

8.2. Performance of Parallel Execution Using Standard Libraries

Figure 10 shows the AET of the Klein's algorithm and MKA algorithm paralle-

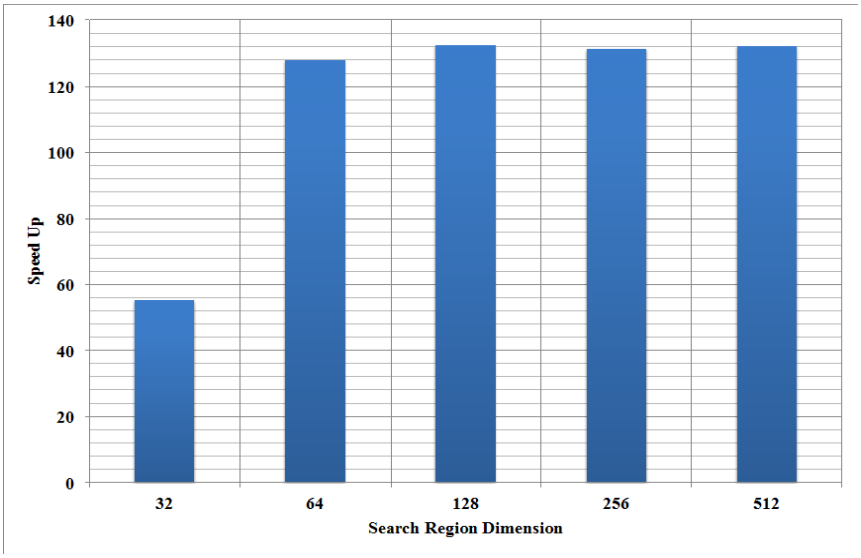

Figure 9. Speed-up of MKA in comparison with Klein's algorithm.

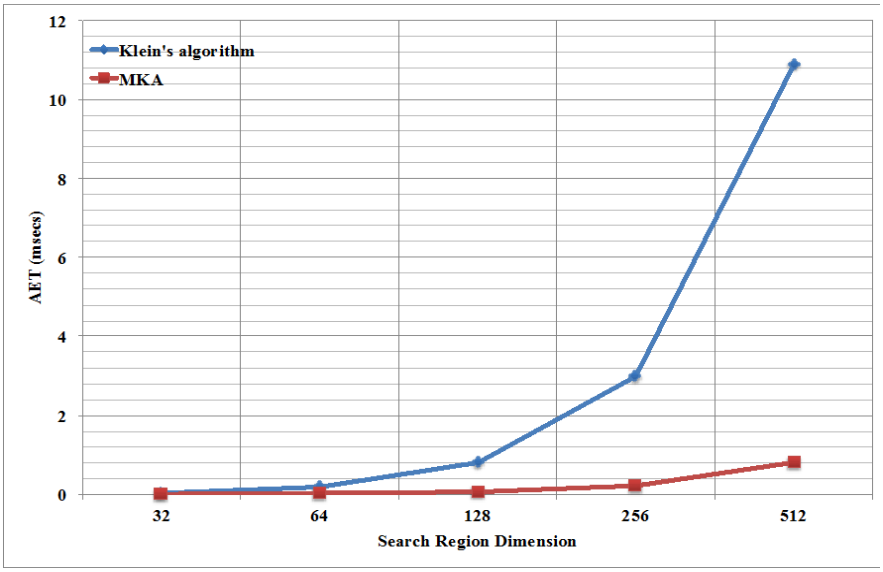

Figure 10. Performance comparison of parallel implementations of Klein's algorithm and MKA using standard libraries.

lized using standard libraries. Similar the sequential version, the parallel MKA performs better than Klein's algorithm. This is to be expected due to the difference in the number of computations in both algorithms, as mentioned before. As the search region dimension region increases, the AET of Klein's algorithm increases non-linearly. This is because of the higher amount of a) computations and b) data transfers occurring between library calls. For MKA, these are much lower, providing a near linear increase in AET.

8.3. Performance of Parallel MKA

Figure 11 shows the AET of our version of parallel MKA for different search dimensions. Due to a single data transfer between CPU and GPU in each direction, and highly optimized code, the AET is found under a millisecond, even for bigger search regions. The low AET and good scalability indicate the suitability of our implementation of parallel MKA for real-time applications.

8.4. Application of HMCS to Elbow PTE

The video frames of a patient performing the elbow flexion and extension exercise shown in Figure 2 is presented again for the completion. In Figure X, the patient elbow has five markers stuck on the hand, which are tracked by the HMCS. The patient in Figure 12, starts at full extension or supine position and bends the elbow to reach the full flexion position.

The patient when performing the elbow exercise is expected to maintain the arm (structure between shoulder and elbow joint) rigid and at the horizontal position. The ROM is measured from the zero value position. The ROM angle for flexion is measured ranging from 0˚ to +90˚ to the left of the zero value position. The ROM angle for extension is 0˚ to −90˚ to the right of the zero value position. Using the HMCS, the markers are tracked in Figure X between subsequent

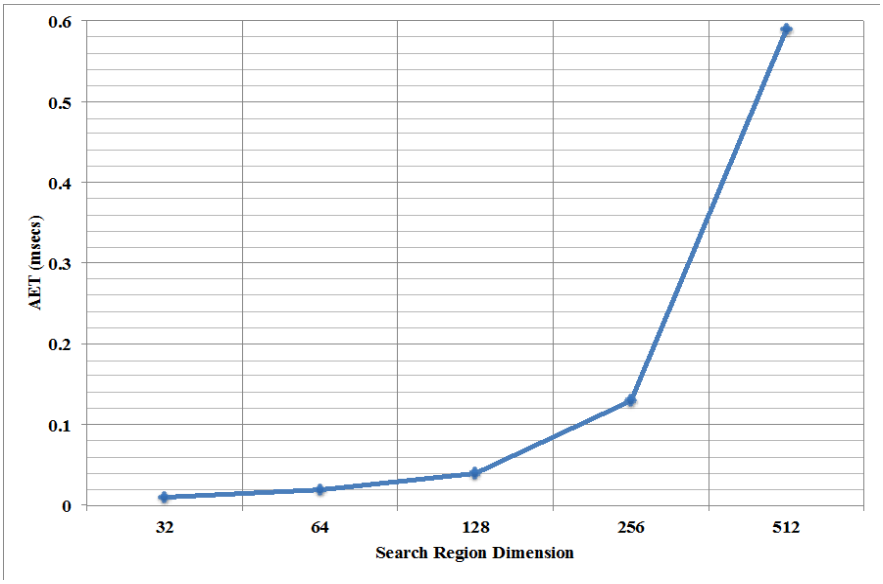

Figure 11. AET of MKA parallelized using framework for different region dimensions.

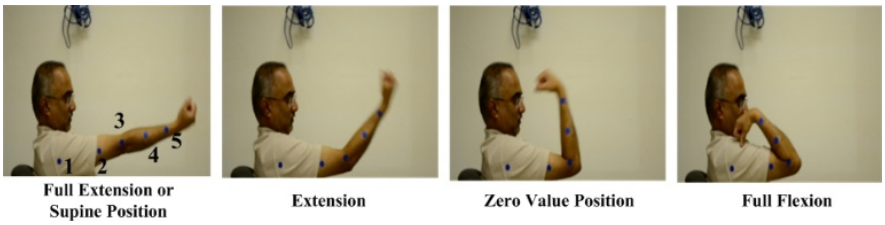

Figure 12. Video frames of flexion and extension exercises.

frames and the ROM angle between marker pairs (1, 3), (3, 4) and (3, 5) are determined. The measured ROM angles for the exercise in Figure X are shown in Figure X1. In **Figure 13**, we can notice that the patient extension and flexion ROMs are consistently close to −90˚ and 40˚ respectively. According to the American Medical Association [22] normal flexion ROM values for adults range from 50˚ to 60˚ from the zero value position.

Therefore, the PT based on the measured ROM can see the patient has some physical issues with elbow flexion and provide appropriate feedback. Furthermore in **Figure 13**, the ROM measurements between markers (1, 3) indicate that the patient is not maintaining the arm in the rigid position. In **Figure 14** and **Figure 15**, the angular velocity and acceleration of the elbow motion while performing the flexion and extension exercise are shown.

In **Figure 14**, it can be seen that the patient is performing the elbow motion with a significantly large variation in angular velocity and correspondingly with an angular acceleration and deceleration as shown in **Figure 15**. The large variation in angular velocity indicates the patient is not performing the exercises at a sustained rate. Varying angular acceleration indicates the patient having to exert different forces during the flexion and extension parts of the elbow PTE. Using the angular velocity and acceleration/deceleration data, a PT can now provide a feedback to the patient in real-time about whether the motion is smooth or jerky

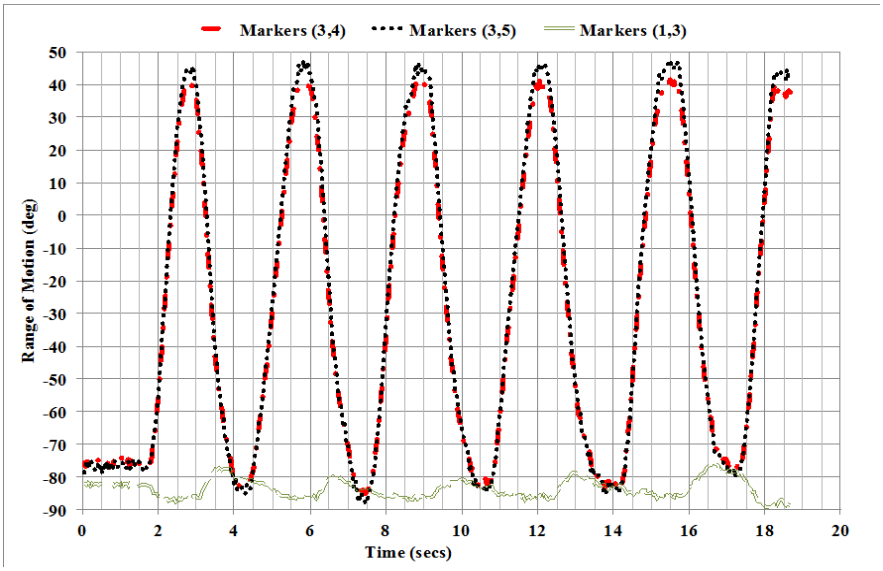

Figure 13. ROM measurements between pair of markers.

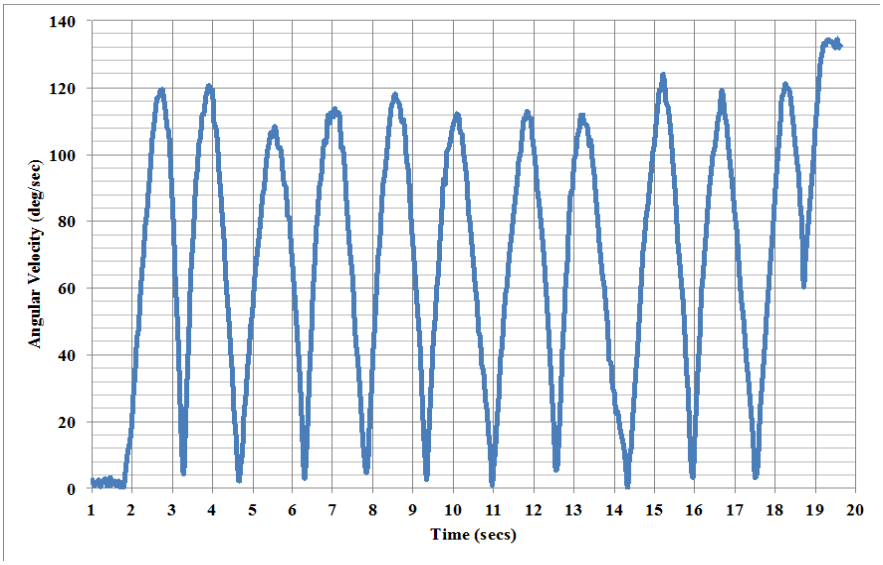

Figure 14. Angular velocity of the elbow motion.

and whether the patient is experiencing fatigue.

In Figure 16, the error in ROM measurement between using the HMCS and a Goniometer is shown. The error varies anywhere from −2.6˚ to +1.4˚ which is well within accepted error value of 5˚ [1]. This low value of error when compared with the Goniometer measurements indicates our technique of measuring the patient performance in near real-time can be used in clinical settings. Furthermore, our implementation on the GPU allows us to track all of the markers in a HD frame in 1.77 msecs making the HMCS capable of operating in real-time.

9. Conclusions & Future Work

An analysis of the Klein's algorithm indicated that there was a potential to re-

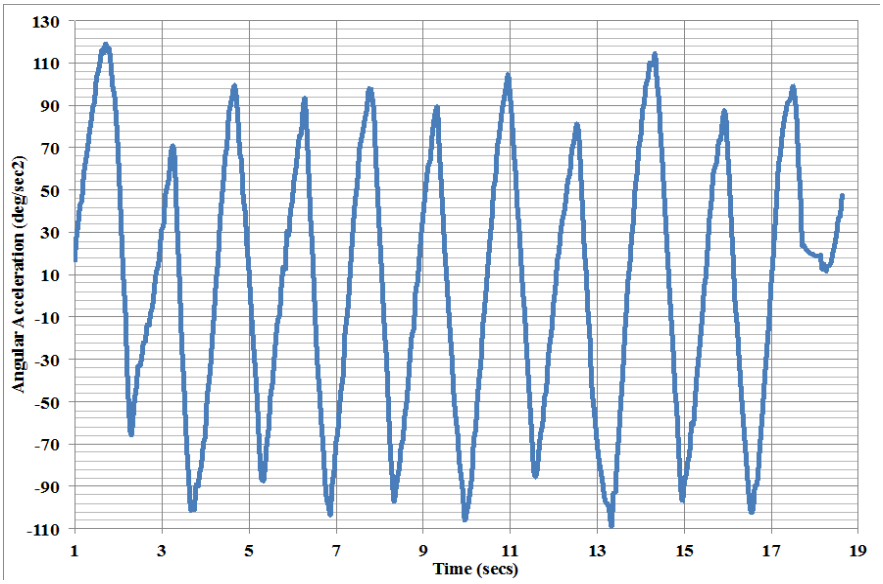

Figure 15. Angular acceleration of the elbow motion.

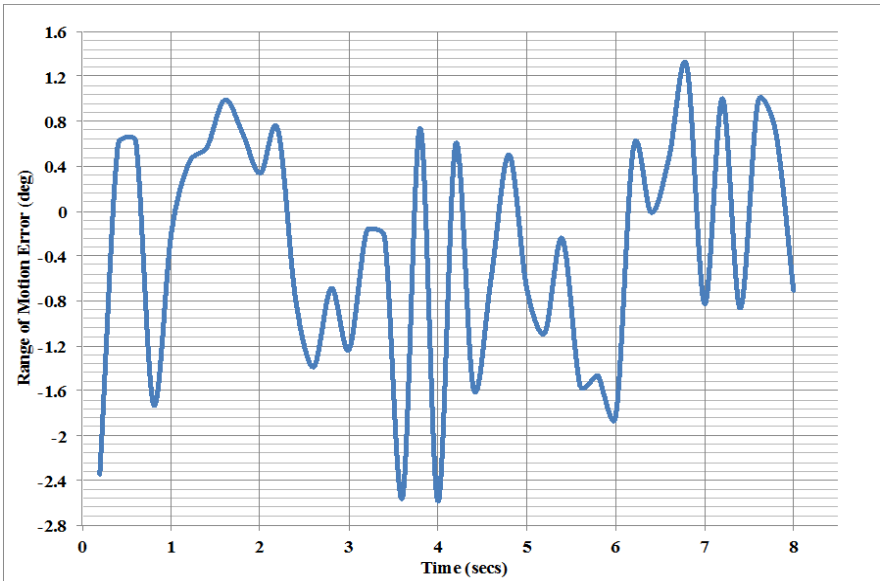

Figure 16. Measurement error between HMCS and Goniometer.

duce the number of computations. We modified the algorithm, to facilitate better parallelization on GPU architectures in two phases—first, refactoring the algorithm to have lesser number of operations and enhanced parallelism, and secondly, optimizing the data to obtain better parallelism for GPU architectures. We compared the effectiveness of our algorithm, with Klein's algorithm for sequential and Klein's algorithm and MKA for parallel implementations. For sequential implementation, MKA performed much better than its predecessor algorithm. To understand the effectiveness of our parallel implementations, Klein's algorithm and MKA were parallelized using the cuLA and cuBLAS library and compared with our parallel implementation. The results showed that our parallel implementation of MKA has the lowest execution time. Hence, our

parallel implementation of MKA could be implemented on GPUs for real-time applications using high-resolution frames with a high number of markers per frame. We have also demonstrated the use of the MKA in determining the ROM of the elbow of a patient performing PTE in real-time. Using the real-time ROM measurement, a PT can identify any issues that a patient is experiencing while performing the PTE and provide feedback in real-time.

We have also shown that it is possible to measure the ROM using a single camera with high accuracy.

The next stage of work would involve constructing 3D images at the clinical end for a PT and MD to examine subtle muscle movements. We envision determining the point cloud of a body part and manipulate the point cloud in 3D using the tracked markers.

References

[1] Norkin, C.C. and White, D.J. (2017) Measurement of Joint Motion: A Guide to Goniometry. 5th Edition, F. A. Davis Company, Philadelphia.

[2] Muller, P., Begin, M.A., Schauer, T. and Seel, T. (2016) Alignment-Free, Self-Calibrating Elbow Angles Measurement Using Inertial Sensors. *IEEE-EMBS International Conference on Biomedical and Health Informatics*, **21**, 312-319. https://doi.org/10.1109/JBHI.2016.2639537

[3] Chang, R.K.Y., Lau, S.H., Sim, K.S. and Too, M.S.M. (2016) Kinect-Based Framework for Motor Rehabilitation. *International Conference on Robotics*, *Automation and Sciences*, Malacca City, 9 March 2017, 1-4.

[4] Zhao, W. (2016) On Automatic Assessment of Rehabilitation Exercises with Real-time Feedback. *IEEE International Conference on Electro Information Technology*, Grand Forks, 8 August 2016, 376-381.

[5] Chen, Y.C., Lee, H.J. and Lin, K.H. (2015) Measurement of Body Joint Angles for Physical Therapy Based on Mean Shift Tracking Using Two Low Cost Kinect Images. 37*th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Milan, 25-29 August 2015, 703-706.

[6] Liu, Z., Huang, J., Han, J., Bu, S. and Lv, J. (2016) Human Motion Tracking by Multiple RGBD Cameras. *IEEE Transactions on Circuits and Systems for Video Technology*, 6 May 2016, 1-1.

[7] Penelle, B. and Debeir, O. (2013) Human Motion Tracking for Rehabilitation Using Depth Images and Particle Filter Optimization. 2*nd International Conference on Advances in Biomedical Engineering*, Tripoli, 11-13 September 2013, 211-214. https://doi.org/10.1109/icabme.2013.6648885

[8] Klein, G. and Murray, D. (2007) Parallel Tracking and Mapping for Small AR Workspaces. 6*th IEEE and ACM International Symposium on Mixed and Augmented Reality*, Nara, 13-16 November 2007, 225-234. https://doi.org/10.1109/ismar.2007.4538852

[9] Scherer, S.A., Dube, D. and Zell, A. (2012) Using Depth in Visual Simultaneous Localisation and Mapping. *IEEE International Conference on Robotics and Automation*, Saint Paul, 14-18 May 2012, 5216-5221.

[10] Schauwecker, K., Ke, N.R., Scherer, S.A. and Zell, A. (2012) Markerless Visual Control of a Quad-Rotor Micro Aerial Vehicle by Means of On-Board Stereo Processing. *Autonomous Mobile Systems Conference*, 11-20.

[11] Jang, H.-S., Jeong, J.-Y., Kim, Y.-H., Yoon, Y.-J. and Ko, S.-J. (2011) Augmented

Reality with High Frame Rate for Low Computational Power Devices. *IEEE International Conference on Consumer Electronics*, Berlin, 6-8 September 2011, Berlin, 274-275. https://doi.org/10.1109/ICCE-Berlin.2011.6031808

[12] Riazuelo, L., Civera, J. and Montiel, J.M.M. (2014) C2TAM: A Cloud Framework for Cooperative Tracking and Mapping. *Robotics and Autonomous Systems*, **62**, 401-413. https://doi.org/10.1016/j.robot.2013.11.007

[13] Shimamura, J., Morimoto, M. and Koike, H. (2011) Robust vSLAM for Dynamic Scenes. *IAPR Conference on Machine Vision Applications*, Nara, 13-15 June 2011, 344-347.

[14] Klein, G. and Murray, D. (2008) Improving the Agility of Keyframe-Based SLAM. *Computer Vision-ECCV*, Berlin, 12-18 October 2008, 802-815. https://doi.org/10.1007/978-3-540-88688-4_59

[15] Klein, G. and Murray, D. (2008) Compositing for Small Cameras. 7*th IEEE/ACM International Symposium on Mixed and Augmented Reality*, Washington DC, 15-18 September 2008, 57-60. https://doi.org/10.1109/ismar.2008.4637324

[16] Klein, G. and Murray, D. (2009) Parallel Tracking and Mapping on a Camera Phone. 8*th IEEE International Symposium on Mixed and Augmented Reality*, Washington DC, 19-22 October 2009, 83-86. https://doi.org/10.1109/ismar.2009.5336495

[17] Fung, J. and Mann, S. (2004) Using Multiple Graphics Cards as a General Purpose Parallel Computer: Applications to Computer Vision. *Proceedings of the* 17*th International Conference on Pattern Recognition*, **1**, 805-808. https://doi.org/10.1109/ICPR.2004.1334339

[18] NVIDIA, NVIDIA CUDA. http://www.nvidia.com/object/cuda_home_new.html

[19] Kumar, R., Muknahallipatna, S. and McInroy, J. (2016) An Approach to Parallelization of SIFT Algorithm on GPUs for Real-Time Applications. *Journal of Computer and Communications*, **4**, 18-50. https://doi.org/10.4236/jcc.2016.417002

[20] Ramaswamy, S., Hodges, E.W. and Banerjee, P. (1996) Compiling MATLAB Programs to ScaLAPACK: Exploiting Task and Data Parallelism. *Proceedings of International Conference on Parallel Processing*, Honolulu, 15-19 April 1996, 613-619. https://doi.org/10.1109/IPPS.1996.508120

[21] Dongarra, J.J., Moler, C.B. and Bunch, J.R. (1979) LINPACK Users' Guide. Version 1.12, Society for Industrial and Applied Mathematics, Philadelphia. https://doi.org/10.1137/1.9781611971811

[22] Rondinelli, R.D. (2007) Guides to the Evaluation of Impairment. 6th Edition, American Medical Association, Chicago.

Nomenclature

BLAS—Basic Linear Algebra Subprograms;
CUDA—Compute Unified Device Architecture;
FAST—Features from Accelerated Segment Test;
GPGPU—General Purpose Graphics Processing Unit;
GPU—Graphics Processing Unit;
HD—High Definition;
HMCS—Human Motion Capture System;
IMU—Inertial Measurement Units;
LA—Linear Algebra;
MKA—Modified Klein's Algorithm;
PT—Physical Therapist;
PTE—Physical Therapeutic Exercise;
RGB—Red Green Blue;
ROM—Range of Motion;
S1—Stage 1;
S2—Stage 2;
SIFT—Scale Invariant Feature Transform;
SM—Streaming Multiprocessor;
SSD—Squared Summation of Differences.